Computer Aided Medical Procedures
Prof. Dr. Nassir Navab

Dissertation

Learning-based Single View 3D Completion

Yida Wang

Fakultät für Informatik
Technische Universität München

Technische Universität München
TUM School of Computation, Information and Technology

Learning-based Single View 3D Completion

Yida Wang

Vollständiger Abdruck der von der TUM School of Computation, Information and Technology der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitz: Prof. Dr. Claudia Eckert

Prüfer der Dissertation:
1. Priv.-Doz. Dr. Federico Tombari
2. Prof. Dr. Daniel Cremers
3. Asst. Prof. Dr. Shuran Song

Die Dissertation wurde am 18.05.2022 bei der Technischen Universität München eingereicht und durch die School of Computation, Information and Technology am 03.11.2022 angenommen.

Yida Wang
Learning-based Single View 3D Completion
Dissertation, Version 3.0

Technische Universität München
TUM School of Computation, Information and Technology
Boltzmannstraße 3
85748, Garching bei München, Deutschland

Abstract

Fed with a partial scan, single-view 3D completion models are expected to generate the completed geometry as an inference result. Applications such as AR/VR and autonomous driving could benefit from targets being presented in their complete structures, while occluded geometries are missing due to limited camera view. Learning-based 3D completion approach can reveal the occluded regions conditioned on a single view sample with the help of large-scale training data.

It is straightforward to adapt existing image-space methods to design encoder-decoder architectures constructed with convolutions in volumetric space to process local and global information. Nonetheless, the slow inference speed limits the practicality of the volumetric completion. The computation and memory costs increase exponentially regarding the resolution because of the convolutions. Apart from the methodology designs, the lack of a large amount of real 3D supervision, because of the difficulty in annotating, also makes it hard to optimize models in volumetric space for real scenarios.

Due to the memory cost, the volumetric completion is struggling to reconstruct finer details practically. Such difficulty motivated us further to investigate the works on a sparse data format, the point cloud, for 3D completion. Since the point cloud does not present information in empty spaces, such a sparse 3D representation can be with flexible local resolution depending on specific local geometric complexity. However, a straightforward adaptation with a similar encoder-decoder architecture as the volumetric methods does not work because of the unorganized structure of point cloud. This situation implies that the well-known operators employed on images or volumes cannot be applied to point cloud.

Therefore, in this dissertation, we propose solutions to address all the aforementioned problems in both volumetric data and point cloud for the task of 3D completion from a single view. Notably, while other point cloud methods are limited to reconstructing a synthetic single object, our proposed approach alleviated this limitation by reconstructing real scenes. Inspired by the improvements from all the completion methods, we also demonstrate their generalizability by expanding their applications to other tasks, e.g. , hand pose estimation, point cloud segmentation, and natural language processing.

Zusammenfassung

Gegeben ein Teilscan, versuchen 3D-Vervollständigungsmodelle von der Einzelansicht die vollständige Geometrie zu erzeugen. Anwendungen wie AR/VR und autonomes Fahren könnten davon profitieren, dass Objekte in ihrer vollständigen 3D Struktur dargestellt werden, obwohl verdeckte Geometrien eigentlich aufgrund der begrenzten Kamerasicht fehlen. Ein auf Lernen basierender Ansatz zur 3D-Vervollständigung kann die verdeckten Regionen auf der Grundlage einer einzigen Ansichtsstichprobe mit Hilfe von umfangreichen Trainingsdaten aufdecken.

Es ist einfach, bestehende Bildverarbeitungsmethoden zu adaptieren, um Encoder-Decoder-Architekturen zu entwickeln, die durch Convolutions im volumetrischen Raum lokale und globale Informationen verarbeiten. Dennoch schränkt die langsame Inferenzgeschwindigkeit die praktische Anwendbarkeit der volumetrischen Vervollständigung ein. Die Rechen- und Speicherkosten steigen aufgrund der Convolutions exponentiell mit der Auflösung an. Abgesehen von der Methodik erschwert das Fehlen einer großen Menge echter 3D-Trainingsdaten die Optimierung von Modellen im volumetrischen Raum für reale Szenarien, da es sehr schwierig ist, solche 3D-Daten zu annotieren.

Aufgrund der Speicherkosten ist die volumetrische Vervollständigung nur schwer in der Lage, feinere Details praktisch zu rekonstruieren. Diese Schwierigkeit hat uns dazu motiviert, die Arbeiten an einem spärlichen Datenformat, der Punktwolke, für die 3D-Vervollständigung weiter zu untersuchen. Da die Punktwolke keine Informationen in leeren Räumen enthält, kann eine solche spärliche 3D-Darstellung eine flexible lokale Auflösung in Abhängigkeit von der spezifischen lokalen geometrischen Komplexität handhaben. Eine einfache Anpassung mit einer ähnlichen Encoder-Decoder-Architektur wie bei den volumetrischen Methoden funktioniert jedoch aufgrund der unorganisierten Struktur der Punktwolke nicht. Dies bedeutet, dass die bekannten Operatoren, die für Bilder oder Volumen verwendet werden, nicht auf Punktwolken angewendet werden können.

Daher schlagen wir in dieser Dissertation Lösungen vor, um alle oben genannten Probleme sowohl bei volumetrischen Daten als auch bei Punktwolken für die Aufgabe der 3D-Vervollständigung aus einer einzigen Ansicht zu lösen. Während andere Punktwolken-Methoden auf die Rekonstruktion eines einzelnen synthetischen Objekts beschränkt sind, wird diese Einschränkung durch den von uns vorgeschlagenen Ansatz dank der Rekonstruktion von realen Szenen gemildert. Inspiriert durch die Verbesserungen aller Vervollständigungsmethoden demonstrieren wir auch ihre Verallgemeinerbarkeit, indem wir ihre Anwendungen auf andere Aufgaben ausweiten, z. B. auf die Schätzung der Handhaltung, die Segmentierung von Punktwolken und die Verarbeitung natürlicher Sprache.

Acknowledgments

It has been a pleasant experience spending my time in Munich as a Ph.D. Thanks to my supervisor Federico, mentor Nassir, and David for your support and advice. Thanks, Fede, for my impressive tours during academic visits to University of Nottingham and Google Zurich. As the leader for CAMP, Nassir organizes the entire lab while being willing to help whenever we have specific issues. CAMP gives me more than enough freedom to focus on my interests. David is a magician; thanks for your effort in getting me out of the mist when I do not have a straightforward way to direct my work. All these vivid presentations of our works supplement our papers with the soul. The *Project Yoda* have been more than 50 pages. Moreover, Martina and Frau Fischer; you have played significant roles during my Ph.D. study to help solve problems such as making all these applications in Germany.

I am proud to have all these works shared during my research at TUM, and they would only be so successful with others' help. A big thumb for Shuncheng's fantastic volumetric cuboids, which are used to visualize results in my ForkNet submission vividly; Fabi's contribution to my Zusammenfassung is crucial for this dissertation. Johanna and Helisa are always willing to help me check the visual satisfaction of the qualitative figures. Big thanks to Han for many efforts in clarifying whether my works are clearly presented in the end. As my office mate, Leslie and Anees organized the Netflix family plan for years, excellent content. I also appreciated the discussion with Yan and Shun-Cheng on topics of general 3D, which lighted some of my blind points in 3D vision. I am glad to learn from supervising my students, including Yafei, Peter, and Mahsa, for their thesis. Also, thanks for all those small yet fruitful discussions with colleagues in CAMP, including Nikolas, Iro, Keisuke, Christian, Yafei, Pietro, Peter, Evin, Yanyan, Johanna, Helisa, Huseyin, Mahdi, Sergey, Mahsa, Benni and others.

My officemates, Anees, Leslie, Mahdi, and Yan, always tolerate my messy desks. May the big smiles on your face be with you all the time. I would also be grateful for the support from my family to try out new opportunities; the strong willingness to explore everything makes great things happen. All good memories from my summer school in Sicily, have enjoyed my time at the football court there and the basketball court here as well in Garching, ski resorts in Austria, and the crystal clear lake in Faaker See. Thanks for the in-depth career coaching from Fede and Nan, the precious fellow opportunity provided by Irman, and the solid intern support from Yiru, Sachin, Markos, and Matthias.

The cat and hedgehog coming from nowhere are always interested in visiting me when they are hungry for snacks; there are other incredible creatures, including the spider, who can tolerate the high temperature of my office in the summer.

Contents

Chronological List of Authored Publications

2022

[1] **Y. Wang**, Y. Shen, D. J. Tan, F. Tombari, and S. S. Talathi. "SecNet: Semantic Eye Completion in Implicit Field". In: *Annual Conference on Neural Information Processing Systems (NeurIPS).*

[2] A. Savkin, **Y. Wang**, S. Wirkert, N. Navab, and F. Tombari. "Lidar Upsampling With Sliced Wasserstein Distance". In: *IEEE Robotics and Automation Letters (RAL).*

[3] **Y. Wang**, D. J. Tan, N. Navab, and F. Tombari. "Self-supervised Latent Space Optimization with Nebula Variational Coding". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI).*

[4] **Y. Wang**, D. J. Tan, N. Navab, and F. Tombari. "Learning Local Displacements for Point Cloud Completion". In: *Conference on Computer Vision and Pattern Recognition (CVPR).*

[5] **Y. Wang**, D. J. Tan, N. Navab, and F. Tombari. "SoftPool++: An Encoder-Decoder Network for Point Cloud Completion". In: *International Journal of Computer Vision (IJCV).*

2020

[6] **Y. Wang**, D. J. Tan, N. Navab, and F. Tombari. "Softpoolnet:Shape descriptor for point cloud completion and classification.". In: *European Conference on Computer Vision (ECCV).* **[Oral]**

[7] Y. Li, N. Brasch, **Y. Wang**, N. Navab, F. Tombari. "Structure-SLAM: Low-Drift Monocular SLAM in Indoor Environments." In: *International Conference on Intelligent Robots and Systems (IROS).*

2019

[8] **Y. Wang**, D. J. Tan, N. Navab, and F. Tombari. " Forknet: Multi-branch volumetric semantic completion from a single depth image." In: *International Conference on Computer Vision (ICCV).*

[9] Y. Gao*, **Y. Wang***, F. Pietro, N. Navab, and F. Tombari. "Variational Object-Aware 3-D Hand Pose From a Single RGB Image" In: *IEEE Robotics and Automation Letters (IROS-RAL).*

2018

[10] **Y. Wang**, D. J. Tan, N. Navab, and F. Tombari. "Adversarial Semantic Scene Completion from a Single Depth Image." In: *International Conference on 3D Vision (3DV).*

Part I

Introduction

1 Introduction

In the starting chapter of this dissertation, Section 1.1 briefly introduces the motivations and background of applying machine learning to tasks in 3D space, especially addressing the advantages of adopting the 3D methods to solve real-world tasks compared to 2D methods. Given a specific type of input in an inference system, which is partial scans, the overview of single-view 3D understanding is then introduced in Section 1.2. Finally, Section 1.3 lists the structure of this dissertation, which focuses more specifically on our contributions.

1.1 Background

Intuitively, the world is perceived as a 3D Euclidean space. 3D space reveals more useful information for complex tasks than 2D information because objects naturally exist in 3D. The eyes of most herbivores are positioned on the sides of their heads so they can see as much of their environment as possible. But the predators have a stereoscopic vision, which is helpful for complicated tasks, e.g. gauging the distance to their prey when they hunt. Such a natural fact shows that a good understanding of detailed 3D structures and distances is crucial if actions are more complicated. Human vision works similarly to predators' vision. The simultaneous integration of the images from both of our eyes results in a 3D understanding which is also enhanced by the memorized related informations in our brain.

Human societies depend more on 3D understanding instead of 2D for industrial applications such as autonomous driving, automatic sorting in the factory, license plate recognition, Simultaneous Localization and Mapping (SLAM), augmented reality (AR), virtual reality (VR), etc. By comparing information towards a scene from two vantage points, 3D information can be extracted by examining the relative positions of objects in the two panels. This is similar to the biological process of stereopsis. Specifically for scenarios involving autonomous driving, there are indeed solutions that work solely in 2D space. For example, LaneNet [11] segments the lanes and marks on the road in 2D space, which guilds the 3D maneuver of the vehicle so that the vehicle can cruise on the current lane on the highway. To avoid a collision, object detectors such as DETR [12] predict 2D bounding boxes using features extracted from images such as ResNet [13].

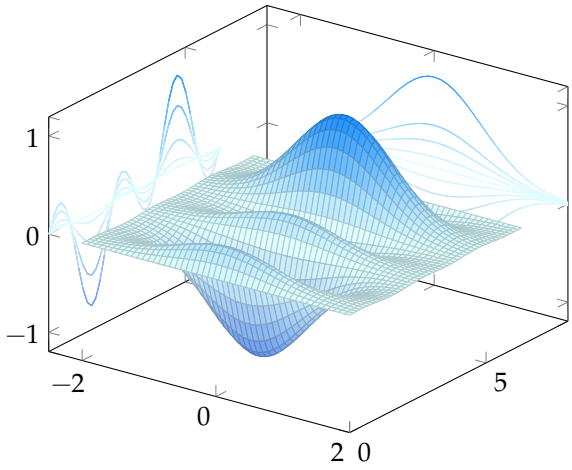


Figure 1.1 An illustration of information lost due to 2D projections from the edges of a 3D mesh with limited views.

However, a colletion of projected 2D data lost part of the original 3D structures, especially when the projection directions are limited. As shown in Fig. 1.1, neither of these two 2D projection from a 3D curvature mesh can reveal the correct 3D geometry without ambiguity. So instead, some follow-up models supplemented with the 3D inference results can be more confidently used in solving object detection and segmentation tasks, e.g. Anchor DETR [14] and PETR [15]. Such a trend proves that using 3D information helps improve recognition accuracy without over-consuming computational resources. Besides autonomous driving, AR and VR also build applications on top of learned parametric models, which depend more frequently on 3D approaches. Augmenting the observed scenes with realistic visual effects involves estimating spaces that are not observable. Such estimation for occluded structures could be achieved by either positioning known 3D models to align them with an observed partial scan [16] or reconstructing the occluded area with a learned completion model [17, 18]. This dissertation focuses on the second approach, which takes on the observed structures to infer the structures that are out of view.

Ideally, the entire 3D space of interest could be progressively constructed with the help of 3D scanners for an autonomous robot to process and take action once the movement of the scanners is not restricted by either the environment or the mechanical components themselves. In special scenarios, like moving vehicles and drones, only 3D information partially captured from a certain camera view is feasible. Resulting in problems in which crucial structures are occluded so that cars or drones can not take further actions reasonably. To handle such a problem of occlusions, 3D completion from a single view plays an important role in extrapolating the complete structures.

Although the RGB image is the most feasible single-view data, such input has no geometric information that could be used as part of expected 3D structures in the output. Especially for scene completion, image-to-scene completion does not satisfy results because the performance of many deep architectures is highly dependent on a global latent feature in which local structures are not explicitly included. Indeed, there are single-view works progressively completing scenes from RGB [19] with several steps which include room layout estimation [20], object detection, and object completion [21], yet such progressive reconstruction is tedious and does not perform well when the camera movement is constrained. Another solution is to rely on depth prediction from RGB [22, 23, 24]. However, the predicted depth from RGB is less precise than a back-projected 3D surface from the depth scan. In addition, off-the-shelf depth sensors such as Kinect [25] and RealSense [26] can deliver scans in real-time, which are more efficient than depth prediction.

1.2 3D Reconstruction from a Single View

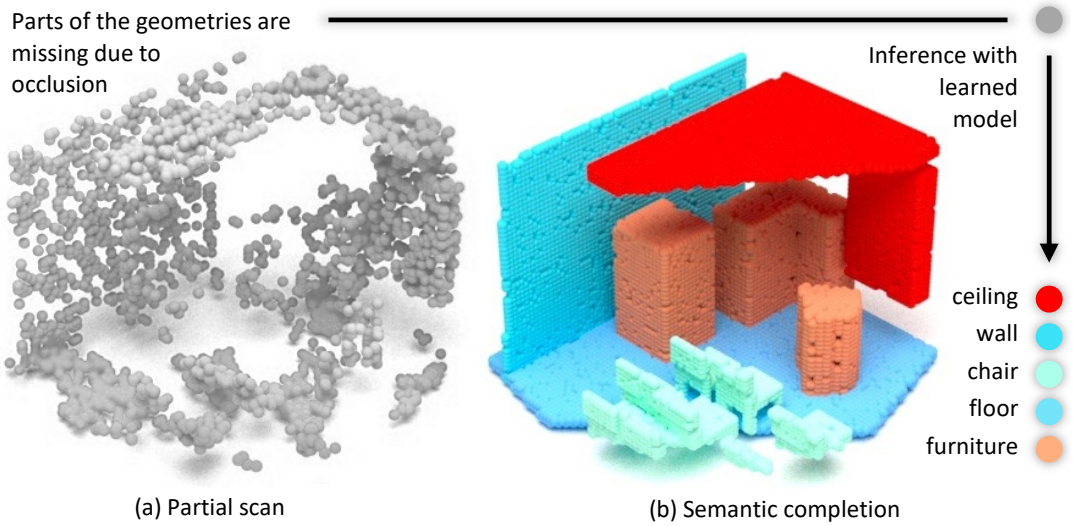


Figure 1.2 An exemplar semantic completion from a 3D partial scan in (a) presented in point cloud which is back-projected from a single depth image with camera intrinsic, and the expected semantically completed 3D scene in (b).

As mentioned in Section 1.1, 3D understanding is critical for modern life. Here, we refer to 3D understanding specifically for tasks with the partial scans as input, e.g. single-view 3D reconstruction. Given a depth image as illustrated in Fig. 1.2, the expected 3D understanding enhances the partial observations with semantics and completed geometries. These approaches are focused on two aspects of the topic of 3D machine vision, which are the capacity of

1. accurately recognizing the surrounding objects, together with,
2. building the geometric structure of the scene which assembles the 3D relation of the objects of interest.

The former is referred to as semantic segmentation, while the latter is the 3D completion. By merging two tasks into the topic of semantic completion with an end-to-end model, semantic segmentation and completion could be estimated by the same deep architecture. In such a manner, the estimated semantics could be leveraged to improve the completion as well.

In this dissertation, we focus on 3D object completion and 3D semantic scene completion from a single view. The bulk of our research is conducted on 3D completion, while segmentation is only used for scenes.

1.3 Structure of the Dissertation

Given the introduced research background and an overview of current progress on the topic of our targeted tasks, this section briefly outlines the main structure of this dissertation starting from Chapter 2 to 7.

Chapter 2: Fundamental Theories. This chapter starts by describing deep learning related topics in Section 2.1, including the basic operations in Section 2.1.1 and the common architectures built from these operators in Section 2.1.4. Apart from the architecture, we also explain how to train it with an optimizer in Section 2.1.4. The succeeding parts of this chapter introduce some major learning techniques that are related to this dissertation, such as adversarial learning in Section 2.2.1, variational inference in Section 2.2.3 and metric learning in Section 2.2.2.

Chapter 3: Learning-based Single View 3D Completion. We start the chapter by introducing the 3D completion in Section 3.1.1. We then integrate the semantic segmentation into the completion as an end-to-end model in Section 3.1.2. After that, we discuss the common 3D data formats used for 3D semantic completion in Section 3.2. At the end of this chapter, we introduce some existing datasets in Section 3.3.1 and the corresponding metrics for evaluation in Section 3.3

Chapter 4: Recent History of 3D Completion from a Single View. We begin with an overview of 3D completion in Section 4.1. This chapter then maps out the related work in 3D completion from a single view. Considering that there are different data formats that can be used in 3D completion, this chapter also investigates the input from RGB or depth image as well as the output to volumetric data in Section 4.2, and point cloud in Section 4.3. At the end of the chapter, we discuss common latent features learned from 3D completion models in Section 4.4.

Chapter 5: Summary of Contributions. This chapter summarizes the main contributions and additionally provides the associated publication for each work. Our works are mostly focused on reconstructing 3D in two data formats, which are volumetric data in Section 5.1 and point cloud in Section 5.2. Inspired by some insights during the model designs in these two sections for 3D completion. We further propose some feature learning methods in Section 5.3 to other 3D-related tasks and even tasks which are not targeted in 3D.

Chapter 6 and 7: Conclusion and Future Works. Eventually, we summarize the contribution of this dissertation and the solved related tasks in Chapter 6. Then, we discuss some future works in Chapter 7.

2 Fundamental Theories

This dissertation focuses on solving single-view 3D completion with learning-based approaches, whereas deep learning plays an important role. Deep learning is part of a broader family of machine learning approaches based on parameterized architectures with many weights.

In this chapter, we first introduce some fundamental deep learning theories in Section 2.1, including basic operators, famous architectures, different types of losses according to specific tasks, and some standard optimizers. Although how the converged deep model performs during inference depends on the loss function, there are still ways to let the parametric model perform differently by setting some constraints as optimization targets. Practically, given a task to solve with a parameterized deep architecture, we may encounter problems, including difficulty sampling in latent space in a generative model, lack of ground truth data for training, wrongly provided annotations, etc. So next, in Section 2.2, we introduce some training techniques targeted at solving problems in specific application scenarios, which can significantly improve the performance of trained deep architectures. Those techniques include variational constraint, adversarial training, unsupervised training, and metric learning.

2.1 Deep Learning

As a branch of machine learning approaches based on large-scale parameterized models, usually referred to as deep architectures, deep learning is widely adopted in a large variety of tasks, including natural language processing, visual recognition, speech recognition, neural rendering, etc. Leveraging the power of a large amount of training data, deep architectures, such as an MLP shown in Fig. 2.1, are usually built with different sequentially connected operators and modules which will be introduced in Section 2.1.1 including fully-connected layers, convolutional layers, pooling operators, etc. Those individual operators are used to compose deep architectures in Section 2.1.2 targeting different types of tasks. While most initialized deep models are not expected to perform a satisfactory inference, the parametric models are validated by loss functions and optimized with optimizers, which are introduced further in Section 2.1.3 and Section 2.1.4.

2.1.1 Operators

Processing input data with large amounts of intermediate latent features, deep architectures are composed of repetitively appearing operations to process input data, including fully-connected layers, convolutional layers, pooling, and non-linear activations. In this section, we briefly introduce each single operator that is used to construct deep architectures in Section 2.1.2.

Fully-connected layer. Fully-connected layers serve as one of the most basic layers in deep learning, all output units from one layer are projected to every unit of the next layer with a parameterized projection function. Performing as dense projections, they compile the input data extracted by previous layers to form the final output to get the expected inference results. Using the fully-connected layers to construct the last few layers is a common design in deep architectures.

Convolutional layers. Convolutions are initially proposed for signal processing, such as noise filtering and high-frequency filtering. Given an input signal $s(x)$, the convolved signal $\hat{y}(x)$ can be formulated as

$$\hat{y}(x) = \int_{-\infty}^{\infty} \omega_{\infty}(\tau) s_{\infty}(x - \tau) d\tau \, . \tag{2.1}$$

To practically implement the convolution from its continuous form to the discrete form with the filter ω, the infinite version

$$\hat{y}(x) = \sum_{d_x=-\infty}^{\infty} \omega_{\infty}(d_x) s_{\infty}(x - d_x) \tag{2.2}$$

is usually adapted to its finite form

$$y(x) = \sum_{d_x=-k_x}^{k_x} \omega(d_x) s(x - d_x) \, , \tag{2.3}$$

where the kernel ω is parameterized to be smaller, resulting in a much smaller model compared to fully-connected layers. The convolutions can be converted to fully-connected layers when the filter ω is with the same length of the signal s.

Local convolutional operations are widely used when data s is organized such as audio, images, and videos. Considering images are well-gridded data where common local patterns are easier to learn from a large amount of data. Processing 2D images is a typical applicable scenario for local convolutions. A fundamental expression of producing a single convolution output targeted on a single 2D position x, y with the value of $s(x, y)$ could be presented as

$$y(x, y) = \sum_{d_x=-k_x}^{k_x} \sum_{d_y=-k_y}^{k_y} \omega_{2d}(d_x, d_y) s(x - d_x, y - d_y) \, , \tag{2.4}$$

where $y(x, y)$ is the output and ω_{2d} is the convolutional kernel. Every element of the filter kernel is considered in the range of $[-k_x, k_x]$ and $[-k_y, k_y]$. Here the kernel size k_x and k_y

determines the scale of the receptive field; once it is set to be larger than a single element, it is possible to extract some local features by considering several local input elements.

Some exceptional cases may happen when the data format has difficulty getting local neighborhoods explicitly organized, e.g. , natural languages and point cloud. In those works, a single down-sampling operator such as max-pooling, introduced in the next paragraph, could even be with a receptive field of the whole input data. Alternatively, works like PointConv [27] and PointCNN [28] index each sample with k-nearest neighbor (KNN) search to find local patches, where they then apply the convolution kernels on those local patches.

Given the basic convolutional operation (2.4), some hyperparameters apart from kernel size k_x are also crucial such as stride, padding, and dilation rate, which could be adjusted to practically solve some real-world problems. While information is sometimes expected to get compressed in the spatial domain to reduce noises, the convolution stride could be larger than 1. In one particular case for image convolution, the spatial dimensionality would be gradually reduced by at least $k_x - 1$ per-layer because the kernel cannot be centered on the edge element on the image, resulting in a tiny feature patch in the end. To overcome this issue, spatial paddings, such as zero padding and repetitive padding, could be used. For object detection and 3D completion tasks, some crucial targets in the input data are with a large variety of scales, so that convolutional kernels in the same layer with different dilation rates will make it possible to attend to the objects on different scales.

To generalize 2D convolution into other dimensions, e.g. volumetric data in 3D, similar patterns could be followed in terms of convolution operation as described in (2.4), where one additional dimension should be added for both local and patch and convolutional kernel. Similarly, convolution can also be applied for audio along the time axis by changing the operational dimension to 1.

While convolution is used to extract features with down-sampled results, a similar operation called transposed convolution is used to up-sample the input feature map to a desired output feature map to fit for the requirement of specific output spatial dimensions such as super-resolution applications [29] and image segmentation [30]. Specifically regarding point cloud deconvolutional operations, FoldingNet [31] uses a 2D grid to help generate a 3D point cloud from a single feature. PCN [17] further uses local FoldingNet to obtain a fine-grained output from a coarse point cloud with a low resolution, which could be regarded as an alternative to point cloud deconvolution.

Pooling. Aiming at making extracted features more compact, pooling functions replace the network output at a specific location with a summarized statistic of the nearby outputs. For example, the 2D max-pooling operation reports the maximum output within a rectangular neighborhood. One crucial factor that makes the pooling results different is the way to determine the samples fed into the pooling operator. While max-pooling is applied for a local set of elements, graph max-pooling (GMP) [32] takes the element-wise maximum value of the feature across all the vectors, where we select the subset of feature vectors with the highest activations. Other popular pooling functions include the average of a rectangular neighborhood, the L_2 norm of a rectangular neighborhood, or a weighted average based

on the distance from the central pixel. In all cases, pooling helps make the representation approximately invariant to small input translations, which means that if we translate the input by a small amount, the values of most of the pooled outputs do not change.

Non-linear activations. The activation function of a node defines the output of that node given an input or set of inputs. The most common activation functions can be divided into three categories: ridge functions, radial functions, and fold functions.

Ridge functions are multivariate functions acting on a linear combination of the input variables, e.g. ReLU [33] function presented in $\max(0, x)$, Heaviside [34, 35] function presented in $1_{x>0}$, Logistic [36, 37] function presented in $(1 + \exp^{-x})^{-1}$, etc.

Radial activation functions are a particular class of activation functions known as radial basis functions (RBFs), including Gaussian function, multi quadratics, inverse multi quadratics, polyharmonic splines, etc.

Folding activation functions are extensively used in the pooling layers in convolutional neural networks and output layers of multi-class classification networks. These activations aggregate the inputs, such as taking the mean, minimum, or maximum. In multi-class classification, the softmax activation is widely used which is presented as

$$\text{softmax}(x)_i = \frac{\exp^{x_i}}{\sum_{j=1}^{c} \exp^{x_j}}, \tag{2.5}$$

where c is the number of classes and x is a c-dimensional vector.

Multi-head attention. The attention mechanism [38] describes a weighted average of elements, particularly sequential elements, with the weights dynamically computed based on a set of queries and the keys of these elements. The difference between weighting each element with attention and straightforward matrix multiplication is that attention modules dynamically decide on which elements are focused more compared to the others.

In terms of implementation, the attention layer takes its input in the form of three parameters, known as the query Q, key K and value V. Multi-head Attention [38] is composed of multiple attention heads allowing for attending to parts of the input sequential data with different lengths. Weighted by W_{out}, a multi-head attention $\mathcal{A}_{mul}$

$$\mathcal{A}_{mul}(Q, K, V) = [\mathcal{A}_1, \ldots, \mathcal{A}_h] W_{out} \tag{2.6}$$

is usually produced by concatenating several individual scaled dot-product attention $\{\mathcal{A}_i\}$

$$\mathcal{A}_i(Q, K, V) = \text{softmax}(\frac{QW_i^Q (KW_i^K)^T}{\sqrt{d_k}}) VW_i^V \tag{2.7}$$

where $\sqrt{d_k}$ is the dimension of the key and query vectors. Such multi-head attention overcomes the difficulty of representing multiple different aspects to which a sequence of elements wants to attend with the help of different sets of $\{W_i^Q, W_i^K, W_i^V\}$.

2.1.2 Architecture

By implementing some inference models in deep architectures, layer-by-layer operations as mentioned in Section 2.1.1 are used. The performance of deep architectures can surpass human expert performance usually because of its large-scale parametric model, where different operators should be carefully connected to form some standard modules. Regarding different tasks, especially given with different input and output pairs, different architectures are proposed, such as deep belief networks, deep reinforcement learning, recurrent neural networks, and convolutional neural networks have been applied to fields including computer vision, speech recognition, natural language processing, machine translation, bioinformatics, drug design, medical image analysis, climate science, material inspection, and board game programs. In the following paragraphs, we illustrate some basic architectures.

Multi-layer Perceptron (MLP)

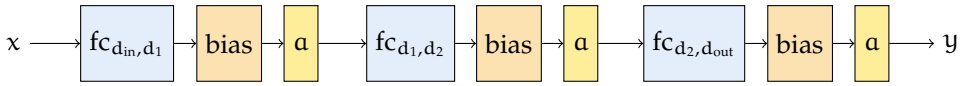


Figure 2.1 An example of a multi-layer perceptron (MLP) composed of 3 fully-connected layers with biases, which are followed by non-linear activations a.

Multi-layer perceptron (MLP), e.g. the one shown in Fig. 2.1, consists of several fully-connected layers with nonlinearly activating functions a for the output of each fully-connected layer. Disregarding forward passing data with a batch size of more than 1, a single input for MLP is always flattened as a vector; each element in the vector is connected with a certain weight to every element in the following layer.

Two of the most important factors of MLP are the number of layers and output dimension. As shown as a 3-layer MLP in Fig. 2.1, d_{in} and d_{out} are determined by the problem on hand. The sizes of d_1 and d_2 are flexible hyperparameters according to other factors such as expected inference accuracy and efficiency. The multi-layer perceptron (MLP) represents a function $y(x;\Theta)$ parameterized by the weights Θ in the fully-connected layers and their biases. Usually, a deeper MLP gives the model more capacity to fit the training data. An example could be shown in Fig. 2.2 where more fully connected layers are used to build the whole auto-encoder (AE) architecture with dimensional hyperparameters d_1, d_2 and d_{lat}. Here the output dimension d_{out} is set to be the same as the input dimension d_{in} to build the reconstruction loss $\mathcal{L}(\tilde{x}, x)$, which quantifies the similarity between the reconstruction $\tilde{x}$ and the input x itself.

Convolutional Neural Network (CNN)

Convolutional neural networks (CNN) are regularized versions of multi-layer perceptrons. Each neuron in one layer of MLP is connected to all neurons in the next layer. The complete connectivity in these layers makes them prone to over-fitting data. CNN performs regularization by taking advantage of the hierarchical pattern in data and assembling increasingly complex patterns using smaller and simpler patterns embossed in their filters. Therefore,

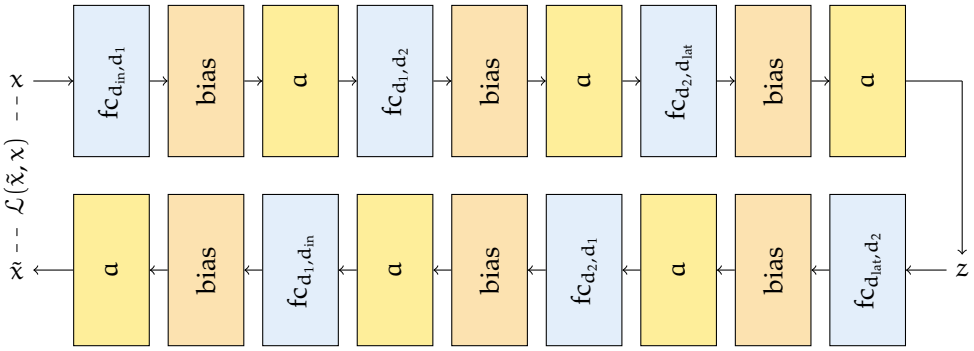


Figure 2.2 An example of an auto-encoder with a d_{in}-dimensional input x, built with a multi-layer perceptron, where both the encoder and decoder are composed of a 3-layer-perceptron connected by a d_{lat}-dimensional latent code z.

on a scale of connectivity and complexity, CNNs are on the lower extremity. One of the earliest works of CNN is LeNet-5 [39] which comprises seven convolutional layers aiming at classifying digits from an image with 32×32 pixels.

Local operators are one of the most critical components in CNN, which takes several elements in a local neighborhood. The output is usually less than the input processed by local operators. To efficiently process information and remove noises, the dimensionality reduction in the encoder-decoder architecture, local operators make it possible to present the original information from the previous layer into a few elements in the output. For example, convolutions with a stride larger than one can compress information in images and videos. Here stride is a convolution operation parameter that modifies the movement over the image or video.

Deconvolutional networks. Similar to the auto-encoder formed with MLP, if transposed convolution is involved, the convolutional network could be modified as the deconvolutional network where spatial dimensions are increased in the decoder. In some tasks, the input information needs to be firstly summarized by an encoder so that a decoder can generate satisfying information without being misled by noises. In such a scenario, the number of elements in input is first decreased and then increased to the other high-dimensional space, which differs for different tasks, e.g. , completion, segmentation, and reconstruction. The dimension reduction in the encoder is usually not applied for all dimensions, e.g. , in image processing with CNN, the spatial dimensions are decreased while the pixel-wise feature dimension is usually increased from initial RGB-α channels. Such a manner allows the model to store potentially useful information even when spatial information is filtered to be less. The other example is point cloud processing; usually, the dimension of the points is significantly decreased, e.g. , into a small amount in SoftPoolNet [6] and 1-dimension in PointNet [40]. While the number of points is significantly decreased, the feature dimension is greatly increased from the three dimensions of the input's (x, y, z) coordinates.

Skip-connections. Skip-connections are also called shortcut connections, feeding output from early layers to later layers which skips some layers in neural networks. It is formulated

as a piece of additional information fed together with straightforward passed features, which can be used in either summation or concatenation.

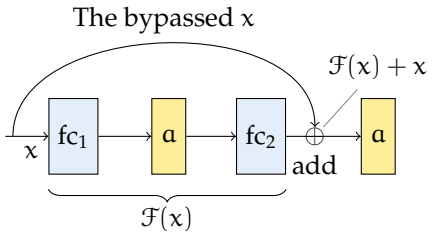


Figure 2.3 A skip-connection formed by feature summation in ResNet [13] variants fed with input x, where two fully-connected layers are skipped.

Residual networks (ResNet) [13] were proposed to solve the image classification problem. In ResNet, the information from the initial layers is passed to deeper layers by matrix summation as shown in Fig. 2.3 between the encoded feature $\mathcal{F}(x)$, which is produced by two fully-connected layers $\{fc_1, fc_2\}$ connected with a non-linear activation a, and the input x. This operation has no additional parameters, as the output from the previous layer is added to the layer ahead. By doing concatenation instead of summation, U-Net [41] concatenates layers in the encoder with layers in the decoder. Such concatenation makes the U-Nets use fine-grained details learned in the encoder to construct an image in the decoder. Regarding the number of skipped layers, U-Net is with long skip-connections, whereas ResNet was with short skip-connections.

Recurrent Neural Network (RNN)

To handle tasks dealing with sequential data, it is intuitive that what was memorized before can be modeled within the context of a recurrent model so that the current inference can be well-conditioned on previous information. Also optimized by gradient descending for their parametric models, Recurrent Neural Network [42] (RNN) is structured with loops, which allow information to persist.

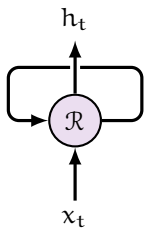


Figure 2.4 An exemplar RNN module.

As shown in Fig. 2.4, the recurrent parametric module $\mathcal{R}$ processes the input x_t and its outputs h_t at the same time in each loop which could be formulated as

$$h_t = \mathcal{R}(x_t, h_{t-1}) \,. \tag{2.8}$$

RNN module stores memory information directly in the neurons. Such behavior is described as the hidden state of the RNN, which is a vector of floating point numbers that keep track of how active each neuron is. These constantly changing hidden states will keep on getting updated once new materials are added. RNN is widely adopted to process time-related data, such as natural languages and financial statistics. As an example, words in sentences can be coded with tokens. It could be trained to use past information and infer the next word for this sentence. In terms of training, the loss function $\mathcal{L}$ for RNN of all time steps t is defined based on the loss at every time step. Practically, the gradient vanishing and exploding problems commonly exist in RNNs, because of a multiplicative gradient that can be exponentially decreasing or increasing with respect to the number of layers.

Two strongly related works, Gated Recurrent Unit [43] (GRU) and Long Short-Term Memory units [44] (LSTM) both target solving the vanishing gradient problem encountered by tradi-

tional RNNs, while LSTM is a generalization of GRU. Interestingly, RNN can be adapted for different kinds of data generation tasks such as Music generation, sentiment classification, name entity recognition, and machine translation.

Transformers

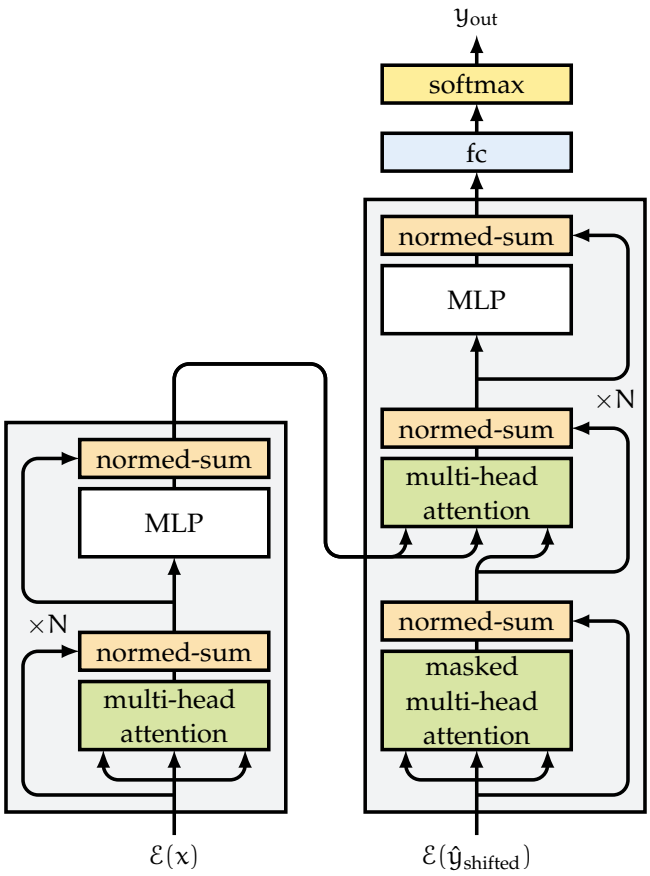


Figure 2.5 An example of a transformer [38] with encoded input $\mathcal{E}(x)$. Two cascaded fully-connected layers which are connected by a ReLU activation form the MLP.

Like recurrent neural networks (RNNs), widely used to process sequential data such as natural language, transformers are also designed to handle sequential input data for tasks such as translation and text summarizing. However, unlike RNNs, transformers [38, 45] do not necessarily process the data in order. Instead, the attention mechanism provides context for any position in the input sequence coded with positional encodings. For example, if the input data is a natural language sentence, the transformer does not need to process the beginning of the sentence before the end. Rather, it identifies the context that confers meaning to each word in the sentence. This feature allows for more parallelization than RNNs and therefore reduces training times. Such a data processing technique also makes it useful for extracting features from unordered data, such as the feature map of the point cloud [46]. As illustrated in Fig. 2.5, the input x and the shifted output $\hat{y}_{shifted}$ are both encoded as latent embeddings $\mathcal{E}(x)$ and $\mathcal{E}(\hat{y}_{shifted})$, which are the summed features with their positional encodings. Benchmarked with satisfying performance, the pre-trained models like GPT-3 [47], BERT [48], and RoBERTa have demonstrated the potential of transformers to find real-world applications such as document summarization, document generation, biological sequence analysis, and video understanding. In chemistry and biology, transformers are also utilized to gain a deeper understanding of the relationships between genes and amino acids in DNA and proteins. This greatly boosts the progress for faster drug design and development. Transformers variants are also employed in some special scenarios to identify patterns and detect unusual activity to prevent fraud, optimize manufacturing processes, suggest personalized recommendations, and enhance healthcare.

2.1.3 Loss Functions

To numerically evaluate how deep architecture performs given specific tasks, we can validate how output y differs from the expected ground truth y_{gt} in terms of different metrics.

Common losses. The most common metric is norm 2 Euclidean distance which evaluates the distance between two vectors in Euclidean space, which is described by the square root of the summed squares of each dimensionality in y and y_{gt}. The norm 2 distance is then presented as

$$\mathcal{L}_{norm} = \sqrt{\sum_{i=1}^{n} (y^i - y_{gt}^i)^2} \,. \tag{2.9}$$

To validate models designed for pose-related tasks, *e.g.* pose estimation, cosine distance

$$\mathcal{L}_{cosine} = \frac{\sum_{i=1}^{n} y^i \times y_{gt}^i}{\|y\| \|y_{gt}\|} \tag{2.10}$$

is widely adopted, which focuses more on the direction of the estimated feature instead of the anchored position of the estimation.

Reward-based loss. In procedures such as reinforcement learning [49], the artificial agents learn to get rewards that they will receive when their actions have to lead to the successful completion of a task. The network is trained to handle tasks to get as many rewards as possible. Notice that such a network comprises two main parts: a decision network that uses sensory information to select actions that lead to the greatest reward and a value network that predicts how rewarding an action will be. The loss to get more rewards is then composed of two parts: maximizing the rewards and predicting the expected numerical reward according to specific actions as accurately as possible.

Binary cross entropy loss. In some scenarios, quantized estimations are expected, such as classification and segmentation. Classification presents a categorical label for each sample as a one-hot code with a length of the number of categories. Similarly, the segmentation networks present the segmentation result as a multi-channel binary mask with the same spatial size as the input data, where a one-hot code feature is assigned for each single input element. The binary cross-entropy loss $\mathcal{L}_{entropy}$ is presented as

$$\mathcal{L}_{entropy} = \sum_{i=1}^{n} -y^i \log y_{gt}^i - (1 - y^i) \log(1 - y_{gt}^i) \,. \tag{2.11}$$

During training, those metrics could be used as loss functions to back-propagate gradients to optimize network parameters. More recently, the binary cross entropy losses have also been widely served as regularization to improve the confidence of some implicitly learned parameters in more complex tasks, such as distinguishing foreground space and background space apart from each other in the radiance field of NeRF [50] and signed distance field in NeuS [51] to reconstruct unbounded 3D scenes.

2.1.4 Optimizer

Learning-based algorithms involve optimization in many contexts. For example, performing inference in models such as PCA involves solving an optimization problem. We often use analytical optimization to write proofs or design algorithms. Concerning optimization of deep architectures, it could be referred to as finding the parameters Θ of a neural network that significantly reduces a cost function $\mathcal{L}_{\text{opt}}$ which could be presented as

$$\mathcal{L}_{\text{opt}} = \underset{\Theta}{\mathcal{L}}(\{S\}) + \lambda \underset{\Theta}{\mathcal{R}}(\Theta) \,, \tag{2.12}$$

where the training set $\{S\}$ is used to produce a performance measure. To overcome the problem of overfitting, additional regularization terms $\mathcal{R}(\Theta)$, e.g. L1 and L2 norms [52], weighted by λ are also added during training, which modify Θ to favor simple solutions of minimizing $\mathcal{L}(\{S\})$.

Optimizing Deep Architecture

Optimization algorithms that use only a single example at a time are sometimes called stochastic or sometimes online methods. The term online is usually reserved for cases where the examples are drawn from a stream of continually created examples rather than from a fixed-size training set over which several passes are made. Most algorithms used for deep learning fall somewhere in between, using more than one but less than all the training examples. These were traditionally called mini-batch or mini-batch stochastic methods, and it is now common to call them stochastic methods.

Stochastic gradient descent (SGD) and its variants are the most used optimization algorithms for general machine learning and deep learning. The learning rate ϵ is necessary to get gradually decreased over time. In practice, it is common to decay the learning rate linearly until iteration τ, so that the learning rate for iteration k would be

$$\epsilon_k = (1 - \frac{k}{\tau})\epsilon_0 + \frac{k}{\tau}\epsilon_\tau \,, \tag{2.13}$$

while for iterations more than τ, ϵ could be simply defined as a constant.

While stochastic gradient descent remains a popular optimization strategy, learning with it can sometimes be slow. The method of momentum, which is derived from a physical analogy, is designed to accelerate learning, especially in the face of high curvature, small but consistent gradients, or noisy gradients. The momentum algorithm accumulates an exponentially decaying moving average of past gradients and continues to move in their direction. Although the momentum algorithm can mitigate these issues related to the learning rate somehow, it does so at the expense of introducing another hyperparameter. There are also optimizers proposed to use adaptive learning rates, e.g. , AdaGrad [53], RMSProp [54] and Adam [54] and their variants [55, 56, 57].

2.2 Training Techniques

Targeted on solving some real problems, training with some special techniques is useful to improve the performance of the converged model even without changing the structural properties of deep architectures.

2.2.1 Adversarial Training

Deep learning works well when the training data is similar to the testing data. In some scenarios, the training data is in a pretty different domain compared to the testing data, causing a data migration problem. This implies that the feature domains extracted by the same encoder are different. Then there is an objective to make both domains similar so that the network trained from one type of data could be applied to the test data.

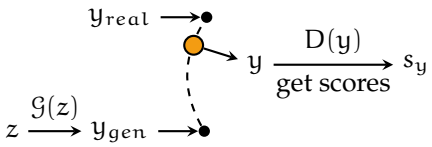


Figure 2.6 Discriminator $\mathcal{D}$ helps optimize generator $\mathcal{G}$. Notice that z could be extracted from an encoder $\mathcal{E}$ from an additional input x

Such tasks could be referred to as domain adaptation [58, 59, 60] or domain generalization [61], and practically could be solved by discriminative training [62, 63]. For instance, the domain discriminator in DANN [64] is used to distinguish the source domain from the target domain using a binary code, while ADDA [65] uses two different discriminators for the source feature and the target features. A discriminator $\mathcal{D}$ is crucial in the architectural implementation and training. As shown in Fig. 2.6, $\mathcal{D}$ which produces realistic scores s_y is used to optimize the generator $\mathcal{G}$ through a loss function

$$\underset{(\mathcal{G})}{\mathcal{L}_{\text{gen}}} = -\log\left(\mathcal{D}(\mathcal{G}(z))\right) \tag{2.14}$$

by randomly sampling the latent features z from certain distributions, e.g. Gaussian distribution, Bernoulli distribution, or even distribution defined by a parametric encoder $\mathcal{E}$ from another input x. Such optimization works only when the parameters of the discriminator are optimized by the other loss function

$$\underset{(\mathcal{D})}{\mathcal{L}_{\text{dis}}} = -\log(\mathcal{D}(y_{\text{real}})) - \log\left(1 - \mathcal{D}(\mathcal{G}(z))\right), \tag{2.15}$$

which serves as a counterpart of $\mathcal{L}_{\text{gen}}$. Practically, $\mathcal{L}_{\text{gen}}$ and $\mathcal{L}_{\text{dis}}$ are usually optimized in alternating training.

If additional category labels are available, SymNets [66] can further improve the domain adaptation by using the two-level domain confusion losses from the domain level to the category level. By making the latent feature extracted by the encoder in the same dimension as the input, GVB [67] uses discriminative training to make the latent features from the two different domains in the same subspace.

2.2.2 Metric Learning

Instead of evaluating the final output of deep architectures, latent features could also be set with some constraints, potentially beneficial to improving the performance of the final output. Such optimization is specifically beneficial when latent features are expected to be within a given distribution so that new samples can be sampled, which produces reasonable output.

Several methods have applied metric learning in the latent space optimization [68, 69, 70]. Assuming supervised learning, these methods optimize the distance among the samples so that they reflect their ground truth semantic similarity. They formulate the pair-wise distance metrics, which include: the triplet loss and its derivatives [68, 71, 72, 73], the contrastive loss and its derivatives [74, 69], and the Neighborhood Component Analysis and its derivatives [75, 76, 70]. Among all those losses, triplet learning [77, 68, 78, 71, 79] is one of the typical learning strategies where the pair-wise distances are further labeled as positive or negative based on the pair-wise relationships, resulting in clusters in the latent space. Focused on latent features extracted by encoder $\mathcal{E}(\cdot)$, the loss functions training with Siamese term [80] could be defined as

$$\mathcal{L}_{\text{pair}}(x_i, x_{\text{pos}}) = |\mathcal{E}(x_i) - \mathcal{E}(x_{\text{pos}})|^2 \,, \tag{2.16}$$

so that distances between the latent samples $\mathcal{E}(x_{\text{pos}})$ from the same category with $\mathcal{E}(x_i)$ become smaller while the distances between the samples from different categories become larger; and, the loss function for triplet learning [77] is defined as

$$\mathcal{L}_{\text{triplet}}(x_i, x_{\text{pos}}, x_{\text{neg}}) = \ln\left(\max\left(1, 2 - \frac{|\mathcal{E}(x_i) - \mathcal{E}(x_{\text{neg}})|^2}{|\mathcal{E}(x_i) - \mathcal{E}(x_{\text{pos}})|^2 + \omega}\right)\right) \,, \tag{2.17}$$

where ω is a small margin to ensure that the final loss function is not infinite so that samples from the same category, i.e. the positive pairs x_i and x_{pos}, are closer than samples from distinct clusters, i.e. the negative pairs x_i and x_{neg}. We apply these loss functions to all the possible permutations in the batch.

2.2.3 Variational Constraint

Given an encoder-decoder architecture such as Fig. 2.7, the variational inference uses the encoder to approximate the posterior of the latent features produced by the encoder that is conditioned on the expected network output. Such approximation is governed by a simple distribution of the data in latent space, e.g. Gaussian distribution. These methods are derived from VAE, where [81] proposes an assumption that the latent feature of VAE behaves like PCA components. This, in effect, makes the convergence in training more efficient.

To solve the problem caused by the over-simplified latent space of VAE, categorical labels are integrated into conditional VAE (CVAE), which is used in generation [82] and prediction [83].

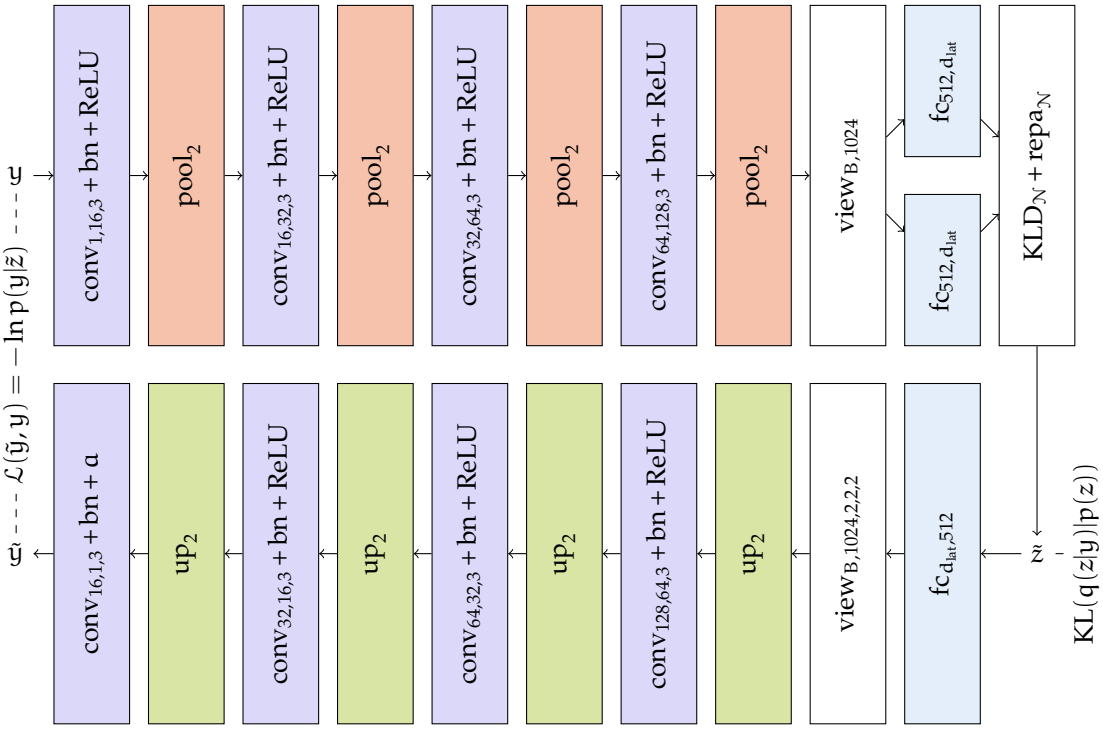


Figure 2.7 An example of a variational auto-encoder (VAE) with four convolutional modules in both the encoder and decoder individually.

With a better performance than the standard VAE, they prove that such information can potentially improve the generative models.

Using variational inference in clustering, GMVAE [84] and its graph embedding version [85] demonstrate that multi-modal Gaussian can reveal categorical information better than VAE. Another notable method is from VQ-VAE [86], which aims at solving the “posterior collapse" by discretizing the trained features into a table. Moreover, InfoVAE [87] improves the evidence lower bound (ELBO) objective since they observed that the ELBO favors optimizing the distribution over the inference.

Beyond the assumption of simple distributions in the latent space, Auxiliary Deep Generative Models [88] adds auxiliary latent variables. However, the disadvantage of such work is that the auxiliary latent variables are necessary during training and inference time.

Besides visual data, VAE can also be used for natural language processing (NLP), such as machine translation. VAE-LSTM [89] composes both the encoder and the decoder with LSTM operations, while VAE-CNN [90] uses LSTM in the encoder and the dilated CNN to form the decoder. In these methods, to solve the latent variable collapse problem [89, 91] of VAEs, HR-VAE [91] imposes regularization for all the hidden states of the LSTM encoder.

The variational inference model imposes a pre-defined distribution on the latent feature to optimize the encoder-decoder architecture. Unlike other methods, we implement the variation constraint through the nebula anchors.

We first adopt the variation translation model from GNMT [92] where the expected Y differs from input X. The difference between input and expected output implies that we can denote a given problem through the probabilistic model $P(Y|X)$.

Given the encoder $\mathcal{E}(X)$ which produces the latent feature z and the generator $\mathcal{G}(z)$, the objective is to make the expectation $\mathbb{E}_{z\sim Q}P(Y|z)$ of the likelihood $P(Y|z)$ to be close to the true probability $P(Y)$, where the probability $Q(z)$ is determined by $\mathcal{E}(X)$ while $P(Y|z)$ is determined by $\mathcal{G}(z)$. We then use the Kullback-Leibler (KL) divergence $\mathbb{D}$ from posterior $P(z|Y)$ to $Q(z|X)$, written as

$$\mathbb{D}_{KL}[Q(z|X,\mathcal{A})||P(z|Y)] = \mathbb{E}_{z\sim Q}\left[\log\left(\frac{Q(z|X)}{P(z|Y)}\right)\right], \tag{2.18}$$

to measure the difference between those two distributions. Thus, by minimizing the KL divergence, we evaluate the capacity of the encoder to generate latent variables that are likely to produce the expected target. Since $P(z|Y)$ is intractable, VAE [93, 94] rewrites KL-divergence from (2.18) as

$$\mathbb{D}_{KL}[Q(z|X)||P(z|Y)] = \mathbb{E}_{z\sim Q}\left[\log\left(\frac{Q(z|X)}{P(Y|z)\cdot P(z)}\right)\right] + \log P(Y)\,. \tag{2.19}$$

If we reorganize it as

$$\mathbb{D}_{KL}[Q(z|X)||P(z|Y)] = \log P(Y) - \left(-\mathbb{E}_{z\sim Q}\left[\log\left(\frac{Q(z|X)}{P(Y|z)\cdot P(z)}\right)\right]\right), \tag{2.20}$$

the second term is then called evidence lower bound (ELBO) of $\log P(Y)$. Considering that the first term is independent of $Q(z|X)$, the optimization then focuses on

$$\text{ELBO} = -\mathbb{E}_{z\sim Q}\left[\log\left(\frac{Q(z|X)}{P(Y|z)\cdot P(z)}\right)\right] \tag{2.21}$$

alone, which could be reformulated as

$$\begin{aligned}\text{ELBO} &= \mathbb{E}_{z\sim Q}\left[\log P(Y|z)\right] - \mathbb{E}_{z\sim Q}\left[\log\left(\frac{Q(z|X)}{P(z)}\right)\right] \\ &= \mathbb{E}_{z\sim Q}\left[\log P(Y|z)\right] - \mathbb{D}_{KL}[Q(z|X)||P(z)]\,.\end{aligned} \tag{2.22}$$

Finally, the loss function of the generative model is formulated as $\mathcal{L}_{\text{enc-gen}} = -\text{ELBO}$ which is written as

$$\mathcal{L}_{\text{enc-gen}} = \underbrace{\mathbb{D}[Q(z|X))||P(z)]}_{\mathcal{L}_{\text{enc}}} \underbrace{-\mathbb{E}_{z\sim Q}[\log P(Y|z)]}_{\mathcal{L}_{\text{gen}}}, \tag{2.23}$$

where the first term $\mathcal{L}_{\text{enc}}$ enforces the encoder to produce latent features which satisfy a Gaussian distribution while the second term $\mathcal{L}_{\text{gen}}$ enforces the predicted output from the latent feature fits the expected ground truth.

3 Learning-based Single View 3D Completion

The 3D completion task takes a single view of the targeted object, depicted by a depth image, an SDF volume, a point cloud, etc. to predict its 3D representation. Considering that only a limited view of the entire scene is captured, constrained by the sensor's viewpoint, the objective of 3D completion inference models is to complete the target area by revealing the hidden structures that are not visible in the input while keeping the observed geometries as precise as possible. Learning-based models are popularly used due to their flexibility of adaptation to different training data.

Here we first demonstrate our focused task of 3D semantic completion in Section 3.1 which starts with a pure geometric completion in Section 3.1.1. Then combining semantic segmentation with completion, the topic of doing semantic completion with an end-to-end model is introduced in Section 3.1.2. Practically, there are many optional ways to present the 3D semantic completion results. Different 3D data formats have their advantages and disadvantages for 3D semantic completion which will be introduced in Section 3.2. At the end of this chapter, we introduce some existing datasets in Section 3.3.1 and metrics to evaluate the related approaches in Section 3.3.

3.1 3D Semantic Completion

Given the task of constructing completed 3D geometry y from a single view of the target x, which is practically depicted by a 2.5D input such as a depth image, an SDF volume, or a point cloud. We solve it with a deep architecture which is usually implemented by encoder-decoder architecture $\mathcal{E}$ and $\mathcal{G}$. Both architectural and training designs could be involved to improve the performance of the inference $\mathcal{G}(\mathcal{E}(x))$ in terms of completion accuracy and inference efficiency.

3.1.1 3D Completion

To start with the tasks of doing 3D completion, we would mention that real-world RGB images are much more than the number of existing depth images practically because depth sensors are less popular in daily life. So we would first mention that an easy solution for 3D completion is to use multiple frames and RGB images as input. For example, the recurrent networks 3D-R2N2 [95] fuse multiple RGB feature maps sequentially into a decoder to predict

the complete 3D geometries. To get 3D structures presented in higher resolutions, a coarse-to-fine 3D decoder presented in Pix2Vox [96] and a residual refiner in Pix2Vox++ [97] are proposed. Due to the recent popularity of the attention mechanisms, AttSets [98] proposed to build attention layers to correlate the image features from different views. Although RGB images are easy to get, the lack of 3D geometries makes deep architectures less robust to background noises, which is a problem for complicated targets such as indoor scenes. Our 3D reconstruction in this paper focuses on only a single depth image.

Taking a depth image of an object from an arbitrary camera pose, the objective of 3D object completion is to complete its missing structure and build its full reconstruction. The advantage of a 2.5D image is the fact that it could be back-projected into 3D space.

3D back-projection from depth. In the depth image captured from devices such as Kinect and RealSense, the depth value z together with the pixel-wise coordinates (x, y) could be used to present a 3D surface or structural volume such as signed distance field (SDF) volume. Here we have an introduction to the way to back-project depth images into 3D.

Considering the simplest pinhole camera model with no skew or distortion factor. A 3D point $p_{3d} \in \mathbb{R}^3$, which could be explicitly coded as (x, y, z), is mapped to the image plane by the function $f(x, y, z)$ as a 2D point $p_{2d} \in \mathbb{R}^2$ which is presented as (u, v). Such procedure can be described as a transformation

$$p_{2d} \cong K \times [R|t] \times p_{3d} \,, \tag{3.1}$$

where p_{2d} is the projected point on the 2D image plane, K is the intrinsic camera matrix that projects a 3D point to the image plane, and $[R|t]$ is the extrinsic parameters describing the relative transformation of the point in the world coordinate to the camera coordinate. Here p_{3d} represents the 3D point expressed in a predefined world coordinate system in Euclidean space. Consider the equation in (3.1), a 3D point p_{3d} can be projected into a 2D position (u, v). On the other hand, the 3D points can also be recovered with a given z from a depth map and solving x and y in the world frame for further processing, to get a 3D point p_{3d} correctly positioned.

By repetitively calculating a set of 3D points $\{p_{3d}\}$ from all valid depth values on an image, a back-projected 3D surface $\mathbf{S}$ can be described by $\{p_{3d}\} \in \mathbf{S}$. Given such an incomplete surface as input, a completion model can be structured, conditioning on the observed geometries.

3.1.2 Semantic Completion

Semantic completion supplements geometric completion with estimated semantic labels. Sometimes it is practically referred to as semantic scene completion (SSC) [99] because scenes are one of the best targets which need semantic labels to get objects separately presented from the room layout. Here the objective is not only to build the full reconstruction of the scene but also to semantically label each component. Before introducing semantic completion, 3D segmentation is also an important topic to start with.

3D segmentation. Apart from 3D completion, 3D segmentation is also a fundamental and challenging task. Traditionally, 3D segmentation could be performed with hand-crafted features, but hand-crafted features are hard to get generalized to large-scale data. Deep learning based approaches significantly improves the performance of 3D segmentation for different data formats such as sparse convolution [100] for volumetric data and PointNet++ [101] for point cloud.

End-to-end semantic completion. Instead of estimating the complete geometries and the semantic segmentation with cascaded completion and segmentation architectures, we focus on semantic completion with end-to-end models which jointly estimate the complete geometry and semantics. Cascade semantic completion training usually tends to separately optimize completion and segmentation models with the back-propagated gradients. Assuming that the model is cascaded by a completion head and concatenated with a segmentation sub-network, the segmentation gradient has far less influence on completion model. The specialty of end-to-end training is that the semantic labels are usually produced by the same layer of deep architecture so that the completion and segmentation supervisions enforce the networks jointly producing geometric estimation for occluded areas and segmentation labels. In this case, simultaneously learning the geometric structure and the semantic information allows the algorithm to learn the contextual cues that can in turn represent the objects in the reconstruction.

Focusing on learning-based semantic completion, most related work can be categorized depending on the input data they process – voxelized grid or point cloud.

3.2 3D Representations

Some major data formats in which the expected 3D completion is presented have quite different characteristics. They differ in the efficiency of processing, scale to get stored and simplicity to get processed. In topics of 3D semantic completion, three types of data are utilized the most which are volumetric data in Section 3.2.1, point cloud in Section 3.2.2 and implicit surface 3.2.3.

3.2.1 Volumetric Data

Volumetric data is a set of samples (x, y, z, v), representing the value v of some property of the data, e.g. 0 or 1 for occupancy and one-hot code for categories. Occupancy of 0 or 1 is commonly used to indicate whether a voxel at (x, y, z) belongs to the background or object. Similar to 2D image segmentation, volumetric semantic completion can also be presented by element-wise categorical one-hot coding. The collection of 3D location (x, y, z) are usually fixed across all samples in the same dataset for easy processing.

Given a depth image, its volumetric representation could be made by back-projecting all pixels in the depth image into 3D space with the help of the camera's intrinsic and extrinsic

matrix. The values v for volumetric data are not necessarily to be integers, e.g. SDF values in the signed distance field describe the interior and surrounding volume of an object. The negative and positive signs for SDF values are used to denote where the position is located from the surface of the object. The SDF values sometimes are quite useful for 3D completion because they can provide an offset in empty space indicating the distance to the closest structures.

Volumetric data is easy to get processed because of its gridded format. Such a fixed voxel grid usually makes it easy to apply 3D convolution inspired by the popularity of 2D convolution operations in CNNs [102, 103, 104] for RGB images.

For object completion, works such as 3D-EPN [105] and 3D-RecGAN [106] are proposed to present complete shapes in volumetric space generated by 3D deconvolutions. Built upon the architecture of 3D-RecGAN [106], 3D-RecGAN++ [107] utilizes adversarial training with a 3D discriminator to improve the reconstruction. Also supplemented with discriminative training, our ForkNet [8] further improves the completion accuracy for both synthetic and real objects by self-supervised training with synthetically generated training pairs from two of our decoders during training.

As for the topic of semantic scene completion, there have been several methods for semantic scene completion based on voxel grids that were initiated by SSCNet [99]. Using a similar volumetric data with 3D convolutions [105, 106, 107], VVNet [108] convolves on the 3D volumes which are back-projected from the depth images, revealing the camera view instead of a TSDF volume. Since 3D convolutions are heavy in terms of memory consumption especially when the input is presented in high resolution, SketchSSC [109] learns the 3D boundary of all objects in the scene to quickly estimate the resolution of the invariant features.

Losses. Some related loss functions used for volumetric data reconstruction are usually focused on 3 objectives in topic of semantic completion: (1) reconstructing SDF volumes in (3.2), (2) estimating completion occupancy volumes in (3.3) and (3) estimating semantic completion volumes in (3.5).

First, the loss function $\mathcal{L}_{\text{SDF}}$ used to enforce a reconstructed SDF volume x_{output} to be similar to its ground truth x is

$$\mathcal{L}_{\text{SDF}} = \left\| x_{\text{output}} - x \right\|^2 . \tag{3.2}$$

Next, the loss functions $\mathcal{L}_{\text{complete}}$ and $\mathcal{L}_{\text{SSC}}$ are used to optimize the completion result g_{output} and the semantic completion result s_{output} to be similar to their ground truth g_{gt} and s_{gt}, binary cross entropy could be adapted with weights in loss functions.

$$\mathcal{L}_{\text{complete}} = \sum_{i=0}^{1} (\epsilon(g_{\text{output}}, g_{\text{gt}})) , \tag{3.3}$$

where $\epsilon(\cdot,\cdot)$ is the per-category error

$$\epsilon(q,r) = -\lambda r \log q - (1-\lambda)(1-r)\log(1-q)\,. \tag{3.4}$$

Notice that λ in (3.4) ranges between 0 and 1, which weighs the importance of reconstructing true positive regions in the volume. The larger λ is, the less penalty for the false positive predictions will be enforced.

Similar to (3.3), a loss function to estimate semantic completion could be presented as

$$\mathcal{L}_{SSC} = \sum_{i=0}^{N} (\epsilon(s_{output}, s_{gt})) \tag{3.5}$$

where N is the number of categories in the semantic scene. With the help of some common optimizers such as Adam optimizer [110], the parametric models would be with the ability to produce expected 3D reconstruction results.

Volumetric completion architectures are comparably easy to get designed concerning the fact that many 2D convolutional networks proposed for image-related tasks could be referred. One disadvantage of representing completion in volumetric data is the limited local resolution which makes it hard to reveal details without consuming many resources. Such a problem could be solved by adopting a point cloud for 3D completion.

3.2.2 Point Cloud

A point cloud is a set of data points in space. The points may represent a 3D shape or object. Each point position has its set of Cartesian coordinates (x, y, z). A point cloud has the potential to reconstruct the object at a higher resolution. Simple tasks like translation and rotation on point cloud can be done with point-wise rotation and translation matrix, which could be presented with MLP if such matrix is expected to be estimated. For more complicated tasks, e.g. point cloud segmentation and completion need to get solved by learning-based methods such as some trending deep learning approaches.

Point cloud's limitation to getting it easily used in deep learning is its unstructured data. Unlike RGB images or voxel maps, point clouds do not have a particular order, and the number of points varies as we change the camera pose or the object.

Dealing with unstructured data. To get encoded point cloud features possible to get used as input for a decoder in deep architectures, the problem introduced by its unordered structure needs to be solved. Targeted to solve the unordered structure of point clouds, presenting the whole point cloud into a global feature that is permutational invariant is a practical way. Applying pooling for the entire point cloud, e.g. max-pooling or average pooling results in a single vector as a latent feature that is constantly given any point cloud presenting the same shape. Max-pooling is more commonly used instead of average pooling because it also gets rid of the bad influence of point duplication. PointNet [40] is among the first few works

which propose to adopt max-pooling to achieve a permutation invariant latent feature for further classification and segmentation.

Targeted on point cloud reconstruction and completion tasks, there are many works built upon a PointNet feature. FoldingNet [31] proposes an object completion solution that deforms 2D rectangular grids by multi-layer perceptron (MLP). By increasing the number of 2D rectangular grids, AtlasNet [111] and PCN [17] added more complexity as well as details into the reconstruction. MSN [18] then further improves the completion by adding restrictions to separate different patches apart from each other. Moreover, Cycle4Completion [112] is also based on PointNet features but solves the problem by training with an unsupervised cycle transformation.

Some works solve the problem of unstructured point cloud without summarizing them into a single vector. SoftPoolNet [6] builds local groups of features by sorting them into a feature map which is convolved by transposed 2D convolutions to produce complete point cloud. Consequently, this approach can deal with unorganized point clouds and achieve reconstruction results at high resolution. RFNet [113] and PointTr [46] produce latent features presented in several vectors by encoder. On one hand, RFNet [113] uses its features to complete the object in a recurrent way by concatenating the incomplete input and the predicted points level by level. On the other, PointTr [46] relies on transformers to produce a set of queries directly from the observed points with the help of positional coding, which is further fed for the decoder to produce another set of queries to present the occluded regions.

Extracting local features. Apart from the global features, some local features are originally proposed for point cloud segmentation to exploit local neighborhoods such as PointNet++ [101], which focuses on extracting features from local point groups which are sampled with farthest point sampling (FPS). Using the nearest neighbor search for feature extraction, KCNet [114] further aggregates the local features to investigate more complex relationships. KPConv [115] and 3D-GCN [32] make the kernel of the point cloud convolution deformable to generate better matches with different local geometries for segmentation.

For the aim of doing point cloud completion, PointNet++ [101] is used in PMP-Net [116] which completes the entire object gradually from the observed regions to the nearest occluded regions. The other recent work which uses the PointNet++ feature is SnowflakeNet [117], which split points in the coarsely reconstructed object to execute the completion progressively.

Processing point cloud in voxel grid. Point cloud can also be discretized into voxel grid so that some volumetric operations such as 3D convolution and transposed 3D convolution could be used to extract local features. The recent work from PVD [118], GRNet [119] and VE-PCN [120] leverage both the point cloud and the voxel grid representations. Unlike most works that rely on Chamfer distance to optimize the model, PVD [118] uses a simple Euclidean loss to optimize the shape generation model from the voxelized point cloud representation. GRNet [119] first voxelizes the point cloud, processes the voxel grid with deep learning and converts the results back to point cloud. While this solves the unorganized

structure of the point clouds, its discretization removes its advantage on reconstructing in higher resolutions. VE-PCN [120] improves the completion by supplementing the features of the decoder in the volumetric completion with the edges. This method then converts the voxels to point clouds by Adaptive Instance Normalization [121].

Skip-connection. If we want to keep some observed geometries from input to output, a skip connection is a good solution that passes input points or features in the decoder. Skip connection for point cloud cannot be built directly because the points before the global feature are not indexed explicitly with points in the decoder in many works. The skip connection could be presented in either point or feature space. To point-wise skip connection, resampling approaches among the fused point cloud of output and input could be applied, e.g. MDS [18] and FPS [101]. As for building up feature-wise skip connection, we need to project the encoder feature to the space of the decoder feature. SoftPool++ [5] solves such feature projection by a matrix which is learned from a large scale of training data.

Losses. The most common loss to match the geometries between two-point clouds is Chamfer distance and Earth-mover distance. Because point cloud datasets present 3D data efficiently with flexible local resolutions, point cloud completion is one of the most important topics. Due to the unorganized characteristic of the point cloud, point-to-point supervision cannot be enforced directly. To calculate point-wise distance Chamfer distance and Earth Mover's Distance are commonly used. Chamfer distance uses the nearest neighbor searching to assign points from one point cloud to the other. Earth Mover's Distance (EMD), sometimes referred to as Earth Moving Distance, additionally makes sure that one point in a point cloud is assigned to only one point in the other point cloud once to calculate the point-wise distance in EMD.

3.2.3 Implicit Surface

The other alternative 3D representation is implicit surfaces [122, 123, 124, 125]. Such a surface $\mathbf{S}$ is defined as a surface in Euclidean space defined with an implicit function $f(\cdot)$

$$\mathbf{S} = \{p \in \mathbb{R}^3 \mid f(p) = \tau\}\,. \tag{3.6}$$

where τ is a threshold used to distinguish the inner and outer space of the targets and p is the input 3D position, usually presenting the x, y and z coordinates. Particularly, extracting the mesh from a signed implicit field $f(\cdot)$, *e.g.* the signed distance field (SDF) [126, 127] or truncated SDF (TSDF) [128], is determined by the values of the sampled positions at a threshold τ of 0.

So by querying a sufficient amount of points (x, y, z), the surface of the targeted objects could be smoothly presented. To estimate point-wise implicit values, some approaches use point cloud features [124, 125] which consider both global and local geometric information. Unlike point cloud which only focuses on reconstructing expected structures while empty space is disregarded, implicit 3D reconstruction such as DeepSDF [122], IF-Net [129] and Points2Surf [124] creates a fine-grained 3D shape by estimating an object surface which

distinguishes the inner and outer space. Such a format not only produces smoother surface reconstruction but also reveals more local structural details compared to traditional mesh reconstruction approaches such as screened Poisson reconstruction (SPR) [130]. For this work, we do not investigate this form of 3D data, only volumetric and point cloud data – mainly for the reason of efficiency concern, especially for applications of scene completion.

3.3 Evaluation

To get approaches evaluated, some popular datasets are constructed, which will be introduced in Section 3.3.1. Some metrics are adopted or proposed specifically for different datasets which will be introduced in Section 3.3.2.

3.3.1 Dataset

Here we introduce some major dataset for evaluating 3D completion performance targerted on single object in paragraph 3.3.1 and indoor scenes in paragraph 3.3.1

Object completion. ShapeNet [131] is kept updated during these years to establish a highly-annotated, large-scale dataset of 3D shapes. Such single object 3D dataset is widely used in computer graphics, computer vision, robotics, and other related areas. Targeted on object completion task from a single view, so far there are mainly 3 datasets that are established based on the original ShapeNet dataset, which are PCN [17], TopNet [132], and MVP [133] datasets. As the first two completion targeted point cloud datasets, object point cloud in PCN [17] and TopNet [132] are sampled from ShapeNet meshes, supplementing two sets of datasets individually for low and high resolutions evaluation, which contain 2,048 and 16,384 points on the complete object, respectively, where the inputs are provided with 2,048 points for each camera view. The low-resolution dataset provided by TopNet is also commonly referred to Completion3D benchmark. Since some related works report their results in terms of the L1 and L2 metric of the Chamfer distance separately, we also report our results in both resolutions (2,048 and 16,384) and metrics (L1 and L2) in this dissertation. By providing much more partial scan samples from 26 uniformly distributed camera poses for each 3D CAD model, MVP [133] dataset establishes a multi-view partial point cloud dataset which contains over 100,000 scans in total.

Apart from those mentioned datasets which are targeted on 3D completion from a single view, ABC [134] dataset introduces a collection of one million Computer-Aided Design (CAD) models for research of geometric deep learning methods and applications, which can be used for evaluating object completion from noise scans. Each model in the ABC dataset is a collection of explicitly parameterized curves and surfaces, providing ground truth for differential quantities, patch segmentation, geometric feature detection, and shape reconstruction. Sampling the parametric descriptions of surfaces and curves allows for generating data in different formats and resolutions, enabling fair comparisons for a wide range of geometric learning algorithms. As a use case for our dataset, we perform a large-

scale benchmark for the estimation of the surface normal, comparing existing data-driven methods and evaluating their performance against both the ground truth and traditional normal estimation methods.

Semantic scene completion. The SUNCG [99] and NYU [135] are two of the earliest benchmarks for semantic scene completion and include a paired depth image and the corresponding semantically labeled volume. Based on an online interior design platform, the evaluation of SUNCG contains more than 130,000 paired depth images and voxel-wise semantic labels taken from 45,622 houses with realistic rooms and furniture layouts [99]. Like SUNCG, each image in NYU [135] dataset is also annotated with 3D semantic labels. While SUNCG comprises synthetically rendered depth data, NYU includes real scenes acquired with a Kinect depth sensor. The NYU [135] dataset is composed of 1,449 indoor depth images captured with a Kinect depth sensor. This makes the evaluation of NYU more challenging, due to the presence of real nuisances, as well as due to a limited training set of fewer than 1000 samples. The ground truth resolution of both datasets is given as voxels on the scale of $240 \times 144 \times 240$. Due to memory issues, most approahes decrease the 3D resolution to get their proposed methods evaluated, e.g. [136, 108, 137, 138, 99] produce $60 \times 36 \times 60$ semantic volumes for evaluation and [10, 107, 8] produce a resolution of $80 \times 48 \times 80$. Notice that both SUNCG and NYU datasets suggest two types of evaluation as introduced in SSCNet [99]. One evaluates the semantic segmentation accuracy on the observed surface reconstruction, while the other considers the semantic segmentation of the predicted full volumetric reconstruction.

In terms of the categorical semantics, both SUNCG [99] and NYU [135] present semantic categories of 12 classes of varying shapes and sizes, i.e. : *empty space*, *ceiling*, *floor*, *wall*, *window*, *chair*, *bed*, *sofa*, *table*, *tvs*, *furniture* and *other objects*.

Although SUNCG supplements large amount of data, they are all rendered from synthetic models. Considering that NYU dataset only contains limited number of real samples, another relevant dataset for single view 3D completion for real data is ScanNet [139] which is supplemented with its ground truth semantic completion supervision by CompleteScanNet [140] dataset. Such real dataset contains a total of 45,451 paired partial scans and semantic completion.

3.3.2 Metrics

To validate the performance of completion approaches, some metrics are proposed to validate results presented in different data formats.

Intersection over union (IoU). Intersection over Union (IoU) and the mean average precision (mAP) are commonly used to validate semantic completion results presented in a volumetric format [10, 8, 99], the predicted voxel labels are compared to ground truth labels for each object class on both the observed and occluded voxels for semantic scene completion.

Chamfer distance (CD). Chamfer Distance (CD) is the most common evaluation metric to evaluate whether geometries from two point clouds match with each other or not [17, 6]. It takes the distance of each point into account. For each point in each cloud, CD finds the nearest point in the other point set and sums the square of distance up.

Earth mover's distance (EMD). Earth Mover's Distance (EMD), sometimes referred to as Earth Moving Distance, is another popular loss metric for comparing point clouds alongside Chamfer Distance [17, 6]. One point in a point cloud is only assigned to only one point in the other point cloud once to calculate the point-wise distance in EMD. One disadvantage of the Chamfer distance and earth mover's distance is the fact that they can hardly reflect the errors in the local geometry because some crucial structures are only with a few points from which the errors will be averaged by a large amount of other points on the same target.

F-Score@1%. Since the Chamfer distance hardly reflects the errors in the local geometry as suggested in [141], the evaluation in GRNet [119] uses the metric F-Score@1% to validate the performance of point cloud completion. In general, F-Score is calculated from the precision and recall of the test, where the precision is the number of true positive results divided by the number of all positive results, including those not identified correctly, and the recall is the number of true positive results divided by the number of all samples that should have been identified as positive. Specifically, F-Score@1% computes the F-Score after matching the predicted point cloud to the ground truth with a distance threshold of 1% of the side length of the reconstructed volume.

4 Recent History

Single-view 3D completion could be presented from different types of input and output. The output is expected to be presented in multiple different 3D data formats, while the input can also be RGB. This chapter reviews works targeting 3D completion tasks with and without semantics from a single view. Considering that 3D data could be presented in different formats, different completion approaches are proposed, particularly for individual common 3D data. Here, we start with an overview of 3D completion in Section 4.1. Then we present works using single view data as input, targeting presenting the completion in the form of volumetric data in Section 4.2 and point cloud in Section 4.3, where the topics concerning both geometric completion and semantic completion are discussed. Concerning that most 3D completion models are structured using encoder-decoder architectures, we discuss some popular features connecting the encoder and decoder in Section 4.4.

4.1 3D Completion

Single RGB is one of the simplest input data to get used to produce complete 3D data, e.g. Im2Struct [142] is one of the first few works using CNN to process an image to produce cuboids to present a 3D shape. Single RGB image does not explicitly present observed structures and the background usually introduces much noise to decode objects.

To improve the completion accuracy furthermore by changing input, a sequence of RGB images and a set of depth images can be used as input. The former could be referred to as solutions revealing occluded regions with more images, and the latter solution provides 3D structures directly which can be part of output. 3D-R2N2 [95] builds recurrent neural networks to fuse multiple feature maps extracted from input RGB images sequentially to recover the 3D geometries. To further improve the reconstruction, a coarse-to-fine 3D decoder was proposed in Pix2Vox [96] as well as the residual refiner in Pix2Vox++ [97]. Due to the recent popularity of the attention mechanisms, AttSets [98] proposed to build attention layers to correlate the image features from different views. Considering a sequence of RGB images is less effective to collect, and more camera positions are not feasible in some scenarios, in contrast, 3D reconstruction in this dissertation focuses only on a single scan.

Focusing on learning-based completion depth images, most related work can be categorized depending on the input data they process – voxelized grid, point cloud, and meshes. In the following sections, we separately introduce approaches which are used for completion in different data formats, e.g. volumetric approaches in Section 4.2 and point cloud approaches in Section 4.3.

4.2 Volumetric Inference

Volumetric data is a set of samples (x, y, z, v), where the value v presents some property of the data such as occupancy values or signed distance function values for voxel at position (x, y, z). Given a depth image, its volumetric representation could be made by back-projecting all pixels in the depth image into 3D space with the help of the camera's intrinsic and extrinsic matrix. The convenience of processing the volumetric data is because of its gridded format. Such a fixed voxel grid usually makes it easy to apply 3D convolution inspired by the popularity of 2D convolution operations in CNNs [102, 103, 104] for RGB images.

Object completion. Due to the advantages in extracting meaningful local patterns while removing noises introduced by the 2D convolution operations in CNNs [102, 103, 104] for images, its straightforward extension to 3D convolutions on volumetric data also rose to fame. 3D-EPN [105] and 3D-RecGAN [106] are the first few works on this topic, where they extended the typical encoder-decoder architecture [30] to 3D. Although 3D-EPN produces coarse 3D shapes, the fine shape needs to be with the help of a correlation between coarse shape with known 3D geometry from a shape database. Such retrieval introduces an additional cost of time and limits the ability to reconstruct atypical examples. 3D-RecGAN [106] and 3D-RecGAN++ [107] on the other hand improve the fine 3D completion by applying a discriminator composed of 3D convolutions.

Semantic scene completion. Semantic scene completion is distinguishable for its end-to-end inference, which simultaneously produces geometric completion and semantic segmentation. It does not necessarily use cascaded architecture, which produces geometric completion first and then does semantic segmentation based on the completed shape. For this topic, SSCNet [99] is the first few works that demonstrated promising results in the joint task of scene completion and semantic segmentation utilizing a CNN [143, 144]. SSCNet [99] proposes a CNN-based architecture that carries out jointly the 3D scene completion and the semantic labeling from a single depth image. A voxel-wise softmax loss function is proposed as the optimizer for learning semantic segmentation of volumetric elements. SSCNet encodes the depth image into volumetric space using the Truncated Signed Distance Function (TSDF) from KinectFusion [145] so that the input of such inference pipeline is presented in 3D space for easier processing. It uses dilated convolutions to make it possible to extract features from the object with different scales. Based on SSCNet, VVNet [108] applies view-based 3D convolutions to replace SDF back-projections, resulting in more effective geometric information extraction from the input depth image. SaTNet [138] relies on the RGB-D images. They initially predict the 2D semantic segments with the RGB. The depth image then back-projects the semantically labeled pixels to a 3D volume, which goes through another architecture for 3D scene completion. ScanComplete [146] also targets semantic scene completion. However, instead of starting from a single depth image, they assume to process a large-scale reconstruction of a scene acquired via a consumer depth camera. They suggest a coarse-to-fine scheme based on an auto-regressive architecture [147], where each level predicts the completion and the per voxel semantic labeling at a different voxel resolution. Focusing on refining produced volumetric completion, 3D-RecGAN++ [148] suggests using an adversarial approach to learn how to realistically complete partial object shapes from

common classes. Later on, other generative models have also been proposed to generate 3D data directly from 2D images, such as the 3D Inductor [149]. Due to the limited amount of 3D annotation for real data, previous works have performed poorly on real depth images. Also adopting 3D discriminators to optimize the output, ForkNet [8] performs better on real datasets such as NYU [135] dataset by enforcing multiple decoders producing paired unseen partial scan and its semantic completion for training. Such a generated training pair makes it possible to get the model trained well even if there is training data. As mentioned works present input partial scan into a volumetric latent feature which potentially loses some local details, SketchSSC [109] extract contour sketches of a target to be used as local priors to complete the whole target.

In summary, the main advantage of volumetric completion is its data structure, such that deep learning methods developed for RGB images can be extended to 3D. However, this advantage is also its limitation. The fixed local resolution makes it hard to reconstruct the object's finer details without consuming much memory. The performance of recent works on the topic of volumetric semantic completion is reported in Table 4.1 on SUNCG [99] dataset and Table 4.2 on NYU [135] dataset, both reported in intersection over union (IoU in %).

SUNCG [99] dataset, intersection over union (IoU in %)

Method	ceil.	floor	wall	win.	chair	bed	sofa	table	tvs	furn.	objs.	*Avg.*
SSCNet [99] - *surf*	97.7	94.5	66.4	30.0	**36.9**	60.2	62.5	**56.3**	12.1	46.7	**33.0**	54.2
ForkNet [8] - *surf*	**98.2**	**96.9**	**67.8**	**37.4**	35.9	**72.9**	**69.6**	48.8	**20.5**	**48.4**	32.4	**57.2**
AdversarialSSC [10]	41.4	37.7	45.8	26.5	26.4	21.8	25.4	23.7	20.1	16.2	5.7	26.4
3D-RecGAN [107]	79.9	75.2	48.2	28.9	20.2	64.4	54.6	25.7	17.4	33.7	24.4	43.0
SSCNet [99]	96.3	84.9	56.8	28.2	21.3	56.0	52.7	33.7	10.9	44.3	25.4	46.4
VVNet [108]	**98.4**	**87.0**	61.0	54.8	**49.3**	**83.0**	**75.5**	**55.1**	43.5	**68.8**	**57.7**	**66.7**
SaTNet [138]	97.9	82.5	57.7	**58.5**	45.1	78.4	72.3	47.3	**45.7**	67.1	55.2	64.3
ForkNet [8]	95.0	85.9	**73.2**	54.5	46.0	81.3	74.2	42.8	31.9	63.1	49.3	63.4

Table 4.1 Evaluation on semantic scene completion. *surf* indicates that only observed geometries are validated.

NYU [135] dataset, Intersection over Union (IoU in %)

Method	res.	whole	ceil.	floor	wall	win.	chair	bed	sofa	table	tvs	furn.	objs	*Avg.*
SSCNet [99]	60	55.1	15.1	94.6	24.7	10.8	17.3	53.2	45.9	15.9	13.9	31.1	12.6	30.5
VVNet [108]	60	61.1	19.3	94.8	28.0	12.2	19.6	57.0	50.5	17.6	11.9	35.6	15.3	32.9
SaTNet [138]	60	60.6	17.3	92.1	28.0	16.6	19.3	57.5	53.8	17.7	18.5	38.4	18.9	34.4
ForkNet [8]	80	37.1	36.2	93.8	29.2	18.9	17.7	61.6	52.9	23.3	19.5	45.4	20.0	37.1
CCPNet [150]	240	63.5	23.5	96.3	35.7	20.2	25.8	61.4	56.1	18.1	28.1	37.8	20.1	38.5
SketchSSC [109]	60	71.3	43.1	93.6	40.5	24.3	30.0	57.1	49.3	29.2	14.3	42.5	28.6	41.1
SISNet [151]	60	**78.2**	**54.7**	93.8	**53.2**	**41.9**	**43.6**	**66.2**	**61.4**	**38.1**	**29.8**	**53.9**	**40.3**	**52.4**

Table 4.2 Evaluation on semantic scene completion. The value of res. (r) indicates the output volumetric resolution, which is $r \times 0.6r \times r$.

4.3 Point Cloud Inference

A point cloud is a sparse 3D representation that presents every single 3D element in a set of Cartesian coordinates (x, y, z). Unlike volumetric data which need to be gridded in fixed 3D volumes, point cloud naturally does not need to be discretized into fixed resolution so that complex local structures can be presented with higher local resolutions. Although point cloud can potentially reconstruct the object at a higher resolution, it exhibits a limited application in deep learning due to its unstructured data [40]. Such unstructured data form makes it difficult to get local features extracted because the local region is not explicitly defined. Such unstructured data does not have a particular order. Practically, if we back-project the depth image into a point cloud in 3D space, the number of points varies as we change the camera pose or the object.

Targeted to solve the unordered structure of point clouds, PointNet [40] proposes to implement max-pooling to achieve a permutation invariant latent feature presented in a single vector. Notice that such a feature can still describe the object; interestingly, a structure of contour and edges could be revealed if we trace the kept values in the feature back into the original point cloud.

Point cloud completion. Based on the latest feature of PointNet, FoldingNet [31] proposes an object completion solution that deforms 2D rectangular grids by multi-layer perceptron (MLP). Such a large grid limits the ability to produce slim and sharp structures. By decreasing the number of samples in a single 2D grid and increasing the number of grids, AtlasNet [111] and PCN [17] added more complexity as well as more details to the reconstruction. MSN [18] then further improves the completion by adding restrictions to separate different patches. Moreover, Cycle4Completion [112] is also based on PointNet features but solves the problem by training with an unsupervised cycle transformation. In addition, building a similar feature as PointNet, ME-PCN [152] takes both the occupied and the empty regions on the depth image as input for 3D completion, showing the advantage of masking the empty regions in completion.

Moving away from the global feature representation, PointNet++ [101] samples the local subset of points with the farthest point sampling (FPS) and then feeds it into PointNet [40]. Based on this feature, PMP-Net [116] completes the entire object gradually from the observed regions to the nearest occluded regions. SnowflakeNet [117] also uses the PointNet++ features to split points in the coarsely reconstructed object to execute the completion progressively.

Unlike the methods that are dependent on a vectorized global feature to solve the permutation invariant problem, RFNet [113] and PointTr [46] produce several global features in their encoder. On one hand, RFNet [113] uses its features to complete the object recurrently by concatenating the incomplete input and the predicted points level by level. On the other, PointTr [46] relies on transformers to produce a set of queries directly from the observed points with the help of positional coding. In effect, PointTr [46] does not need to compress the input into a single vector. The recent work from PVD [118], GRNet [119], and VE-PCN [120] leverage both the point cloud and the voxel grid representations. Unlike most works that use Chamfer distance to optimize the model, PVD [118] uses a simple Euclidean loss to optimize

the shape generation model from the voxelized point cloud representation. GRNet [119] first voxelizes the point cloud, processes the voxel grid with deep learning, and converts the results back to a point cloud. While this solves the unorganized structure of the point clouds, its discretization removes its advantage of reconstructing in higher resolutions. VE-PCN [120] improves the completion by supplementing the features of the decoder in the volumetric completion with the edges. This method converts the voxels to point clouds by Adaptive Instance Normalization [121]. Another solution is presented in our previous work SoftPoolNet [6], that builds local groups of features by sorting them into a feature map. 2D convolutions are then applied to the feature map. Consequently, this approach can deal with unorganized point clouds and achieve high-resolution reconstruction results. We build upon SoftPoolNet [6] and generalize the feature extraction into a module called SoftPool++. This then allows us to connect multiple modules in an encoder-decoder architecture. As a consequence, we achieve better quantitative and qualitative results. Some selected recent works validating on point cloud completion are reported in different metrics, i.e. , L1 distance in Table 4.3 and Table 4.5, L2 distance in Table 4.4 and Table 4.6, and F-Score@1% in Table 4.7 and Table 4.8.

Completion3D [132] benchmark, output resolution = 2,048, L1 distance

Method	plane	cabinet	car	chair	lamp	sofa	table	vessel	*Avg.*
FoldingNet [31]	11.18	20.15	13.25	21.48	18.19	19.09	17.80	10.69	16.48
AtlasNet [111]	10.37	23.40	13.41	24.16	20.24	20.82	17.52	11.62	17.69
PCN [17]	8.09	18.32	10.53	19.33	18.52	16.44	16.34	10.21	14.72
TopNet [132]	5.50	12.02	8.90	12.56	9.54	12.20	9.57	7.51	9.72
SA-Net [153]	**2.18**	**9.11**	**5.56**	**8.94**	9.98	**7.83**	9.94	7.23	7.74
SoftPoolNet [6]	4.76	10.29	7.63	11.23	8.97	10.08	7.13	6.38	8.31
SoftPool++ [5]	3.50	9.95	7.01	10.48	**8.45**	8.86	**5.99**	**5.60**	**7.48**

Table 4.3 Evaluation on the object completion based on the Chamfer distance trained with L1 distance (multiplied by 10^4) with the output resolution of 2,048.

Point cloud segmentation. Unlike generating more 3D structures in 3D completion, introduced in Section 4.1, the segmentation task assigns categorical labels without changing the observed geometries. We focus on introducing point cloud segmentation approaches here in this section. Point cloud segmentation depends on point-wise features; in PointNet [40], 3D features are processed into a point-wise feature with MLP. Such a point-wise feature is further concatenated with a global feature produced by max-pooling in latent space for semantic decoding, resulting in one-hot code that presents categorical information. Some features are proposed for point cloud segmentation to exploit local neighborhoods, such as PointNet++ [101] focusing on extracting features from local point groups. Also using the nearest neighbor search for feature extraction, KCNet [114] further aggregates the local features to investigate more complex relationships. KPConv [115] and 3D-GCN [32] make the kernel of the point cloud convolution deformable to generate better matches with different local geometries for segmentation.

Completion3D [132] benchmark, output resolution = 2,048, L2 distance

Method	plane	cabinet	car	chair	lamp	sofa	table	vessel	*Avg.*
FoldingNet [31]	12.83	23.01	14.88	25.69	21.79	21.31	20.71	11.51	19.07
PointSetVoting [154]	6.88	21.18	15.78	22.54	18.78	28.39	19.96	11.16	18.18
AtlasNet [111]	10.36	23.40	13.40	24.16	20.24	20.82	17.52	11.62	17.77
PCN [17]	9.79	22.70	12.43	25.14	22.72	20.26	20.27	11.73	18.22
TopNet [132]	7.32	18.77	12.88	19.82	14.60	16.29	14.89	8.82	14.25
GRNet [119]	6.13	16.90	8.27	12.23	10.22	14.93	10.08	5.86	10.64
SA-Net [153]	5.27	14.45	7.78	13.67	13.53	14.22	11.75	8.84	11.22
SoftPoolNet [6]	6.39	17.26	8.72	13.16	10.78	14.95	11.01	6.26	11.07
SoftPool++ [5]	4.59	15.82	6.78	11.41	8.82	13.37	9.15	4.93	9.36
PMP-Net [116]	**3.99**	**14.70**	**8.55**	**10.21**	**9.27**	**12.43**	**8.51**	**5.77**	**9.23**

Table 4.4 Evaluation on the object completion based on the Chamfer distance trained with L2 distance (multiplied by 10^4) with the output resolution of 2,048.

PCN [17] dataset, output resolution = 16,384, L1 distance

Method	plane	cabinet	car	chair	lamp	sofa	table	vessel	*Avg.*
3D-EPN [105]	13.16	21.80	20.31	18.81	25.75	21.09	21.72	18.54	20.15
ForkNet [8]	9.08	14.22	11.65	12.18	17.24	14.22	11.51	12.66	12.85
PointNet++ [101]	10.30	14.74	12.19	15.78	17.62	16.18	11.68	13.52	14.00
FoldingNet [31]	9.49	15.80	12.61	15.55	16.41	15.97	13.65	14.99	14.31
AtlasNet [111]	6.37	11.94	10.11	12.06	12.37	12.99	10.33	10.61	10.85
TopNet [132]	7.61	13.31	10.90	13.82	14.44	14.78	11.22	11.12	12.15
PCN [17]	5.50	10.63	8.70	11.00	11.34	11.68	8.59	9.67	9.64
MSN [18]	5.60	11.96	10.78	10.62	10.71	11.90	8.70	9.49	9.97
GRNet [119]	6.45	10.37	9.45	9.41	7.96	10.51	8.44	8.04	8.83
PMP-Net [116]	5.65	11.24	9.64	9.51	6.95	10.83	8.72	7.25	8.73
SoftPoolNet [6]	6.93	10.91	9.78	9.56	8.59	11.22	8.51	8.14	9.20
CRN [155]	**4.79**	**9.97**	**8.31**	9.49	8.94	10.69	**7.81**	8.05	8.51
SoftPool++ [5]	5.50	10.02	8.73	**9.05**	**7.53**	**10.24**	8.01	**7.43**	**8.31**

Table 4.5 Evaluation on the object completion based on the Chamfer distance trained with L1 distance (multiplied by 10^3) with the output resolution of 16,384.

Point cloud semantic completion. Unlike volumetric data, semantic completion for point cloud keeps changing the geometries, so it could be presented in a cascaded way, where completion is done before head segmentation.

PCN [17] dataset, output resolution = 16,384, L2 distance

Method	plane	cabinet	car	chair	lamp	sofa	table	vessel	*Avg.*
FoldingNet [31]	3.15	7.94	4.68	9.23	9.23	8.90	6.69	7.33	7.14
TopNet [132]	2.15	5.62	3.51	6.35	7.50	6.95	4.78	4.36	5.15
MSN [18]	1.54	7.25	4.71	4.54	6.48	5.89	3.80	3.85	4.76
AtlasNet [111]	1.75	5.10	3.24	5.23	6.34	5.99	4.36	4.18	4.52
NSFA [156]	1.75	5.31	3.43	5.01	4.73	6.41	4.00	3.56	4.28
PCN [17]	1.40	4.45	**2.45**	4.84	6.24	5.13	3.57	4.06	4.02
PF-Net [157]	1.55	4.43	3.12	3.96	4.21	5.87	3.35	3.89	3.80
CRN [155]	1.46	4.21	2.97	3.24	5.16	5.01	3.99	3.96	3.75
SoftPoolNet [6]	1.63	3.79	3.05	3.27	2.95	3.78	2.59	2.25	2.91
GRNet [119]	1.53	3.62	2.75	**2.95**	**2.65**	3.61	2.55	2.12	2.72
SoftPool++ [5]	**1.27**	**3.43**	2.65	2.98	2.67	**3.38**	**2.27**	**1.85**	**2.55**

Table 4.6 Evaluation on the object completion based on the Chamfer distance trained with L2 distance (multiplied by 10^3) with the output resolution of 16,384.

PCN [17] dataset, output resolution = 16,384, F-Score@1%

Method	plane	cabinet	car	chair	lamp	sofa	table	vessel	*Avg.*
FoldingNet [31]	0.642	0.237	0.382	0.236	0.219	0.197	0.361	0.299	0.322
TopNet [132]	0.771	0.404	0.544	0.413	0.408	0.350	0.572	0.560	0.503
AtlasNet [111]	0.845	0.552	0.630	0.552	0.565	0.500	0.660	0.624	0.616
SoftPoolNet [6]	0.831	0.605	0.685	0.649	0.715	0.601	0.746	0.721	0.694
PCN [17]	0.881	0.651	**0.725**	0.625	0.638	0.581	0.765	0.697	0.695
MSN [18]	**0.885**	0.644	0.665	0.657	0.699	0.604	0.782	0.708	0.705
GRNet [119]	0.843	0.618	0.682	0.673	0.761	0.605	0.751	0.750	0.708
SoftPool++ [5]	0.867	**0.693**	0.706	**0.712**	**0.794**	**0.689**	**0.825**	**0.804**	**0.761**

Table 4.7 Evaluation on the object completion based on the F-Score@1% trained with L2 Chamfer distance and the output resolution of 16,384.

4.4 Embedding of the 3D Completion Model

Recently, 3D reconstruction with the help of a radiance field can deliver high-fidelity surface reconstruction. Several methods using neural networks have been proposed for volume rendering and surface reconstruction, such as VolSDF [159] that applied the cumulative distribution function of Laplacian distribution to evaluate the density function from SDF for volume rendering and surface reconstruction. Another notable work of NeuS [160] adopts an unbiased density function for signed distance field (SDF) for more accurate reconstruction; and, NeuralWarp [161] improved the accuracy on low-textured areas by optimizing consistency between warped views of different images. Moreover, there have been numerous works that were developed from NeuS [160]. For example, SparseNeuS [162] extends NeuS

MVP [133] dataset, output resolution = 16,384, F-Score@1%

Method	plane	cabinet	car	chair	lamp	sofa	table	vessel	*Avg.*
TopNet [132]	0.789	0.621	0.612	0.443	0.387	0.506	0.639	0.609	0.576
PCN [17]	0.816	0.614	0.686	0.517	0.455	0.552	0.646	0.628	0.614
SoftPoolNet [6]	0.843	0.568	0.636	0.623	0.698	0.568	0.680	0.710	0.666
GRNet [119]	0.853	0.578	0.646	0.635	0.710	0.580	0.690	0.723	0.677
MSN [18]	0.879	0.692	0.693	0.599	0.604	0.627	0.730	0.696	0.690
CRN [155]	0.898	0.688	0.725	0.670	0.681	0.641	0.748	0.742	0.724
SoftPool++ [5]	0.862	0.622	0.704	0.695	0.783	0.649	0.776	0.778	0.734
ECG [158]	0.906	0.680	0.716	0.683	0.734	0.651	0.766	0.753	0.736
PoinTr [46]	0.888	0.681	0.716	0.703	0.749	0.656	0.773	0.760	0.741
NSFA [156]	0.903	0.694	0.721	0.737	0.783	**0.705**	**0.817**	0.799	0.770
VRCNet [133]	**0.928**	**0.721**	**0.756**	**0.743**	**0.789**	0.696	0.813	**0.800**	**0.781**

Table 4.8 Evaluation on the object completion based on the F-Score@1% trained with L2 Chamfer distance and the output resolution of 16,384.

to use fewer images for reconstruction. The geometric details can be revealed because of the precisely recorded camera poses and a multi-view stereo dataset towards a single object.

Single-view completion usually tackles the task of various targets compared to these works; single-view completion is usually inferencing based on a learned categorical prior so that the estimated geometry could be regarded as a conditional reconstruction given a partial scan. Such a conditional inference usually presents the categorical prior with a latent feature. While there are many point cloud completion works carried out in encoder-decoder architectures, e.g. FoldingNet [31], AtlasNet [111], PCN [17] and MSN [18], which is connected with only a single latent feature such as PointNet [40] or PointNet++[101] feature, the feature itself usually contains strong shape and categorical priors. Such implicitly learned categorical prior can be helpful for other tasks such as classification. A notable work from OcCo [163] demonstrates that the weights trained for completion are also valuable for other tasks like segmentation and classification.

Contributions 5

This chapter summarizes the main contributions, supplemented with the associated publication for each work. The primary focus of our work is completing 3D geometries in two data formats: volumetric data in Section 5.1 and point cloud in Section 5.2. First, to leverage the well-investigated 2D operators such as 2D convolution to 3D data, Section 5.1 demonstrates solutions to make completion presenting in 3D grids from a single view scan by adapting existing operations, architectures, and training techniques to 3D completion. Because volumetric representation limits the resolution of the completed 3D targets, we also focus on completing the target object as a point cloud with flexible local resolutions in Section 5.2, where the reconstructed objects are with finer details. Eventually, inspired by the improved performance on 3D completion by feature learning, especially metric learning and variational inference, we further try to generalize feature optimization methods in Section 5.3 for other 1D, 2D and 3D tasks, including language translation planar reconstruction, hand pose estimation, etc.

5.1 Volumetric Completion

This section focuses on semantic volumetric completion from a single view. The volumetric 3D completion is usually presented in a 3D grid with a size of $d_1 \times d_2 \times d_3$. Incorporating the semantics for N_c categories estimated simultaneously, N_c 3D grids are then used in total, resulting in 4D data with the shape of $d_1 \times d_2 \times d_3 \times N_c$. To estimate such expected completion, Section 5.1.1 and Section 5.1.2 introduce two of our proposed approaches focused on two different formats of input partial scan: depth image and signed distance field. Section 5.1.1 starts with a simple 2.5D image input while Section 5.1.2 leverages a 3D input which is a signed distance field (SDF).

5.1.1 Adversarial Semantic Scene Completion from a Single Depth Image (International Conference on 3D Vision 2018)

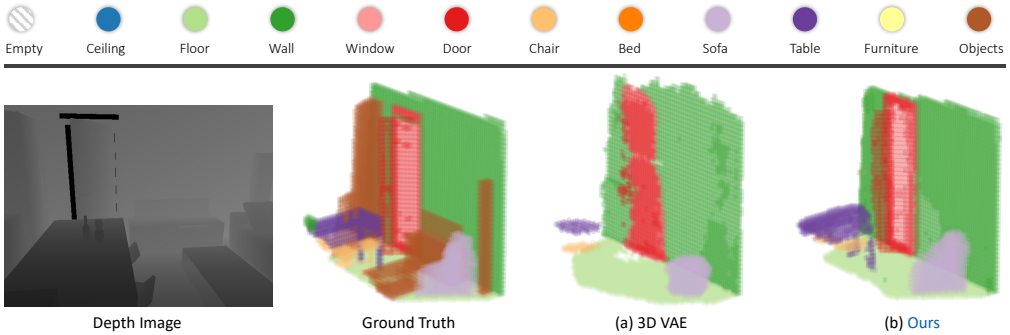


Figure 5.1 Taking a single depth image as input, our proposed AdversarialSSC [10] outperforms previous works on revealing more furniture.

Considering that an image has a limited view of the entire scene, constrained by the sensor's viewpoint, our deep learning approach aims to complete the scene by revealing the hidden structures that are not visible in the input. A leisurely start to tackle the task of semantic completion is presenting output in volumetric space so that existing 2D convolutional architectures could be adapted by changing 2D operators to 3D.

While there are existing approaches that solve single-view 3D semantic completion, they operate entirely through 3D convolutions during inference which could be more efficient. In contrast, during inference, our model recovers the 3D structure in latent space from the 2.5D input by down-sampling through layers of 2D convolutions to improve the efficiency of our encoder. However, having end-to-end training with a 2D encoder and a 3D decoder cannot produce a satisfying reconstruction. This is because the 2D latent feature does not contain the spatial 3D geometries. We then propose to match the reshaped 2D feature to a 3D feature of 3D VAE. To make the feature matching easier compared to 3D VAE, a variational inference is used as a constraint on the latent variables. During training, the adversarial loss is used to make the two types of features similar. We additionally use adversarial learning on the output to make our model able to reconstruct and classify every type of object, including small objects.

Contributions

Yida Wang proposed and implemented the architecture, and adjusted the optimizer to fit both feature learning from two domains and the completion inference from the image to the expected semantic 3D reconstruction. Additionally, he evaluated on synthetic datasets.

David Tan helped rephrase the methodology and gave suggestions about the evaluations.

Federico Tombari corrected the evaluation in terms of fairness on common metrics.

Nassir Navab financed this work on behalf of the leader of TUM CAMP.

Adversarial Semantic Scene Completion from a Single Depth Image

Yida Wang[1], David Joseph Tan[2], Nassir Navab[1], Federico Tombari[1,2]

[1] Technical University of Munich
[2] Google

Published version: https://doi.org/10.1109/3DV.2018.00056

Adversarial Semantic Scene Completion from a Single Depth Image

Yida Wang, David Joseph Tan, Nassir Navab, Federico Tombari
Technische Universität München
Boltzmannstraße 3, 85748 Garching bei München

Abstract

We propose a method to reconstruct, complete and semantically label a 3D scene from a single input depth image. We improve the accuracy of the regressed semantic 3D maps by a novel architecture based on adversarial learning. In particular, we suggest using multiple adversarial loss terms that not only enforce realistic outputs with respect to the ground truth, but also an effective embedding of the internal features. This is done by correlating the latent features of the encoder working on partial 2.5D data with the latent features extracted from a variational 3D auto-encoder trained to reconstruct the complete semantic scene. In addition, differently from other approaches that operate entirely through 3D convolutions, at test time we retain the original 2.5D structure of the input during downsampling to improve the effectiveness of the internal representation of our model. We test our approach on the main benchmark datasets for semantic scene completion to qualitatively and quantitatively assess the effectiveness of our proposal.

1. Introduction

Inspired by the way humans can imagine the structure of a room by looking at an image, we propose an algorithm that reconstructs the entire scene geometry and semantics from a single depth image. By directly reconstructing the scene from one view, the challenge is to plausibly complete the scene in place of the hidden structures that are not visible from the input depth image. To this end, we utilize a learning strategy that allows the algorithm to simultaneously perceive the objects in the scene and use its contextual shape to fill the hidden structures. In addition, we simultaneously estimate a semantic segmentation of the completed 3D scene geometry.

Reconstructing the environmental information in 3D space from a single viewpoint is relevant for a lot of tasks in the field of augmented reality [24], robotic perception [14] and scene understanding [15], where users and autonomous agents often have only a limited set of observations of the surrounding, and would benefit from a complete semantic

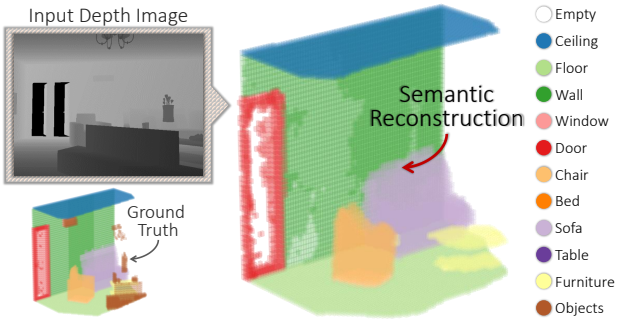


Figure 1: The input depth image and the output semantic 3D reconstruction.

reconstruction of the scene geometry. To push the scientific effort along these directions, recently large-scale benchmark datasets, such as SUNCG [22], NYU [20], ScanNet [5] and SceneNet [8], have been proposed to evaluate different visual scene understanding tasks including those of scene completion and semantic segmentation.

A few methods have recently been proposed in the direction of 3D shape completion. In particular, SSCNet [22] demonstrated good results in the joint task of scene completion and semantic segmentation by means of a CNN [23, 9]. They encode the depth image into volumetric space using the Truncated Signed Distance Function (TSDF) from KinectFusion [17]. Differently, 3D-RecGAN++ [26] suggests using an adversarial approach to learn how to realistically complete partial object shapes from common classes. Generative models have also been proposed to generate 3D data directly from 2D images such as the 3D Inductor [7].

In this work, we focus on the data acquired from depth cameras, with the goal of reconstructing and semantically labeling the whole scene from one single range image. As a scene may contain small objects and complicated shapes, we apply a generative adversarial model for this semantic completion task. Combined with an encoder and a generator, our architecture uses depth images directly as the input information and generate 3D volumetric data whose elements are labeled with object categories. Specifically, we use two discriminators to train the architecture to back-

project the depth information into the 3D volumetric space with semantic labels. One discriminator is used to optimize the entire architecture by comparing the reconstructed semantic scene with the ground truth. Since the 3D variational auto-encoders models the latent features of the volumetric data very well, we designed our architecture such that our encoder for depth images learns similar latent features. To do so, we introduce another discriminator for optimizing the learnt latent features.

To summarize, we have two main contributions. Firstly, we propose the first generative adversarial network aimed at semantic 3D scene completion, and we demonstrate how the adversarial approach is a meaningful choice for the task at hand. Secondly, we enforce adversarial learning not just on the output reconstruction, but also on the latent space to improve the quality of the results. We evaluate our approach on the main benchmark dataset for semantic scene completion to qualitatively and quantitatively assess the effectiveness of our proposal.

2. Related work

Being particularly difficult and training intensive, the task of shape completion from 2.5D has only, with the recent explosion of deep learning, started to become a main research trend in the community. SSCNet [22] proposes a CNN-based architecture that carries out jointly the 3D scene completion and the semantic labeling from a single depth image. A voxel-wise softmax loss function is proposed as the optimizer for learning semantic segmentation of volumetric elements. For training, the method assumes to know the viewpoint as well as the alignment of the depth maps and the reconstructed volumes to a common 3D reference frame. Differently, our approach drops such assumptions and can work without the information regarding the camera pose or the global alignment. On a different task, 3D-RecGAN++ [26] suggests learning a 3D adversarial generative model to complete partial 3D shapes of common object classes. The use of the adversarial loss is motivated to provide realistic and plausible interpolations of the missing shape parts.

Scene and object completion has been investigated also from RGB data. MarrNet [25] proposes to reconstruct 3D object from 2.5D sketches with normal, depth and silhouette information extracted from 2D images. Inspired by MarrNet, the encoder of our architecture is mainly composed of 2D convolutional operators while the generator is mainly composed of 3D deconvolutional operators. The difference lies in the fact that the latent variables of our model are learned to be similar to the feature extracted from a 3D VAE trained on the complete volumetric data.

PointOutNet [6] proposes an encoder-decoder deep architecture to complete 3D objects from RGB images in the form of 3D coordinates. 3D-R^2N^2 [3] tries to reconstruct a volumetric representation of an object from an RGB image by training a recurrent neural network over a latent representation of the RGB data. In addition, by combining scene reconstruction and GAN, 3D-Scene-GAN [27] is introduced for reconstructing complicated 3D scenes from RGB views with mesh and texture by applying a discriminator to distinguish between the rendered 2D images of the scene and real ones.

On a different topic, feature representations for generative models has been often deployed for reconstruction tasks, *e.g.* by means of Variational Auto-Encoders (VAE) [1, 13] and conditional VAE (CVAE) [12, 21], which are two popular methods to learn features from an input data in continuous latent spaces trained via variational inference. 3D VAE [2] is also introduced by replacing 2D convolutional kernels with 3D kernels for auto-encoding voxel data.

3. Semantic reconstruction

The semantic reconstruction algorithm takes a single view of the scene, depicted by a depth image x, to predict its 3D volumetric representation y. The voxels of y are semantically labeled with N_c object classes, denoted as an $N_c \times 1$ one-hot vector, *i.e.* a binary vector where one of its element has a value of 1 to indicate the object category while the other elements remain zero. Considering that the image has a limited view of the entire scene, constrained by the sensor's viewpoint, the objective of our deep learning approach is also to complete the scene by revealing the hidden structures that are not visible in the input. Therefore, simulateneously learning the geometric structure and the semantic information allows the algorithm to learn the contextual cues that can in turn represent the objects in the reconstruction.

Specifically, the depth image is a 640×480 image that represents the z-axis of the camera coordinate system. As input to our deep learning architecture, this image is downsampled to 320×240 in order to conserve GPU memory. The resulting volumetric reconstruction is represented by N_c grids of size $40 \times 80 \times 80$ filled with binary elements presenting the labels for each of the N_c objects. For simplicity, we denote this 4D data as $40 \times 80 \times 80 \times N_c$.

From the depth image to the 3D volume, our architecture is a concatenation of an encoder $\mathcal{E}_{\text{dep}}$ with 2D convolutional operators that convert the input depth image into a lower-dimensional latent feature l_{dep}; and, a generator $\mathcal{G}$ with 3D deconvolutional kernels that takes l_{dep} to build the semantic reconstruction. This architecture is illustrated in Fig. 2.

Encoder for depth image. The encoder $\mathcal{E}_{\text{dep}}$ compresses the depth image into a feature in the latent space. Its architecture is a concatenated network that sequentially combines 2D convolutional layers and max-pooling layers. The operators for the paired convolutional and pooling layers

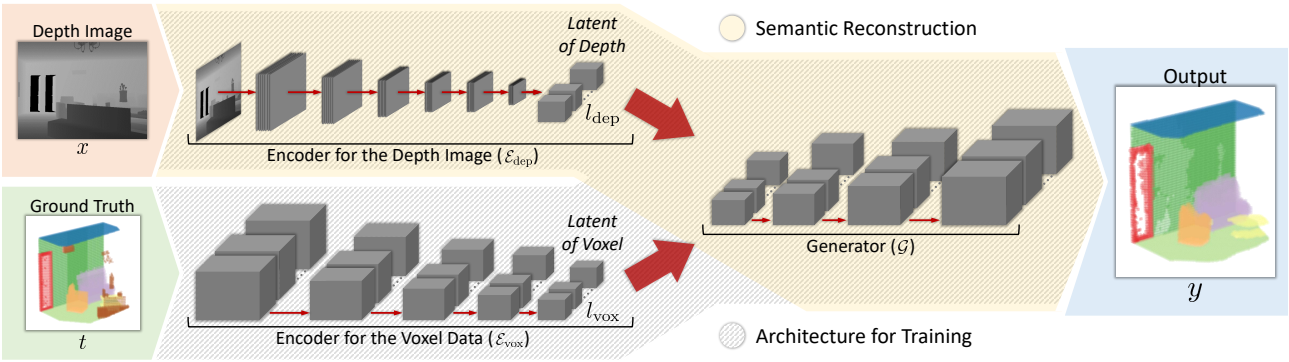


Figure 2: Deep architecture for the semantic reconstruction in Sec. 3 and for the training procedure in Sec. 4. The former is the concatenated architecture of the encoder $\mathcal{E}_{\text{dep}}$ and the generator $\mathcal{G}$ reconstructs from the depth image to a voxel data while the latter is the concatenated architecture of $\mathcal{E}_{\text{vox}}$ and $\mathcal{G}$ is a 3D variational auto-encoder [2] for self-reconstruction.

are 2D convolutional kernels with, respectively, the size of 3×3 and stride of 1×1 and the size of 2×2 with stride of 2×2. Each of these paired layers is processed by a leaky ReLU activation function [16]. Therefore, the output of every ReLU activation is a multi-channel 2D image. After six convolutions operations, the result is an 80-channel 5×3 image which is reshaped into a set of 3D volume of size $5 \times 3 \times 5 \times 16$. The output of the encoder represents the latent feature l_{dep} of the semantic reconstruction architecture.

Generator. With the goal of regressing the semantic reconstruction, the generator $\mathcal{G}$ unwraps the latent feature to a higher dimensional voxel data. We assemble the generator with 3D deconvolutional layers with the size of $3 \times 3 \times 3$ and stride of $2 \times 2 \times 2$ which are processed by the ReLU function as activation. After four deconvolutional layers, the output of the generator is the voxel-wise classification y. By doing this, y is presented in the shape of $80 \times 48 \times 80 \times N_c$.

4. Architecture for training

Although our semantic reconstruction algorithm in Sec. 3 could be optimized only with encoder $\mathcal{E}_{\text{dep}}$ and generator $\mathcal{G}$, the performance after training this way is subpar (see Sec. 7.1). Hence, we include three components during the training process to improve the performance – (1) the encoder for the voxel data, (2) the discriminator for the reconstruction and (3) the discriminator for the latent features.

Specifically, we introduce another encoder $\mathcal{E}_{\text{vox}}$ to extract the feature l_{vox} such that the latent feature from the encoder $\mathcal{E}_{\text{dep}}$ is driven to be similar to a feature extracted from $\mathcal{E}_{\text{vox}}$. Thus, a discriminator $\mathcal{D}_l$ is used to optimize this similarity as illustrated in Fig. 3 and consequently updates the parameters in $\mathcal{E}_{\text{dep}}$. Notably, $\mathcal{E}_{\text{vox}}$ is optimized together with the generator $\mathcal{G}$ as a 3D variational auto-encoder (3D VAE) [2] to learn meaningful weights from training samples representing complete 3D semantic volumes.

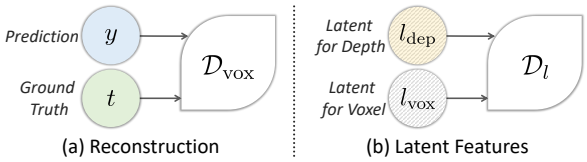


Figure 3: Variables associated to each discriminator.

Encoder for the voxel data. Since reconstructing from one image has a restrictive view of the scene, we want to make the latent features l_{dep}, extracted from the depth image, to be similar to the complete volumetric data in order to incorporate structures that are not visible from the input. We introduce another encoder $\mathcal{E}_{\text{vox}}$ to extract the feature l_{vox} into the architecture for learning. The input of $\mathcal{E}_{\text{vox}}$ is the ground truth volumentric data with semantic labels such that all the operators in its architecure are 3D convolution kernels with the size of $3 \times 3 \times 3$ and stride of $2 \times 2 \times 2$ as shown in Fig. 2. The last layer of the encoder $\mathcal{E}_{\text{vox}}$ produces 16 blocks. The size of the output from $\mathcal{E}_{\text{vox}}$ is set to be the same as $\mathcal{E}_{\text{dep}}$ because we want to make the latent representation of the input depth images to be as similar as possible to that of the ground truth volumetric representations. Thus, measuring the similarity between l_{dep} and l_{vox} is possible because the latent representation compresses the results for both the depth and voxel data. As illustrated in Fig. 2, the latent features from both the encoders $\mathcal{E}_{\text{dep}}$ and $\mathcal{E}_{\text{vox}}$ go through the same generator $\mathcal{G}$ to predict semantic volumetric data.

Discriminator for the reconstruction. Encouraged by the benefits of the generative models trained with adversarial techniques [4], we introduce the discriminator $\mathcal{D}_{\text{vox}}$ in training to optimize our semantic reconstruction by comparing our prediction against the ground truth as shown in Fig. 3. The architecture of $\mathcal{D}_{\text{vox}}$ is similar to the encoder $\mathcal{E}_{\text{vox}}$ except for the last layer. In Fig. 4 (a), all the four 3D

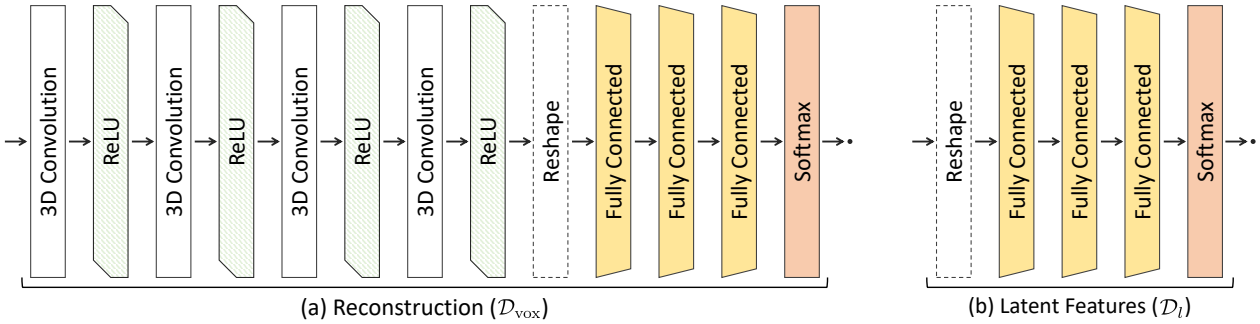


Figure 4: Architecture of the two discriminators.

convolutions have $3 \times 3 \times 3$ kernels with stride of $2 \times 2 \times 2$. Then, the output of the last convolutional layer with the size of $5 \times 3 \times 5 \times 16$ is reshaped to a vector of 1200 dimensions. This is processed by three fully-connected layers with output sizes, respectively, of 256, 128 and 1. Hence, the final logit is a binary indicator to determine whether the predicted volumetric data is the expected ones or not, which is widely used in GAN [19].

Discriminator for the latent features. Since the output of $\mathcal{E}_{\text{dep}}$ and $\mathcal{E}_{\text{vox}}$ are passed to the same generator $\mathcal{G}$, the resulting latent feature from the depth image l_{dep} is driven to be similar to the feature extracted from the ground truth volumetric data l_{vox}. We introduce another discriminator $\mathcal{D}_l$ aiming at distinguishing the latent descriptors illustrated in Fig. 3 and consequently updating the parameters in $\mathcal{E}_{\text{vox}}$. As input to $\mathcal{D}_l$, the latent variables are reshaped from $5 \times 3 \times 5 \times 16$ to a vector of 1200 dimensions. The architecture of $\mathcal{D}_l$ in Fig. 4 (b) is constructed purely by three fully-connected layers with output sizes of 256, 128 and 1. Finally, the output of the discriminator is also a logit where 1 indicates that the latent feature from the depth image is similar to the feature of the 3D VAE; otherwise, the value is zero.

5. Optimization

The goal of the optimization is to enforce the latent features from the depth image (l_{dep}) and the predicted reconstruction (y) to resemble the latent features of the 3D VAE (l_{vox}) and the ground truth volumetric data (t), respectively. Since the architecture for the semantic reconstruction and the 3D VAE share the same generator (see Fig. 2), we distinguish their results by denoting y_x as the prediction from the semantic reconstruction while y_t from the 3D VAE.

Loss functions. When we solely consider the semantic reconstruction architecture (see Sec. 3), the loss that compares the prediction and the ground truth for all the N_c objects is represented as

$$\mathcal{L}_{x \to y}(\mathcal{E}_{\text{dep}}, \mathcal{G}) = \sum_{c=1}^{N_c} [\epsilon(y_x(c), t(c))] \tag{1}$$

where we define the per-object error as

$$\epsilon(q, r) = -\gamma r \log q - (1-\gamma)(1-r) \log(1-q) \tag{2}$$

with γ as the hyper-parameter which weighs the relative importance of false positives against false negatives. Consequently, the error penalizes when the prediction and the ground truth are distinct.

To further improve the reconstruction performance using GAN (see Sec. 4), we include an adversarial loss

$$\mathcal{L}_{\text{GAN-}y}(\mathcal{E}_{\text{dep}}, \mathcal{G}) = -\log(\mathcal{D}_{\text{vox}}(y_x)) \tag{3}$$

based on the trained discriminator $\mathcal{D}_{\text{vox}}$ that optimizes the semantic reconstruction architecture by updating the parameters of its encoder and generator. On the other hand, training for the parameters in $\mathcal{D}_{\text{vox}}$ entails a loss function

$$\mathcal{L}_{\text{GAN-}y}(\mathcal{D}_{\text{vox}}) = -\log(\mathcal{D}_{\text{vox}}(t)) - \log(1 - \mathcal{D}_{\text{vox}}(y_x)) \tag{4}$$

so that the discriminator $\mathcal{D}_{\text{vox}}$ could be further optimized to be capable of distinguishing the generated volumetric data from the ground truth.

As for the 3D VAE, we can train this architecture by minimizing a loss similar to (1). However, since we use the ground truth reconstruction as the input, the loss function

$$\mathcal{L}_{t \to y}(\mathcal{E}_{\text{vox}}, \mathcal{G}) = \sum_{c=1}^{N_c} [\epsilon(y_t(c), t(c))] \tag{5}$$

enforces the predicted reconstruction y_t to be similar to its input. By training with variational inference by optimizing the evidence lower bound (ELBO) [1, 13], the latent variables are distributed in a simple Gaussian distribution.

In reference to semantic reconstruction architecture, the 3D VAE influences the latent variable l_{dep} to be as similar

to l_{vox} as possible by using discriminator $\mathcal{D}_l$ to determine whether l_{dep} is presented similar to l_{vox}. Therefore, similar to $\mathcal{D}_{\text{vox}}$, optimizing the similarity between the latent features uses another discriminator $\mathcal{D}_l$ such that the loss function to update the encoder $\mathcal{E}_{\text{dep}}$ is

$$\mathcal{L}_{\text{GAN-}l}(\mathcal{E}_{\text{dep}}) = -\log(\mathcal{D}_l(l_{\text{dep}})) \tag{6}$$

while training for $\mathcal{D}_l$ involves

$$\mathcal{L}_{\text{GAN-}l}(\mathcal{D}_l) = -\log(\mathcal{D}_l(l_{\text{vox}})) - \log(1 - \mathcal{D}_l(l_{\text{dep}}))\,. \tag{7}$$

Minimization. Now that we have all the loss functions, our optimization is defined as a combination of five components. The first two are based on the architecture for training in Fig. 2. From the depth image x and the ground truth t, we separately train them one after the other for the samples in a mini-batch with

$$\min(\mathcal{L}_{x\to y}(\mathcal{E}_{\text{dep}}, \mathcal{G})) \text{ and} \tag{8}$$
$$\min(\mathcal{L}_{t\to y}(\mathcal{E}_{\text{vox}}, \mathcal{G})) \tag{9}$$

so that the parameters of the architectures are updated alternatively. At the same time, the variational inference sets a constraint on the latent variables as a Gaussian distribution which makes it easier for the output of both of the encoders to match with each other.

Assuming that the discriminators are trained, we can fix their parametric model in order to update the encoder to move towards

$$\min(\mathcal{L}_{\text{GAN-}l}(\mathcal{E}_{\text{dep}})) \tag{10}$$

while update both the encoder and the generator toward

$$\min(\mathcal{L}_{\text{GAN-}y}(\mathcal{E}_{\text{dep}}, \mathcal{G})) \tag{11}$$

which are also optimized alternatively.

Finally, the two discriminators are trained by minimizing

$$\min(\mathcal{L}_{\text{GAN-}y}(\mathcal{D}_{\text{vox}})) \text{ and} \tag{12}$$
$$\min(\mathcal{L}_{\text{GAN-}l}(\mathcal{D}_l)) \tag{13}$$

such that the former is used to penalize poorly reconstructed voxel data in reference to the ground truth while the latter makes the latent codes computed from the depth image similar to the latent feature extracted from a well trained 3D VAE. Notably, the discriminators are updated when the accuracy in distinguishing the generated outputs are lower than specific level [4]. We set this threshold to 15% in our experiments.

In practice, we use the Adam optimizer [11] with a learning rate of 0.0001.

6. Implementation details

We use the paired depth image and semantically labeled volumes provided by SUNCG [22] and NYU [20]. The size of volumetric data with the object labels is $240 \times 240 \times 240 \times N_c$ where N_c is set to 12. Due to the limited GPU memory, we down-sample the data to $80 \times 48 \times 80 \times N_c$ by max-pooling with $3 \times 3 \times 3$ kernel and $3 \times 3 \times 3$ stride. In this manner, the original volumetric data is presented in a space with a lower resolution which is suitable for training in a single GPU with no more than 12 GB memory. In our experiments, we use a single NVIDIA TITAN Xp for training and the batch size is set to be 8. The depth images are also resized from 640×480 to 320×240 with a bilinear interpolation.

The 12 object classes in our experiments are based on SUNCG [22] that includes: empty space, ceiling, floor, wall, window, door, chair, bed, sofa, table, furniture and small objects. Since the ratios of samples in each categories are not balanced, we redesign the evaluation strategy in Sec. 7 to concentrate on reconstructing important objects in the indoor condition with small amount of voxels such as furnitures and small objects.

7. Experiments

We evaluated on the SUNCG dataset [22] that includes pairs of depth images and the corresponding semantically labelled 3D reconstructions.

Evaluation Strategy. Considering that this dataset is for the indoor environments, over 90% of the reconstructed scene is empty. Then, when we exclude the empty spaces, simple structures such as the wall, floor and ceiling dominate the voxels in the scene. This means that the ratio of the number of voxels for different object classes is not balanced. For instance, we noticed that the SUNCG test sets [22] do not have enough small objects and furnitures. In this case, if the learned architecture enhances its ability to predict the empty spaces and the simple structures, their accuracy is significantly higher than the results predicted by an architecture that focuses on distinguishing the other object classes.

Since the ratio of voxels for small objects and furnitures in the training dataset are higher than the one in the test set in SUNCG [22], we design a 10-fold cross validation by splitting the training data which was introduced by [10]. The entire dataset is divided into ten folds with the same amount of samples, the evaluation procedure then uses 1 of the 10 folds as the test set and the remaining 9 as the training dataset. Thereafter, the final result is the average of the ten evaluations.

	empty	ceil.	floor	wall	win.	door	chair	bed	sofa	table	furn.	objs.	*Avg.*
3D VAE [2]	49.3	26.1	33.2	29.7	14.4	4.6	0.7	16.4	13.9	0.0	0.0	0.0	30.8
3D-RecGAN++ [26]	49.3	32.6	37.7	36.0	23.6	13.6	8.7	20.3	16.7	9.6	0.2	3.6	36.1
Ours without $\mathcal{D}_l$	49.6	42.0	35.9	44.8	28.5	25.5	15.4	28.6	20.1	21.5	11.5	6.5	42.7
Ours without $\mathcal{D}_{\text{vox}}$	49.6	39.0	35.7	43.4	26.8	23.8	18.5	29.2	22.4	16.8	10.4	5.3	41.7
Ours (*Proposed*)	49.7	41.4	37.7	45.8	26.5	26.4	21.8	25.4	23.7	20.1	16.2	5.7	**44.1**

Table 1: Semantic scene completion results on the SUNCG test set with depth map for IoU in percentage.

	empty	ceil.	floor	wall	win.	door	chair	bed	sofa	table	furn.	objs.	*Avg.*
3D VAE [2]	99.6	18.8	68.9	63.6	25.0	8.5	4.2	16.4	9.5	1.3	0.4	2.6	65.6
3D-RecGAN++ [26]	99.9	21.5	76.2	78.8	31.9	15.3	8.1	18.7	10.2	2.9	1.4	4.3	79.4
Ours without $\mathcal{D}_l$	100.0	29.1	72.8	92.9	29.7	20.2	9.9	20.8	13.5	2.6	6.2	3.0	92.3
Ours without $\mathcal{D}_{\text{vox}}$	99.9	28.6	70.3	91.5	28.3	18.8	9.1	20.2	12.7	2.6	4.9	2.6	90.1
Ours (*Proposed*)	100.0	29.1	76.2	94.2	32.0	22.7	11.4	21.9	14.2	3.1	7.6	3.6	**94.5**

Table 2: Semantic scene completion results on the SUNCG test set with depth map for mAP in percentage.

Metric. We evaluate the performance of the reconstructor based on the intersection over union (IoU) and the mean average precision (mAP) of the predicted voxel labels compared to ground truth labels [22] where we evaluate the IoU of each object classes on both the observed and occluded voxels for semantic scene completion. Notably, instead of taking the average IoU and mAP as the mean of the results from individual categories, we calculate the average with respect to the number of voxels in each category.

Comparison. We compare our results against 3D VAE [2] and 3D-RecGAN++ [26]. In order to directly estimate the volumetric reconstruction solely from the input depth image, we modify [2, 26] by scaling the surface generated by the depth image through bilinear interpolation to fit the $80 \times 48 \times 80$ volumetric grid which serves as the input to [2, 26]. Furthermore, we added the loss function from (1) in training to perform semantic segmentation. Notably, the U-Net [18] connection between encoder and decoder in 3D-RecGAN++ [26] are still applied by resizing the scale of every layers. In addition, we further investigate the advantage of the discriminators by evaluating our approach without $\mathcal{D}_l$ and $\mathcal{D}_{\text{vox}}$. Based on Sec. 5, when implementing our approach without $\mathcal{D}_l$, (10) and (13) are discarded in the optimization; while, the implementation without $\mathcal{D}_{\text{vox}}$ discards (11) and (12).

7.1. SUNCG

SUNCG [22] is a dataset of 3D scenes which contains pairs of depth image and its corresponding volumetric scene where all objects in the scene are semantically annotated. We implemented the 10-fold validation on the pairs for the 111,697 different scenes.

Comparison against other approaches. The evaluation on both the IoU and mAP in Table 1 and Table 2 shows that our generative model performs better than 3D VAE [2] and 3D-RecGAN++ [26] which are the recent works on 3D generative architectures. We acquired an IoU of 44.1% and an mAP of 94.5% that is 8% and 15.1% better than the next best performing approach.

Comparison on the architecture for learning. To understand the advantage of incorporating the components in learning, we investigate learning our method without the discriminators. Without the discriminator for the latent features, our performance decreases by 1.4% in IoU and 2.2% in mAP; while, without the discriminator for the reconstruction, the results decrease by 2.4% in IoU and 4.4% in mAPo. However, it is noteworthy to mention that, even without these discriminators, our method still achieves better results compared to both 3D VAE [2] and 3D-RecGAN++ [26].

Performance on smaller objects. If we look closely on Table 1, our approach has a significant improvement over 3D VAE [2] and 3D-RecGAN++ [26] on smaller objects like the class of table, furniture and objects wherein [2] produced an IoU of zero. The reason behind this improvement is because the adversarial training is especially helpful in reconstructing and completing small objects compared to 3D VAE [2]. Note that these results are also validated by evaluating the mAP in Table 2.

Since the latent space is continouos, this implies that it reserves regions for the object classes with a smaller amount of voxels in the scene or a fewer samples in the training dataset. Therefore, while all methods can reconstruct the common objects such as the ceiling, floor and walls with

	empty	ceil.	floor	wall	win.	door	chair	bed	sofa	table	furn.	objs.	*Avg.*
3D VAE [2]	49.4	33.3	25.3	32.4	16.9	9.3	5.6	19.2	14.7	1.1	0.0	0.0	31.5
3D-RecGAN++ [26]	49.6	35.1	31.8	39.2	23.7	17.9	11.5	26.1	22.6	18.1	5.1	3.0	37.7
Ours without $\mathcal{D}_l$	49.6	42.4	35.8	44.4	29.2	24.8	17.2	30.6	24.2	19.5	11.5	4.4	42.4
Ours without $\mathcal{D}_{\text{vox}}$	49.7	43.9	37.3	45.9	26.7	29.2	20.1	24.0	24.6	26.1	19.8	9.0	44.3
Ours (*Proposed*)	49.8	49.6	42.7	51.2	24.2	34.9	23.0	28.1	30.4	29.9	22.0	11.5	**51.4**

Table 3: Semantic scene completion results finetuned on the NYU training set with real world depth map for IoU in percentage.

	empty	ceil.	floor	wall	win.	door	chair	bed	sofa	table	furn.	objs.	*Avg.*
3D VAE [2]	99.8	25.0	53.8	70.9	19.3	7.4	4.2	14.3	9.4	1.1	1.2	0.9	68.4
3D-RecGAN++ [26]	99.9	27.3	67.5	87.6	27.0	15.8	8.0	19.2	12.0	2.2	3.4	1.8	86.5
Ours without $\mathcal{D}_l$	100.0	28.9	72.1	92.7	29.6	19.8	9.9	20.8	13.3	2.7	6.6	2.9	91.9
Ours without $\mathcal{D}_{\text{vox}}$	100.0	29.2	76.8	94.5	31.9	22.6	11.5	21.9	14.2	3.2	8.2	4.1	94.8
Ours (*Proposed*)	100.0	30.8	79.1	96.6	35.4	26.9	17.0	13.9	15.8	3.6	9.7	5.5	**97.2**

Table 4: Semantic scene completion results finetuned on the NYU training set with real world depth map for mAP in percentage.

correct labels as illustrated in both Table 1 and Table 2, the main advantage of our work is the capacity to reconstruct and classify every type of object labels.

Qualitative results. We illustrate the qualitative results in Fig. 5 and compare them with 3D VAE [2] and the ground truth. Based on these voxel representations, we can clearly visualize the superiority of our algorithm to reconstruct more detailed structures compared to [2]. Therefore, this confirms the advantage of our approach to reconstruct not only the larger structures but also the smaller objects in the scene.

7.2. Fine-tune with NYU

The objective of this section is to investigates whether an increase in the size of the learning dataset from a different source can improve the performance of the algorithm or confuse the learned model.

In this section, we include the NYU dataset [20] which is also an indoor scene dataset. It contains both the depth images captured by Kinect and the 3D models. This includes the volumetric 3D data with the annotated object labels for every voxels in 1,449 scenes. The semantic annotations for the volumetric data in this dataset consist of 33 objects in 7 categories. Note that, due to the limited amount of 1,449 volumetric scenes from the NYU dataset, this size is insufficient to learn a deep learning architecture. Thus, we only use the NYU to supplement our training dataset while testing on SUNCG for the 12 categories. This requires us to relabel the object classes of the volumetric data in NYU to match the labels provided by SUNCG dataset.

Comparison against other approaches. Similar to Sec. 7.1, this procedure is implemented on all the five approaches that we are comparing. While fine-tuning with the NYU dataset, our experiments show that the combination of the two datasets improve the perfomance of our algorithm. From Table 1 to Table 3 and Table 2 to Table 4, we experience an increase in IoU by 7.3% and in mAP by 2.7%. Although both the 3D VAE [2] and 3D-RecGAN++ [26] also experienced an increase in performance, the difference is not significant which counts for a maximum of 1.6% increase in IoU. Note that, in Table 3, the results on smaller objects for 3D VAE [2] remains close to zero or zero.

Comparison on the architecture for learning. When we learn our architecture with the discriminators, the effect of the improvement is negligible. Without the discriminator for the latent features, the IoU even decreased from 42.7% to 42.4%; while, without the discriminator for the reconstruction, the IoU increases only from 41.7% to 44.3%. Therefore, based on this experiment, we can attribute the significant improvement of our work's perfomance to the discriminators in the training architecture.

8. Conclusion

We have proposed a novel approach for semantic scene completion from a single depth map, which exploits the power of adversarial training to regress accurate reconstructions without the need of additional assumptions or the camera pose information. Our proposal relies on the enforcement of two adversarial losses – one aimed at mak-

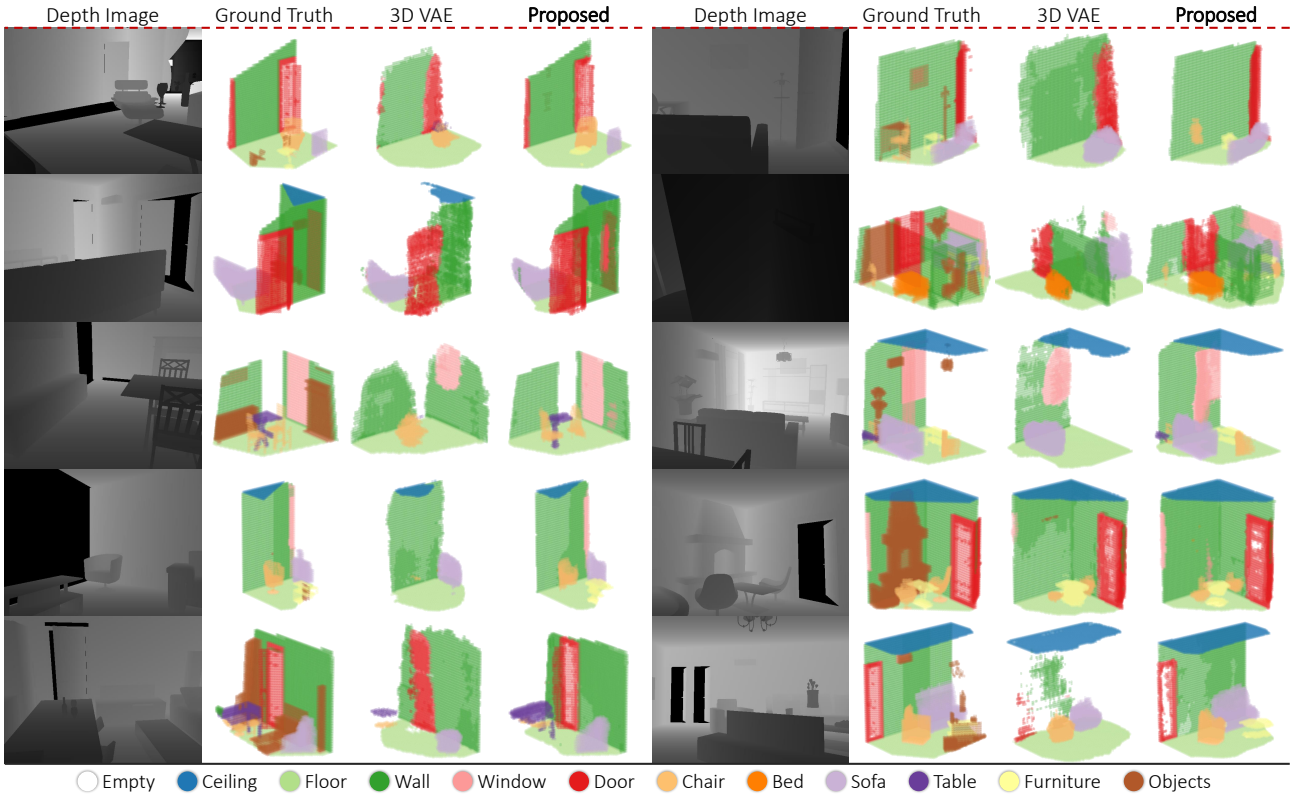


Figure 5: GAN for semantic 3D reconstruction from depth images.

ing the output realistc; while, the other aimed at imitating the embedding learned via auto-encoder from the complete volumetric data. We have demonstrated the effectiveness of our approach on a reference benchmark dataset such as SUNCG. The future work aims at modifying our architecture to overcome the memory limitation so to process higher resolution samples, this allowing a direct comparison with approaches such as SSCNet [22].

References

[1] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, (just-accepted), 2017. 2, 4

[2] A. Brock, T. Lim, J. M. Ritchie, and N. Weston. Generative and discriminative voxel modeling with convolutional neural networks. *arXiv preprint arXiv:1608.04236*, 2016. 2, 3, 6, 7

[3] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016. 2

[4] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath. Generative adversarial networks: An overview. *IEEE Signal Processing Magazine*, 35(1):53–65, 2018. 3, 5

[5] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume 1, 2017. 1

[6] H. Fan, H. Su, and L. Guibas. A point set generation network for 3d object reconstruction from a single image. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 38, 2017. 2

[7] M. Gadelha, S. Maji, and R. Wang. 3d shape induction from 2d views of multiple objects. *arXiv preprint arXiv:1612.05872*, 2016. 1

[8] A. Handa, V. Patraucean, V. Badrinarayanan, S. Stent, and R. Cipolla. Scenenet: Understanding real world indoor scenes with synthetic data. *arXiv preprint arXiv:1511.07041*, 2015. 1

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1

[10] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical report, Technical Report 07-49, University of Massachusetts, Amherst, 2007. 5

[11] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[12] D. P. Kingma, D. J. Rezende, S. Mohamed, and M. Welling. Semi-supervised learning with deep generative models. *Advances in Neural Information Processing Systems*, 4:3581–3589, 2014. 2

[13] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 2, 4

[14] J. Kober, J. A. Bagnell, and J. Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013. 1

[15] L.-J. Li, R. Socher, and L. Fei-Fei. Towards total scene understanding: Classification, annotation and segmentation in an automatic framework. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2036–2043. IEEE, 2009. 1

[16] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010. 3

[17] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*, pages 127–136. IEEE, 2011. 1

[18] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 6

[19] M. Rosca, B. Lakshminarayanan, D. Warde-Farley, and S. Mohamed. Variational approaches for auto-encoding generative adversarial networks. *arXiv preprint arXiv:1706.04987*, 2017. 4

[20] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *European Conference on Computer Vision*, pages 746–760. Springer, 2012. 1, 5, 7

[21] K. Sohn, H. Lee, and X. Yan. Learning structured output representation using deep conditional generative models. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 3483–3491. Curran Associates, Inc., 2015. 2

[22] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser. Semantic scene completion from a single depth image. *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 1, 2, 5, 6, 8

[23] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014. 1

[24] D. Van Krevelen and R. Poelman. A survey of augmented reality technologies, applications and limitations. *International journal of virtual reality*, 9(2):1, 2010. 1

[25] J. Wu, Y. Wang, T. Xue, X. Sun, W. T. Freeman, and J. B. Tenenbaum. MarrNet: 3D Shape Reconstruction via 2.5D Sketches. In *Advances In Neural Information Processing Systems*, 2017. 2

[26] B. Yang, S. Rosa, A. Markham, N. Trigoni, and H. Wen. 3d object dense reconstruction from a single depth view. *arXiv preprint arXiv:1802.00411*, 2018. 1, 2, 6, 7

[27] C. Yu and Y. Wang. 3d-scene-gan: Three-dimensional scene reconstruction with generative adversarial networks. 2018. 2

5.1.2 ForkNet: Multi-Branch Volumetric Semantic Completion From a Single Depth Image (International Conference on Computer Vision 2019)

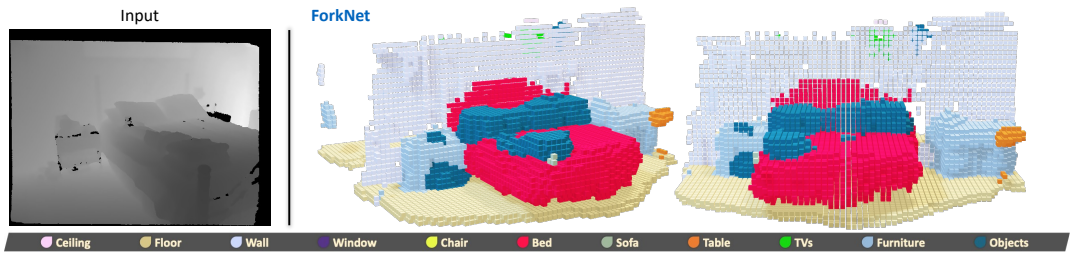


Figure 5.2 The proposed ForkNet [8] can semantically complete a real scene from a depth image captured from a single camera view. Fine structures of an object can be reconstructed with details with our model trained on ShapeNet [164] dataset.

We use back-projected SDF volumes as input for semantic completion. Our proposed ForkNet is composed of three generators together with an encoder. It is trained by a mix of supervised and unsupervised stages, which leverage the power of generative models and the annotations provided by benchmark datasets.

To overcome a common limitation of available benchmarks e.g. SUNCG [99] that provide imprecise semantic labels, we design our geometric completion to be independent of semantic completion. Existed 3D annotations for real examples are insufficient to train large-scale deep architectures, we constantly generate paired SDF volumes and semantic completion volumes to act as the training data. By training the entire encoder and decoder architecture with our proposed *SDF-Semantic consistency*, the generators are trained to produce SDF volumes and semantic scenes while being optimized to produce realistic data by the discriminator. Aiming at revealing both the small and large objects in the scene, we structured our network, which is composed of many 3D residual (Res3D) modules, where the first Res3D module activates small objects. In contrast, the larger objects are captured on the subsequent Res3D modules.

We numerically show a significant performance boost when evaluating the NYU dataset [135] Interestingly, the synthetically generated SDF volume can correctly present occluded regions due to the limited camera position.

Contributions

Yida Wang proposed and implemented the ForkNet architecture and evaluated the performance in synthetic and real datasets.

Federico Tombari proposed the idea of adapting a cycle consistency, which is initially used to optimize two different domains.

David Tan helped investigate why *SDF-semantic consistency* is beneficial for evaluation.

Nassir Navab financed on behalf of the leader of TUM CAMP.

ForkNet: Multi-Branch Volumetric Semantic Completion From a Single Depth Image

Yida Wang[1], David Joseph Tan[2], Nassir Navab[1], Federico Tombari[1,2]

[1] Technical University of Munich
[2] Google

Published version: https://doi.org/10.1109/ICCV.2019.00870

ForkNet: Multi-branch Volumetric Semantic Completion from a Single Depth Image

Yida Wang[1], David Joseph Tan[2], Nassir Navab[1], Federico Tombari[1,2]
[1]Technische Universität München [2]Google Inc.

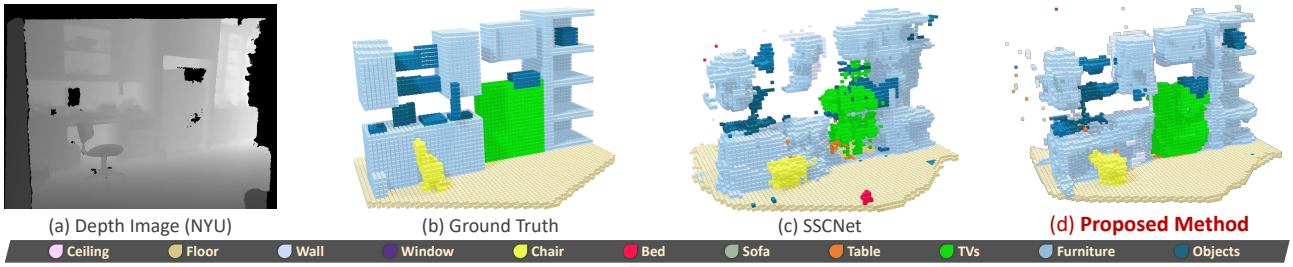


Figure 1: Our 3D semantic completion model (right-most) generates realistic yet accurate volumetric scene representations from a single depth image (left-most) affected by occlusion and noise, even if acquired from a real depth sensor.

Abstract

We propose a novel model for 3D semantic completion from a single depth image, based on a single encoder and three separate generators used to reconstruct different geometric and semantic representations of the original and completed scene, all sharing the same latent space. To transfer information between the geometric and semantic branches of the network, we introduce paths between them concatenating features at corresponding network layers. Motivated by the limited amount of training samples from real scenes, an interesting attribute of our architecture is the capacity to supplement the existing dataset by generating a new training dataset with high quality, realistic scenes that even includes occlusion and real noise. We build the new dataset by sampling the features directly from latent space which generates a pair of partial volumetric surface and completed volumetric semantic surface. Moreover, we utilize multiple discriminators to increase the accuracy and realism of the reconstructions. We demonstrate the benefits of our approach on standard benchmarks for the two most common completion tasks: semantic 3D scene completion and 3D object completion.

1. Introduction

The increasing abundance of depth data, thanks to the widespread presence of depth sensors on devices such as robots and smartphones, has recently fostered big advancements in 3D processing for augmented reality, robotics and scene understanding, unfolding new applications and technology that relies on the geometric rather than just the appearance information. Since 3D devices sense the environment from one specific viewpoint, the geometry that can be captured in one shot is only partial due to occlusion caused by foreground objects as well as self-occlusion from the same object.

As for many applications, this partial 3D information is insufficient to robustly carry-out 3D tasks such as object detection and tracking or scene understanding. A recent research direction has emerged that leverages deep learning to "complete" the depth images acquired by a 3D sensor, *i.e.* filling in the missing geometry that the sensor could not capture due to occlusion. The capability of deep learning to determine a latent space that captures the global context from the training samples proved useful in regressing completed 3D scenes and 3D shapes even when big portion of the geometry are missing [3, 4, 28, 30, 39]. Also, some of these approaches have been extended to jointly learn how to infer geometry and semantic information, in what is referred to as semantic 3D scene completion [4, 30, 34]. Nevertheless, current approaches are still limited by different factors, including the difficulty of regressing fine and sharp details of the completed geometry, as well as to generalize to shapes that significantly differ from those seen during training.

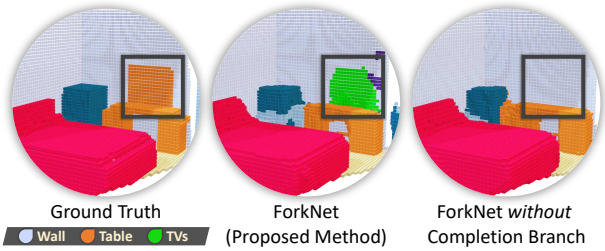


Figure 2: This figure shows the ground truth reconstruction where we notice the incorrect labels from SUNCG [30] dataset on the TVs, *i.e*. enclosed in the black box.

In this work, we aim to tackle 3D completion from a single depth image based on a novel learned model that relies on a single encoder and multiple generators, each trained to regress a different 3D representation of the input data: (*i*) a voxelized depth map, (*ii*) a geometric completed volume, (*iii*) a semantic completed volume. This particular architecture aims at two goals. The first is to supplement the lack of paired input-output data, *i.e*. a depth map and the associated completed volumetric scene, with novel pairs directly generated from the latent space, *i.e*. by means of (*i*) and (*iii*). The second goal is to overcome a common limitation of available benchmarks that provide imprecise semantic labels, by letting the geometric completion remain unaffected from it, *i.e*. by means of (*i*) and (*ii*). By means of specific connections between corresponding neural layers in the different branches, we let the semantic completion model be conditioned on geometric reconstruction information, this being beneficial to generate accurate reconstructions with aligned semantic information.

Overall, the proposed learning model uses a mix of supervised and unsupervised training stages which leverage the power of generative models in addition to the annotations provided by benchmark datasets. Additionally, we propose to further improve the effectiveness of our generative model by employing discriminators able to increase the accuracy and realism of the produced output, yielding completed scenes with high level details even in the presence of strong occlusion, as witnessed by Fig. 1 that reports an example from a real dataset (NYU [24]).

Our contributions can be summarized as follows: (*i*) a novel architecture, dubbed ForkNet, based on a single encoder and three generators built upon the same shared latent space, useful to generate additional paired training samples; (*ii*) the use of specific connections between generators to let geometric information condition and drive the completion process over the often imprecise ground truth annotations (see Fig. 2); and, (*iii*) the use of multiple discriminators to regress fine details and realistic completions. We demonstrate the benefits of our approach on standard benchmarks for the two most common completion tasks: semantic 3D scene completion and 3D object completion. For the former, we rely on SUNCG [30] (synthetic) and NYU [24] (real). For the latter, instead, we test on ShapeNet [1] and 3D-RecGAN [38]. Notably, we outperform the state of the art for both scene reconstruction and object completion on the real dataset.

2. Related work

Semantic scene completion. 3D semantic scene completion starts from a depth image or a point cloud to provide an occlusion-free 3D reconstruction of the visible scene within the viewpoint's frustrum while labeling each 3D element with a semantic class from a pre-defined category set. Scene completion could be in principle achieved by exploiting simple geometric cues such as plane consistency [23] or object symmetry [17]. Moreover, meshing approaches such as Poisson reconstruction [16] as well as purely geometric works [7] can also be employed for this goal.

Recent approaches suggested to leverage deep learning to predict how to fill-in occluded parts in a globally coherent way with respect to the training set. SSCNet [30] carries out semantic scene completion from a single depth image using dilated convolution [40] to capture 3D spatial information at multiple scales. They rely on a volumetric representation to represent both input and output data. Based on SSCNet, VVNet [12] applies view-based 3D convolutions as a replacement for SDF back-projections, this resulting more effective in extracting geometric information from the input depth image. SaTNet [21] relies on the RGB-D images. They initially predict the 2D semantic segments with the RGB. The depth image then back-projects the semantically labelled pixels to a 3D volume which goes through another architecture for 3D scene completion. ScanComplete [4] also targets semantic scene completion but, instead of starting from a single depth image, they assume to process a large-scale reconstruction of a scene acquired via a consumer depth camera. They suggest a coarse-to-fine scheme based on an auto-regressive architecture [27], where each level predicts the completion and the per-voxel semantic labeling at a different voxel resolution. The work in [34] proposes to use GANs for the task of semantic scene completion from a single depth image. In particular, it proposes to use adversarial losses applied on both the output and latent space to enforce realistic interpolation of scene parts. The work in [34] proposes to use GANs for the task of semantic scene completion from a single depth image. In particular, it proposes to use adversarial losses applied on both the output and latent space to enforce realistic interpolation of scene parts. Partially related to this field, the work in [31] leverages input object proposals in the form of 2D bounding boxes to extract the layout of a 3D scene from a single RGB image, while estimating the pose of the objects therein. A similar task is tackled by [9] starting from an

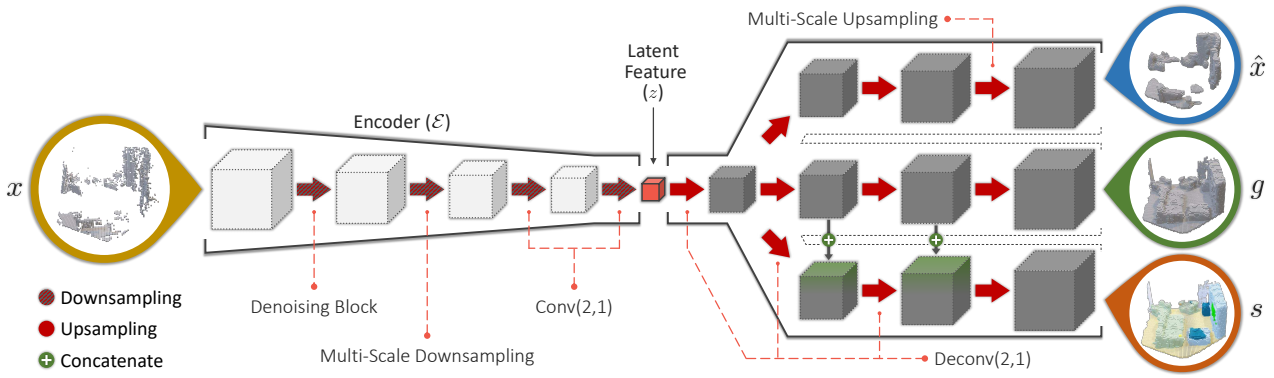


Figure 3: ForkNet – the proposed volumetric network architecture for semantic completion relies on a shared latent space encoded from SDF volume x reconstructed from the input depth image. The two decoding paths are trained to generate, respectively, incomplete surface geometry ($\hat{x}$), completed geometric volume (g) and completed semantic volumes (s).

RGB-D image.

Object completion. 3D object completion aims at obtaining a full 3D object representation from either a single depth or RGB image. While several RGB-based approaches have been recently proposed [2, 6, 35], in this section, we will focus only on those based on depth images as input since they are more related to the scope of this work. The work in [28] uses a hybrid architecture based on a CNN and an autoencoder to learn completing 3D shapes from a single depth map. 3D-RecGAN [38, 39] proposes to complete an observed object from a single depth image using a network based on skip connections [29] between the encoder and the generator so to fetch more spatial information from the input depth image to the generator. 3D-EPN [3] performs shape completion based on a latent feature concatenated with object classification information via one-hot coding, so that this additional semantic information could drive an accurate extrapolation of the missing shape parts. Han *et al*. [13] complete shapes with multiple depth images fused via LSTM Fusion [19] and process the fused data using a 3D fully convolutional approach. MarrNet [35] reconstructs the 3D shape by applying reprojection consistency between 2.5D sketch and 3D shape.

GANs for 3D shapes. Although the use of GANs for 3D semantic scene completion tasks is almost an unexplored territory, GANs have been frequently employed in recent proposals for the task of learning a latent space for 3D shapes, useful for object completion as well as for tasks such as object retrieval and object part segmentation. For instance, 3D-VAE-GAN [36] trains a volumetric GAN in an unsupervised way from a dataset of 3D models, so to be able to generate realistic 3D shapes by sampling the learned latent space. ShapeHD [37] tackles the difficult problem of reconstructing 3D shapes from a single RGB image and suggests to overcome the 2D-3D ambiguity by adversarially learning a regularizer for shapes. PrGAN [8] learns to generate 3D volumes in an unsupervised way, trained by a discriminator that distinguishes whether 2D images projected from a generated 3D volume are realistic or fake. 3D-ED-GAN [33] transforms a coarse 3D shape into a more complete one using a Long Short-term Memory (LSTM) Network by interpreting 3D volumes as sequences of 2D images.

3. Proposed semantic completion

Taking the depth image as input, we reconstruct the visible surface by back-projecting each pixel onto a voxel of the volumetric data. Denoted as x, we represent the surface reconstruction from the depth image as a signed distance function (SDF) [25] with $n_l \times n_w \times n_h$ voxels such that the value of the voxel approaches zero when it is closer to the visible surface.

Our task then is to produce the completed reconstruction of the scene with a semantic label for each voxel. Having N object categories, the class labels are assigned as $\mathcal{C} = \{c_i\}_{i=0}^{N}$ where c_0 is the empty space. Thus, denoted as s, we represent the resulting semantic volume as a one-hot encoding [22] with $N+1$ dimensional feature. Similarly, we define g as the completed reconstruction of the scene without the semantic information by setting N to 1.

3.1. Model architecture

We assemble an encoder-generator architecture [36] that builds the completed semantic volume from the partial scene derived from a single depth image. As illustrated in Fig. 3, the encoder $\mathcal{E}(\cdot)$ is composed of 3D convolutional operators where the spatial resolutions are decreased by a factor of two in each layer. In effect, this continuously reduces the volume into its simplest form, denoted by the latent feature z such that $z = \mathcal{E}(x)$.

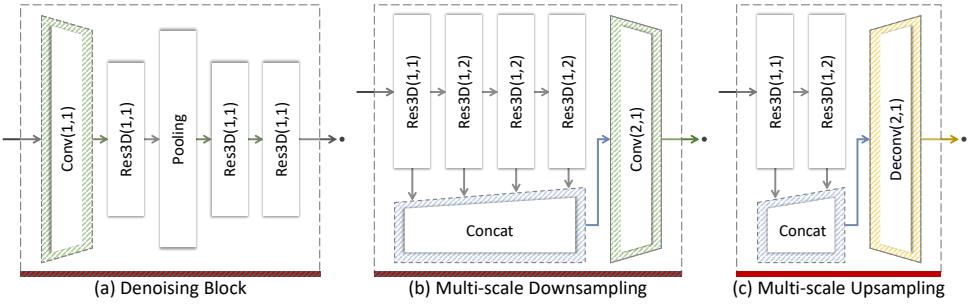


Figure 4: (a-b) Downsampling and (c) upsampling convolutional layers in our architecture (see Fig. 3). Note that the two parameters (s, d) in all the functions are the stride and dilation while the kernel size is set to 3.

In detail, the encoder is composed of four downsampling operators. The first aims at denoising [30] the SDF volumes as illustrated in Fig. 4(a). This involves a combination of a 3D convolutional operator, several 3D ResNet blocks [14], denoted as Res3D(s, d) where s is the stride while d is the dilation, and a pooling layer. The second layer aims at including different objects in the scene even with varying sizes by concatenating the output of four sequentially connected 3D ResNet blocks in Fig. 4(b). Consequently, the information from the smaller objects are captured on the first Res3D$(\cdot, \cdot)$ while the larger object are captured on the subsequent blocks. Notably, the first block is parameterized with a dilation of 1 while the other three with dilations of 2. The concatenated result is then downsampled by a 3D convolutional operator. In the final two layers, we further downsample the volume with 3D convolutional operators until we form the latent feature with a size of $16 \times 5 \times 3 \times 5$.

Branching from the same latent feature, we design three generators that reconstructs:

(i) the *SDF volume* ($\hat{x}$) which, with respect to x, formulates as an autoencoder;
(ii) the *completed volume* (g) which focuses on reconstructing the geometric structure of the scene; and,
(iii) the *completed semantic volume* (s) which is the desired outcome.

We assign these generators as the functions $\mathcal{G}_{\hat{x}}(\cdot)$, $\mathcal{G}_g(\cdot)$ and $\mathcal{G}_s(\cdot)$, respectively. Notably, we distinguish x, which is the SDF volume obtained from the input depth image, from $\hat{x}$, which is the inferred SDF volume obtained from the generator. The structure of each generator is composed of 3D deconvolutional operators that increases the spatial resolution by two in each layer.

While the first 3 convolutional upsampling layers in the generators are composed of 3D deconvolutional operators as shown in Fig. 3, the last layer is a multi-scale upsampling which is sketched in Fig. 4(c). This layer is similar to the multi-scale downsampling of the encoder where the goal is to consider the variation of sizes from different objects. In this case, we concatenate the results of two sequentially connected 3D ResNet blocks then end with a 3D deconvolution operator. With the same operations as the other generators, the generator that builds the completed semantical volume $\mathcal{G}_s$ additionally incorporates the data from the generator of the geometric scene reconstruction $\mathcal{G}_g$ as shown in Fig. 3 by concatenating the results from the second and the third layers. Since the resulting $\hat{x}$, g and s have different number of channels, only the dimension of the output from the deconvolutional operator in the last layer changes for each structure.

Giving a holistic perspective, we can simplify the sketch of the architecture in Fig. 3 to Fig. 5 by plotting the relation of the variables x, $\hat{x}$, g, s and z. When we focus on certain structures, we notice that we have an autoencoder that builds an SDF volume in Fig. 5(a), the reconstruction of the scene in Fig. 5(b) and the volumetric semantic completion in Fig. 5(c), where all of these structures branch out from the same latent feature. Later in Sec. 3.2, these plots are used to explain the loss terms in training.

The rationale of having multiple generators is twofold. First, in contrast to the typical encoder-decoder architecture, we introduce the connection that relates the two generators. Taking the output from the $\mathcal{G}_{\hat{x}}$ in each layer, we concatenate the results to the data from $\mathcal{G}_s$ as shown in Fig. 3. By establishing this relation, we incorporate the SDF reconstruction from the $\mathcal{G}_{\hat{x}}$ into the semantic completion in order to capture the geometric information of the observed scene.

Second, the latent feature can generate a pair of SDF and completed semantic volumes. Through this set of paired volumes, we can supplement the learning dataset in an unsupervised manner. This becomes a significant component in evaluating the NYU dataset [24] in Sec. 4.1 where the amount of learning dataset is limited because, since they use a consumer depth camera to capture real scenes, annotation becomes difficult. However, evaluating on this dataset is more essential compared to the synthetic dataset because it brings us a step closer to real applications. Relying on this idea in Sec. 3.2, we propose an unsupervised loss term that optimizes the entire architecture based on its own learning dataset.

Discriminators. Inspired by GANs [11, 26], we introduce the discriminator $\mathcal{D}_x$ that evaluates whether the generated SDF volumes from $\mathcal{G}_{\hat{x}}$ are realistic or not by comparing them to the learning dataset. Here, $\mathcal{D}_x$ is constructed by a

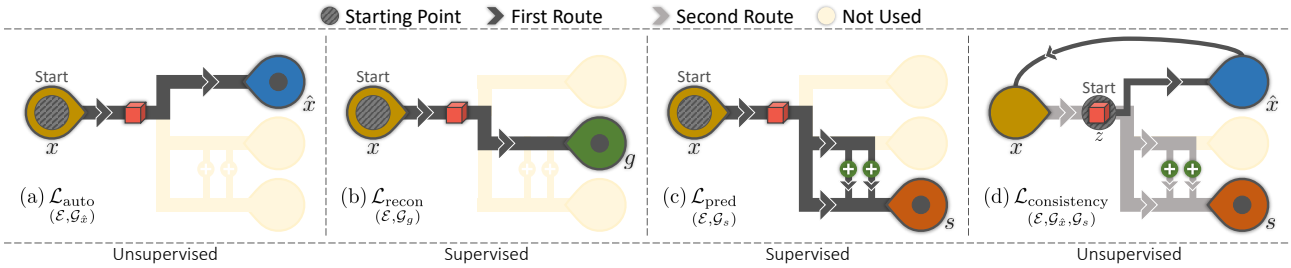


Figure 5: Graphical models of the 4 data flows (and the associated loss terms) used during training and derived from Fig. 3.

sequentially connected 3D convolutional operators with the kernel size of 3×3×3 and stride of 2. This implies that the resolution of the input volume is sequentially decreased by a factor of two after each operation. To capture the local information of the volume [5], the results from $\mathcal{D}_x$ is set to a resolution of 5×3×5.

With a similar architecture as $\mathcal{D}_x$, we also introduce a second discriminator $\mathcal{D}_s$ that evaluates the authenticity of the generated volume s. Notably, the two discriminators are evaluated in the loss terms in Sec. 3.2 to optimize the generators.

3.2. Loss terms

Leveraging on the forward passes of smaller architectures in Fig. 5, we can optimize the entire architecture by simultaneously optimizing different paths. We also optimize the architecture of the two discriminators that distinguishes whether the generated volumes are realistic or not. During training, the learning dataset is given by a set of the pairs $\{(x, s_{\text{gt}})\}$, where we distinguish s_{gt} as the ground truth from the generated s. Note that the ground truth for the geometric completion g_{gt} is the binarized summation of non-empty space in s_{gt} and an occupancy volume from the SDF surface.

SDF autoencoder. Motivated to reconstruct as similar SDF volume from the generator $\mathcal{G}_{\hat{x}}$ as the original input, we define the loss function

$$\underset{(\mathcal{E},\mathcal{G}_{\hat{x}})}{\mathcal{L}_{\text{auto}}} = \|\mathcal{G}_{\hat{x}}(\mathcal{E}(x)) - x\|^2 \tag{1}$$

for the autoencoder in Fig. 5(a) in order to minimize the difference between the observed x and the inferred $\hat{x}$.

Geometric completion. In Fig. 5(b), a conditional generative model combines the encoder $\mathcal{E}(\cdot)$ and the generator $\mathcal{G}_g(\cdot)$ in order to reconstruct the scene (*i.e.* without the semantic labels). Since the reconstruction is a two-channel volume that represents the empty and non-empty category, we use a binary cross-entropy loss

$$\underset{(\mathcal{E},\mathcal{G}_g)}{\mathcal{L}_{\text{recon}}} = \sum_{i=0}^{1}(\epsilon(\mathcal{G}_g(\mathcal{E}(x)), g_{\text{gt}})) \tag{2}$$

to train the inference network, where $\epsilon(\cdot,\cdot)$ is the per-category error

$$\epsilon(q, r) = -\lambda r \log q - (1-\lambda)(1-r)\log(1-q)\,. \tag{3}$$

In (3), λ, which ranges from 0 to 1, weighs the importance of reconstructing true positive regions in the volume. If $\lambda = 1$, the penalty for the false positive predictions will not be considered; while, if λ is set to 0, the false negatives will not be corrected.

Semantic completion. Similar to (2), in Fig. 5(c), we train a conditional generative model that is composed of the encoder $\mathcal{E}(\cdot)$ and generator $\mathcal{G}_s(\cdot)$ linking x and s. Hence, we also use a binary cross-entropy loss

$$\underset{(\mathcal{E},\mathcal{G}_s)}{\mathcal{L}_{\text{pred}}} = \sum_{i=0}^{N}(\epsilon(\mathcal{G}_s(\mathcal{E}(x)), s_{\text{gt}})) \tag{4}$$

where N is the number of categories in the semantic scene.

Discriminators on the architecture. In relation to the architecture, we use two discriminators to optimize the generators [36] through

$$\begin{aligned}\underset{\mathcal{G}_{\hat{x}}}{\mathcal{L}_{\text{gen-}\hat{x}}} &= -\log\left(\mathcal{D}_x(\mathcal{G}_{\hat{x}}(z))\right)\\ \underset{\mathcal{G}_s}{\mathcal{L}_{\text{gen-}s}} &= -\log\left(\mathcal{D}_s(\mathcal{G}_s(z))\right)\,.\end{aligned} \tag{5}$$

In this manner, we optimize the two generative models including both the SDF encoder and the semantic scene generator by randomly sampling the latent features. On the other hand, when we update the parameters of both discriminators, we optimize the loss functions

$$\begin{aligned}\underset{(\mathcal{D}_x)}{\mathcal{L}_{\text{dis-}x}} &= -\log(\mathcal{D}_x(x)) - \log\left(1 - \mathcal{D}_x(\mathcal{G}_{\hat{x}}(z))\right)\\ \underset{(\mathcal{D}_s)}{\mathcal{L}_{\text{dis-}s}} &= -\log(\mathcal{D}_s(s_{\text{gt}})) - \log\left(1 - \mathcal{D}_s(\mathcal{G}_s(z))\right)\,.\end{aligned} \tag{6}$$

During training, we apply the set of equations in (5) and (6) alternatingly to optimize the generators and the discriminators separately. Note that we use the KL-divergence from the variational inference [10, 15] to penalize the deviation

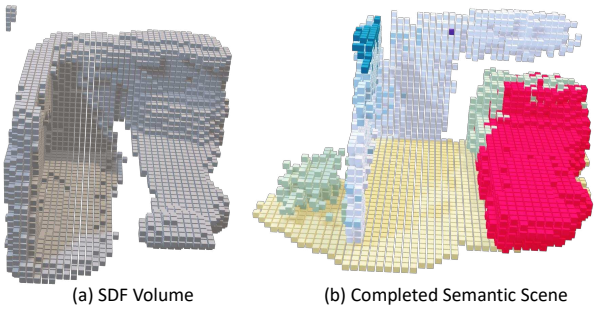


Figure 6: An example of the generated SDF volume and the corresponding completed semantic scene parameterized from the latent feature, which are used to supplement the existing learning dataset.

between the distribution of $\mathcal{E}(x)$ and a normal distribution with zero mean and identity variance matrix. The advantage of such is the capacity to easily sample from the latent space in the generative model, which becomes helpful in the succeeding loss term.

SDF-Semantic consistency. Since the generators are trained to produce SDF volumes and semantic scenes while being optimized to produce realistic data by the discriminator, we can build a new set of paired volumes to act as the learning dataset in order to supplement the existing one. Thus, we propose to generate paired volumes directly from the latent feature in order to optimize the architecture in an unsupervised learning.

Exploiting the latent space, we reconstruct the set of pairs $\{(\mathcal{G}_{\hat{x}}(z), \mathcal{G}_s(z))\}$, where z is randomly sampled from a Gaussian distribution centered on the average of latent features of a batch of samples. Following the inference model in Fig. 5(c), we formulate a similar loss function as (4) but with the newly acquired data such that

$$\mathcal{L}_{\substack{\text{consistency} \\ (\mathcal{E},\mathcal{G}_{\hat{x}},\mathcal{G}_s)}} = \sum_{i=0}^{N} (\epsilon(\mathcal{G}_s(\mathcal{E}(\mathcal{G}_{\hat{x}}(z))), \mathcal{G}_s(z))) \; . \tag{7}$$

By drawing the data flow of the first term $\mathcal{G}_s(\mathcal{E}(\mathcal{G}_{\hat{x}}(z)))$ in Fig. 5(d), we observe that the loss term in (7) optimizes the entire architecture.

Interestingly, when we take a closer look at the newly generated pairs $\{(\mathcal{G}_{\hat{x}}(z), \mathcal{G}_s(z))\}$ in Fig. 6, we can easily notice the realistic results. The SDF volume in Fig. 6(a) considers missing regions due to the camera position while the semantic scene in Fig. 6(b) generates lifelike structures and reasonable positions of the objects in the scene (*e.g.* the bed in red). By adding the newly generated pairs, we numerically show in Sec. 4.1 that there is a significant boost in performance when evaluating the NYU dataset [24] where the size of the learning dataset is small.

Optimization. With all the loss terms given, achieving the optimum parameters in our architecture requires us to simultaneously minimize them. We start by optimizing (1), (2), (4) and (5) altogether. Then, the loss functions in (6) for the two discriminators are optimized alternatively (*i.e.* batch-by-batch) with (1), (2), (4) and (5). In practice, we employ the Adam optimizer [18] with a learning rate of 0.0001. For the data flows, Fig. 5(a) and (d) are both unsupervised while Fig. 5(b) and (c) are supervised. In addition, for the discriminators, (5) is unsupervised while (6) is supervised.

4. Experiments

There are two tasks at hand – (1) 3D semantic scene completion; and, (2) 3D object completion. Although they perform similar tasks in reconstructing from a single view, the former completes the structure of a scene with semantic labels while the latter requires a more detailed completion with the assumption of a single category.

Metric. For each of the N classes, the accuracy of the predicted volumes is measured based on the Intersection over Union (IoU). Analogously to the evaluation carried out by other methods, the average IoU is taken from all the categories except for the empty space.

Implementation details. We learn our model with an Nvidia Titan Xp with a batch size of 8. We applied batch normalization after every convolutional and deconvolutional operations except for the convolutional operations in the last deconvolutional layers in 3 generators. Leaky ReLU with a negative slope of 0.2 is applied on the output of each convolutional layer in the Res3D$(\cdot,\cdot)$ modules in Fig. 4. In addition, ReLU is applied on the output of deconvolutional operations in the generators except for the last deconvolution operation in the Multi-Scale Upsampling. Finally, the sigmoid operation is applied to the last deconvolution layer of the generators for the geometric and semantic completion. Notably, the factor λ from (3) is set to be 0.5 for the geometric completion in (2). For the semantic completion, it is initially set to 0.9 in (4). However, when the network is capable of revealing objects from the depth image, more and more false positive predictions in the empty space appears. Due to this, we set λ to 0.6 after five epochs.

4.1. Semantic scene completion

The SUNCG [30] and NYU [24] datasets are currently the most relevant benchmarks for semantic scene completion, and include a paired depth image and the corresponding semantically labeled volume. While SUNCG comprises synthetically rendered depth data, NYU includes real scenes acquired with a Kinect depth sensor. This makes the evaluation of NYU more challenging, due to the presence of real nuisances, as well as due to a limited training set of

	ceil.	floor	wall	win.	chair	bed	sofa	table	tvs	furn.	objs.	*Avg.*
SSCNet [30] (observed)	97.7	94.5	66.4	30.0	**36.9**	60.2	62.5	**56.3**	12.1	46.7	**33.0**	54.2
Proposed Method (observed)	**98.2**	**96.9**	**67.8**	**37.4**	35.9	**72.9**	**69.6**	48.8	**20.5**	**48.4**	32.4	**57.2**
Wang *et al*. [34]	41.4	37.7	45.8	26.5	26.4	21.8	25.4	23.7	20.1	16.2	5.7	26.4
3D-RecGAN [38]	79.9	75.2	48.2	28.9	20.2	64.4	54.6	25.7	17.4	33.7	24.4	43.0
SSCNet [30]	96.3	84.9	56.8	28.2	21.3	56.0	52.7	33.7	10.9	44.3	25.4	46.4
VVNet [12]	**98.4**	**87.0**	61.0	54.8	**49.3**	**83.0**	75.5	**55.1**	43.5	**68.8**	**57.7**	**66.7**
SaTNet [21]	97.9	82.5	57.7	**58.5**	45.1	78.4	72.3	47.3	**45.7**	67.1	55.2	64.3
Proposed Method	95.0	85.9	**73.2**	54.5	46.0	81.3	74.2	42.8	31.9	63.1	49.3	63.4
– *without completion branch*	94.1	83.5	68.2	49.6	43.1	80.5	**77.7**	41.8	33.8	61.7	51.7	62.3
– *without scene consistency*	89.6	79.5	63.4	46.3	39.0	77.5	73.2	37.7	29.8	57.4	46.7	58.2

Table 1: Semantic scene completion results on the SUNCG test set with depth map for IoU (in %).

	ceil.	floor	wall	win.	chair	bed	sofa	table	tvs	furn.	objs.	*Avg.*
SSCNet [30] (observed)	37.7	**91.9**	**75.4**	64.0	29.0	51.1	63.3	43.7	**29.7**	73.3	54.5	50.8
Proposed Method (observed)	**41.5**	90.8	69.6	**54.8**	**27.7**	**53.1**	**66.3**	**44.4**	27.1	**74.7**	**57.5**	**55.2**
Lin *et al*. [20] (NYU only)	0.0	11.7	13.3	14.1	9.4	29.0	24.0	6.0	7.0	16.2	1.1	12.0
3D-RecGAN [38]	35.3	70.3	24.1	3.8	11.9	47.4	43.1	11.4	16.9	30.6	7.2	27.5
Geiger and Wang [9] (NYU only)	10.2	62.5	19.1	5.8	8.5	40.6	27.7	7.0	6.0	22.6	5.9	19.6
SSCNet [30]	15.1	94.6	24.7	10.8	17.3	53.2	45.9	15.9	13.9	31.1	12.6	30.5
VVNet [12]	19.3	**94.8**	28.0	12.2	**19.6**	57.0	50.5	17.6	11.9	35.6	15.3	32.9
SaTNet [21]	17.3	92.1	28.0	16.6	19.3	57.5	**53.8**	17.7	18.5	38.4	18.9	34.4
Proposed Method	36.2	93.8	**29.2**	18.9	17.7	**61.6**	52.9	**23.3**	**19.5**	45.4	**20.0**	**37.1**
– *without completion branch*	35.8	94.1	28.9	**19.2**	16.8	61.4	53.5	23.0	14.0	**45.6**	18.9	36.5
– *without scene consistency*	**36.8**	91.7	28.0	18.3	8.3	58.8	49.5	13.0	16.7	42.6	17.6	34.7

Table 2: Semantic scene completion results on the NYU test set with depth map for IoU (in %).

	SUNCG	NYU
Lin *et al*. [20]	–	36.4
3D-RecGAN [38]	72.1	51.3
Geiger and Wang [9]	–	44.4
SSCNet [30]	73.5	56.6
VVNet [12]	84.0	61.1
SaTNet [21]	78.5	60.6
Proposed Method	**86.9**	**63.4**
– *without completion branch*	82.3	62.6
– *without scene consistency*	82.0	61.1

Table 3: Scene completion results on the SUNCG and the NYU test set in terms of IoU (in %).

less than 1000 samples. We compare our method against Wang *et al*. [34], Lin *et al*. [20], 3D-RecGAN [38], Geiger and Wang [9], SSCNet [30], VVNet [12], and SaTNet [21]. The resolution of our input volume is given in the scale of 80×48×80 voxels. While [9, 12, 20, 21, 30] produce 60×36×60 semantic volumes for evaluation, [34, 38] and us produce a slightly higher resolution of 80×48×80.

Following SUNCG [30], the semantic categories include 12 classes of varying shapes and sizes, *i.e.*: *empty space*, *ceiling*, *floor*, *wall*, *window*, *chair*, *bed*, *sofa*, *table*, *tvs*, *furniture* and *other objects*. We follow two types of evaluation as introduced by [30]. One evaluates the semantic segmentation accuracy on the observed surface reconstruction, while the other considers the semantic segmentation of the predicted full volumetric reconstruction.

SUNCG dataset. Based on an online interior design platform, the evaluation of SUNCG contains more than 130,000 paired depth images and voxel-wise semantic labels taken from 45,622 houses with realistic rooms and furniture layouts [30]. Focusing on the semantic segmentation on the observed surface, our approach performs at an IoU of 57.2% which is 3.0% higher than SSCNet [30]. On the other hand, when we evaluate the IoU measure on the entire volume in Table 1, our method reaches an average IoU of 63.4% which is significantly better than Wang *et al*. [34], 3D-RecGAN [38] and SSCNet [30] but slightly worse than VVNet [12] and SaTNet [21].

NYU dataset (real). The NYU dataset [24] is composed of 1,449 indoor depth images captured with a Kinect depth sensor. Like SUNCG, each image is also annotated with

	bench	chair	couch	table	*Avg.*
Varley *et al.* [32]	65.3	61.9	81.8	67.8	69.2
3D-EPN [3]	75.8	73.9	83.4	77.2	77.6
Han *et al.* [13]	54.4	46.9	48.3	56.0	51.4
3D-RecAE [38]	73.3	73.6	83.2	75.0	76.3
3D-RecGAN [38]	74.5	74.1	84.4	77.0	77.5
Proposed Method	**79.1**	**80.6**	**92.4**	**84.0**	**84.1**
– without scene consistency	76.3	76.4	87.5	81.2	80.4

Table 4: Object completion results on the ShapeNet test set in terms of IoU (in %). The resolution for Varley *et al.* [32] and 3D-EPN [3]: 32×32×32, for others: 64×64×64.

3D semantic labels. Due to its size, training our network on this dataset alone is insufficient. As a solution already used in [30], we take the network trained on the SUNCG then refine it by supplementing the training data from NYU with 1,500 randomly selected samples from SUNCG in each epoch of training.

Although we achieved slightly worse results than VVNet [12] and SaTNet [21] on the synthetic dataset, we performed better than the state of the art on the real images, reaching an IoU measure of 37.1% as shown in Table 2. Consequently, we attain a 4.2% improvement compared to VVNet [12] and 2.7% to SaTNet [21].

Looking at the other approaches, we achieve even more significant improvements with at least 6.6% increase in IoU. For the evaluation on the semantic labels on the observed surface, we gained 4.4% increase in IoU against SSCNet. Notably, our approach outperforms other works not only on the average IoU but also on individual object categories. In addition, we also achieve similar improvements in the scene completion task in Table 3 with approximately 2.8% better in IoU compared to SaTNet [21].

Moreover, while the re-implementation SSCNet [30] in our experiments does not fit into any of our contributions, we used it in order to qualitatively compare our results with them (see Fig. 1).

Ablation study for loss terms. In Tables 1 and 2, we investigate the contribution of $\mathcal{L}_{\text{recon}}$ from the supervised learning and $\mathcal{L}_{\text{consistency}}$ from the unsupervised learning. Our ablation study indicates that $\mathcal{L}_{\text{consistency}}$ prompts the highest boost in IoU with 5.2% in Table 1. When using the $\mathcal{L}_{\text{recon}}$ in the geometric completion, it improves by 1.1% on the SUNCG dataset. A similar conclusion for the loss terms is presented in Table 1 for NYU.

4.2. 3D object completion

Adapting the assessment data and strategy from 3D-RecGAN [38], we use ShapeNet [1] to generate the training and test data for 3D object completion, wherein each reconstructed object surface x is paired with a corresponding ground truth voxelized shape with a size of 64×64×64. The dataset comprises four object classes: *bench*, *chair*, *couch* and *table*. [38] prepared an evaluation for both synthetic and real input data. Notably, for both synthetic and real test data, we can express the same conclusions as the ablation studies in Sec. 4.1 (see Tables 4 and 5).

	bench	chair	couch	table	*Avg.*
Han *et al.* [13]	18.4	14.8	10.1	12.6	14.0
3D-RecAE [38]	23.1	17.8	10.7	14.8	16.6
3D-RecGAN [38]	23.0	17.4	10.9	14.6	16.5
Proposed Method	**32.7**	**24.1**	**15.9**	**22.5**	**23.8**
– without scene consistency	26.1	21.5	14.9	18.6	20.3

Table 5: Object completion results on the real-world test set provided by 3D-RecGAN [38] in terms of IoU (in %). The resolution for all methods is 64×64×64.

Synthetic test data. We perform two evaluations in Table 4. The first is a single category test [38] such that each category is trained and tested separately while the second considers the categories in order to label the voxels. We compare our results against [3, 13, 32, 38].

In the single category test, we achieve the best results with 84.1%. This result is 6.5% higher than 3D-EPN [3], 6.6% higher than 3D-RecGAN [38], 7.8% higher than 3D-RecAE [38], 32.7% higher than Han *et al.* [13] and 14.9% higher than Varley *et al.* [32]. Moreover, this table also shows the we achieve the best results across all categories.

Real test data. Using the single category test in Table 5, we also evaluate the 3D object completion task on the real world test data provided by [38]. In this evaluation, we generate the state-of-the art results with 23.8% IoU measure, which is higher than 3D-RecAE [38] by 7.2%, 3D-RecGAN [38] by 7.3% and Han *et al.* [13] by 9.8%.

5. Conclusion

We propose ForkNet, a novel architecture for volumetric semantic 3D completion that leverages a shared embedding encoding both geometric and semantic surface cues, as well as multiple generators designed to deal with limited paired data and imprecise semantic annotations. Experimental results numerically demonstrate the benefits of our approach for the two tasks of scene and object completion, as well as the effectiveness of the proposed contributions in terms of architecture, loss terms and use of discriminators. However, since we compress the input SDF volume into a lower resolution through the encoder then increase the resolution through the generator, small or thin structures such as the legs of the chair or TVs tend to disappear during compression. This is an aspect we plan to improve in the future work. In addition, for 3D scene understanding, the volumetric representations are typically memory and power-hungry, we also plan to extend our model for completion of efficient and sparse representations such as point clouds.

References

[1] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 2, 8

[2] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *European conference on computer vision (ECCV)*. Springer, 2016. 3

[3] Angela Dai, Charles Ruizhongtai Qi, and Matthias Nießner. Shape completion using 3d-encoder-predictor cnns and shape synthesis. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume 3, 2017. 1, 3, 8

[4] Angela Dai, Daniel Ritchie, Martin Bokeloh, Scott Reed, Jürgen Sturm, and Matthias Nießner. Scancomplete: Large-scale scene completion and semantic segmentation for 3d scans. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume 1, 2018. 1, 2

[5] Ugur Demir and Gözde B. Ünal. Patch-based image inpainting with generative adversarial networks. *CoRR*, abs/1803.07422, 2018. 5

[6] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume 2, 2017. 3

[7] Michael Firman, Oisin Mac Aodha, Simon Julier, and Gabriel J Brostow. Structured prediction of unobserved voxels from a single depth image. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[8] Matheus Gadelha, Subhransu Maji, and Rui Wang. 3d shape induction from 2d views of multiple objects. In *International Conference on 3D Vision (3DV)*. IEEE, 2017. 3

[9] Andreas Geiger and Chaohui Wang. Joint 3d object and layout inference from a single rgb-d image. In *German Conference on Pattern Recognition (GCPR)*. Springer, 2015. 2, 7

[10] Zoubin Ghahramani and Matthew J Beal. Variational inference for bayesian mixtures of factor analysers. In *Advances in Neural Information Processing Systems (NIPS)*, 2000. 5

[11] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)*, 2014. 4

[12] Yuxiao Guo and Xin Tong. View-volume network for semantic scene completion from a single depth image. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. AAAI Press, 2018. 2, 7, 8

[13] Xiaoguang Han, Zhen Li, Haibin Huang, Evangelos Kalogerakis, and Yizhou Yu. High-resolution shape completion using deep neural networks for global structure and local geometry inference. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2017. 3, 8

[14] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 6546–6555, 2018. 4

[15] Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013. 5

[16] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics (ToG)*, 32(3):29, 2013. 2

[17] Young Min Kim, Niloy J Mitra, Dong-Ming Yan, and Leonidas Guibas. Acquiring 3d indoor environments with variability and repetition. *ACM Transactions on Graphics (TOG)*, 31(6):138, 2012. 2

[18] Diederick P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015. 6

[19] Zhen Li, Yukang Gan, Xiaodan Liang, Yizhou Yu, Hui Cheng, and Liang Lin. Lstm-cf: Unifying context modeling and fusion with lstms for rgb-d scene labeling. In *European Conference on Computer Vision (ECCV)*. Springer, 2016. 3

[20] Dahua Lin, Sanja Fidler, and Raquel Urtasun. Holistic scene understanding for 3d object detection with rgbd cameras. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2013. 7

[21] Shice Liu, YU HU, Yiming Zeng, Qiankun Tang, Beibei Jin, Yinhe Han, and Xiaowei Li. See and think: Disentangling semantic scene completion. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 263–274. Curran Associates, Inc., 2018. 2, 7, 8

[22] Pauline Luc, Camille Couprie, Soumith Chintala, and Jakob Verbeek. Semantic segmentation using adversarial networks. *CoRR*, abs/1611.08408, 2016. 3

[23] Aron Monszpart, Nicolas Mellado, Gabriel J Brostow, and Niloy J Mitra. Rapter: rebuilding man-made scenes with regular arrangements of planes. *ACM Trans. Graph.*, 34(4):103–1, 2015. 2

[24] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *European Conference on Computer Vision (ECCV)*, 2012. 2, 4, 6, 7

[25] Stanley Osher and Ronald Fedkiw. *Level set methods and dynamic implicit surfaces*, volume 153. Springer Science & Business Media, 2006. 3

[26] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015. 4

[27] Scott Reed, Aäron van den Oord, Nal Kalchbrenner, Sergio Gómez Colmenarejo, Ziyu Wang, Yutian Chen, Dan Belov, and Nando de Freitas. Parallel multiscale autoregressive density estimation. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*. JMLR. org, 2017. 2

[28] Dario Rethage, Federico Tombari, Felix Achilles, and Nassir Navab. Deep learned full-3d object completion from single view. *CVPR '16 Scene Understanding Workshop (SUNw)*, 2016. 1, 3

[29] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmen-

tation. In *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015. 3

[30] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017. 1, 2, 4, 6, 7, 8

[31] Shubham Tulsiani, Saurabh Gupta, David Fouhey, Alexei A Efros, and Jitendra Malik. Factoring shape, pose, and layout from the 2d image of a 3d scene. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[32] Jacob Varley, Chad DeChant, Adam Richardson, Joaquín Ruales, and Peter Allen. Shape completion enabled robotic grasping. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017. 8

[33] W. Wang, Q. Huang, S. You, C. Yang, and U. Neumann. Shape inpainting using 3d generative adversarial network and recurrent convolutional networks. In *IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 3

[34] Y. Wang, D. J. Tan, N. Navab, and F. Tombari. Adversarial semantic scene completion from a single depth image. In *2018 International Conference on 3D Vision (3DV)*, 2018. 1, 2, 7

[35] Jiajun Wu, Yifan Wang, Tianfan Xue, Xingyuan Sun, Bill Freeman, and Josh Tenenbaum. Marrnet: 3d shape reconstruction via 2.5 d sketches. In *Advances in neural information processing systems (NIPS)*, 2017. 3

[36] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Advances in Neural Information Processing Systems (NIPS)*, 2016. 3, 5

[37] Jiajun Wu, Chengkai Zhang, Xiuming Zhang, Zhoutong Zhang, William T Freeman, and Joshua B Tenenbaum. Learning 3D Shape Priors for Shape Completion and Reconstruction. In *European Conference on Computer Vision (ECCV)*, 2018. 3

[38] Bo Yang, Stefano Rosa, Andrew Markham, Niki Trigoni, and Hongkai Wen. Dense 3d object reconstruction from a single depth view. *IEEE transactions on pattern analysis and machine intelligence*, 2018. 2, 3, 7, 8

[39] Bo Yang, Hongkai Wen, Sen Wang, Ronald Clark, Andrew Markham, and Niki Trigoni. 3d object reconstruction from a single depth view with adversarial learning. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017. 1, 3

[40] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015. 2

ForkNet: Multi-branch Volumetric Semantic Completion from a Single Depth Image (*Supplementary Materials*)

Yida Wang[1], David Joseph Tan[2], Nassir Navab[1], Federico Tombari[1,2]
[1]Technische Universität München [2]Google Inc.

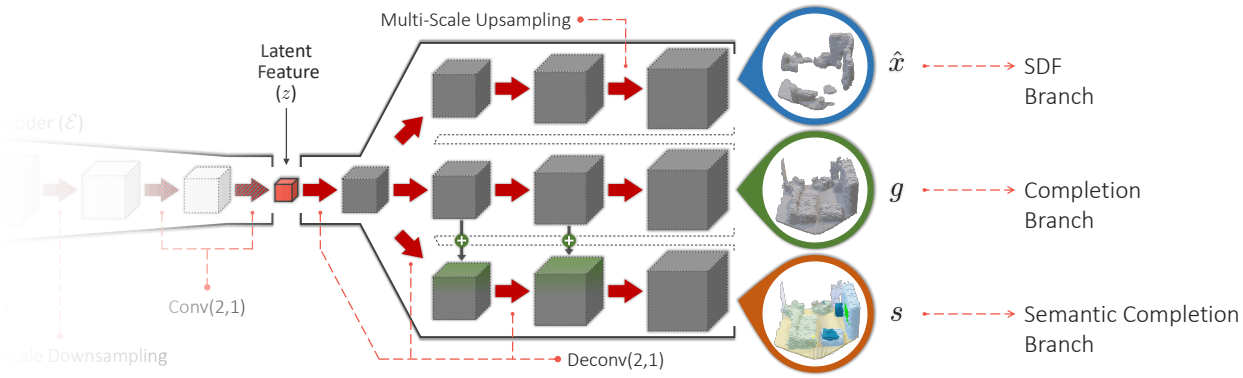


Figure 1: Illustration of the branches in the ForkNet architecture.

1. Ablation study

Based on the architecture from the main submission, the goal is to evaluate the influence of each component proposed in this paper. This section highlights the ablation study in reference to the two main contributions of the paper. In Sec. 1.1, we focus on different cost functions in relation to the branches and discriminators; while, in Sec. 1.2, we investigate the concatenated layers in detail and focus on the advantage of each.

1.1. Branches and discriminators

Investigating the branches in our architecture through Fig. 1, we evaluate on the following scenarios:

(a) *Proposed method* – With all the branches in Fig. 1, this case utilizes all the cost functions and connections discussed in the main submission during learning.

(b) *Semantic completion branch only* – This follows the basic architecture of taking the SDF volume as input and building the completed semantic volume as output. Hence, this utilizes the semantic completion branch in Fig. 1. Removing the other branches as well as the concatenation of the layers from the completion branch, learning in this case only relies solely on $\mathcal{L}_{\text{pred}}$ and $\mathcal{D}_s$.

(c) *Without completion branch* – Starting from the proposed architecture, this case removes the completion branch in Fig. 1, effectively discarding the concatenated layers and the cost function $\mathcal{L}_{\text{recon}}$.

(d) *Without scene consistency* – Aiming at deactivating the cost function $\mathcal{L}_{\text{consistency}}$, this scenario inherently simplifies the entire architecture by removing the SDF branch since, without $\mathcal{L}_{\text{consistency}}$, this branch no longer influences the main semantic completion task. Therefore, this case includes both the completion and semantic completion braches while removing $\mathcal{L}_{\text{auto}}$ and $\mathcal{L}_{\text{consistency}}$ as well as the discriminator $\mathcal{D}_x$.

Notably, these cases are evaluated in Tables 1, 2, 3 and 4. Compared to the tables in the main submission, we completed the entries by adding the evaluation on utilizing the semantic completion architecture only from (b).

Based on the five scenarios, the influence of the main contributions are evaluated on (c) and (d).

3D scene completion. Although one can argue to simplify the architecture by using the semantic completion

	ceil.	floor	wall	win.	chair	bed	sofa	table	tvs	furn.	objs.	*Avg.*
SSCNet [7] (observed)	97.7	94.5	66.4	30.0	**36.9**	60.2	62.5	**56.3**	12.1	46.7	**33.0**	54.2
Proposed Method (observed)	**98.2**	**96.9**	**67.8**	**37.4**	35.9	**72.9**	**69.6**	48.8	**20.5**	**48.4**	32.4	**57.2**
Wang *et al*. [9]	41.4	37.7	45.8	26.5	26.4	21.8	25.4	23.7	20.1	16.2	5.7	26.4
3D-RecGAN [10]	79.9	75.2	48.2	28.9	20.2	64.4	54.6	25.7	17.4	33.7	24.4	43.0
SSCNet [7]	96.3	84.9	56.8	28.2	21.3	56.0	52.7	33.7	10.9	44.3	25.4	46.4
VVNet [3]	**98.4**	**87.0**	61.0	54.8	**49.3**	**83.0**	75.5	**55.1**	43.5	**68.8**	**57.7**	**66.7**
SaTNet [6]	97.9	82.5	57.7	**58.5**	45.1	78.4	72.3	47.3	**45.7**	67.1	55.2	64.3
Proposed Method	95.0	85.9	**73.2**	54.5	46.0	81.3	74.2	42.8	31.9	63.1	49.3	63.4
– *without completion*	94.1	83.5	68.2	49.6	43.1	80.5	**77.7**	41.8	33.8	61.7	51.7	62.3
– *without scene consistency*	89.6	79.5	63.4	46.3	39.0	77.5	73.2	37.7	29.8	57.4	46.7	58.2
– *without discriminators*	88.4	79.1	63.5	46.7	39.3	77.3	73.6	37.2	30.9	57.3	45.7	58.1

Table 1: Semantic scene completion results on the SUNCG test set with depth map for IoU (in %).

	ceil.	floor	wall	win.	chair	bed	sofa	table	tvs	furn.	objs.	*Avg.*
SSCNet [7] (observed)	37.7	**91.9**	**75.4**	64.0	29.0	51.1	63.3	43.7	**29.7**	73.3	54.5	50.8
Proposed Method (observed)	**41.5**	90.8	69.6	**54.8**	**27.7**	**53.1**	**66.3**	**44.4**	27.1	**74.7**	**57.5**	**55.2**
Lin *et al*. [5] (NYU only)	0.0	11.7	13.3	14.1	9.4	29	24	6.0	7.0	16.2	1.1	12.0
3D-RecGAN [10]	35.3	70.3	24.1	3.8	11.9	47.4	43.1	11.4	16.9	30.6	7.2	27.5
Geiger and Wang [2] (NYU only)	10.2	62.5	19.1	5.8	8.5	40.6	27.7	7.0	6.0	22.6	5.9	19.6
SSCNet [7]	15.1	94.6	24.7	10.8	17.3	53.2	45.9	15.9	13.9	31.1	12.6	30.5
VVNet [3]	19.3	**94.8**	28.0	12.2	**19.6**	57.0	50.5	17.6	11.9	35.6	15.3	32.9
SaTNet [6]	17.3	92.1	28.0	16.6	19.3	57.5	**53.8**	17.7	18.5	38.4	18.9	34.4
Proposed Method	36.2	93.8	**29.2**	18.9	17.7	**61.6**	52.9	**23.3**	19.5	45.4	**20.0**	**37.1**
– *semantic completion branch only*	28.3	81.6	24.0	2.2	12.9	49.9	44.9	13.5	**19.9**	30.9	7.4	28.7
– *without completion*	35.8	94.1	28.9	19.2	16.8	61.4	53.5	23.0	14.0	**45.6**	18.9	36.5
– *without scene consistency*	**36.8**	91.7	28.0	18.3	8.3	58.8	49.5	13.0	16.7	42.6	17.6	34.7
– *without discriminators*	35.1	91.7	28.1	**19.3**	8.7	58.8	49.5	13.1	15.6	42.5	16.7	34.5

Table 2: Semantic scene completion results on the NYU test set with depth map for IoU (in %).

branch alone, we illustrate in Tables 1 and 2 that this situation is insufficient to perform the task. Taking the IoU of this situation as baseline, we demonstrate in these tables that each component – the completion branch, the scene consistency and the discriminators – contributes to reach 63.4% for SUNCG and 37.1% for NYU, which is the state of the art.

While the completion branch aims at incorporating the geometric structures on the semantic completion branch while the scene consistency aims at building a new learning dataset to help the limited amounts especially in the NYU, the goal of the discriminator is less obvious.

By disentangling the discriminators from the scene consistency, we notice that we achieve lower IoU without the discriminators (*i.e*. with scene consistency) than without the scene consistency (*i.e*. with discriminators). This observation emphasizes the value of the discriminators while performing the scene consistency since the objective of the scene consistency is to supplement the learning dataset with newly generated volumes while the objective of the discriminator is to enforce that the generated volumes are realistic.

3D object completion. In contrast to the 3D scene completion task, the goal of the 3D object completion is to construct the volume of the object from a single view but with a known object category. Thus, while the semantic completion branch with one object category produces the same

	bench	chair	couch	table	*Avg.*
Varley *et al*. [8]	65.3	61.9	81.8	67.8	69.2
3D-EPN [1]	75.8	73.9	83.4	77.2	77.6
Han *et al*. [4]	54.4	46.9	48.3	56.0	51.4
3D-RecAE [10]	73.3	73.6	83.2	75.0	76.3
3D-RecGAN [10]	74.5	74.1	84.4	77.0	77.5
Proposed method	**79.1**	**80.6**	**92.4**	**84.0**	**84.1**
– *without scene consistency*	76.3	76.4	87.5	81.2	80.4
– *without discriminators*	69.9	75.8	87.1	81.3	78.5

Table 3: Object completion results on the Shapenet test set in terms of IoU (in %). The resolution for Varley *et al*. [8] and 3D-EPN [1]: 32×32×32, for others: 64×64×64.

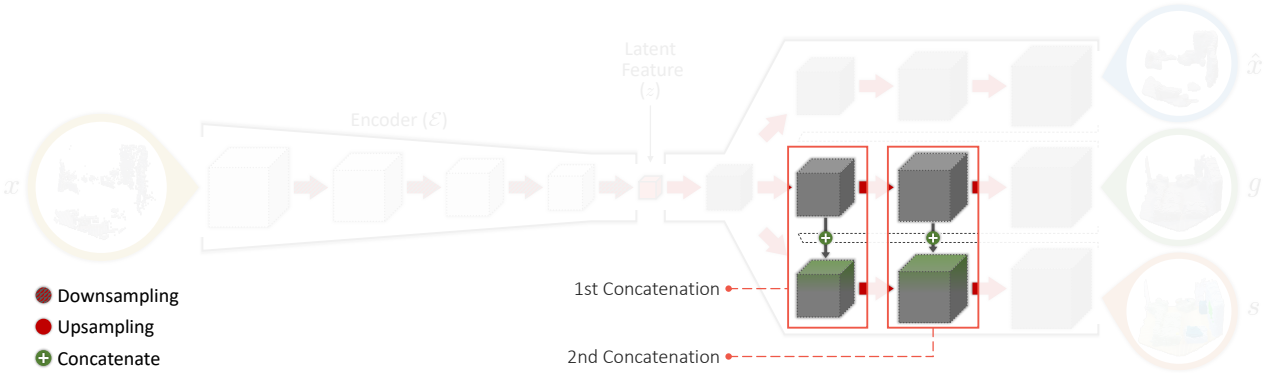


Figure 2: Illustration of the first and second concatenation in the ForkNet architecture.

results as the completion branch, we merge them together; thus, removing the concatenation. As a result, we only have two branches stemming from the latent feature. For the ablation study in Tables 3 and 4, since the task is altered to geometric reconstruction, we highlight the cases where scene consistency and the discriminators are not used. Moreover, since there are only two branches for the 3D object completion, when we evaluate without the scene consistency, this is the same as evaluating with the completion branch alone.

In these tables, we also show the advantage of each component that contributes to achieve better IoU. Notably, we can formulate the same conclusion regarding the relation between the scene consistency and discriminator as the 3D scene completion.

1.2. Concatenations from the completion branch

Between the completion and the semantic completion branches, we connect the them by concatenating the resulting layers from the completion to the semantic as shown in Fig. 2. In effect, we are incorporating the geometric structure of the scene into the semantic completion task.

In this ablation study, we investigate the influence of each layer that is concatenated in Table 5. By completely removing the concatenations, the IoU drops from 63.4% to 62.3% in SUNCG and 37.1% to 36.5% in NYU. Between the two, the second one improves the semantic completion task the most, allowing to add larger geometric information into the semantic completion branch.

	bench	chair	couch	table	*Avg.*
Han *et al.* [4]	18.4	14.8	10.1	12.6	14.0
3D-RecAE [10]	23.1	17.8	10.7	14.8	16.6
3D-RecGAN [10]	23.0	17.4	10.9	14.6	16.5
Proposed method	**32.7**	**24.1**	**15.9**	**22.5**	**23.8**
– without scene consistency	26.1	21.5	14.9	18.6	20.3
– without discriminators	25.7	21.1	12.9	18.9	19.7

Table 4: Object completion results on the real world test set provided by 3D-RecGAN [10] in terms of IoU (in %). The resolution for all methods is 64×64×64.

	SUNCG	NYU
Proposed method	**63.4**	**37.1**
– 1st concatenation only	62.1	36.4
– 2nd concatenation only	62.8	36.9
– without concatenations	62.3	36.5

Table 5: Semantic scene completion results on the SUNCG and the NYU test set in terms of IoU (in %).

2. Qualitative results

We also show the qualitative results for 3D object completion in Fig. 3 and compare with 3D-RecGAN [10]. Notably, these results confirm our numerical performance wherein we reconstruct better structures and semantic labels than the state of the art.

References

[1] Angela Dai, Charles Ruizhongtai Qi, and Matthias Nießner. Shape completion using 3d-encoder-predictor cnns and shape synthesis. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume 3, 2017. 2

[2] Andreas Geiger and Chaohui Wang. Joint 3d object and layout inference from a single rgb-d image. In *German Conference on Pattern Recognition (GCPR)*. Springer, 2015. 2

[3] Yuxiao Guo and Xin Tong. View-volume network for semantic scene completion from a single depth image. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. AAAI Press, 2018. 2

[4] Xiaoguang Han, Zhen Li, Haibin Huang, Evangelos Kalogerakis, and Yizhou Yu. High-resolution shape completion using deep neural networks for global structure and local geometry inference. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2017. 2, 3

[5] Dahua Lin, Sanja Fidler, and Raquel Urtasun. Holistic scene understanding for 3d object detection with rgbd cameras. In

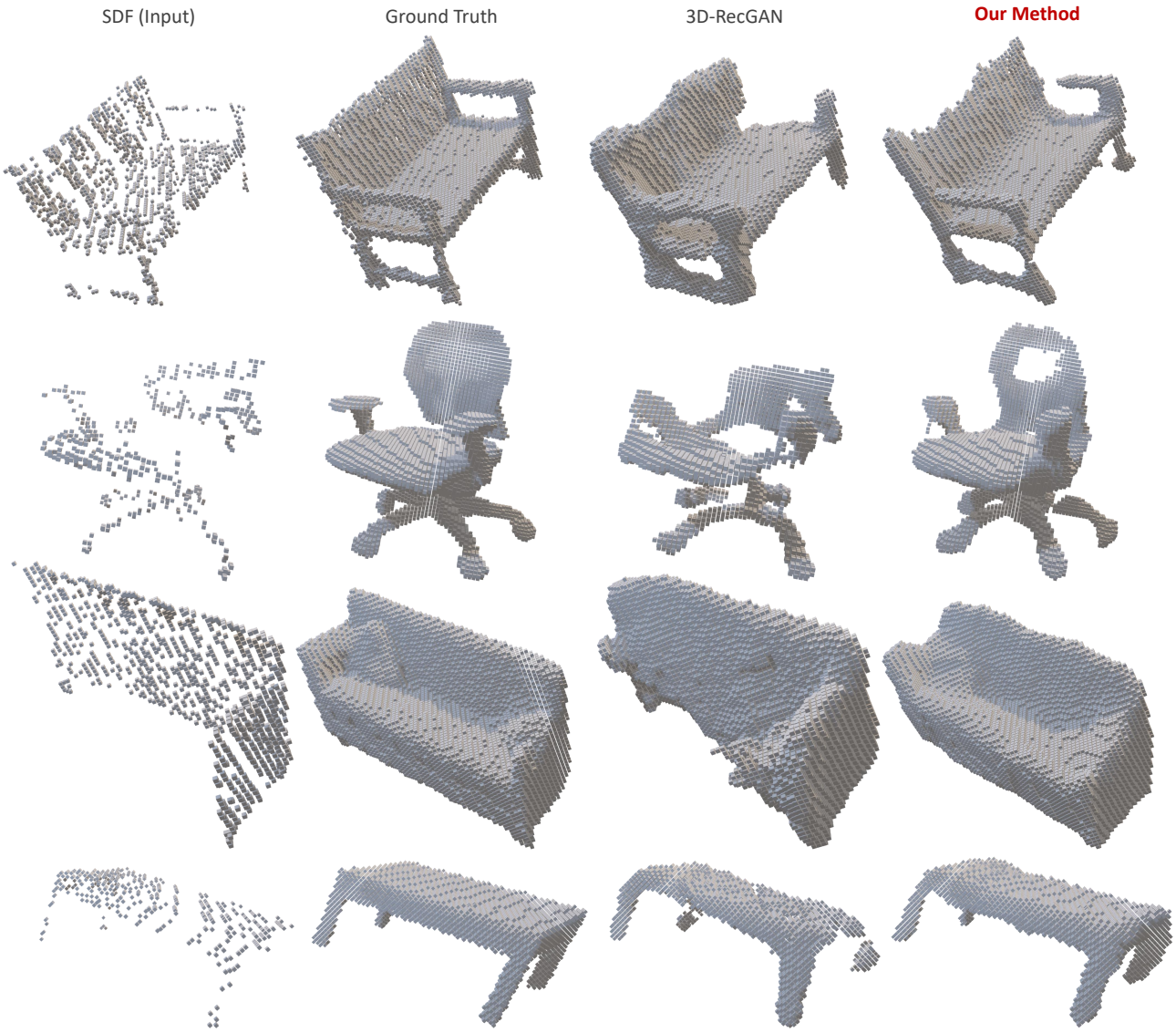


Figure 3: Comparison of the qualitative results of 3D semantic completion results from 3D-RecGAN [10] and our method, evaluated on the dataset provided by [7].

Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2013. 2

[6] Shice Liu, YU HU, Yiming Zeng, Qiankun Tang, Beibei Jin, Yinhe Han, and Xiaowei Li. See and think: Disentangling semantic scene completion. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 263–274. Curran Associates, Inc., 2018. 2

[7] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017. 2, 4

[8] Jacob Varley, Chad DeChant, Adam Richardson, Joaquín Ruales, and Peter Allen. Shape completion enabled robotic grasping. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017. 2

[9] Y. Wang, D. J. Tan, N. Navab, and F. Tombari. Adversarial semantic scene completion from a single depth image. In *2018 International Conference on 3D Vision (3DV)*, 2018. 2

[10] Bo Yang, Stefano Rosa, Andrew Markham, Niki Trigoni, and Hongkai Wen. Dense 3d object reconstruction from a single depth view. *IEEE transactions on pattern analysis and machine intelligence*, 2018. 2, 3, 4

5.2 Point Cloud Completion

Volumetric data relies on a 3D grid with a fixed resolution to discretize the 3D geometry at hand so that the completed shapes produced by our works in Section 5.1 are limited in terms of details. By increasing the resolution, finer details of the surface can be stored. Nevertheless, memory requirements increase exponentially with a linear increase in spatial resolution. Achieving a higher resolution is the driving force to redirect our research to point cloud completion.

5.2.1 SoftPoolNet: Shape Descriptor for Point Cloud Completion and Classification (European Conference on Computer Vision 2020 Oral)

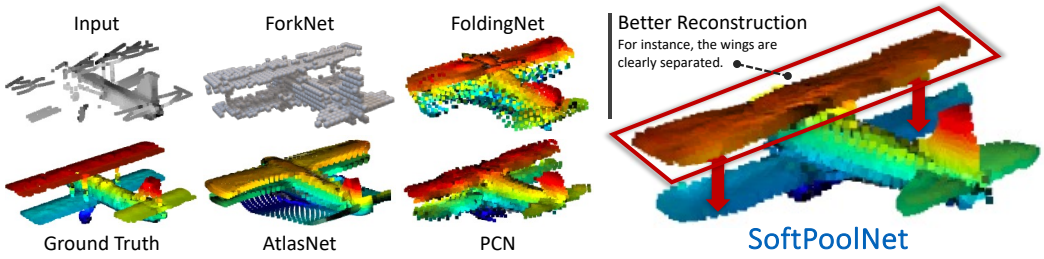


Figure 5.3 The single view object completion with the proposed SoftPoolNet [6] shows more smoothness than our previous volumetric method [8]. It accurately presents the separation between upper and lower wings of the plane compared to other point cloud completion works based on PointNet [40] feature.

Focusing on geometric completion presented in a point cloud, we propose SoftPoolNet [6] to complete a partial scan to a complete shape with detailed surfaces. Structured with flexible 3D local resolutions, the point cloud does not store information about empty spaces, where complex local structures could be presented with more points. Given the partial scan of an object, a point cloud input is presented by N_{in} points written in a matrix formed as $\mathbf{P}_{in} = [\mathbf{x}_i]_{i=1}^{N_{in}}$ where each point is represented as the 3D coordinates $\mathbf{x}_i = [x_i, y_i, z_i]$. The expected complete output point cloud should also be presented as a matrix $\mathbf{P}_{out}$, which a completion model produces.

The main challenge when processing a point cloud is its unstructured arrangement. This fact implies that changing the order of the points in $\mathbf{P}_{in}$ describes the same point cloud but generates a different feature matrix that flows into decoders for completion. PointNet [40] solves this problem by using max-pooling to project the 2D feature into a vector so that it is permutation invariant, but such operation loses too much information from the input, potentially helpful in completion. In SoftPoolNet, we propose to organize the extracted features from the encoder of the point cloud based on their activation, which is a procedure we name soft-pooling. The max-pooling in PointNet is then replaced with soft-pooling so that more information is retained while keeping the required permutation invariance.

Based on the proposed *soft-pool* operation, we build an encoder-decoder architecture called SoftPoolNet, targeted at point cloud completion. Because our latent feature reveals spatial relationships in the feature map, we propose regional convolutions to construct our decoder. Compared to MLP-based approaches, It convolves local features to improve the completion tasks with finer details. Since such convolution depends on local regions, we find local regions by optimizing the inter-regional and intra-regional relationships, resulting in similarly distributed amounts of points throughout the regions, while the confidence of each feature to be in a single region is increased. In practice, we minimize the Chamfer distance between boundary points between 2 regions to restrict the connection between adjacent regions to their boundaries. Finally, we show that this approach benefits classification and completion.

Contributions

Yida Wang implemented and proposed the SoftPool operator and the way to compose the SoftPoolNet feature from features extracted from different regions. He also implemented the regional convolutions used in the decoder to form the whole SoftPoolNet architecture for point cloud completion. He evaluates all experiments.

David Tan suggested focusing more on atypical objects to find the specific advantages of our work. He noticed a problem when regional convolution goes through two regions during inference, which should be solved by carefully setting the hyperparameters of the convolutional kernels.

Federico Tombari involved in the discussion about the advantages and disadvantages of presenting the completed target in point cloud instead of volumetric data.

Nassir Navab financed this work on behalf of the leader of TUM CAMP.

SoftPoolNet: Shape Descriptor for Point Cloud Completion and Classification

Yida Wang[1], David Joseph Tan[2], Nassir Navab[1], Federico Tombari[1,2]

[1] Technical University of Munich
[2] Google

Published version: https://doi.org/10.1007/978-3-030-58580-8_5

SoftPoolNet: Shape Descriptor for Point Cloud Completion and Classification

Yida Wang[1], David Joseph Tan[2], Nassir Navab[1], and Federico Tombari[1,2]

[1] Technische Universität München
[2] Google Inc.

Abstract. Point clouds are often the default choice for many applications as they exhibit more flexibility and efficiency than volumetric data. Nevertheless, their unorganized nature – points are stored in an unordered way – makes them less suited to be processed by deep learning pipelines. In this paper, we propose a method for 3D object completion and classification based on point clouds. We introduce a new way of organizing the extracted features based on their activations, which we name soft pooling. For the decoder stage, we propose regional convolutions, a novel operator aimed at maximizing the global activation entropy. Furthermore, inspired by the local refining procedure in Point Completion Network (PCN), we also propose a patch-deforming operation to simulate deconvolutional operations for point clouds. This paper proves that our regional activation can be incorporated in many point cloud architectures like AtlasNet and PCN, leading to better performance for geometric completion. We evaluate our approach on different 3D tasks such as object completion and classification, achieving state-of-the-art accuracy.

1 Introduction

Point clouds are unorganized sparse representations of a 3D point set. Compared to other common representations for 3D data such as 3D meshes and voxel maps, they are simple and flexible, while being able to store fine details of a surface. For this reason, they are frequently employed for many applications within 3D perception and 3D computer vision such as robotic manipulation and navigation, scene understanding, and augmented/virtual reality. Recently, deep learning approaches have been proposed to learn from point clouds for 3D perception tasks such as point cloud classification [4,14,18,19] or point cloud segmentation [11,13,14,17,23]. Among them, one of the key breakthroughs in handling unorganized point clouds was proposed by PointNet [18], introducing the idea of a max pooling in the feature space to yield permutation invariance.

An interesting emerging research trend focusing on 3D data is the so-called 3D completion, where the geometry of a partial scene or object acquired from a single viewpoint, *e.g.* through a depth map, is completed of the missing part due to (self-)occlusion as visualized in Fig. 1. This can be of great use to aid standard 3D perception tasks such as object modeling, scene registration, part-based segmentation and object pose estimation. Most approaches targeting 3D completion

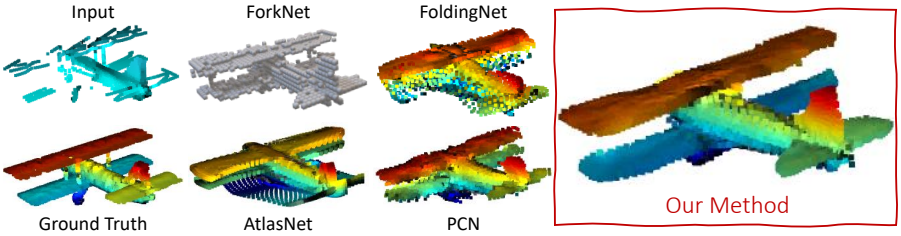


Fig. 1: This paper proposes a method that reconstructs 3D point cloud models with more fine details.

have been proposed for volumetric approaches, since 3D convolutions are naturally suited to this 3D representation. Nevertheless, such approaches bring in the limitations of this representation, including loss of fine details due to discretization and limitations in scaling with the 3D size. Recently, a few approaches have explored the possibility of learning to complete a point cloud [9,25,26].

This paper proposes an encoder-decoder architecture called SoftPoolNet, which can be employed for any task that processes a point cloud as input in order to regress another point cloud as output. One of the tasks and a main focus for this work is 3D object completion from a partial point cloud.

The theoretical contribution of SoftPoolNet is twofold. We first introduce soft pooling, a new module that replaces the max-pooling operator in PointNet by taking into account multiple high-scoring features rather than just one. The intuition is that, by keeping multiple features with high activations rather than just the highest, we can retain more information while keeping the required permutation invariance. A second contribution is the definition of a regional convolution operator that is used within the proposed decoder architecture. This operator is designed specifically for point cloud completion and relies on convolving local features to improve the completion task with fine details.

In addition to evaluating SoftPoolNet for point cloud completion, we also evaluate on the point cloud classification to demonstrate its applicability to general point cloud processing tasks. In both evaluations, SoftPoolNet obtains state of the art results on the standard benchmarks.

2 Related work

Volumetric completion. Object [7] and scene completion [20,22] are typically carried out by placing all observed elements into a 3D grid with fixed resolution. 3D-EPN [7] completes a single object using 3D convolutions while 3D-RecGAN [24] further improves the completion performance by using discriminative training. As scene completion contains objects in different scales and more random relative position among all of them, SSCNet [20] proposes a 3D volumetric semantic completion architecture using dilated convolutions to recognize

objects with different scales. ForkNet [22] designs a multi-branch architecture to generate realistic training data to supplement the training.

Point cloud completion. Object completion based on point cloud data change partial geometries without using a 3D fixed grid. They represent completed shapes as a set of points with 3D coordinates. For instance, FoldingNet [25] deforms a 2D grid from a global feature such as PointNet [18] feature to an output with a desirable shape. AtlasNet [9] generates an object with a set of local patches to simulate mesh data. But overlaps between different local patches makes the reconstruction noisy. MAP-VAE [10] predicts the completed shape by joining the observed part with the estimated counterpart.

CNNs for point clouds. Existing works like PointConv [23] and PointCNN [13] index each point with k-nearest neighbour search to find local patches, where they then apply the convolution kernels on those local patches. Regarding point cloud deconvolutional operations, FoldingNet [25] uses a 2D grid to help generate a 3D point cloud from a single feature. PCN [26] further uses local FoldingNet to obtain a fine-grained output from a coarse point cloud with low resolution which could be regarded as an alternative to point cloud deconvolution.

3 Soft pooling for point features

Given the partial scan of an object, the input to our network is a point cloud with N_{in} points written in the matrix form as $\mathbf{P}_{\text{in}} = [\mathbf{x}_i]_{i=1}^{N_{\text{in}}}$ where each point is represented as the 3D coordinates $\mathbf{x}_i = [x_i, y_i, z_i]$. We then convert each point into a feature vector $\mathbf{f}_i$ with N_f elements by projecting every point with a point-wise multi-layer perceptron [18] (MLP) $\mathbf{W}_{\text{point}}$ with three layers. Thus, similar to $\mathbf{P}_{\text{in}}$, we define the $N_{\text{in}} \times N_f$ feature matrix as $\mathbf{F} = [\mathbf{f}_i]_{i=1}^{N_{\text{in}}}$. Note that we applied a softmax function to the output neuron of MLP so that the elements in $\mathbf{f}_i$ ranges between 0 and 1.

The main challenge when processing a point cloud is its unstructured arrangement. This implies that changing the order of the points in $\mathbf{P}_{\text{in}}$ describes the same point cloud, but generates a different feature matrix that flows into our architecture with convolutional operators. To solve this problem, we propose to organize the feature vectors in $\mathbf{F}$ so that their k-th element are sorted in a descending order, which is denoted as $\mathbf{F}'_k$. Note that k should not be larger than N_f. A *toy example* of this process is depicted in Fig. 2(a) where we assume that there are only five points in the point cloud and arrange the five feature vectors from $\mathbf{F} = [\mathbf{f}_i]_{i=1}^{5}$ to $\mathbf{F}'_k = [\mathbf{f}_i]_{i=\{3,5,1,2,4\}}$ by comparing the k-th element of each vector. Repeating this process for all the N_f elements in $\mathbf{f}_i$, all $\mathbf{F}'_k$ together result to a 3D tensor $\mathbf{F}' = [\mathbf{F}'_1, \mathbf{F}'_2, \dots \mathbf{F}'_{N_f}]$ with the dimension of $N_{\text{in}} \times N_f \times N_f$. As a result, any permutation of the points in $\mathbf{P}_{\text{in}}$ generate the same $\mathbf{F}'$.

Sorting the feature vectors in a descending order highlights the ones with the highest activation values. Thus, by selecting the first N_r feature vectors from all the $\mathbf{F}'_k$ as shown in Fig. 2(b), we assemble $\mathbf{F}^*$ that accumulates the features

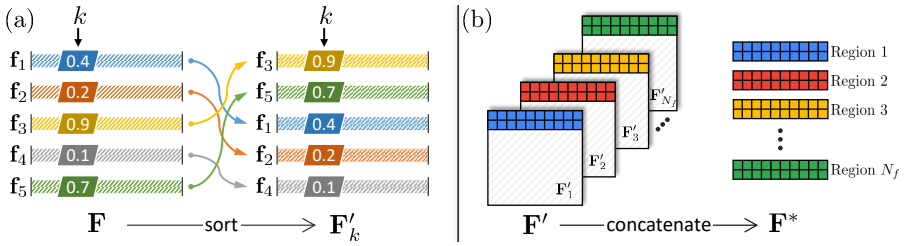


Fig. 2: Toy examples of (a) sorting the the k-th element of the vectors in the feature matrix $\mathbf{F}$ to build $\mathbf{F}'_k$ and consequently $\mathbf{F}'$ and (b) concatenation of the first N_r rows of $\mathbf{F}'_k$ to construct the 2D matrix $\mathbf{F}^*$ which corresponds to the regions with high activations.

with the highest activations. Altogether, the output of soft pooling is the $N_f \cdot N_r$ point features. Since each feature vector corresponds to a point in $\mathbf{P}_{\text{in}}$, we can interpret the first N_r feature vectors as a region in the point cloud. The effects of the activations on the 3D reconstruction are illustrated in Fig. 3, where the point cloud is divided into N_f regions. Later in Sec. 6, we discuss on how to learn $\mathbf{W}_{\text{point}}$ by incorporating these regions. That section introduces several loss functions which optimize towards entropy, Chamfer distance and earth-moving distance such that each point is optimized to fall into only one region and to be selected for $\mathbf{F}^*$ by maximizing the k-th element of the feature vector associated to the same region.

Similar to PointNet [18], we also rely on MLP to build the feature matrix $\mathbf{F}$. However, PointNet directly applies max-pooling on $\mathbf{F}$ to produce a vector while we try to generalize this approach and sort the feature vectors in order to assemble a matrix $\mathbf{F}^*$ as illustrated in Fig. 2. Considering the distinction between the two approaches, we refer our approach as *soft pooling*. Fundamentally, in addition to the increased amount of information from our feature vectors, the advantage of our method is the ability to apply regional convolutional operations to $\mathbf{F}^*$, as discussed in Sec. 4. The differences are evident in Fig. 4, where the proposed method achieves detailed results on reconstructing all the six legs while

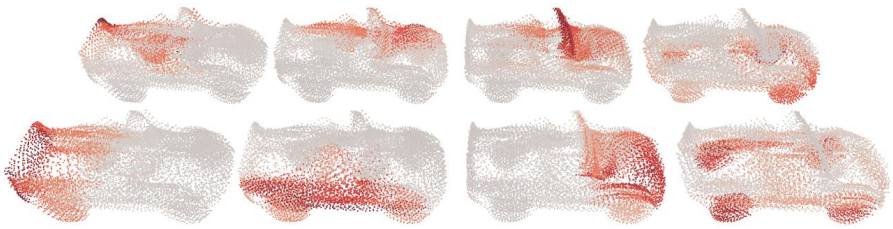


Fig. 3: Deconstructing the learned regions (unsupervised) that correspond to different parts of the car.

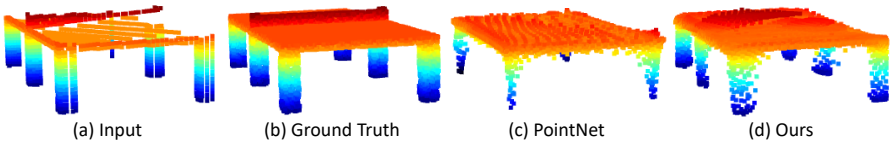


Fig. 4: Comparison of our method and PointNet [18] where PointNet reconstructs the more typical four-leg table instead of six in (c).

PointNet follows the more generic structure of the table with four. This proves that *soft pooling* makes our decoder able to take all observable geometries into account to complete the shape, while the max-pooled PointNet feature cannot reveal the rarely seen geometry.

4 Regional convolution

Operating on $\mathbf{F}^*$, we introduce the convolutional kernel $\mathbf{W}_{\text{conv}}$ that transforms $\mathbf{F}^*$ to a new set of points $\mathbf{P}_{\text{conv}}$ by taking several point features into consideration. We structure $\mathbf{W}_{\text{conv}}$ with a dimension of $N_p \times N_f \times 3$ where N_p represent the number of points which are taken into consideration such that

$$\mathbf{P}_{\text{conv}}(i,j) = \sum_{l=1}^{N_f} \sum_{k=1}^{N_p} \mathbf{F}^*(i+k,l)\mathbf{W}_{\text{conv}}(j,k,l) \ . \tag{1}$$

Here, the kernel slides only inside each region of features without taking features from two different regions in one convolutional operation. As the kernel size allows it to cover N_p features, we pad each region with $N_p - 1$ duplicated samples at the end of each region in order to keep the output resolution the same as N_{in}. Experimentally, we tried different numbers of N_p ranging from 4 to 64 and evaluated that 32 generates the best results. Learning the values in $\mathbf{W}_{\text{conv}}$ is discussed in Sec. 6.

In addition, we use a convolution stride which is set as a value smaller than N_p to change the output resolution in terms of the number of point features. With a stride of S, we then take samples every S point feature in $\mathbf{F}^*$. Notice that, by using a stride which is smaller than 1, we can also upsample $\mathbf{F}^*$ by interpolating $\frac{1}{S} - 1$ new points between two points then apply the convolution kernel again. This is an essential tool in reconstructing the object from a partial scan.

5 Network architecture

We build an encoder-decoder architecture which consists of MLP and our regional convolutions, respectively. Serving as the input to our network, we permutate the input scans and resample 1,024 points. If the partial scans have less than 1,024 points, we then duplicate the missing samples.

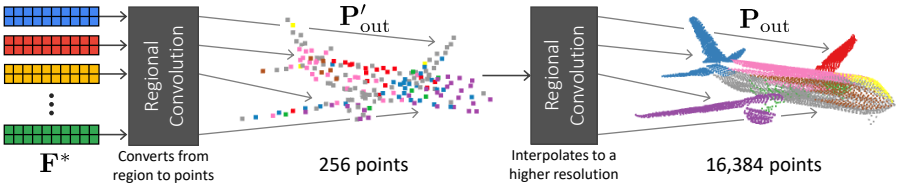


Fig. 5: Decoder architecture of SoftPoolNet with two regional convolution that converts the features from the regions to point clouds and interpolates from the coarse 256 points to a higher resolution with 16,384 points.

Our encoder is a point-wise MLP that generates the output neuron with a dimension of $[512, 512, 8]$. We then perform soft pooling as described in Sec. 3 that produces $\mathbf{F}^*$ with the size of $[256, 8]$ by setting N_r to 32 and N_f to 8, resulting an output of $N_f \cdot N_r = 256$ features.

Finally, for the decoder, we propose a two-stage point cloud completion architecture which is trained end-to-end. The output of the first is used as the input of the second point cloud completion network. Both of them produces the completed point cloud but with different resolutions. Illustrated in Fig. 5, we construct the decoder with two regional convolutions from Sec. 4. The first output $\mathbf{P}'_{\text{out}}$ is fixed at 256 while the second $\mathbf{P}_{\text{out}}$ produces a maximum resolution of 16,384.

6 Loss functions

During learning, we evaluate whether the predicted point feature $\mathbf{P}_{\text{out}}$ matches the given ground truth $\mathbf{P}_{\text{gt}}$ through the Chamfer distance. Similar to [9,25,26], we use the regression loss function for the shape completion from a point cloud

$$\mathcal{L}_{\text{complete}}(\mathbf{W}_{\text{point}}, \mathbf{W}_{\text{conv}}) = \text{Chamfer}(\mathbf{P}_{\text{out}}, \mathbf{P}_{\text{gt}}) \ . \tag{2}$$

We observed that there are two major drawbacks in using this loss function alone – the reconstructed surface tends to be either curved on the sharp edges such FoldingNet [25] or having noisy points appear on flat surfaces such as AtlasNet [9] and PCN [26]. In this work, we tackle these problems by finding local regions first, then by optimizing the inter- and intra-regional relationships.

Moreover, while FoldingNet [25] sacrifices local details to present the entire model with a single mesh having smooth surface, AtlasNet [9] and PCN [26] use local regions (or patches) to increase the details in the 3D model. However, both of them [9,26] have severe overlapping effects between adjacent regions which makes the generated object noisy and the regions discontinuous. To solve this problem, we aim at reducing the overlaps between two adjacent regions.

6.1 Learning activations through regional entropy

Considering that the dimension of a single feature is N_f, we can directly define N_f regions for all features. Given the probabilities of regions to which the feature

$\mathbf{f}_i$ belong, we want to optimize the inter- and intra-regional relationships among the features. We directly present the probability of the feature $\mathbf{f}_i$ belonging to all N_f regions by applying the softmax function to $\mathbf{f}_i$ as

$$P(\mathbf{f}_k, i) = \frac{\mathbf{f}_k[i]}{\sum_{j=1}^{N_f} \mathbf{f}_k[j]} \ . \tag{3}$$

Since the information entropy evaluates both the distribution and the confidence of the probabilities of a set of data, we define the feature entropy and the regional entropy based on the regional probability of the feature.

The goal of the inter-regional loss function is to similarly distribute the number of points throughout the regions. We define the regional entropy as

$$\mathcal{E}_r = -\frac{1}{B}\sum_{j=1}^{B}\sum_{i=1}^{R}\left[\left(\frac{1}{N}\sum_{k=1}^{N} P(\mathbf{f}_k, i)\right) \cdot \log\left(\frac{1}{N}\sum_{k=1}^{N} P(\mathbf{f}_k, i)\right)\right] \tag{4}$$

where B is the batch-size. Here, we want to maximize $\mathcal{E}_r$. Considering that the upper-bound of $\mathcal{E}_r$ is $-\log\frac{1}{R} = \log(R)$, we can then define the inter-regional loss function as

$$\mathcal{L}_{\text{inter}}(\mathbf{W}_{\text{point}}) = \log(R) - \mathcal{E}_r \tag{5}$$

in order to acquire a positive loss function. Once $\mathcal{E}_r$ is close to $\log(R)$, each region would contain similar amount of point features. Interestingly, we can select the number of regions by evaluating how much the regional entropy $\mathcal{E}_r$ differs from its upper-bound. The best number of regions should be the one with a small $\mathcal{L}_{\text{inter}}$. This is evaluated later in Table 6.

On the other hand, the goal of the intra-regional loss function is to boost the confidence of each feature to be in a single region. The intra-regional loss function then minimize the feature entropy

$$\mathcal{L}_{\text{intra}}(\mathbf{W}_{\text{point}}) = -\frac{1}{N}\frac{1}{B}\sum_{k=1}^{N}\sum_{j=1}^{B}\sum_{i=1}^{N_f} P(\mathbf{f}_k, i)\log P(\mathbf{f}_k, i) \ . \tag{6}$$

The optimum case of the feature entropy is for each feature to be a one-hot code, *i.e.* when only one element is 1 while the others are zero.

6.2 Reducing the overlapping regions

Although $\mathcal{L}_{\text{intra}}$ tries to make each point feature confident about the region to which it belongs, instances exist where many adjacent points would fall under different regions. For example, we observe in Fig. 6 that patches from different regions are stacked on top of each other, producing noisy reconstructions. Notably, this introduces unexpected results when fitting a mesh to the point cloud. Thus, we want to minimize region overlap by optimizing the network to restrict the connection between adjacent regions to their boundaries.

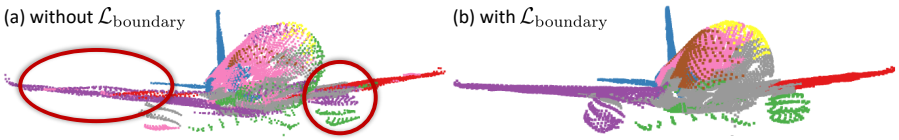


Fig. 6: Effects of without and with $\mathcal{L}_{\text{boundary}}$ where the wings are not planar and the engines are less visible in (a). Note that the colors represent different regions.

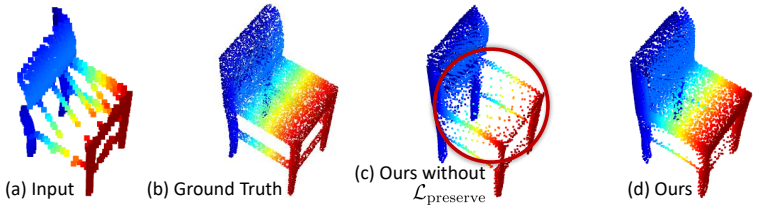


Fig. 7: Effects of without and with $\mathcal{L}_{\text{preserve}}$ where the seat is missing in (c).

First, each point is assigned to a region with the highest activation. All points that belong to region i but has activation for region j larger than a threshold τ are included in the set $\mathcal{B}_i^j$. Inversely, the points that belong to region j but have activation for region i larger than τ are added in the set $\mathcal{B}_j^i$. Note that, if both sets $\mathcal{B}_i^j$ and $\mathcal{B}_j^i$ are not empty, the regions i and j are then adjacent. Thus, by minimizing the Chamfer distance between $\mathcal{B}_i^j$ and $\mathcal{B}_j^i$, we can make the overlapping sets of points smaller such that the optimal result is a line. We then define the loss function for the boundary as

$$\mathcal{L}_{\text{boundary}}(\mathbf{W}_{\text{point}}, \mathbf{W}_{\text{conv}}) = \sum_{i=1}^{N_f} \sum_{j=i}^{N_f} \text{Chamfer}(\mathcal{B}_i^j, \mathcal{B}_j^i) \tag{7}$$

where both $\mathbf{W}_{\text{point}}$ and $\mathbf{W}_{\text{conv}}$ are optimized. After experimenting on different values of τ from 0.1 to 0.9, we set τ to be 0.3.

6.3 Preserving the features from MLP

After sorting and filtering the features to produce $\mathbf{F}^*$, some feature vectors in $\mathbf{F}^*$ are duplicated while some vectors from $\mathbf{F}$ are missing in $\mathbf{F}^*$. To avoid these, we introduce the loss function

$$\mathcal{L}_{\text{preserve}}(\mathbf{W}_{\text{point}}) = \text{Earth-moving}(\mathbf{F}^*, \mathbf{F}) \ . \tag{8}$$

Since the earth moving distance [12] is not efficient when the size of the samples is large, we then randomly select 256 vectors from $\mathbf{F}$ and $\mathbf{F}^*$. Considering that the feature dimensions in $\mathbf{F}$ and $\mathbf{F}^*$ are both N_f, the earth moving distance

then takes features with N_f dimension as input. In practice, Fig. 7 visualizes the effects of $\mathcal{L}_{\text{preserve}}$ in the reconstruction, where removing this loss produce a large hole on the seat while incorporating this loss builds a well-distributed point cloud.

7 Experiments

For all evaluations, we train our model with an NVIDIA Titan V and parameterize it with a batch size of 8. Moreover, we apply the Leaky ReLU with a negative slope of 0.2 on the output of each regional convolution output.

7.1 Object completion on ShapeNet

We evaluate the performance of the geometric completion of a single object on the ShapeNet [5] database where training data are paired point clouds of the partial scanning and the completed shape. To make it comparable to other approaches, we adopt the standard 8 category evaluation [26] for a single object completion. As rotation errors are common in the partial scans, we further evaluate our approach against other works on the ShapeNet database with rotations. We also evaluate the performance on both high and low resolutions which contain 16,384 and 2,048 points, respectively.

We compare against other point cloud completion approaches such as PCN [26], FoldingNet [25], AtlasNet [9] and PointNet++ [19]. To show the advantages over volumetric completion, we also compare against 3D-EPN [24] and ForkNet [22] with an output resolution of $64 \times 64 \times 64$. Notably, we achieve the best results on most objects and in all types of evaluations as presented in Table 1, Table 2 and Table 3.

An interesting hypothesis is the capacity of $\mathcal{L}_{\text{boundary}}$ to be integrated in other existing approaches. Thus, Table 1 and Table 2 also evaluate this hypothesis and prove that this activation helps FoldingNet [25], PCN [26] and AtlasNet [9]

Output Resolution = 16,384

Method	plane	cabinet	car	chair	lamp	sofa	table	vessel	*Avg.*
3D-EPN [7]	13.16	21.80	20.31	18.81	25.75	21.09	21.72	18.54	20.15
ForkNet [22]	9.08	14.22	11.65	12.18	17.24	14.22	11.51	12.66	12.85
PointNet++ [19]	10.30	14.74	12.19	15.78	17.62	16.18	11.68	13.52	14.00
FoldingNet [25]	5.97	10.80	9.27	11.25	12.17	11.63	9.45	10.03	10.07
FoldingNet + $\mathcal{L}_{\text{boundary}}$	5.79	10.61	8.62	10.33	11.56	11.05	9.41	9.79	9.65
PCN [26]	5.50	10.63	8.70	11.00	11.34	11.68	8.59	9.67	9.64
PCN + $\mathcal{L}_{\text{boundary}}$	5.13	9.12	7.58	9.35	9.40	9.31	7.30	8.91	8.26
Our Method	**4.01**	**6.23**	**5.94**	**6.81**	**7.03**	**6.99**	**4.84**	**5.70**	**5.94**

Table 1: Completion evaluated by means of the Chamfer distance (multiplied by 10^3) with the output resolution of 16,384.

Output Resolution = 2,048

Method	plane	cabinet	car	chair	lamp	sofa	table	vessel	*Avg.*
FoldingNet [25]	11.18	20.15	13.25	21.48	18.19	19.09	17.80	10.69	16.48
FoldingNet + $\mathcal{L}_{\text{boundary}}$	11.09	19.95	13.11	21.27	18.22	19.06	17.62	10.10	16.30
AtlasNet [9]	10.37	23.40	13.41	24.16	20.24	20.82	17.52	11.62	17.69
AtlasNet + $\mathcal{L}_{\text{boundary}}$	9.25	22.51	12.12	22.64	18.82	19.11	16.50	11.53	16.56
PCN [26]	8.09	18.32	10.53	19.33	18.52	16.44	16.34	10.21	14.72
PCN + $\mathcal{L}_{\text{boundary}}$	6.39	16.32	9.30	18.61	16.72	16.28	15.29	9.00	13.49
TopNet [21]	5.50	12.02	8.90	12.56	9.54	12.20	9.57	7.51	9.72
Our Method	**4.76**	**10.29**	**7.63**	**11.23**	**8.97**	**10.08**	**7.13**	**6.38**	**8.31**
– *without* $\mathcal{L}_{\text{inter}}$	10.82	20.45	15.21	20.19	18.05	18.58	15.65	8.81	15.97
– *without* $\mathcal{L}_{\text{intra}}$	5.23	16.10	12.49	14.62	13.90	12.37	12.96	5.72	11.67
– *without* $\mathcal{L}_{\text{inter}}$, $\mathcal{L}_{\text{intra}}$	10.91	20.54	15.27	20.28	18.16	18.66	15.75	8.91	16.06
– *without* $\mathcal{L}_{\text{boundary}}$	5.46	10.98	8.27	11.95	9.51	10.92	7.78	7.40	9.03
– *without* $\mathcal{L}_{\text{preserve}}$	10.29	19.75	14.13	19.35	17.88	18.21	15.23	8.11	15.37

Table 2: Completion evaluated using the Chamfer distance (multiplied by 10^3) with the output resolution of 2,048.

Output Resolution = 1,024

Method	plane	cabinet	car	chair	lamp	sofa	table	vessel	*Avg.*
3D-EPN [7]	6.20	7.76	8.70	7.68	10.73	8.08	8.10	8.17	8.18
PointNet++ [19]	5.96	11.62	6.69	11.06	18.58	10.26	8.61	8.38	10.14
FoldingNet [25]	15.64	22.13	17.46	29.74	32.00	24.57	18.99	21.88	22.80
PCN [26]	3.88	7.07	5.50	6.81	8.46	7.24	6.01	6.27	6.40
DeepSDF [16]	3.88	-	-	5.63	-	**4.68**	-	-	-
LGAN [3]	3.32	-	-	5.59	-	-	-	-	-
MAP-VAE [10]	3.23	-	-	5.57	-	-	-	-	-
Our Method	**2.52**	**5.49**	**4.08**	**5.20**	**6.17**	5.25	**4.61**	**5.80**	**4.89**

Table 3: Completion results using the Earth-Moving distance (multiplied by 10^2) with the output resolution of 1,024. We report the values of DeepSDF [16] from their original paper by rescaling according to the difference of point density.

perform better. Nevertheless, even with such improvements, the complete version of the proposed method still outperforms them.

7.2 Car completion on KITTI

The KITTI [8] dataset present partial scans of real-world cars using Velodyne 3D laser scanner. We adopt the same training and validating procedure for car completion as proposed by PCN [26]. We train a car completion model based on the training data generated from ShapeNet [5] and test our completion method on sparse point clouds generated from the real-world LiDAR scans. For each sample, the points within the bounding boxes are extracted with 2,483 partial

Method	Fidelity	MMD	Consistency
FoldingNet [25]	0.03155	0.02080	0.01326
AtlasNet [9]	0.03461	0.02205	0.01646
PCN [26]	0.02800	0.01850	0.01163
Our Method	**0.02171**	**0.01465**	**0.00922**
PCN [26] (rotate)	0.03352	0.02370	0.01639
Our Method (rotate)	**0.02392**	**0.01732**	**0.01175**

Table 4: Car completion on LiDAR scans from KITTI.

point clouds. Each point cloud is then transformed to the box's coordinates to be completed by our model then transformed back to the world frame. PCN [26] proposed three metrics to evaluate the performance of our model: (1) *Fidelity*, *i.e.* the average distance from each point in the input to its nearest neighbour in the output (*i.e.* measures how well the input is preserved); (2) *Minimal Matching Distance* (MMD), *i.e.* the Chamfer distance between the output and the car's point cloud nearest neighbor from ShapeNet (*i.e.* measures how much the output resembles a typical car); and, (3) *Consistency*, the average Chamfer distance between the completion outputs of the same instance in consecutive frames (*i.e.* measures how consistent the networks outputs are against variations in the inputs).

Table 4 shows that we achieve state of the art on the metrics compared to FoldingNet [25], AtlasNet [9] and PCN [26]. When we introduce random rotations on the bounding box in order to simulate errors in the initial stages, we still acquire the lowest errors.

7.3 Classification on ModelNet and PartNet

We evaluate the performance of the features in term of classification on ModelNet10 [27], ModelNet40 [27] and PartNet [15] datasets. ModelNet40 contains 12,311 CAD models in 40 categories. Here, the training data contains 9,843 samples and the testing data contains 2,468 samples. Following RS-DGCNN [2], a linear Support Vector Machine [6] (SVM) is trained on the representations learned in an unsupervised manner on the ShapeNet dataset. RS-DGCNN [2] divides the point cloud of the objects into several regions by positioning the object in a pre-defined voxel grid, then use the regional information to help train latent feature. In Table 5, the proposed method outperforms RS-DGCNN [2] by 1.64% accurracy on ModelNet40 dataset, which shows that our feature contains better categorical information. Notably, similar results are also acquired from ModelNet10 [27] and PartNet [15] with their respective evaluation strategy.

7.4 Ablation study

Loss functions. In the reconstruction and classification experiments, Table 2 and Table 5 also include the ablation study that investigates the effects of the

Method	ModelNet40 [27]	ModelNet10 [27]	PartNet [15]
VConv-DAE	75.50%	80.50%	-
3D-GAN	83.30%	91.00%	74.23%
Latent-GAN	85.70%	95.30%	-
FoldingNet	88.40%	94.40%	-
VIP-GAN	90.19%	92.18%	-
RS-PointNet [2]	87.31%	91.61%	76.95%
RS-DGCNN [2]	90.64%	94.52%	-
KCNet [1]	91.0%	94.4%	-
Our Method	**92.28%**	**96.14%**	**84.32%**
– *without* $\mathcal{L}_{\text{inter}}$	89.40%	95.75%	81.13%
– *without* $\mathcal{L}_{\text{intra}}$	83.70%	90.21%	79.28%
– *without* $\mathcal{L}_{\text{inter}}$, $\mathcal{L}_{\text{intra}}$	82.97%	90.02%	78.41%
– *without* $\mathcal{L}_{\text{boundary}}$	88.26%	95.01%	80.86%
– *without* $\mathcal{L}_{\text{preserve}}$	86.09%	92.27%	79.05%

Table 5: Object classification on ModelNet40 [27], ModelNet40 [27] and PartNet [15] datasets in terms of accuracy.

loss functions from Sec. 6. For both experiments, we notice all loss functions are critical to achieve good results since each of them focuses on different aspects.

Activations. Since the number of regions is one of the hyper-parameters in our approach, we evaluate on the performance with different number of regions quantitatively in Table 6. These results demonstrate that the accuracy for the shape completion is increasing as the number of regions increases from 2 to 8, then the performance gradually drops as the number of regions continues to increase from 8 to 32. By observing $\mathcal{L}_{\text{inter}}$ at the same time, we find that it achieves the minimum value of 0.20 when there are 8 regions as well. This proves that $\mathcal{L}_{\text{inter}}$ can be used as an indicator for whether the expected number of regions could be used or not.

Moreover, Fig. 8 shows the regional activations when we shuffle the sequence of points in the partial scan. We can see that both the reconstructed geometry relative sub-regions are identical. So, it illustrates that, by using the proposed regional activations, our model is permutation invariant, which indicates that the reordered point cloud is suitable to perform convolutions.

(N_f, N_r)	$(2, 128)$	$(4, 64)$	$(8, 32)$	$(16, 16)$	$(32, 8)$
Chamfer Distance	7.80	6.31	**5.94**	6.27	6.75
$\mathcal{L}_{\text{inter}}$	0.41	0.67	**0.20**	0.49	1.33

Table 6: Influence of N_f and N_r on the Chamfer distance (multiplied by 10^3) and $\mathcal{L}_{\text{inter}}$.

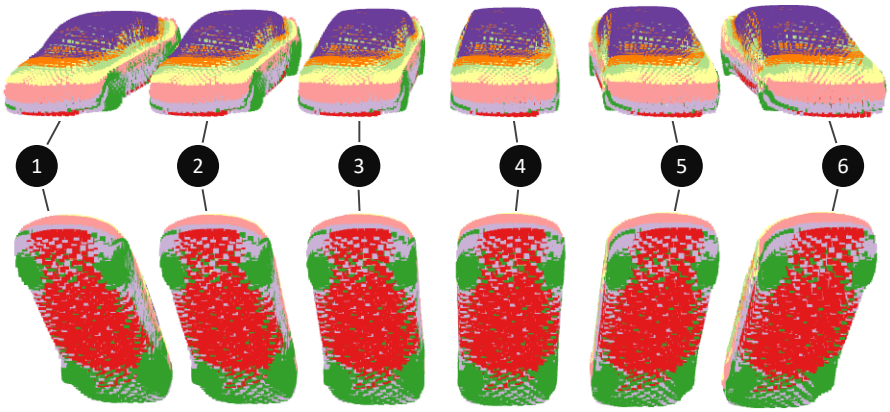


Fig. 8: With identical results, this evaluation shows the robustness of the reconstruction when we randomly shuffle the input point cloud.

Point cloud versus volumetric data. In addition to achieving worse numerical results in Sec. 7.1, volumetric approaches have smaller resolutions than the point cloud approaches due to the memory constraints. The difference becomes more evident in Fig. 9, where ForkNet [22] is limited by a $64 \times 64 \times 64$ grid. Nevertheless, both the volumetric and point cloud approaches have difficulty in reconstructing thin structures. For instance, the volumetric approach tends to ignore the joints between the wheels and car chassis in Fig. 9 while FoldingNet [25] and AtlasNet [9] only use large surface to cover the area of wheels. In contrast, our approach is capable of reconstructing the thin structures quite well. Moreover, in Table 7, we also achieve the lowest inference time compared to all point cloud and volumetric approaches.

Method	Size (MB)	Inference Time (s)	Closed Surface	Type of Data
3D-EPN [7]	420	-	Yes	Volumetric
ForkNet [22]	362	-	Yes	Volumetric
FoldingNet [25]	19.2	0.05	Yes	Points
AtlasNet [9]	2	0.32	No	Points
PCN [26]	54.8	0.11	No	Points
DeepSDF [16]	7.4	9.72	Yes	SDF
Our Method	37.2	0.04	Yes	Points

Table 7: Overview of the object completion methods. The inference time is the amount of time to conduct inference on a single sample.

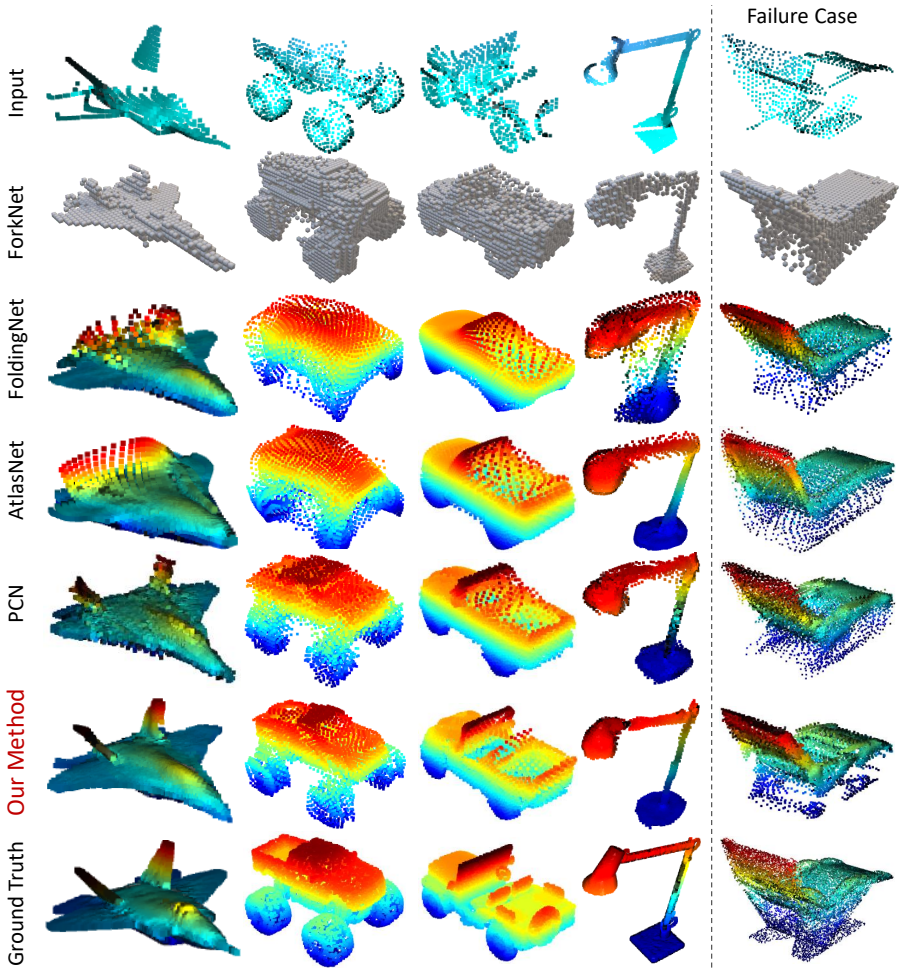


Fig. 9: Evaluated on ShapeNet [5], comparison of shape completion based on ForkNet [22], FoldingNet [25], AtlasNet [9] and PCN [26] against our method.

8 Conclusion

This paper introduced the SoftPool idea as a novel and general way to extract rich deep features from unordered point sets such as 3D point clouds. Also, it proposed a state-of-the-art point cloud completion approach by designing a regional convolution network for the decoding stage. Our numerical evaluation reflects that our approach achieves the best results on different 3D tasks, while our quantitative results illustrate the reconstruction and completion ability of our method with respect to ground truth.

References

1. Mining point cloud local structures by kernel correlation and graph pooling. In: CVPR (2018) 12
2. Self-supervised deep learning on point clouds by reconstructing space. In: NIPS (2019) 11, 12
3. Achlioptas, P., Diamanti, O., Mitliagkas, I., Guibas, L.: Learning representations and generative models for 3D point clouds. In: Dy, J., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 80, pp. 40–49. PMLR, Stockholmsmssan, Stockholm Sweden (10–15 Jul 2018) 10
4. Arief, H.A., Arief, M.M., Bhat, M., Indahl, U.G., Tveite, H., Zhao, D.: Density-adaptive sampling for heterogeneous point cloud object segmentation in autonomous vehicle applications. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. pp. 26–33 (2019) 1
5. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al.: Shapenet: An information-rich 3d model repository. arXiv preprint arXiv:1512.03012 (2015) 9, 10, 14
6. Cortes, C., Vapnik, V.: Support-vector networks. Machine learning **20**(3), 273–297 (1995) 11
7. Dai, A., Qi, C.R., Nießner, M.: Shape completion using 3d-encoder-predictor cnns and shape synthesis. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). vol. 3 (2017) 2, 9, 10, 13
8. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: The kitti dataset. The International Journal of Robotics Research **32**(11), 1231–1237 (2013) 10
9. Groueix, T., Fisher, M., Kim, V.G., Russell, B.C., Aubry, M.: A papier-mch approach to learning 3d surface generation. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2018) 2, 3, 6, 9, 10, 11, 13, 14
10. Han, Z., Wang, X., Liu, Y.S., Zwicker, M.: Multi-angle point cloud-vae: Unsupervised feature learning for 3d point clouds from multiple angles by joint self-reconstruction and half-to-half prediction. In: The IEEE International Conference on Computer Vision (ICCV) (October 2019) 3, 10
11. Landrieu, L., Simonovsky, M.: Large-scale point cloud semantic segmentation with superpoint graphs. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4558–4567 (2018) 1
12. Li, P., Wang, Q., Zhang, L.: A novel earth mover's distance methodology for image matching with gaussian mixture models. In: The IEEE International Conference on Computer Vision (ICCV) (December 2013) 8
13. Li, Y., Bu, R., Sun, M., Wu, W., Di, X., Chen, B.: Pointcnn: Convolution on x-transformed points. In: Advances in Neural Information Processing Systems. pp. 820–830 (2018) 1, 3
14. Liu, Y., Fan, B., Xiang, S., Pan, C.: Relation-shape convolutional neural network for point cloud analysis. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 8895–8904 (2019) 1
15. Mo, K., Zhu, S., Chang, A.X., Yi, L., Tripathi, S., Guibas, L.J., Su, H.: PartNet: A large-scale benchmark for fine-grained and hierarchical part-level 3D object understanding. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019) 11, 12

16. Park, J.J., Florence, P., Straub, J., Newcombe, R., Lovegrove, S.: Deepsdf: Learning continuous signed distance functions for shape representation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 165–174 (2019) 10, 13
17. Pham, Q.H., Nguyen, T., Hua, B.S., Roig, G., Yeung, S.K.: Jsis3d: Joint semantic-instance segmentation of 3d point clouds with multi-task pointwise networks and multi-value conditional random fields. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 8827–8836 (2019) 1
18. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 652–660 (2017) 1, 3, 4, 5
19. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: Advances in Neural Information Processing Systems (NIPS) (2017) 1, 9, 10
20. Song, S., Yu, F., Zeng, A., Chang, A.X., Savva, M., Funkhouser, T.: Semantic scene completion from a single depth image. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). IEEE (2017) 2
21. Tchapmi, L.P., Kosaraju, V., Rezatofighi, H., Reid, I., Savarese, S.: Topnet: Structural point cloud decoder. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 383–392 (2019) 10
22. Wang, Y., Tan, D.J., Navab, N., Tombari, F.: Forknet: Multi-branch volumetric semantic completion from a single depth image. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 8608–8617 (2019) 2, 3, 9, 13, 14
23. Wu, W., Qi, Z., Fuxin, L.: Pointconv: Deep convolutional networks on 3d point clouds. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 9621–9630 (2019) 1, 3
24. Yang, B., Rosa, S., Markham, A., Trigoni, N., Wen, H.: Dense 3d object reconstruction from a single depth view. IEEE transactions on pattern analysis and machine intelligence (2018) 2, 9
25. Yang, Y., Feng, C., Shen, Y., Tian, D.: Foldingnet: Point cloud auto-encoder via deep grid deformation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 206–215 (2018) 2, 3, 6, 9, 10, 11, 13, 14
26. Yuan, W., Khot, T., Held, D., Mertz, C., Hebert, M.: Pcn: Point completion network. In: 2018 International Conference on 3D Vision (3DV). pp. 728–737. IEEE (2018) 2, 3, 6, 9, 10, 11, 13, 14
27. Zhirong Wu, Song, S., Khosla, A., Fisher Yu, Linguang Zhang, Xiaoou Tang, Xiao, J.: 3d shapenets: A deep representation for volumetric shapes. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1912–1920 (2015) 11, 12

SoftPoolNet: Shape Descriptor for Point Cloud Completion and Classification

Yida Wang[1], David Joseph Tan[2], Nassir Navab[1], and Federico Tombari[1,2]

[1] Technische Universität München
[2] Google Inc.

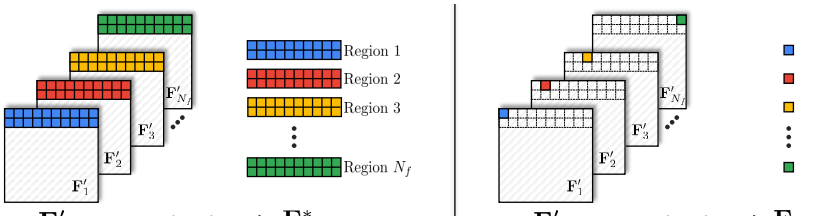
1 Comparison of SoftPoolNet to PointNet and PCN

Our architecture is composed by two parts: *encoder* and *decoder*. The encoder takes the partial scan as input. We process the partial scans with our novel soft pooling to produce the ordered feature F^*. Then, the decoder takes the feature F^* as input and apply our regional convolution twice to produce the point clouds with resolutions of 256 and 16,384 successively.

Notably, there are some similar components between our encoder and PointNet [1], as well as our decoder and PCN [3]. The following sections discuss the distinction in more detail.

1.1 Distinction of our encoder from PointNet

Each point on the cloud goes through the multi-layer perceptron (MLP) to accumulate the feature vectors into the matrix $\mathbf{F}$. Then, we sort the feature vectors in a descending order based on the k-th element of each vector. The sorted matrix is denoted as $\mathbf{F}'_i$. After independently sorting all N_f elements, we collect the matrices to form the tensor $\mathbf{F}'$ as shown in Fig. 1(a). We then build our softpool feature by taking the first N_r elements of each matrix and concatenate them to $\mathbf{F}^*$.



(a) *Soft*-pooling from the proposed (b) *Max*-pooling from PointNet

Fig. 1: Comparison between (a) our soft-pool operation and (b) max-pooling from PointNet, where the feature from PointNet is only a subset of our feature.

When comparing our softpool feature $\mathbf{F}^*$ with the feature from PointNet [1] denoted as $\mathbf{F}_{\text{PN}}$, PointNet executes a max-pooling operation on $\mathbf{F}'$ as illustrated in Fig. 1(b). Assuming that both features are derived from the same $\mathbf{F}'$ produced by an MLP, we can conclude that $\mathbf{F}_{\text{PN}}$ is a subset of our feature where

$$\mathbf{F}_{\text{PN}} = \left[\mathbf{F}'_1[1],\ \mathbf{F}'_2[2],\ \mathbf{F}'_3[3],\ \ldots\ \mathbf{F}'_{N_f}[N_f]\right] \tag{1}$$

only takes the one value of each matrix while our method takes the first N_r rows. Due to this, the dimensionality of the feature are then distinct. PointNet takes a vector with 1,024 values while we take $N_r \times N_f \times N_f$.

Notably, both our softpool feature and the PointNet feature are permutation invariant, which means that $\mathbf{F}^*$ and $\mathbf{F}_{\text{PN}}$ are the same irrelevant of the order of the input points. This is one of the most important aspect when handling point clouds since this data is unordered.

1.2 Distinction of our decoder from PCN

Based on our decoder architecture in the paper, the resulting feature from the encoder undergoes two successive regional convolution operations. The first converts the features to a course point cloud $\mathbf{P}'_{\text{out}}$ with 256 points. From there, the second regional convolution interpolates from the coarse to a fully-packed point cloud with 16,384 points which is denoted as $\mathbf{P}_{\text{out}}$.

Compared to PCN [3], both approaches execute a coarse-to-fine approach which is performed by our second regional convolution. However, the architecture and the method are different.

Given $\mathbf{P}'_{\text{out}}$, PCN [3] duplicates $\mathbf{P}'_{\text{out}}$ 64 times and appends a 2D coordinates of an 8×8 grid. Then, they use MLP to produce $\mathbf{P}_{\text{out}}$ that locally deforms the 2D grids around each point similar FoldingNet [2]. In contrast, we interpolate 63 samples between every 2 points of $\mathbf{P}'_{\text{out}}$ and use the proposed regional convolution to produce $\mathbf{P}_{\text{out}}$. Compared to MLP in PCN, our regional convolution takes more local samples into account to produce a point in the higher resolution.

2 Ablation study on the softpool feature $\mathbf{F}^*$

Using our architecture trained with $N_r = 32$, we present the qualitative results when only a subset of the rows is selected. The objective is to investigate which parts of the object each region reconstructs first. In Fig. 2, we start by limiting with the first two rows of the feature matrix then increasing N_r to reach 32. By selecting the first 2 features, we observe that the softpool feature focuses on a skeleton of the object without large surfaces. Although the regions reconstruct different parts of the object, they tend to cover the important components like the wings of plane and the wheels of car. As we increase N_r from 2 to 32, the object is slowly completed without huge overlaps between different regions.

In addition to the first 32 rows when setting N_r, we also looked into the rows beyond 32. The lamp in Fig. 3 focuses on the following ranges: [33 : 64],

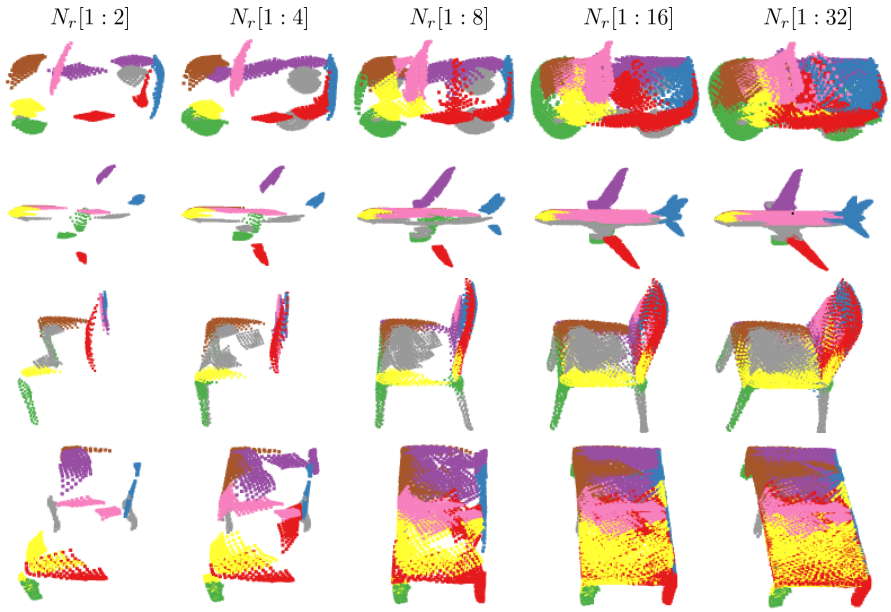


Fig. 2: Results when choosing the first subsets of N_r with the following ranges: $[1:2]$, $[1:4]$, $[1:8]$, $[1:16]$ and $[1:32]$ when the architecture is trained with $N_r = 32$.

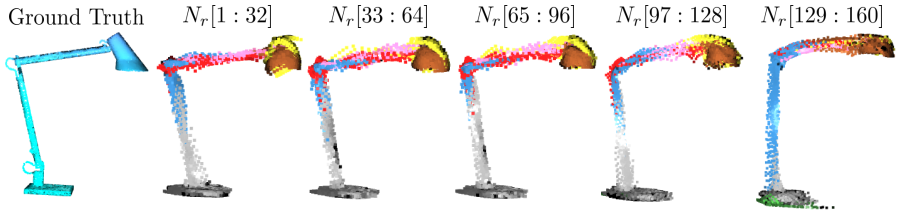


Fig. 3: Results when choosing different ranges of rows from $\mathbf{F}'$ to form $\mathbf{F}^*$ instead of selecting the first $N_r = 32$ rows.

$[65:96]$, $[97:128]$ and $[129:169]$. Although the shape of the lamp starts to deform as we go beyond 32, our reconstruction results still captures its overall shape even when we select the range $[129:169]$. Therefore, this proves that our feature is not constrained to the first 32 rows when sorting and demonstrates the robustness of our softpool feature.

3 Ablation study on τ

When computing for $\mathcal{L}_{\text{boundary}}$, we introduced the threshold τ to compute the sets. In Table 1, we then evaluate different values of τ and investigate its behavior with respect to the Chamfer distance. The table demonstrates that the results are not sensitive to the τ, where the thresholds between 0.2-0.9 generate a small difference in the Chamfer distance (with less than 1) from the chosen threshold of 0.3. Notably, compared to the related work, any threshold between 0.1 to 0.9 outperforms the other methods.

τ	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Chamfer Distance	7.08	5.99	**5.94**	6.12	6.19	6.18	6.21	6.25	6.71

Table 1: Sensitivity of the average Chamfer distance (multiplied by 10^3) to the threshold τ.

4 Ablation study on N_r, N_r and $\mathbf{L_{boundary}}$

We investigate the influence of increasing the weight of $\mathcal{L}_{\text{boundary}}$ on the reconstruction as we change the number of regions N_f and the number selected rows N_r. While we chose the best option with N_r set to 8 and N_r set to 32, Table 2 also shows that a larger weight on $\mathcal{L}_{\text{boundary}}$ improves the performance when the number of regions is larger, *e.g.* when N_f is 32.

(N_f, N_r)	(2, 128)	(4, 64)	(8, 32)	(16, 16)	(32, 8)
$1 \times \mathcal{L}_{\text{boundary}}$	7.80	6.31	5.94	6.27	6.75
$2 \times \mathcal{L}_{\text{boundary}}$	7.80	6.31	**5.91**	6.25	6.72
$10 \times \mathcal{L}_{\text{boundary}}$	7.82	6.29	5.95	6.01	6.19

Table 2: Influence of N_f, N_r and the weight of $\mathcal{L}_{\text{boundary}}$ for object completion on the average Chamfer distance (multiplied by 10^3).

References

1. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 652–660 (2017) 1, 2

2. Yang, Y., Feng, C., Shen, Y., Tian, D.: Foldingnet: Point cloud auto-encoder via deep grid deformation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 206–215 (2018) 2
3. Yuan, W., Khot, T., Held, D., Mertz, C., Hebert, M.: Pcn: Point completion network. In: 2018 International Conference on 3D Vision (3DV). pp. 728–737. IEEE (2018) 1, 2

5.2.2 SoftPool++: An Encoder-Decoder Network for Point Cloud Completion (International Journal of Computer Vision 2022)

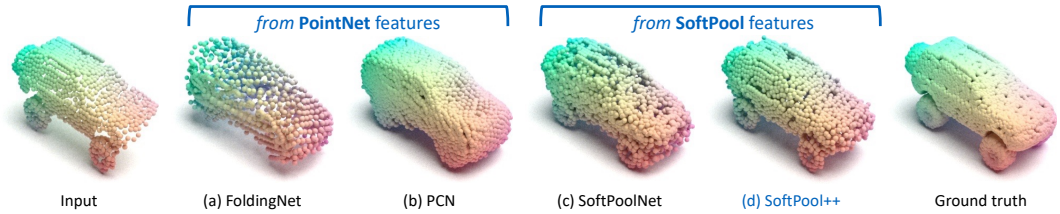


Figure 5.4 Our proposed SoftPool++ [5] shows local structures such as the tires better than approaches built upon PointNet [40] feature, while more input geometries are kept compared to our previous work of SoftPoolNet [6] contributed by the proposed point cloud skip-connection.

As introduced in Section 5.2.1, the input point cloud is described by a 2-dimensional feature map in SoftPoolNet, which is processed by MLP and soft pooling. The drawback of SoftPoolNet is the size of the SoftPool features. With a dimension of $N_r \times N_f \times N_f$, the latent feature cannot have a large N_f in practice because the memory footprint increases with the size of the feature, but the memory size of our off-the-shelf GPU constrains us. For example, SoftPoolNet sets the feature dimension N_f to a small value of 8, which is comparably much smaller than PointNet's latent feature. Such a small value of N_f may limit the decoder's power for completion.

In this work, we are interested in building a series of SoftPool features with larger dimensionalities in the latent space. Hence, we propose a *SoftPool++* operator to further truncate the SoftPoolNet features to $N_r \times N_f \times N_s$, where the third dimension takes the first N_s matrices in $\mathbf{F}^*$. As the pointwise dimensionality of the SoftPool++ feature could be much larger than the SoftPoolNet feature, we can increase it to 1,024, which matches the dimension of the PointNet feature. By replacing the PointNet [40] encoder in MSN [18] with our SoftPoolNet++ encoder, we show that the SoftPool++ feature supplements the MSN's decoder with more than separated local structures which match the ground truth more precise. This proves that *SoftPool++* makes our decoder capable of considering all the observable geometries to complete the shape. At the same time, the max-pooled PointNet feature cannot deal with geometric structures, which are rarely or not at all seen in the training data.

Additionally, we found a problem in existing works where the compression in the encoder tends to lose parts of the input shape structure, especially when latent features are too compact as a single vector, e.g. , the PointNet feature. Using skip connections specifically devised for point clouds, where links between corresponding layers in the encoder and the decoder are established. However, point cloud U-connection cannot be directly implemented by feature concatenation because the encoder and decoder features are in different domains. To solve this problem, we introduce a transformation matrix that projects the features from the encoder to the decoder and vice-versa.

Contributions

Yida Wang tackled the weakness of previous work (SoftPoolNet), where the observed structures are not guaranteed to get kept in the output for some samples. He implemented and evaluated the *U-connection* built for point cloud features between the encoder and decoder to solve this problem.

David Tan rephrased the methodology section and suggested a more proper way to visualize the results.

Federico Tombari revised this work, especially by building up an explicit connection between this work and existing works.

Nassir Navab financed this work on behalf of the leader of TUM CAMP.

SoftPool++: An Encoder–Decoder Network for Point Cloud Completion

Yida Wang[1], David Joseph Tan[2], Nassir Navab[1], Federico Tombari[1,2]

[1] Technical University of Munich
[2] Google

Published version: https://doi.org/10.1007/s11263-022-01588-7

International Journal of Computer Vision (2022) 130:1145–1164
https://doi.org/10.1007/s11263-022-01588-7

SoftPool++: An Encoder–Decoder Network for Point Cloud Completion

Yida Wang[1] · **David Joseph Tan**[2] · **Nassir Navab**[1] · **Federico Tombari**[1,2]

Received: 1 March 2021 / Accepted: 3 February 2022 / Published online: 11 March 2022

Abstract
We propose a novel convolutional operator for the task of point cloud completion. One striking characteristic of our approach is that, conversely to related work it does not require any max-pooling or voxelization operation. Instead, the proposed operator used to learn the point cloud embedding in the encoder extracts permutation-invariant features from the point cloud via a *soft-pooling* of feature activations, which are able to preserve fine-grained geometric details. These features are then passed on to a decoder architecture. Due to the compression in the encoder, a typical limitation of this type of architectures is that they tend to lose parts of the input shape structure. We propose to overcome this limitation by using skip connections specifically devised for point clouds, where links between corresponding layers in the encoder and the decoder are established. As part of these connections, we introduce a transformation matrix that projects the features from the encoder to the decoder and vice-versa. The quantitative and qualitative results on the task of object completion from partial scans on the ShapeNet dataset show that incorporating our approach achieves state-of-the-art performance in shape completion both at low and high resolutions.

Keywords Point cloud · Completion · SoftPool · Skip-connection

1 Introduction

Several data representations exist for 3D shapes. One common choice is the use of spatially discretized representations such as volumetric data (Yang et al. 2017; Wang et al. 2019b; Yang et al. 2018a). Alternative popular choices are implicit descriptions (Park et al. 2018; Chibane et al. 2020) as well as sparse 3D coordinate-based representations such as point clouds (Yang et al. 2018b; Xie et al. 2020b; Yuan et al. 2018) and 3D meshes (Groueix et al. 2018). Among this latter category of 3D data formats, point clouds are arguably the simplest, since they store 3D coordinates without any additional topological information such as faces or edges associated to the vertices. Hence, investigating how to process and learn 3D shape geometry based on these simple, yet effective representations is currently a hot research topic. This has recently motivated several tasks in 3D computer vision such as estimating point cloud deformation (Yang et al. 2018b; Yuan et al. 2018), registration (Aoki et al. 2019; Park et al. 2017), completion (Wang et al. 2020b; Groueix et al. 2018; Yuan et al. 2018), segmentation (Qi et al. 2017a; Lei et al. 2020; Xu et al. 2020) and 3D object detection (Shi et al. 2020; Qi et al. 2019).

This paper focuses on the point cloud completion task. The goal is to fill out occluded parts of the input 3D geometry represented by a partial scan, in a way that is coherent with the global shape while preserving fine local surface details. This is a useful task for many real world applications since occluded regions are normally present as part of most 3D data capture processes within, e.g., SLAM or multi-view reconstruction pipelines. State-of-the-art approaches targeting this task are based on neural networks and mostly rely on learning how to deform a set of 2D grids at different scales into 3D points, based on global shape descriptors typically represented by PointNet (Qi et al. 2017a) features. Examples of

Communicated by Akihiro Sugimoto.

Yida Wang
yida.wang@tum.de

David Joseph Tan
djtan@google.com

Nassir Navab
nassir.navab@tum.de

Federico Tombari
tombari@google.com

[1] Technische Universität München, Munich, Germany

[2] Google, Zurich, Switzerland

these approaches are FoldingNet (Yang et al. 2018b), AtlasNet (Groueix et al. 2018) and PCN (Yuan et al. 2018).

To overcome the aforementioned problem related to information loss due to feature compression at the level of the encoder–decoder bottleneck, GRNet (Xie et al. 2020b) suggests to preserve fine geometry details by discretizing the features via volumetric feature maps used at the different layers of the encoder. It also suggests using volumetric U-Net (Yang et al. 2018a) to build skip connections between the encoder and the decoder, eventually merging the obtained features with the input point cloud. The idea of leveraging skip connections among different layers of an encoder–decoder model follows the successful paradigm already exploited for volumetric shape completion, in particular 3D-RecGAN (Yang et al. 2017) and ForkNet (Wang et al. 2019b). While effective, converting sparse point cloud features into volumetric maps brings in all the disadvantages of discretized 3D data representations with respect to point clouds, in particular the loss of fine shape details, the inability to flexibly deal with local point density variations, as well as the unpractical trade-off between 3D resolution and memory occupancy.

Recently, we have demonstrated how, by means of sorting features based on their activations rather than applying max pooling, we can build up point clouds embeddings that store more informative features for a point cloud with respect to PointNet. This feature-learning approach, named SoftPool (Wang et al. 2020b), obtained state-of-the-art results for different point cloud-related tasks, such as completion and classification. In this work, we build up on our previous work (Wang et al. 2020b) to propose a more complete end-to-end framework. Our contributions are two-folds and are listed as follows:

1. We generalize our feature extraction technique into a module called SoftPool++. This module introduces *truncated softpool features* aimed to decrease the memory requirements of the original method during training, making it compatible with off-the-shelf GPUs. Notably, a disadvantage of the SoftPool features (Wang et al. 2020b) is that each point is processed independently from the rest. Due to this, the proposed module further processes the truncated softpool features with regional convolutions in order to recognize the relationships between the feature points. In contrast to Wang et al. (2020b) that applies their feature once, this module can be applied multiple times as demonstrated in our architecture, which uses it across multiple layers.
2. We propose a novel encoder–decoder architecture characterized by the use of point-wise skip connections. By connecting corresponding layers between encoder and decoder, this has the advantage of preserving fine geometric details from the given partial input cloud. This is to the best of our knowledge the first approach using skip connections for unorganized sets of 3D feature maps, relaxing the need of spatial discretization as deployed in Xie et al. (2020b), with benefits in terms of completion accuracy and memory occupancy. In addition, we also adapt the discriminator from TreeGAN (Shu et al. 2019) for the shape completion problem to further improve our model.

Our method is evaluated on ShapeNet (Chang et al. 2015) for the task of shape completion and on ModelNet (Zhirong et al. 2015) and PartNet (Mo et al. 2019) for the task of classification. Figure 1 illustrates a teaser of the shape completion results. It compares the architectures that are built on PointNet (Qi et al. 2017a) and SoftPool (Wang et al. 2020b) features. Visually, we show the advantage of the reconstructions that rely on SoftPool features as they are remarkably more similar to the ground truth. Moreover, the figure also highlights the improvements of SoftPool++ with respect to our previous approach (Wang et al. 2020b).

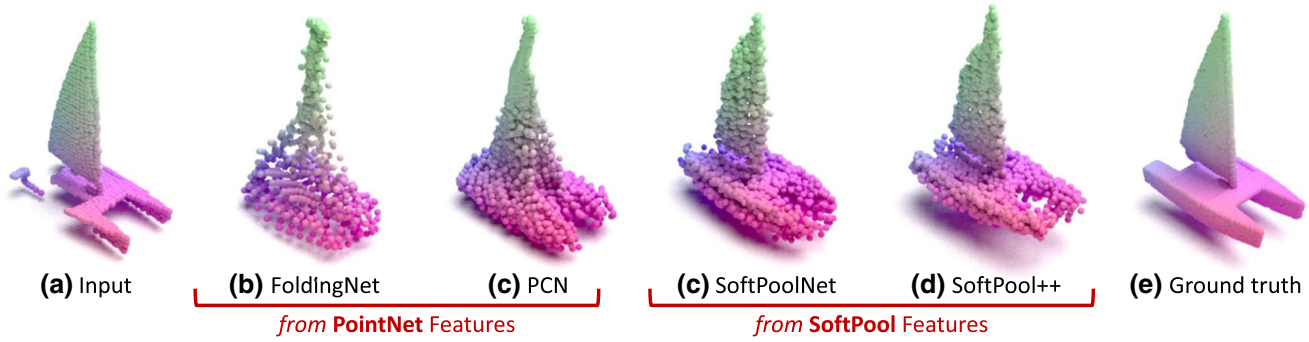


Fig. 1 Object completion results of the PointNet features such as FoldingNet (Yang et al. 2018b) and PCN (Yuan et al. 2018); and, the SoftPool features such as SoftPoolNet (Wang et al. 2020b) and the proposed SoftPool++

2 Related Work

Based on the focus of our contributions, we browse through the relevant methods in 3D object completion from partial scans and the use of skip connections with 3D data.

2.1 3D Object Completion

Inspired by the way humans perceive the 3D world from 2D projections, 3D-R2N2 (Choy et al. 2016) builds recurrent neural networks (RNNs) to fuse multiple feature maps extracted from input RGB images sequentially to recover the 3D geometries. To further improve the reconstruction, a coarse-to-fine 3D decoder was presented in Pix2Vox (Xie et al. 2019) as well as the residual refiner in Pix2Vox++ (Xie et al. 2020a). Due to the recent popularity of the attention mechanisms, AttSets (Yang et al. 2020) proposed to build attention layers to correlate the image features from different views. In contrast, our 3D reconstruction in this paper focuses on only a single depth image.

Taking a depth image of an object from an arbitrary camera pose, the objective of 3D object completion is to complete its missing structure and build its full reconstruction. Focusing on learning-based completion, most related work can be categorized depending on the input data they process—voxelized grid or point cloud. Interestingly, a notable work from OcCo (Wang et al. 2021a) demonstrates that the weights trained for completion are also valuable for other tasks like segmentation and classification.

Voxelized Grid Due to the popularity of 2D convolution operations in CNNs (Azad et al. 2019; Kirillov et al. 2020; Yang et al. 2020) for RGB images, its straightforward extension to 3D convolutions on volumetric data also rose to fame. 3D-EPN (Dai et al. 2017) and 3D-RecGAN (Yang et al. 2017) are the first works on this topic, where they extended the typical encoder–decoder architecture (Noh et al. 2015) to 3D. Adopting a similar architecture, 3D-RecGAN++ (Yang et al. 2018a) and ForkNet (Wang et al. 2019b) utilize adversarial training with 3D discriminator to improve the reconstruction.

The main advantage of volumetric completion is the structure of its data such that deep learning methods developed for RGB images can be extended to 3D. However, this advantage is also its limitation. The fixed local resolution makes it hard to reconstruct the object's finer details without consuming a huge amount of memory.

Point Cloud Having the inverse problem, point clouds have the potential to reconstruct the object at a higher resolutions but exhibited so far a limited application in deep learning due to its unstructured data. Note that, unlike RGB images or voxel maps, point clouds do not have a particular order, and the number of points varies as we change the camera pose or the object.

Targeted to solve the unordered structure of point clouds, PointNet (Qi et al. 2017a) proposes to implement max-pooling in order to achieve a permutation invariant latent feature. Based on this one dimensional feature, FoldingNet (Yang et al. 2018b) proposes an object completion solution that deforms a 2D rectangular grids by multi-layer perceptron (MLP). By increasing the number of 2D rectangular grids, AtlasNet (Groueix et al. 2018) and PCN (Yuan et al. 2018) added more complexity as well as details into the reconstruction. MSN (Liu et al. 2020) then further improves the completion by adding restrictions to separate different patches apart from each other. Moreover, Cycle4Completion (Wen et al. 2021) is also based on PointNet features but solves the problem by training with an unsupervised cycle transformation. Moving away from the global feature representation, PointNet++ (Qi et al. 2017b) samples the local subset of points with farthest point sampling (FPS) then feeds it into PointNet (Qi et al. 2017a). Based on this feature, PMP-Net (Wen et al. 2020b) completes the entire object gradually from the observed regions to the nearest occluded regions. SnowflakeNet (Xiang et al. 2021) also uses the PointNet++ features to split points in the coarsely reconstructed object to execute the completion progressively. In addition, building a similar feature as PointNet, ME-PCN (Gong et al. 2021) takes both the occupied and the empty regions on the depth image as input for 3D completion, showing the advantage of masking the empty regions in completion.

Unlike the methods which are dependent on a vectorized global feature to solve the permutation invariant problem, RFNet (Huang et al. 2021) and PointTr (Yu et al. 2021) produce several global features in their encoder. On one hand, RFNet (Huang et al. 2021) uses their features to complete the object in an recurrent way by concatenating the incomplete input and the predicted points level by level. On the other, PointTr (Yu et al. 2021) relies on transformers to produce a set of queries directly from the observed points with the help of positional coding. In effect, PointTr (Yu et al. 2021) does not need to compress the input into a single vector.

The recent work from PVD (Zhou et al. 2013), GRNet (Xie et al. 2020b) and VE-PCN (Wang et al. 2021b) leverage both the point cloud and the voxel grid representations. Unlike most works that rely on Chamfer distance to optimize the model, PVD (Zhou et al. 2013) uses a simple Euclidean loss to optimize the shape generation model from the voxelized point cloud representation. GRNet (Xie et al. 2020b) first voxelizes the point cloud, processes the voxel grid with deep learning and converts the results back to point cloud. While this solves the unorganized structure of the point clouds, its discretization removes its advantage on reconstructing in higher resolutions. VE-PCN (Wang et al. 2021b) improves the completion by supplementing the features of the decoder

in the volumetric completion with the edges. This method then converts the voxels to point clouds by Adaptive Instance Normalization (Lim et al. 2019).

Another solution is presented in our previous work SoftPoolNet (Wang et al. 2020b) that builds local groups of features by sorting them into a feature map. 2D convolutions are then applied to the feature map. Consequently, this approach is able to deal with unorganized point clouds and achieve reconstruction results at high resolution. We build upon SoftPoolNet (Wang et al. 2020b) and generalize the feature extraction into a module which we call SoftPool++. This then allows us to connect multiple modules in an encoder–decoder architecture. As a consequence, we achieve better quantitive and qualitative results.

2.2 Skip Connections in 3D

Skip connections were initially proposed for image processing (Mazaheri et al. 2019; Kim et al. 2016; Gao et al. 2019; Azad et al. 2019) then later adapted for 3D volumetric reconstruction (Yang et al. 2017, 2018a; Wang et al. 2019b). Given a point cloud as input, the methods like GRNet (Xie et al. 2020b) and InterpConv (Mao et al. 2019) require to convert the input point cloud to voxel grids.

Aiming at alleviating this limitation on point clouds, the work from Std (Yang et al. 2019) bypasses the encoder features into decoder point-by-point while GACNet (Wang et al. 2019a) constructs a graph from the points then constructs the skip connection with the graph. The problem of these point-wise skip connections is that new points cannot be introduced in the decoder. To solve this, SA-Net (Wen et al. 2020a) groups PointNet++ (Qi et al. 2017b) features in different resolutions with KNN. The skip connection from the encoder then matches the resolution of the decoder.

Contrary to these methods, in the context of object completion, the objective of our skip connection is compensate for the lost data in the encoder and bypass the observed geometry to the decoder. We also introduce the concept of feature transformation to compensate for the difference between the features from the encoder and decoder. Later in our evaluation, we found that the skip connection is a crucial step to achieve higher accuracy. Moreover, the SoftPool++ features also contribute to make our skip connection simpler. Since it is an organized feature, we avoid the time-consuming KNN, which significantly decreases our inference time.

3 Feature Extraction

Given the partial scan of an object, the input to our network is a point cloud with N_{in} points written in matrix form as $\mathbf{P}_{\text{in}} = [\mathbf{x}_i]_{i=1}^{N_{\text{in}}}$, where each point is represented as the 3D coordinates $\mathbf{x}_i = [x_i, y_i, z_i]$.

On one hand, the first objective of this section is to build a feature descriptor from the unorganized point cloud such that the feature remains the same for any permutation of the point cloud in $\mathbf{P}_{\text{in}}$. On the other hand, the second objective is to generalize this process into a feature extraction module that takes an arbitrary input $\mathbf{P}_{\text{in}}$. In this way, the proposed module can be implemented at multiple instances in our architecture.

3.1 SoftPool Feature

From the point cloud vector, we then convert each point into a feature vector $\mathbf{f}_i$ with N_f elements by projecting every point with a point-wise multi-layer perceptron (Qi et al. 2017a) with its parameters assembled in $\mathbf{W}_{\text{MLP}}$. Thus, we define the $N_{\text{in}} \times N_f$ feature matrix as $\mathbf{F} = [\mathbf{f}_i]_{i=1}^{N_{\text{in}}}$. Note that we applied a softmax function to the output neuron of the perceptron so that the elements in $\mathbf{f}_i$ range between 0 and 1.

Throughout this section, we refer to the *toy example* in Fig. 2 to visualize the various steps. This example assumes that there are only five points in the point cloud such that $N_{\text{in}} = 5$ as shown in Fig. 2a.

One of the main challenges in processing a point cloud is its unstructured arrangement. If we look at Fig. 2a, changing the order of the points in $\mathbf{P}_{\text{in}}$ reorganizes the rows of the feature map F. There is consequently no guarantee that the feature map remains constant for the same set of points. To solve this problem, we propose to organize the feature vectors in $\mathbf{F}$ so that their k-th elements are sorted in a descending order, which is denoted as $\mathbf{F}'_k$. Note that k should not be larger than N_f. This is demonstrated in Fig. 2a where we arrange the five feature vectors from $\mathbf{F} = [\mathbf{f}_i]_{i=1}^{5}$ to $\mathbf{F}'_k = [\mathbf{f}_i]_{i=\{3,5,1,2,4\}}$ by comparing the k-th element of each vector.

The features in SoftPoolNet (Wang et al. 2020b) repeat this process for all of the N_f elements in $\mathbf{f}_i$. Altogether, the feature is a 3D tensor with the dimension of $N_{\text{in}} \times N_f \times N_f$ denoted as $\mathbf{F}' = [\mathbf{F}'_1, \mathbf{F}'_2, \dots \mathbf{F}'_{N_f}]$ in Fig. 2b. Finally, we assemble the SoftPool features $\mathbf{F}^*$ by taking the N_r rows with the highest activations of all $\mathbf{F}'_i$ in $\mathbf{F}'$. Since each row in $\mathbf{F}'_i$ is equivalent to a point, we can then interpret the N_r rows of $\mathbf{F}'_i$ as one region in the point cloud, summing up to all N_f regions in $\mathbf{F}^*$.

Although both PointNet (Qi et al. 2017a) and SoftPoolNet (Wang et al. 2020b) utilize MLP in their architecture, they have significant differences on handling the results thereof. Compared to the max-pooling operation in PointNet (Qi et al. 2017a), the motivation of the SoftPool feature is to capture a larger amount of information and to further process it with regional convolution operations, as explained later in Sect. 4.

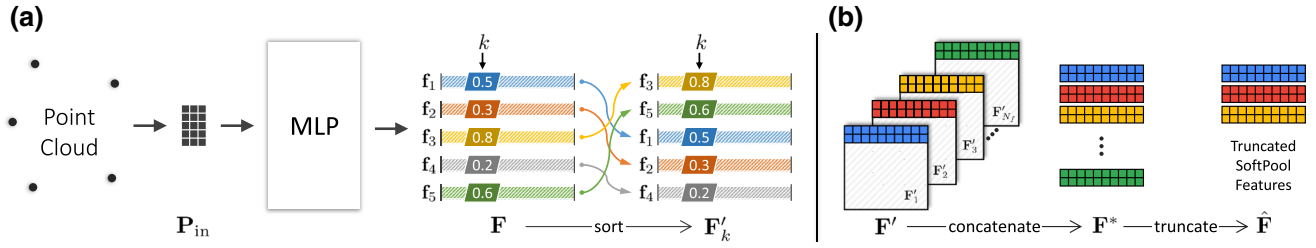


Fig. 2 Toy examples of the truncated SoftPool feature. Given 5 points in (**a**), they go through Multi-layer Perceptron (MLP) to produce $\mathbf{F}$. At the k-th element, the vectors are sorted to build $\mathbf{F}'_k$ and consequently $\mathbf{F}'$. In (**b**), we concatenation of the first N_r rows of $\mathbf{F}'_k$ to construct the 3D tensor $\mathbf{F}^*$ which corresponds to the regions with high activations then truncated to assemble $\hat{\mathbf{F}}$

3.2 Generalizing and Truncating the SoftPool Feature

In practice, we noticed that we can generalize the SoftPool feature formulation to an arbitrary input feature $\mathbf{P}_{\text{in}}$—thus, alleviating the definition of points—to produce the SoftPool features $\mathbf{F}'$. From this perspective, we can construct an architecture with a series of SoftPool feature extractions. Therefore, we take the point cloud as the input to the architecture and extract the first SoftPool features. Then, after processing the first features, we can then extract the second features from them and so on. This is discussed later in Sect. 4 with an encoder–decoder architecture.

However, the drawback of such architecture is the size of the SoftPool features. With a dimension of $N_r \times N_f \times N_f$, the memory footprint increases with the size of the feature but we are constrained by the memory size of our off-the-shelf GPU. Notably, in Wang et al. (2020b), they set the feature dimension N_f to a small value of 8. In this work, since we are interested in building a series of SoftPool features in an encoder–decoder architecture, N_f increases up to 256 in the latent space.

Hence, we propose to further truncate the SoftPool features to $N_r \times N_f \times N_s$, where the third dimension takes the first N_s matrices in $\mathbf{F}^*$ as illustrated in Fig. 2b. To distinguish from Wang et al. (2020b), we refer this as the *Truncated SoftPool feature*, denoted as $\hat{\mathbf{F}}$ in Fig. 2b.

3.3 Regional Convolutions

Considering that each point in the cloud independently goes through MLP while the operations thereafter to produce the truncated SoftPool features rely on sorting, each row of our feature remains independent from each other. However, in contrast to max-pooling which produces a vector, our feature is a 3D tensor which can undergo convolutional operations.

Instead of applying the same kernel to all regions as Wang et al. (2020b), we generalize the regional convolutions and impose distinct kernels for each region. We first split $\hat{\mathbf{F}} = [\hat{\mathbf{F}}_r]_{r=1}^{N_s}$ into separate regions $\hat{\mathbf{F}}_r$ and correspondingly apply a set of kernels $\mathbf{W}_{\text{conv}} = \{\mathbf{W}_r\}_{r=1}^{N_s}$. Assigning the concatenated output tensor as $\mathbf{F}_{\text{out}} = [\mathbf{P}_r]_{r=1}^{N_s}$, we can formally describe this operation as

$$\mathbf{P}_r(i, j) = \sum_{l=1}^{N_f} \sum_{k=1}^{N_k} \hat{\mathbf{F}}_r(i + k, l)\mathbf{W}_r(j, k, l) \tag{1}$$

for the r-th region.

The dimension of each kernel is $N_k \times N_f \times N_{\text{out}}$, where N_k indicates the number of neighbors to consider and N_{out} is the desired size of the output $\mathbf{P}_r$. Note that the kernels convolves on the entire width of $\hat{\mathbf{F}}_r$, i.e.corresponding to its width N_f. This implies that we only pad on the vertical axis. Similar to other convolutional operators, the stride s distinguishes between a convolutional and deconvolutional operation. If the stride is greater than 1, $\hat{\mathbf{F}}_r$ is downsampled, while it is upsampled if the stride is less than 1.

3.4 SoftPool++ Module

Now, we have all the components to build the feature extraction module as shown in Fig. 3, which we call SoftPool++. Since $\mathbf{P}_{\text{in}}$ is defined as the input point cloud, we generalize the input of the module as $\mathbf{F}_{\text{in}}$ where we set $\mathbf{F}_{\text{in}} = \mathbf{P}_{\text{in}}$ in the first layer. Hence, the input matrix $\mathbf{F}_{\text{in}}$ goes through a 3-layer perceptron then builds the truncated SoftPool features. Thereafter, we perform regional convolution and reshape the results by squeezing the third dimension to finally acquire our output feature matrix $\mathbf{F}_{\text{out}}$.

Fig. 3 Overview of the feature extraction module called SoftPool++

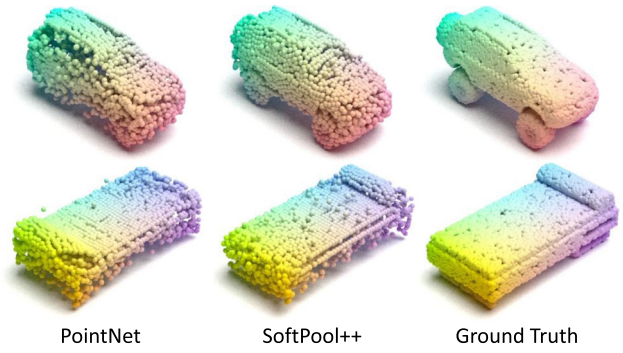


Fig. 4 Object completion results with MSN (Liu et al. 2020) while using PointNet (Qi et al. 2017a) features and SoftPool++ features on its encoder

When constructing our architecture in Sect. 4, the encoder and decoder are distinguished primarily on the stride s. In this paper, we show the versatility of this novel module to act as an encoder and decoder as well as to refine a coarse point cloud with more elaborate details.

The differences between decoding from PointNet features and SoftPool++ features are evident in Fig. 4, where we replace the PointNet feature in MSN (Liu et al. 2020) with a SoftPool++ feature with the same size of 1024. By replacing the PointNet (Qi et al. 2017a) encoder in MSN (Liu et al. 2020) with our SoftPoolNet++ encoder, we show that the SoftPool++ feature supplements the MSN's decoder where all the wheels are clearly separated from the body of the SUV, while the original PointNet feature in MSN follow the more generic structure of a vehicle with tiny gaps between wheel and body. This proves that *SoftPool++* makes our decoder able to take all observable geometries into account to complete the shape, while the max-pooled PointNet feature cannot deal with geometric structures which are rarely or not at all seen in the training data.

4 Network Architecture

The volumetric U-Net (Çiçek et al. 2016; Yang et al. 2018a) in 3D-RecGAN (Yang et al. 2017) and GRNet (Xie et al. 2020b) has shown significant improvements in object completion as it injects more data from the encoder to the decoder in order to supplement the compressed latent feature. Without the skip connection in U-Net, we end up losing most of the input data as it goes through the encoder. Consequently, the decoder starts hallucinating the overall structure without being faithful to the given information. Inspired by this idea, we introduce a novel U-Net connection that directly takes the point cloud as input, i.e.without the need of voxelization at any stage of the network.

Our network architecture is composed of an encoder–decoder structure with a skip connection as shown in Fig. 5. Such connection between encoder and decoder makes the completion more likely to preserve input geometries. The encoder is composed of consecutive feature extraction modules from Sect. 3.4 to downsample the input to the latent feature while the decoder is composed of the similar feature extraction modules to upsample to the output. As discussed in Sect. 3.4, the stride s is a significant parameter to distinguish the two layers. Table 1 lists the values of all the parameters for the module in the convolution and deconvolution layers.

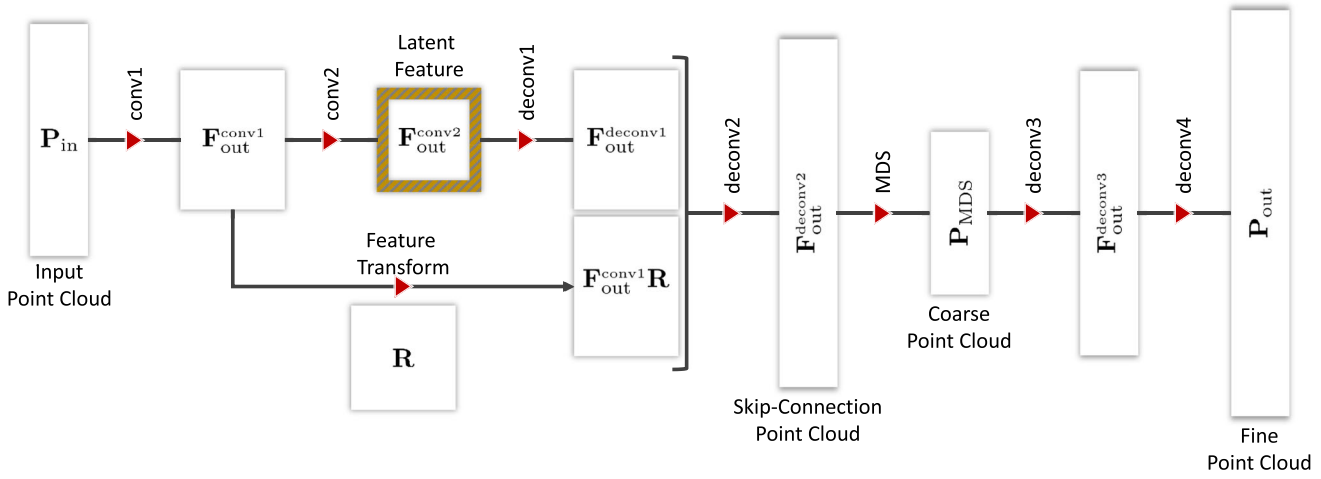


Fig. 5 Overview of our object completion architecture where the parameters for the convolution and deconvolution operations based on the feature extraction module are listed in Table 1. Note that, in our evaluation, we compare three point cloud results from the decoder: (1) the skip-connection output; (2) the coarse output; and, (3) the fine-grained output which is the final reconstruction

Table 1 Dimensions and parameters on each feature extraction module in our architecture

Feature extraction module	Conv1	Conv2	Deconv1	Deconv2	MDS	Deconv3	Deconv4
Input variable	$\mathbf{P}_{\text{in}}$	$\mathbf{F}_{\text{out}}^{\text{conv1}}$	$\mathbf{F}_{\text{out}}^{\text{conv2}}$	$[\mathbf{F}_{\text{out}}^{\text{deconv1}}, \mathbf{F}_{\text{out}}^{\text{conv1}}\mathbf{R}]$	$\mathbf{F}_{\text{out}}^{\text{deconv2}}$	$\mathbf{P}_{\text{MDS}}$	$\mathbf{F}_{\text{out}}^{\text{deconv3}}$
Number of rows	N_{in}	$N_{\text{in}}/8$	256	$512 + N_{\text{in}}/8$	$1024 + N_{\text{in}}/4$	2048	4096
Number of columns	3	256	256	256	3	3	3
Stride (s)	8	2	1/2	1/2	–	1/2	1/4
Feature dimension (N_f)	256	256	256	256	3	256	256
Number of rows in the region (N_r)	N_{in}	32	N_{in}	N_{in}	–	N_{in}	N_{in}
Truncation size (N_s)	1	8	1	1	–	1	1
Kernel size (N_k)	8	8	4	4	–	4	4
Dimension of output feature (N_{out})	256	256	256	3	3	3	3
Output variable	$\mathbf{F}_{\text{out}}^{\text{conv1}}$	$\mathbf{F}_{\text{out}}^{\text{conv2}}$	$\mathbf{F}_{\text{out}}^{\text{deconv1}}$	$\mathbf{F}_{\text{out}}^{\text{deconv2}}$	$\mathbf{P}_{\text{MDS}}$	$\mathbf{F}_{\text{out}}^{\text{deconv3}}$	$\mathbf{P}_{\text{out}}$
Number of rows	$N_{\text{in}}/8$	256	512	$1024 + N_{\text{in}}/4$	2048	4096	16,384
Number of columns	256	256	256	3	3	3	3

Note that the input to the architecture is the point cloud $\mathbf{P}_{\text{in}}$ with a dimension of $N_{\text{in}} \times 3$ while the output is another point cloud $\mathbf{P}_{\text{out}}$ with $16{,}384 \times 3$

Skip Connection with Feature Transform We bridge the encoder and decoder with a skip connection to build a U-Net-like (Çiçek et al. 2016; Yang et al. 2018a) structure. This connection links the results of conv1, denoted as $\mathbf{F}_{\text{out}}^{\text{conv1}}$, to the results of deconv1, denoted as $\mathbf{F}_{\text{out}}^{\text{deconv1}}$. However, instead of simply concatenating them, we introduce a square matrix $\mathbf{R}$ that transforms the features from the encoder as $\mathbf{F}_{\text{out}}^{\text{conv1}}\mathbf{R}$. Note that the multiplication by the transform is on the right side because the points are arranged row-wise in $\mathbf{P}_{\text{in}}$, which implies that the feature vectors are also arranged row-wise. Subsequently, we concatenate the two matrices into $[\mathbf{F}_{\text{out}}^{\text{deconv1}}, \mathbf{F}_{\text{out}}^{\text{conv1}}\mathbf{R}]$ that serves as the input to the feature extraction module, producing $\mathbf{F}_{\text{out}}^{\text{deconv2}}$.

In order to avoid randomly large values in the transformations and attain numerical stability during training, we regularize the transformation matrix to be orthonormal such that all elements are between $[-1, 1]$ and it mathematically satisfies $\mathbf{R}\mathbf{R}^\top = \mathbf{I}$ where $\mathbf{I}$ is an identity matrix. Geometrically, the regularizer imposes to rotate the features by $\mathbf{R}$.

Minimum Density Sampling Since the number of points in the input cloud vary, the results of deconv2 would also produce a varying number of points, i.e.with $1024 + \frac{N_{\text{in}}}{4}$ points from Table 1, since it depends on the input dimension. Thus, we include a Minimum Density Sampling (MDS) (Liu et al. 2020) in the decoder to standardize the output to a coarse resolution of 2048 points. The coarse resolution is then refined with two deconvolutional operations to 16,384 points. The motivation of adding the MDS is to help the final deconvolutional layers to converge faster during training. Later in Sect. 6, we investigate further the differences between the point clouds from the skip-connection as well as the coarse and fine as illustrated in Fig. 5.

5 Loss Functions

Since the main goal here is point cloud completion (Groueix et al. 2018; Yang et al. 2018b; Yuan et al. 2018), we first analyse whether the predicted point feature $\mathbf{P}_{\text{out}}$ matches the given ground truth $\mathbf{P}_{\text{gt}}$ through the Chamfer distance

$$\mathcal{L}_{\text{complete}} = \text{Chamfer}(\mathbf{P}_{\text{out}}, \mathbf{P}_{\text{gt}}) \,. \tag{2}$$

Furthermore, we optimize our architecture with two sets of loss functions that are related to the feature extraction module for all the convolution and deconvolution layers in the architecture from Sect. 3.4 as well as the skip connection with the feature transform from Sect. 4.

5.1 Optimizing the Feature Extraction Module

For the feature extraction module that utilizes SoftPool features, we adopt the same loss terms as in Wang et al. (2020b), where their main objective is to optimize the distribution of the features across different regions.

Intra-regional Entropy The ideal case for the feature vector $\mathbf{f}_i$ is a one-hot code, i.e. each vector gets assigned to only one region. To accomplish this goal, we measure the probability of $\mathbf{f}_i$ belonging to region k in all N_s regions by directly applying the softmax on the elements of the vector as

$$P(\mathbf{f}_i, k) = \frac{e^{\mathbf{f}_i[k]}}{\sum_{j=1}^{N_s} e^{\mathbf{f}_i[j]}} \,. \tag{3}$$

This implies that P is maximized when $\mathbf{f}_i$ is a one-hot code, with the k-th element equal to one. However, in presence of multiple peaks in the vector, $P(\mathbf{f}_i, k)$ might decrease significantly. Therefore, by taking the entropy into account, the

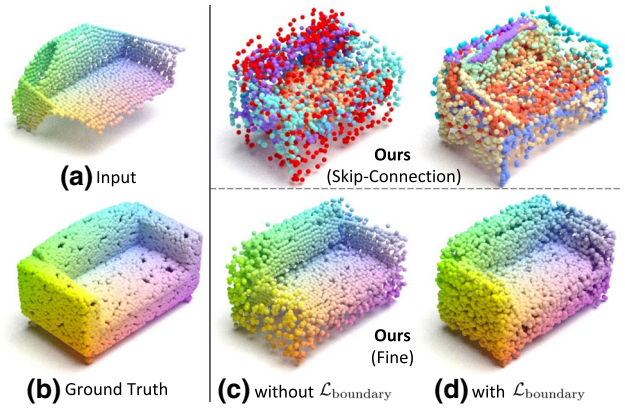


Fig. 6 Our object completion results with and without the influence of $\mathcal{L}_{\text{boundary}}$

intra-regional loss function

$$\mathcal{L}_{\text{intra}} = -\frac{1}{N_{\text{in}}}\frac{1}{B}\sum_{i=1}^{N_{\text{in}}}\sum_{j=1}^{B}\sum_{k=1}^{N_s} P(\mathbf{f}_i, k)\log P(\mathbf{f}_i, k) \tag{4}$$

where B is the batch size, tries to enforce the feature vector to have one peak so that it confidently falls into just one region.

Inter-regional Entropy The drawback of $\mathcal{L}_{\text{intra}}$ is that all feature vectors have the same peak at the k-th element. Looking at a more holistic perspective, the inter-regional loss function aims at distributing the features across different regions. It relies on maximizing the regional entropy

$$\mathcal{E}_r = -\frac{1}{B}\sum_{j=1}^{B}\sum_{k=1}^{N_s}\bar{P}_k \cdot \log \bar{P}_k \tag{5}$$

given that

$$\bar{P}_k = \frac{1}{N_{\text{in}}}\sum_{i=1}^{N_{\text{in}}} P(\mathbf{f}_i, k)\ . \tag{6}$$

We can then define the loss function as

$$\mathcal{L}_{\text{inter}} = \log(N_s) - \mathcal{E}_r \tag{7}$$

since the upper-bound of $\mathcal{E}_r$ is computed as $-\log\frac{1}{N_s}$ or simply $\log(N_s)$.

Boundary Overlap Minimization In addition to optimizing the holistic distribution of the points, we also incorporate a loss function that is applied on pairs of regions i and j. We collect a set of points $\mathcal{B}_j^i$ from region i with activations of region j larger than a threshold τ, i.e.set to 0.3. Similarly, we also take the inverse $\mathcal{B}_i^j$. Consequently, we squeeze the overlaps between the two regions.

By minimizing the Chamfer distance between $\mathcal{B}_j^i$ and $\mathcal{B}_i^j$, we obtain the loss

$$\mathcal{L}_{\text{boundary}} = \sum_{i=1}^{N_s}\sum_{j=i}^{N_s}\text{Chamfer}(\mathcal{B}_i^j, \mathcal{B}_j^i) \tag{8}$$

that tries to make the overlapping sets of points smaller, ideally down to just a line. In Fig. 6, we visualize the difference of optimizing with and without $\mathcal{L}_{\text{boundary}}$, where the distribution of the point cloud is less noisy on the occluded regions such as the armrest.

Notably, this loss function is general enough to be effectively applied also on other methods that rely on a subdivision of the point cloud into different regions, such as AtlasNet (Groueix et al. 2018), PCN (Yuan et al. 2018) and MSN (Liu et al. 2020). In Sect. 7.2, we formally evaluate these methods with and without $\mathcal{L}_{\text{boundary}}$.

Feature Duplicate Minimization The last loss term

$$\mathcal{L}_{\text{preserve}} = \text{Earth-moving}(\hat{\mathbf{F}}, \mathbf{F}) \tag{9}$$

imposes that the resulting truncated SoftPool feature $\hat{\mathbf{F}}$ takes most of the features from original $\mathbf{F}$ so that it avoids duplicates. To make the earth moving distance (Li et al. 2013) more efficient, 256 vectors are randomly selected from $\mathbf{F}$ and $\hat{\mathbf{F}}$. In practice, Fig. 7 visualizes the effects of $\mathcal{L}_{\text{preserve}}$ in the reconstruction, where lower weights of this loss produce a large hole, while incorporating this loss builds a point cloud with similar densities.

5.2 Optimizing the Skip Connection

We first visualize a subset of the architecture and focus on the skip connection as shown in Fig. 8. Here, we define $\mathbf{P}_{\text{partial}}$ as the partial reconstruction on $\mathbf{F}_{\text{out}}^{\text{deconv2}}$ contributed by the skip connection with the feature transform. However, note that $\mathbf{P}_{\text{partial}}$ is not a subset of $\mathbf{F}_{\text{out}}^{\text{deconv2}}$. It is produced by taking the input point cloud through conv1, feature transform and deconv2.

Since the skip connection aims to maintain the given input structure, we define a loss function that acts as an auto-encoder such that

$$\mathcal{L}_{\text{skip}} = \text{Chamfer}(\mathbf{P}_{\text{partial}}, \mathbf{P}_{\text{in}})\ . \tag{10}$$

In addition, based on Sect. 4, we regularize the values in the feature transform such that

$$\mathcal{L}_{\mathbf{R}} = \|\mathbf{R}\mathbf{R}^\top - \mathbf{I}\|^2 \tag{11}$$

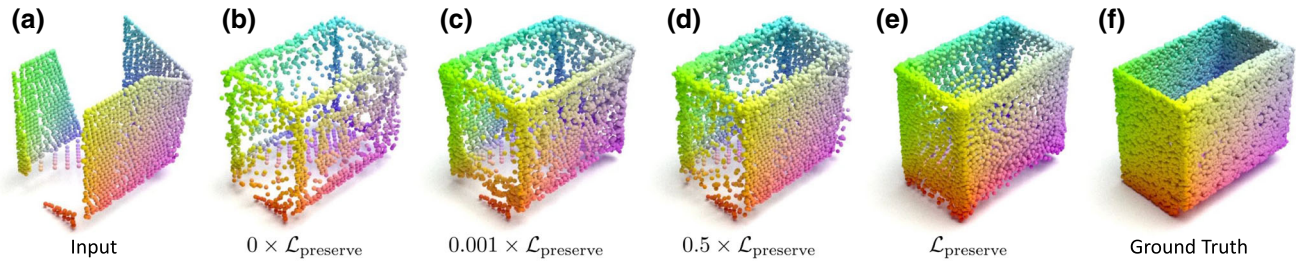


Fig. 7 Our object completion results while increasing the weight of $\mathcal{L}_{\text{preserve}}$

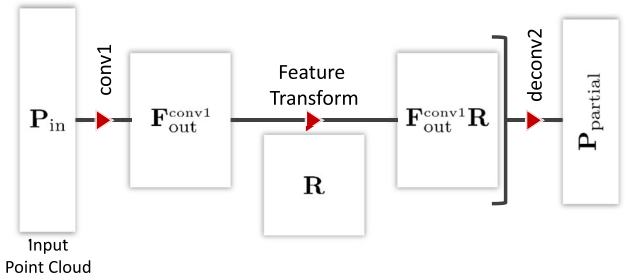


Fig. 8 Subset of the architecture that focuses on the skip connection

makes $\mathbf{R}$ orthonormal.

5.3 Discriminative Training

Recognizing the advantages from TreeGAN (Shu et al. 2019), we also investigate applying discriminative training conditions on the input partial scan $\mathbf{P}_{\text{in}}$. In this case, we first introduce the conditional feature maps $\mathbf{P}_{\text{out}}|\mathbf{P}_{\text{in}}$ and $\mathbf{P}_{\text{gt}}|\mathbf{P}_{\text{in}}$ by concatenating them along the point axis. We build our discriminator $\mathcal{D}$ with the same parametric model proposed in Shu et al. (2019). By restricting the output of the discriminator to a range between 0 and 1, we can then apply

$$\mathcal{L}_{\text{infer}} = -\log(\mathcal{D}(\mathbf{P}_{\text{out}}|\mathbf{P}_{\text{in}})) \,, \tag{12}$$

to optimize our completion architecture while

$$\mathcal{L}_{\text{discri}} = -\log(\mathcal{D}(\mathbf{P}_{\text{gt}}|\mathbf{P}_{\text{in}})) - \log(1 - \mathcal{D}(\mathbf{P}_{\text{out}}|\mathbf{P}_{\text{in}})) \tag{13}$$

to optimize the discriminator $\mathcal{D}$. In practice, we impose the loss functions in (12) and (13) alternatively in order to optimize the completion architecture and the discriminators separately.

6 Experiments

For all evaluations, we train our model with an NVIDIA Titan V and parameterize it with a batch size of 8. Moreover, we apply the Leaky ReLU with a negative slope of 0.2 on the output of each regional convolution.

6.1 Completion on ShapeNet

We evaluate the performance of the geometric completion of a single object on the ShapeNet (Chang et al. 2015) database where they have the point clouds of the partial scanning as input and the corresponding ground truth completed shape. To make it comparable to other approaches, we adopt the standard 8 category evaluation (Yuan et al. 2018) for a single object completion. Both sampled from ShapeNet meshes, PCN (Yuan et al. 2018) and TopNet (Tchapmi et al. 2019) supplement two set of datasets individually for low and high resolutions evaluation, which contain 2048 and 16,384 points, respectively, where the inputs are provided with 2048 points. Notice that the low resolution dataset provided by TopNet is also commonly referred to Completion3D benchmark. Since previous works report their results in terms of L1/L2 metric of the Chamfer distance separately, we also report our results in both resolutions (2048 and 16,384) and metrics (L1 and L2).

We compare against state-of-the-art point cloud completion approaches such as PCN (Yuan et al. 2018), FoldingNet (Yang et al. 2018b), AtlasNet (Groueix et al. 2018), PointNet++ (Qi et al. 2017b), MSN (Liu et al. 2020) and GRNet (Xie et al. 2020b). To show the advantages over volumetric completion, we also compare against 3D-EPN (Dai et al. 2017) and ForkNet (Wang et al. 2019b) with an output resolution of $64 \times 64 \times 64$. As for point cloud resolutions, PCN (Yuan et al. 2018), GRNet (Xie et al. 2020b) and SoftPoolNet (Wang et al. 2020b) report the best performance with 16,384 points while MSN (Liu et al. 2020) presents their final output resolution with 8192 points. Aiming at a fair numerical comparison at different resolutions, we modify the last layers of these architectures so as to attain the same resolution for all methods.

Low Resolution At low resolution, we achieve competitive results, attaining the 0.13×10^{-4} from PMP-Net (Wen et al. 2020b) with the L2-Chamfer distance in Table 2, while we achieve state-of-the-art results when evaluating on the L1-Chamfer distance in Table 3.

Table 2 Evaluation on the object completion based on the Chamfer distance trained with L2 distance (multiplied by 10^4) with the output resolution of 2048

Method	Plane	Cabinet	Car	Chair	Lamp	Sofa	Table	Vessel	Avg.
Completion3D (Tchapmi et al. 2019) benchmark, Output Resolution = 2048, L2 metric									
FoldingNet (Yang et al. 2018b)	12.83	23.01	14.88	25.69	21.79	21.31	20.71	11.51	19.07
PointSetVoting (Zhang et al. 2020a)	6.88	21.18	15.78	22.54	18.78	28.39	19.96	11.16	18.18
AtlasNet (Groueix et al. 2018)	10.36	23.40	13.40	24.16	20.24	20.82	17.52	11.62	17.77
PCN (Yuan et al. 2018)	9.79	22.70	12.43	25.14	22.72	20.26	20.27	11.73	18.22
TopNet (Tchapmi et al. 2019)	7.32	18.77	12.88	19.82	14.60	16.29	14.89	8.82	14.25
GRNet (Xie et al. 2020b)	6.13	16.90	8.27	12.23	10.22	14.93	10.08	5.86	10.64
SA-Net (Wen et al. 2020a)	5.27	14.45	7.78	13.67	13.53	14.22	11.75	8.84	11.22
PMP-Net (Wen et al. 2020b)	**3.99**	**14.70**	**8.55**	**10.21**	**9.27**	**12.43**	**8.51**	**5.77**	**9.23**
SoftPoolNet (Wang et al. 2020b)	6.39	17.26	8.72	13.16	10.78	14.95	11.01	6.26	11.07
Ours	4.59	15.82	6.78	11.41	8.82	13.37	9.15	4.93	9.36
Without skip-connection	4.63	16.35	9.10	13.40	10.55	13.85	10.90	6.23	10.63
Without $\mathcal{D}$	5.07	16.12	6.86	11.56	8.88	13.67	9.21	5.33	9.59
Without $\mathcal{L}_{\mathrm{R}}$	5.38	17.04	9.93	14.13	11.35	14.52	11.63	6.81	11.35

Bold indicates the best performance achieved in certain column

Table 3 Evaluation on the object completion based on the Chamfer distance trained with L1 distance (multiplied by 10^4) with the output resolution of 2048

Method	Plane	Cabinet	Car	Chair	Lamp	Sofa	Table	Vessel	Avg.
Completion3D (Tchapmi et al. 2019) benchmark, output resolution = 2048, L1 metric									
FoldingNet (Yang et al. 2018b)	11.18	20.15	13.25	21.48	18.19	19.09	17.80	10.69	16.48
AtlasNet (Groueix et al. 2018)	10.37	23.40	13.41	24.16	20.24	20.82	17.52	11.62	17.69
AtlasNet + $\mathcal{L}_{\mathrm{boundary}}$	9.25	22.51	12.12	22.64	18.82	19.11	16.50	11.53	16.56
PCN (Yuan et al. 2018)	8.09	18.32	10.53	19.33	18.52	16.44	16.34	10.21	14.72
PCN + $\mathcal{L}_{\mathrm{boundary}}$	6.39	16.32	9.30	18.61	16.72	16.28	15.29	9.00	13.49
TopNet (Tchapmi et al. 2019)	5.50	12.02	8.90	12.56	9.54	12.20	9.57	7.51	9.72
SA-Net (Wen et al. 2020a)	**2.18**	**9.11**	**5.56**	**8.94**	9.98	**7.83**	9.94	7.23	7.74
SoftPoolNet (Wang et al. 2020b)	4.76	10.29	7.63	11.23	8.97	10.08	7.13	6.38	8.31
Ours	3.50	9.95	7.01	10.48	**8.45**	8.86	**5.99**	**5.60**	**7.48**
Without skip-connection	4.29	10.24	7.76	11.10	9.13	9.72	6.33	6.46	8.13
Without $\mathcal{D}$	3.72	10.07	7.23	10.76	8.50	9.15	6.10	5.92	7.68
Without $\mathcal{L}_{\mathrm{R}}$	4.68	10.54	8.06	11.42	9.45	10.03	6.70	6.77	8.46
Without $\mathcal{L}_{\mathrm{inter}}$	9.81	19.49	13.24	18.20	16.83	17.00	15.64	7.16	14.67
Without $\mathcal{L}_{\mathrm{intra}}$	3.70	15.93	10.78	12.97	12.89	11.79	11.33	5.60	10.62
Without $\mathcal{L}_{\mathrm{inter}}$, $\mathcal{L}_{\mathrm{intra}}$	9.07	19.62	14.31	19.17	16.46	17.82	14.34	7.60	14.80
Without $\mathcal{L}_{\mathrm{boundary}}$	4.71	10.43	8.14	11.27	9.27	10.57	7.43	7.36	8.65
Without $\mathcal{L}_{\mathrm{preserve}}$	7.43	18.84	11.58	15.68	17.38	17.53	14.46	6.49	13.68

Bold indicates the best performance achieved in certain column

High Resolution We achieve the best results on most objects with the high resolution as presented in Tables 4 and 5 with 8.31×10^{-3} and 2.55×10^{-3}, respectively. Table 5 also shows that volumetric approaches like 3D-EPN (Dai et al. 2017) and ForkNet (Wang et al. 2019b) having large issues when evaluated in Chamfer distance because the converted point clouds from the fixed volumetric grids are at much smaller local resolutions.

Validating with F-Score@1% Since the Chamfer distance hardly reflect the errors in the local geometry as suggested in Tatarchenko et al. (2019), the evaluation in GRNet (Xie et al. 2020b) uses the metric F-Score@1% that computes the F-Score after matching the predicted point cloud to the ground truth with a distance threshold of 1% of the side length of the reconstructed volume. The evaluations on reconstructing higher resolutions are reported in Tables 6 and 7 on ShapeNet

Table 4 Evaluation on the object completion based on the Chamfer distance trained with L1 distance (multiplied by 10^3) with the output resolution of 16,384

Method	Plane	Cabinet	Car	Chair	Lamp	Sofa	Table	Vessel	Avg.
PCN (Yuan et al. 2018) dataset, Output Resolution = 16,384, L1 metric									
3D-EPN (Dai et al. 2017)	13.16	21.80	20.31	18.81	25.75	21.09	21.72	18.54	20.15
ForkNet (Wang et al. 2019b)	9.08	14.22	11.65	12.18	17.24	14.22	11.51	12.66	12.85
PointNet++ (Qi et al. 2017b)	10.30	14.74	12.19	15.78	17.62	16.18	11.68	13.52	14.00
FoldingNet (Yang et al. 2018b)	9.49	15.80	12.61	15.55	16.41	15.97	13.65	14.99	14.31
AtlasNet (Groueix et al. 2018)	6.37	11.94	10.11	12.06	12.37	12.99	10.33	10.61	10.85
TopNet (Tchapmi et al. 2019)	7.61	13.31	10.90	13.82	14.44	14.78	11.22	11.12	12.15
PCN (Yuan et al. 2018)	5.50	10.63	8.70	11.00	11.34	11.68	8.59	9.67	9.64
PCN + $\mathcal{L}_{\text{boundary}}$	5.13	9.12	7.58	9.35	9.40	9.31	7.30	8.91	8.26
MSN (Liu et al. 2020)	5.60	11.96	10.78	10.62	10.71	11.90	8.70	9.49	9.97
GRNet (Xie et al. 2020b)	6.45	10.37	9.45	9.41	7.96	10.51	8.44	8.04	8.83
PMP-Net (Wen et al. 2020b)	5.65	11.24	9.64	9.51	6.95	10.83	8.72	7.25	8.73
CRN (Wang et al. 2020a)	**4.79**	**9.97**	**8.31**	9.49	8.94	10.69	**7.81**	8.05	8.51
SoftPoolNet (Wang et al. 2020b)	6.93	10.91	9.78	9.56	8.59	11.22	8.51	8.14	9.20
Ours	5.50	10.02	8.73	**9.05**	**7.53**	**10.24**	8.01	**7.43**	**8.31**
Without skip-connection	6.72	10.46	9.70	9.12	8.42	10.85	8.48	7.80	8.95
Without $\mathcal{D}$	5.73	10.19	8.79	9.10	7.55	10.47	8.12	7.75	8.46
Without $\mathcal{L}_{\text{R}}$	5.77	11.92	11.60	11.47	9.02	12.14	11.82	9.87	10.45

Bold indicates the best performance achieved in certain column

Table 5 Evaluation on the object completion based on the Chamfer distance trained with L2 distance (multiplied by 10^3) with the output resolution of 16,384

Method	Plane	Cabinet	Car	Chair	Lamp	Sofa	Table	Vessel	Avg.
PCN (Yuan et al. 2018) dataset, Output Resolution = 16,384, L2 metric									
FoldingNet (Yang et al. 2018b)	3.15	7.94	4.68	9.23	9.23	8.90	6.69	7.33	7.14
FoldingNet + *SoftPool++*	3.02	7.86	4.50	9.07	9.03	8.69	6.49	7.31	7.00
AtlasNet (Groueix et al. 2018)	1.75	5.10	3.24	5.23	6.34	5.99	4.36	4.18	4.52
TopNet (Tchapmi et al. 2019)	2.15	5.62	3.51	6.35	7.50	6.95	4.78	4.36	5.15
NSFA (Zhang et al. 2020b)	1.75	5.31	3.43	5.01	4.73	6.41	4.00	3.56	4.28
PCN (Yuan et al. 2018)	1.40	4.45	2.45	4.84	6.24	5.13	3.57	4.06	4.02
PCN + *SoftPool++*	**1.10**	4.37	**2.40**	4.81	5.67	4.70	3.41	3.82	3.79
MSN (Liu et al. 2020)	1.54	7.25	4.71	4.54	6.48	5.89	3.80	3.85	4.76
MSN + *SoftPool++*	1.13	7.24	4.64	4.21	6.28	5.83	3.57	3.45	4.54
PF-Net (Huang et al. 2020)	1.55	4.43	3.12	3.96	4.21	5.87	3.35	3.89	3.80
CRN (Wang et al. 2020a)	1.46	4.21	2.97	3.24	5.16	5.01	3.99	3.96	3.75
GRNet (Xie et al. 2020b)	1.53	3.62	2.75	**2.95**	**2.65**	3.61	2.55	2.12	2.72
SoftPoolNet (Wang et al. 2020b)	1.63	3.79	3.05	3.27	2.95	3.78	2.59	2.25	2.91
Ours	1.27	**3.43**	2.65	2.98	2.67	**3.38**	**2.27**	**1.85**	**2.55**
Without skip-connection	1.53	3.75	2.96	3.15	2.90	3.59	2.35	1.96	2.77
Without $\mathcal{D}$	1.37	3.59	2.78	3.13	2.74	3.51	2.43	2.03	2.69
Without $\mathcal{L}_{\text{R}}$	1.42	4.74	2.91	4.63	3.66	4.14	2.83	2.29	3.33

Bold indicates the best performance achieved in certain column

Table 6 Evaluation on the object completion based on the F-Score@1% trained with L2 Chamfer distance and the output resolution of 16,384

Method	Plane	Cabinet	Car	Chair	Lamp	Sofa	Table	Vessel	Avg.
PCN (Yuan et al. 2018) dataset, output resolution = 16,384, F-Score@1%									
FoldingNet (Yang et al. 2018b)	0.642	0.237	0.382	0.236	0.219	0.197	0.361	0.299	0.322
FoldingNet + *SoftPool++*	0.687	0.347	0.455	0.237	0.236	0.257	0.377	0.428	0.378
AtlasNet (Groueix et al. 2018)	0.845	0.552	0.630	0.552	0.565	0.500	0.660	0.624	0.616
TopNet (Tchapmi et al. 2019)	0.771	0.404	0.544	0.413	0.408	0.350	0.572	0.560	0.503
PCN (Yuan et al. 2018)	0.881	0.651	0.725	0.625	0.638	0.581	0.765	0.697	0.695
PCN + *SoftPool++*	0.880	0.671	**0.777**	0.723	0.755	0.578	0.819	0.661	0.733
MSN (Liu et al. 2020)	0.885	0.644	0.665	0.657	0.699	0.604	0.782	0.708	0.705
MSN + *SoftPool++*	**0.903**	0.727	0.721	**0.736**	0.718	0.633	0.796	0.750	0.748
GRNet (Xie et al. 2020b)	0.843	0.618	0.682	0.673	0.761	0.605	0.751	0.750	0.708
SoftPoolNet (Wang et al. 2020b)	0.831	0.605	0.685	0.649	0.715	0.601	0.746	0.721	0.694
Ours	0.867	**0.693**	0.706	0.712	**0.794**	**0.689**	**0.825**	**0.804**	**0.761**
Without skip-connection	0.836	0.658	0.670	0.671	0.753	0.652	0.753	0.791	0.723
Without $\mathcal{D}$	0.843	0.672	0.700	0.686	0.767	0.653	0.768	0.796	0.736
Without $\mathcal{L}_{\mathrm{R}}$	0.824	0.634	0.593	0.670	0.695	0.575	0.686	0.755	0.679

Bold indicates the best performance achieved in certain column

objects provided by the Completion3D (Tchapmi et al. 2019) and MVP (Pan et al. 2021), respectively. Here, the average F-Score with SoftPool++ outperforms the other methods. The tables also validate the benefit of our individual contributions in the overall result. In addition, Table 7 shows that, by applying our SoftPool++ module on the variational coarse sub-architecture of VRCNet (Pan et al. 2021), the average performance of the fine reconstruction reached the state-of-the-art with the improvement from 78.1 to 79.9%.

Advantage over SoftPoolNet Wang et al. (2020b).

Compared to SoftPoolNet (Wang et al. 2020b), our contributions in the proposed SoftPool++ features improve (Wang et al. 2020b) by 0.83×10^{-4} in the L1 Chamfer distance and 1.71×10^{-4} for L2. Strikingly, even without the skip connections, we have already outperformed SoftPoolNet (Wang et al. 2020b). This then demonstrate the strength of the proposed SoftPool++ over (Wang et al. 2020b).

Moreover, the results from high resolution reconstruction also validates our conclusion when evaluating against SoftPoolNet (Wang et al. 2020b). With or without the skip connections, our SoftPool++ performs better than (Wang et al. 2020b).

6.2 Qualitative Evaluation

Similar to Sect. 6.1, the objects in this section are also trained from and evaluated on ShapeNet (Chang et al. 2015). However, for the qualitative results in Fig. 9, we show the results in the original points resolution specified in their respective methods.

Comparison against PointNet (Qi et al. 2017a) *feature.*

From Fig. 9, the max-pooling operation from the PointNet (Qi et al. 2017a) feature is embedded in FoldingNet (Yang et al. 2018b), PCN (Yuan et al. 2018) and MSN (Liu et al. 2020). We noticed that these methods are either over-smoothens the reconstruction or start introducing noise.

On one hand, FoldingNet (Yang et al. 2018b) and PCN (Yuan et al. 2018) smoothens out the reconstruction so that the fine details such as the armrest of the chair are no longer visible and the wheels of the car are no longer separated. On the other, MSN (Liu et al. 2020) tries to reconstruct the finer details but produces a noisy point cloud. Contrary to these methods, we achieve a smoother surface reconstruction with with visible geometric details of the object like the armrest and the wheels.

Advantage of Skip Connections We also explore the combination of 3D-GCN (Lin et al. 2020) and TreeGAN (Shu et al. 2019) that uses graph convolutions in an encoder–decoder architecture. Its latent feature is presented as a vector with a length of 1024. Without the skip connection, several inconsistencies emerge. For instance, the shape of the boat is slimmer than the ground truth while one dimension of the bookshelf is thicker. These information are part of the input but are not propagated to the output.

Among these methods, GRNet (Xie et al. 2020b) achieves similar quantitative results compared to our approach in Table 5. They also build skip connections between their encoder and decoder. However, as input to the architecture, they first voxelize the input point cloud. After going through the encoder–decoder, they convert the 3D grid back to point cloud. Due to the discretization of the point cloud, this affects the results of GRNet (Xie et al. 2020b). It fails to reconstruct

Table 7 Evaluation on the object completion based on the F-Score@1% trained with L2 Chamfer distance and the output resolution of 16,384

Method	Plane	Cabinet	Car	Chair	Lamp	Sofa	Table	Vessel	Avg.
MVP (Pan et al. 2021) dataset, Output Resolution = 16,384, F-Score@1%									
TopNet (Tchapmi et al. 2019)	0.789	0.621	0.612	0.443	0.387	0.506	0.639	0.609	0.576
PCN (Yuan et al. 2018)	0.816	0.614	0.686	0.517	0.455	0.552	0.646	0.628	0.614
PCN + *SoftPool++*	0.853	0.643	0.729	0.563	0.472	0.566	0.670	0.643	0.642
MSN (Liu et al. 2020)	0.879	0.692	0.693	0.599	0.604	0.627	0.730	0.696	0.690
MSN + *SoftPool++*	0.914	0.717	0.727	0.620	0.638	0.649	0.765	0.726	0.719
GRNet (Xie et al. 2020b)	0.853	0.578	0.646	0.635	0.710	0.580	0.690	0.723	0.677
ECG (Pan 2020)	0.906	0.680	0.716	0.683	0.734	0.651	0.766	0.753	0.736
NSFA (Zhang et al. 2020b)	0.903	0.694	0.721	0.737	0.783	0.705	0.817	0.799	0.770
CRN (Wang et al. 2020a)	0.898	0.688	0.725	0.670	0.681	0.641	0.748	0.742	0.724
VRCNet (Pan et al. 2021)	0.928	0.721	0.756	0.743	0.789	0.696	0.813	0.800	0.781
VRCNet + *SoftPool++*	**0.947**	**0.745**	**0.768**	**0.759**	**0.810**	**0.720**	**0.829**	**0.813**	**0.799**
PoinTr (Yu et al. 2021)	0.888	0.681	0.716	0.703	0.749	0.656	0.773	0.760	0.741
SoftPoolNet (Wang et al. 2020b)	0.843	0.568	0.636	0.623	0.698	0.568	0.680	0.710	0.666
Ours	0.862	0.622	0.704	0.695	0.783	0.649	0.776	0.778	0.734
Without skip-connection	0.862	0.555	0.648	0.652	0.716	0.603	0.703	0.719	0.682
Without $\mathcal{D}$	0.856	0.624	0.666	0.664	0.732	0.622	0.738	0.770	0.709
Without $\mathcal{L}_R$	0.822	0.488	0.602	0.573	0.661	0.500	0.667	0.696	0.626

Bold indicates the best performance achieved in certain column

thin structures like the antenna on the boat and the vertical stabilizers of the jet. In addition, it tried to fill up the hole in the box which should have remained empty. In contrast, our method that processes directly on the point cloud can handle these cases.

Improvements from SoftPoolNet (Wang et al. 2020b). Moreover, we compared the proposed method against the previous SoftPoolNet (Wang et al. 2020b) to reveal the advantages of our novel approach. From Fig. 9, while the previous method fails to reconstruct the four corners of the box and the wheels of the jet, the new method is more consistent to the ground truth. Overall, our novel approach reconstructs sharper geometries with less noise and less holes.

Other Methods There have been some trend to re-purpose method that were originally tailored for semantic segmentation such as PointCNN (Li et al. 2018) to train for object completion. Since they both use point clouds, the intuition is to use the local convolutions from Li et al. (2018) to upsample the point cloud from its partial scan to its completed structure. Unfortunately, these methods fails to reconstruct the objects because it is not the intended purpose of the architecture—in semantic segmentation, their input and output point cloud remains the same.

6.3 Classification on ModelNet and PartNet

In addition to shape completion, we also evaluate our approach in terms of classification on the ModelNet10 (Zhirong et al. 2015), ModelNet40 (Zhirong et al. 2015) and PartNet (Mo et al. 2019) datasets. Note that ModelNet40 contains 12,311 CAD models classified into 40 categories while PartNet contains 26,671 models with 24 categories.

Similar to the other approaches such as 3D-GAN (Wu et al. 2016), RS-DGCNN (Sauder et al. 2019), VConv-DAE (Sharmaet al. 2016), FoldingNet (Yang et al. 2018b) and KCNet (Shen et al. 2018), we also implemented a self-supervised training to extract features from the input point cloud then a supervised training to train a linear Support Vector Machine (SVM) (Cortes and Vapnik 1995) to predict the categorical classification. The former relies on the 57,448 ShapeNet models (Chang et al. 2015) as its training dataset while the latter relies on ModelNet (Zhirong et al. 2015) and PartNet (Mo et al. 2019).

It is noteworthy to mention that there is a significant difference from RS-DGCNN (Sauder et al. 2019) in the details of the self-supervised training. On one hand, our method randomly subsamples the point cloud while, on the other, Sauder et al. (2019) includes an additional data augmentation step that randomly decomposes the 3D input structure into different parts then repositions these parts by translation. Since we did not include the additional augmentation from Sauder et al. (2019), our evaluation is a fair comparison against other methods.

The evaluation in Table 8 reports that our model outperforms the accuracy of RS-DGCNN (Sauder et al. 2019) by 4.11% on the ModelNet40 dataset, a sign of the higher descriptiveness in terms of categorical information. The

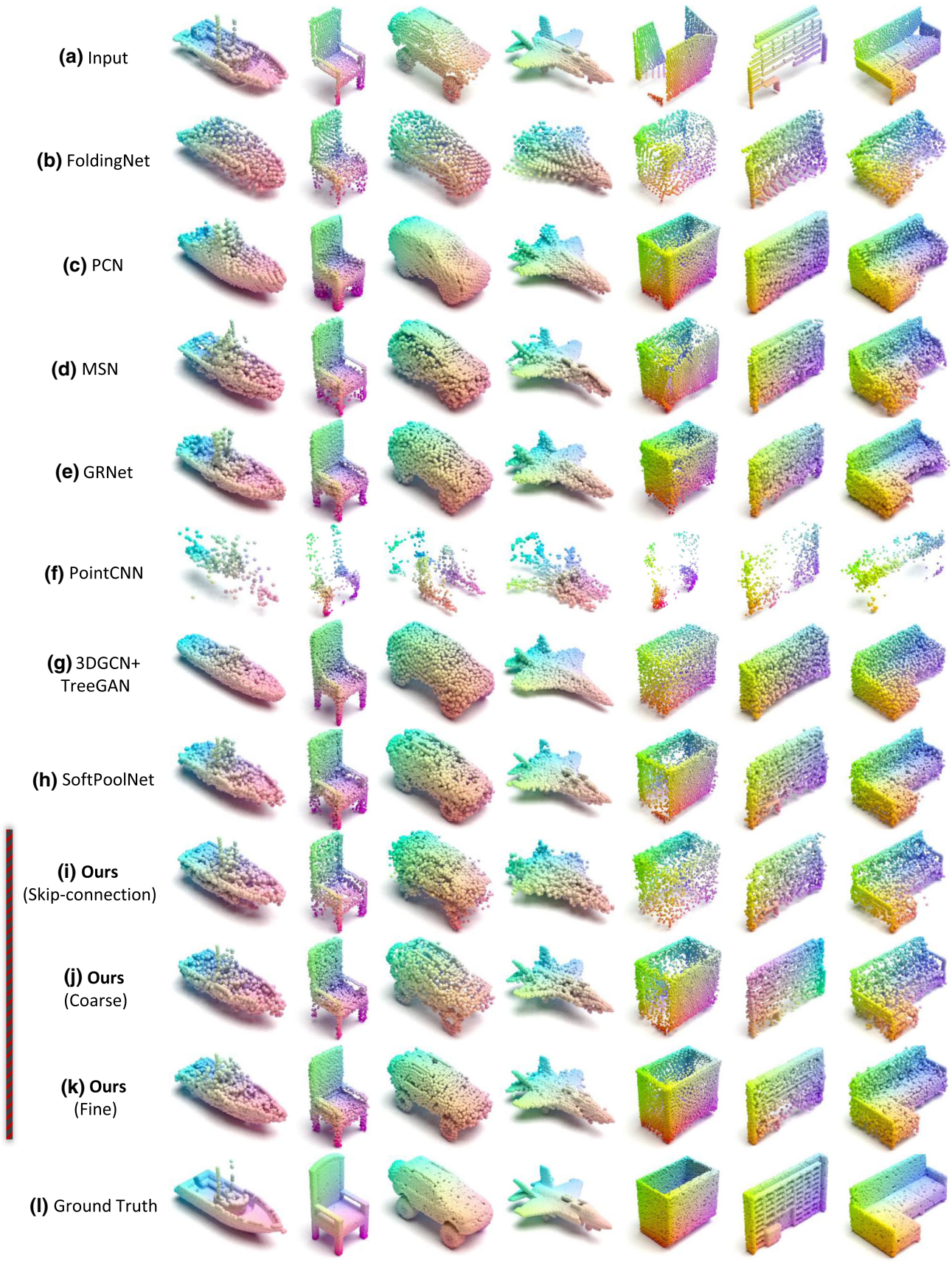


Fig. 9 Qualitative results on the ShapeNet (Chang et al. 2015) dataset. Note that the three results from our method corresponds to different parts of the architecture as explained in Fig. 5. Here, (k) represents our final reconstruction

Table 8 Object classification accuracy on ModelNet40 (Zhirong et al. 2015), ModelNet40 (Zhirong et al. 2015) and PartNet (Mo et al. 2019) datasets

Method	ModelNet40 (%)	ModelNet10 (%)	PartNet (%)
VConv-DAE (Sharmaet al. 2016)	75.50	80.50	–
3D-GAN (Wu et al. 2016)	83.30	91.00	74.23
Latent-GAN	85.70	95.30	–
FoldingNet (Yang et al. 2018b)	88.40	94.40	–
VIP-GAN (Han et al. 2019)	90.19	92.18	–
RS-PointNet (Sauder et al. 2019)	87.31	91.61	76.95
RS-DGCNN (Sauder et al. 2019)	90.64	94.52	–
KCNet (Shen et al. 2018)	91.0	94.4	–
SoftPoolNet (Wang et al. 2020b)	92.28	96.14	84.32
Ours	**94.75**	**96.99**	**87.25**
Without skip-connection	93.17	96.34	85.26
Without $\mathcal{L}_{\text{inter}}$	91.23	95.11	83.07
Without $\mathcal{L}_{\text{intra}}$	84.98	91.35	81.91
Without $\mathcal{L}_{\text{inter}}$, $\mathcal{L}_{\text{intra}}$	84.21	91.77	80.55
Without $\mathcal{L}_{\text{boundary}}$	89.70	94.14	82.39
Without $\mathcal{L}_{\text{preserve}}$	87.84	93.15	81.32
Without $\mathcal{L}_{\text{R}}$	79.22	85.40	76.48

Bold indicates the best performance achieved in certain column

improvement of 2.47% from our approach compared to SoftPoolNet (Wang et al. 2020b) is also obvious, proving that the proposed SoftPool++ feature and skip-connection together are more advantageous for classification. Similar results are also obtained on ModelNet10 (Zhirong et al. 2015) and PartNet (Mo et al. 2019).

6.4 Efficiency

In addition to the evaluation in terms of shape completion and categorical classification, we also compare in Table 9 the properties of our model such as its memory footprint and inference speed, as well as the type of data being processed.

The cost of outperforming SoftPoolNet (Wang et al. 2020b) becomes evident on the memory footprint and the inference time. Compared to SoftPoolNet (Wang et al. 2020b), the memory footprint of our method is approximately doubled due the increase in the number of parameters from the multiple feature extraction modules in our architecture. This also triggers a larger inference time than SoftPoolNet (Wang et al. 2020b) from 0.04 to 0.11 seconds. A similar trend is associated to other approaches that divides the point cloud into regions such as AtlasNet (Groueix et al. 2018) and MSN (Liu et al. 2020), i.e.we achieve significantly higher accuracy in reconstruction but also increase the memory footprint and the inference time.

However, if we look at the overall data, we observe that the proposed method at 61.7MB consumes remarkably less memory than the other point cloud approaches such as GRNet (Xie et al. 2020b) at 293MB and PointCNN (Li et al. 2018) at 497MB, as well as the volumetric approaches such as 3D-EPN (Dai et al. 2017) at 420MB and ForkNet (Wang et al. 2019b) at 362MB. An important reason why their models are so large in memory usage is that 3D convolutions are applied in multiple layers of their architectures, while our approach is mainly composed of 2D convolutions only. Among those approaches with large memory consumption, GRNet (Xie et al. 2020b) is one of the top performers in point cloud completion. Since their architecture relies on volumetric grids where they convert the input point cloud to voxel grid then convert back to a point cloud, this affects not only their memory footprint but also their inference time, which is 8 times higher than ours.

Compared to approaches composed mainly of MLPs, our model reports a comparable size to PCN (Yuan et al. 2018) while having a faster inference time than MSN (Liu et al. 2020). The reason is that although our 2D convolution kernels introduces a additional dimensions, the newly added dimension N_k of 32 is comparably much smaller than the feature dimension N_f of 256 at which MLPs operates. Notably, approaches based on KNN search such as PointCNN (Li et al. 2018) and 3D-GCN (Lin et al. 2020) usually take much longer for inference.

7 Ablation Study

Based on the evaluation from ShapeNet (Chang et al. 2015), we further analyze our proposed method's behavior through an ablation study. In this section, we demonstrate the advantage of SoftPool++ over PontNet; expound on the claims of

Table 9 Overview of different object completion methods. Note that the inference time is represented by the amount of time to conduct inference on a single object

Method	Size (MB)	Inference Time (s)	Core Operator	Data Type	With KNN
3D-EPN (Dai et al. 2017)	420.0	–	3D Conv	Voxels	No
ForkNet (Wang et al. 2019b)	362.0	–	3D Conv	Voxels	No
GRNet (Xie et al. 2020b)	293.0	0.88	3D Conv	Points	No
PointCNN (Li et al. 2018)	497.0	1.20	3D Conv	Points	Yes
DeepSDF (Park et al. 2018)	7.4	9.72	MLP	SDF	No
FoldingNet (Yang et al. 2018b)	19.2	0.05	MLP	Points	No
AtlasNet (Groueix et al. 2018)	2.0	0.32	MLP	Points	No
PCN (Yuan et al. 2018)	54.8	0.11	MLP	Points	No
MSN (Liu et al. 2020)	12.0	0.21	MLP	Points	No
3D-GCN (Lin et al. 2020) (coder)	2.1	0.82	2D Conv	Points	Yes
SoftPoolNet (Wang et al. 2020b)	37.2	0.04	2D Conv	Points	No
Ours	61.7	0.11	2D Conv	Points	No

our loss function; investigate the value of the skip connection with feature transform in our architecture; and; delve deeper on what happens in the SoftPool++ module.

7.1 Replacing PointNet with SoftPool++

In addition to the comparison in Table 5, we also evaluate the results by replacing the latent features in other point cloud completion approaches [i.e.FoldingNet (Yang et al. 2018b), MSN (Liu et al. 2020), and PCN (Yuan et al. 2018)] with our SoftPool++ features, while keeping their decoders unchanged. In this way, we have a one-to-one comparison of PointNet and SoftPool++ features.

Since these works depend on a PointNet features (having a dimensionality of 1024), we also build up our SoftPool++ features with the same size. Remarkably, the use of SoftPool++ features improves performance in all tested methods, i.e.the performance of FoldingNet (Yang et al. 2018b), PCN (Yuan et al. 2018) and MSN (Liu et al. 2020) improves respectively by 0.14×10^{-3}, 0.23×10^{-3} and 0.22×10^{-3}.

7.2 Loss Functions

Tables 3 and 8 include an ablation study that investigates the effects of the individual loss functions from Sect. 5. For both experiments, we notice that all loss functions are critical to achieve state-of-the-art results. Note that we have shown in Fig. 6 and cabinet completion in Fig. 7 to demonstrate the contributions of $\mathcal{L}_{\text{boundary}}$ and $\mathcal{L}_{\text{preserve}}$ in the reconstruction. $\mathcal{L}_{\text{boundary}}$ *in other methods*. An interesting idea is the capacity of $\mathcal{L}_{\text{boundary}}$ to be integrated in other existing methods that join multiple deformed 2D patches together to form the final output. Since the patches in AtlasNet (Groueix et al. 2018), PCN (Yuan et al. 2018) and MSN (Liu et al. 2020) are frequently overlapping nearby patches, we tried to integrate

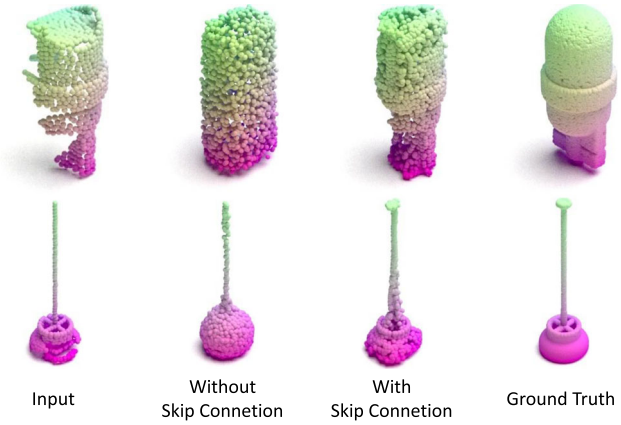


Fig. 10 Object completion results with and without the influence of the skip-connection

$\mathcal{L}_{\text{boundary}}$ into their loss functions. Tables 4 and 3 evaluate this idea and prove that this activation helps FoldingNet (Yang et al. 2018b), PCN (Yuan et al. 2018) and AtlasNet (Groueix et al. 2018) perform better, improving the Chamfer distance with at least 1×10^{-4} on the resolution of 2048 and 1×10^{-3} on resolution of 16,384.

7.3 Skip Connection with Feature Transform

One of the key contributions in this paper is the introduction of skip connections with feature transforms on point cloud. Our ablation study in Tables 3 and 2 also includes the numerical advantage of having the skip connection in our architecture, improving the Chamfer distance by 0.65×10^{-4} in L1 and 1.27×10^{-4} in L2.

In addition to the numerical advantage, we also interpret these values through some examples in Fig. 10 where we reconstruct lamps. Without the skip connection, the model recursively simplifies the given partial scan until it reaches

Fig. 11 Visualization of the first row of **F** on the first SoftPool++ module in our architecture

the latent feature. Due to the oversimplification, the output then builds the closest generic shape of the lamp. Contrary to that, with the skip connection, the model preserves the input structure and incorporates the given partial scan into the final reconstruction. In effect, the result is closer to the ground truth.

We also perform an ablation study on the regularization $\mathcal{L}_{\mathbf{R}}$ of the feature transform **R** in Tables 4, 5, 3 and 2. Compared to our complete framework, the results trained without the skip-connection drops by 0.64×10^{-4}. However, when trained with the skip-connection but without the regularization of **R**, the results drops by 2.14×10^{-4} which is significantly larger. Therefore, it is noteworthy to mention that training with skip-connection but without the regularization performs worse than removing the skip-connection altogether. This clearly shows the advantage of the regularization term on the feature transform.

7.4 Activations from the SoftPool++ Features

Given the input point cloud, we explore how SoftPool++ sorts the points on the first feature extraction module in the architecture. For this experiment, we visualize the points based on the value of the first column in **F** which is the result of MLP as shown in Fig. 2. Therefore, Fig. 11 highlights the activations associated to this feature. Noticeably, due to MLP, the points can undergo much more than just a linear transformation of its absolute coordinates.

Continuing our analysis, we move further into examining how the truncation sizes (N_s, N_r) and the output dimension N_f influence the completion. Table 10 summarizes this evaluation on ShapeNet (Chang et al. 2015) as we vary these values on the second SoftPool++ module in our architecture. As described in Table 1, since our SoftPool++ feature is fixed with 256 rows, we then set $N_r \times N_s = 256$. Note that the next ablation study focuses on changing the number of rows by

Table 10 Influence of N_f and (N_s, N_r) on the L2 Chamfer distance (multiplied by 10^3), evaluated on the output resolution of 2048 points

N_s	N_r	N_f				
		32	64	128	256	512
1	256	17.33	16.50	13.45	11.24	10.39
2	128	17.28	17.06	15.22	14.62	13.10
4	64	16.32	15.66	13.65	11.29	11.31
8	32	17.55	11.18	10.27	9.59	**9.58**
16	16	17.97	11.26	10.21	9.60	9.59
32	8	17.31	11.89	10.32	9.59	**9.58**

Bold indicates the best performance achieved in certain column

Table 11 Influence of N_s and N_r on the L2 Chamfer distance (multiplied by 10^3), evaluated on the output resolution of 2048 points

N_s	N_r					
	8	16	32	64	128	256
1	–	–	14.99	14.82	14.64	11.24
2	–	14.99	14.97	14.73	14.62	11.26
4	14.79	12.85	12.27	11.29	11.08	9.91
8	11.56	10.62	**9.59**	**9.59**	9.62	9.64
16	10.07	**9.59**	9.62	9.62	9.61	9.63
32	9.60	9.61	9.61	9.62	9.61	9.61

Here, N_f is set to 256
Bold indicates the best performance achieved in certain column

independently setting N_r and N_s. For the ease of training and evaluation for all (N_s, N_r) and N_f, we do not apply discriminative training $\mathcal{D}$ for Table 10. The table indicates that we reach the minimum Chamfer distance as soon as N_s reaches 8, N_r reaches 32 and N_f reaches 256. After then, only small improvements of around 0.01×10^{-3} are attained. Therefore, we select $N_s = 8$, $N_r = 32$ and $N_f = 256$ so that there are less parameters in the model to train which consequently lead to less memory footprint.

The next ablation study alleviates the constraint of having a fixed latent feature dimension where we set $N_r \times N_s = 256$ in Table 10. In Table 11, we consider different values of N_r and N_s while setting N_f to 256, where we observe that that the error plateaus when N_s is 8 and N_r is 32. Note that these values matches the optimum values from Table 10 and validates the advantage of truncation.

Considering the numerical advantages of N_s, we also explore it visually while keeping N_f and N_r constant to 256 and 32, respectively. Similar to Fig. 11, Fig. 12 plots the points from the input point cloud that are truncated by N_s. By increasing N_s from 4 to 16, the resulting feature also increases the amount of structures from the plane. For instance, the wings become more and more visible on the figure. This then raises the question of how much information from the partial scans does the network need to reconstruct

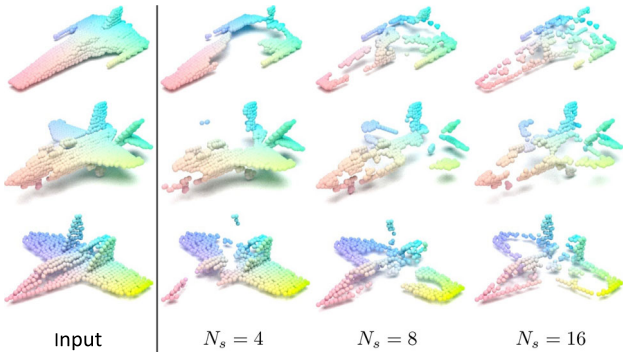


Fig. 12 Visualization of the truncated points on the input point cloud with different values of N_s

the object. Evidently, this question is answered by our ablation study in Table 10 where we found the optimum value of N_s, i.e.8. Comparisons in Fig. 12 shows that larger values of N_s does not further add the points on the body of the plane which is a common part for plane category.

8 Conclusion

We propose a novel feature extraction technique called *SoftPool++* that directly processes the point cloud. Compared to the counterpart that heavily relies on the max-pooling operation in PointNet (Qi et al. 2017a), our feature extraction method captures a higher amount of data from the input point cloud by alleviating the limitation of taking only the maximum while also establishing the relation between different points through our regional convolutions.

Structuring multiple SoftPool++ in an encoder–decoder structure, this paper becomes the first to propose a point-wise skip connection with feature transformation. Considering that the given point cloud is continuously downsampled in the encoder, the main advantage of such connection is the capacity to incorporate the input data into the decoder. This then overcomes the loss of information in the encoder.

Examining our contributions on 3D object completion, we discovered that we perform the state-of-the-art especially on high-resolution reconstructions. We also visually demonstrate our advantage and concluded that our reconstructions are sharper, i.e.with less noise in our point cloud; and, captures the finer details, i.e.without over-smoothing different parts of the object.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

Aoki, Y., Goforth, H., Srivatsan, R. A., & Lucey, S. (2019). Pointnetlk: Robust & efficient point cloud registration using pointnet. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 7163–7172).

Azad, R., Asadi-Aghbolaghi, M., Fathy, M., & Escalera, S. (2019). Bi-directional ConvLSTM U-net with Densley connected convolutions. In *Proceedings of the IEEE/CVF international conference on computer vision workshops* (pp. 406–415).

Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., & Xiao, J. (2015). *Shapenet: An information-rich 3d model repository*. arXiv preprint arXiv:1512.03012.

Chibane, J., Alldieck, T., & Pons-Moll, G. (2020). Implicit functions in feature space for 3d shape reconstruction and completion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 6970–6981).

Choy, C. B., Xu, D., Gwak, J., Chen, K., & Savarese, S. (2016). 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *European conference on computer vision* (pp. 628–644). Springer.

Çiçek, Ö., Abdulkadir, A., Lienkamp, S. S., Brox, T., & Ronneberger, O. (2016). 3d u-net: Learning dense volumetric segmentation from sparse annotation. In *International conference on medical image computing and computer-assisted intervention* (pp. 424–432). Springer.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning, 20*(3), 273–297.

Dai, A., Qi, C. R., & Nießner, M. (2017). Shape completion using 3d-encoder-predictor CNNS and shape synthesis. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (Vol. 3).

Gao, H., Tao, X., Shen, X., & Jia, J. (2019). Dynamic scene deblurring with parameter selective sharing and nested skip connections. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 3848–3856).

Gong, B., Nie, Y., Lin, Y., Han, X., & Yu, Y. (2021). ME-PCN: Point completion conditioned on mask emptiness. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 12488–12497).

Groueix, T., Fisher, M., Kim, V. G., Russell, B. C., & Aubry, M. (2018). A Papier-Mâché approach to learning 3d surface generation. In *The IEEE conference on computer vision and pattern recognition (CVPR)*.

Han, Z., Shang, M., Liu, Y. S., & Zwicker, M. (2019). View inter-prediction gan: Unsupervised representation learning for 3d shapes by learning global shape memories to support local view predictions. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 33, pp. 8376–8384).

Huang, T., Zou, H., Cui, J., Yang, X., Wang, M., Zhao, X., Zhang, J., Yuan, Y., Xu, Y., & Liu, Y. (2021). RFNET: Recurrent forward network for dense point cloud completion. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 12508–12517).

Huang, Z., Yu, Y., Xu, J., Ni, F., & Le, X. (2020). PF-NET: Point fractal network for 3d point cloud completion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 7662–7670).

Kim, J., Lee, J. K., & Lee, K. M. (2016). Deeply-recursive convolutional network for image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1637–1645).

Kirillov, A., Wu, Y., He, K., & Girshick, R. (2020). Pointrend: Image segmentation as rendering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 9799–9808).

Lei, H., Akhtar, N., Mian, A. (2020). Seggcn: Efficient 3d point cloud segmentation with fuzzy spherical kernel. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 11611–11620).

Li, P., Wang, Q., & Zhang, L. (2013). A novel earth mover's distance methodology for image matching with gaussian mixture models. In *The IEEE international conference on computer vision (ICCV)*.

Li, Y., Bu, R., Sun, M., Wu, W., Di, X., & Chen, B. (2018). Pointcnn: Convolution on x-transformed points. In *Advances in neural information processing systems* (pp. 820–830).

Lim, I., Ibing, M., & Kobbelt, L. (2019). A convolutional decoder for point clouds using adaptive instance normalization. In *Computer graphics forum* (Vol. 38, pp. 99–108). Wiley Online Library.

Lin, Z. H., Huang, S. Y., & Wang, Y. C. F. (2020). Convolution in the cloud: Learning deformable kernels in 3d graph convolution networks for point cloud analysis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 1800–1809).

Liu, M., Sheng, L., Yang, S., Shao, J., & Hu, S. M. (2020). Morphing and sampling network for dense point cloud completion. In *Proceedings of the AAAI conference on artificial intelligence* (vol. 34, pp. 11596–11603).

Mao, J., Wang, X., & Li, H.(2019). Interpolated convolutional networks for 3d point cloud understanding. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 1578–1587).

Mazaheri, G., Mithun, N. C., Bappy, J. H., & Roy-Chowdhury, A. K. (2019). A skip connection architecture for localization of image manipulations. In *CVPR workshops* (pp. 119–129).

Mo, K., Zhu, S., Chang, A. X., Yi, L., Tripathi, S., Guibas, L. J., & Su, H. (2019). PartNet: A large-scale benchmark for fine-grained and hierarchical part-level 3D object understanding. In *The IEEE conference on computer vision and pattern recognition (CVPR)*.

Noh, H., Hong, S., Han, B. (2015). Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE international conference on computer vision* (pp. 1520–1528).

Pan, L. (2020). ECG: Edge-aware point cloud completion with graph convolution. *IEEE Robotics and Automation Letters, 5*(3), 4392–4398.

Pan, L., Chen, X., Cai, Z., Zhang, J., Zhao, H., Yi, S., & Liu, Z. (2021). Variational relational point completion network. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 8524–8533).

Park, J., Zhou, Q.Y., Koltun, V. (2017). Colored point cloud registration revisited. In *Proceedings of the IEEE international conference on computer vision* (pp. 143–152).

Park, J. J., Florence, P., Straub, J., Newcombe, R., & Lovegrove, S. (2018). Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 165–174).

Qi, C. R., Litany, O., He, K., & Guibas, L. J. (2019). Deep hough voting for 3d object detection in point clouds. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 9277–9286).

Qi, C. R., Su, H., Mo, K., & Guibas, L. J. (2017a). Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 652–660).

Qi, C. R., Yi, L., Su, H., & Guibas, L. J. (2017b). Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems (NIPS)*.

Sauder, J., & Sievers, B. (2019). *Self-supervised deep learning on point clouds by reconstructing space*. arXiv preprint arXiv:1901.08396

Sharma, A., Grau, O., & Fritz, M. (2016). VCONV-DAE: Deep volumetric shape learning without object labels. In *European conference on computer vision* (pp. 236–250). Springer.

Shen, Y., Feng, C., Yang, Y., & Tian, D. (2018). Mining point cloud local structures by kernel correlation and graph pooling. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4548–4557).

Shi, W., & Rajkumar, R. (2020). Point-GNN: Graph neural network for 3d object detection in a point cloud. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 1711–1719).

Shu, D. W., Park, S. W., & Kwon, J. (2019). 3d point cloud generative adversarial network based on tree structured graph convolutions. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 3859–3868).

Tatarchenko, M., Richter, S. R., Ranftl, R., Li, Z., Koltun, V., & Brox, T. (2019). What do single-view 3d reconstruction networks learn? In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 3405–3414).

Tchapmi, L. P., Kosaraju, V., Rezatofighi, H., Reid, I., & Savarese, S. (2019). Topnet: Structural point cloud decoder. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 383–392).

Wang, H., Liu, Q., Yue, X., Lasenby, J., & Kusner, M. J. (2021a). Unsupervised point cloud pre-training via occlusion completion. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 9782–9792).

Wang, L., Huang, Y., Hou, Y., Zhang, S., & Shan, J. (2019a). Graph attention convolution for point cloud semantic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 10296–10305).

Wang, X., Ang Jr, M. H., & Lee, G. H. (2020a). Cascaded refinement network for point cloud completion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*.

Wang, X., Ang, M. H., & Lee, G. H. (2021b). Voxel-based network for shape completion by leveraging edge generation. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 13189–13198).

Wang, Y., Tan, D. J., Navab, N., & Tombari, F. (2019b). Forknet: Multi-branch volumetric semantic completion from a single depth image. In *Proceedings of the IEEE international conference on computer vision* (pp. 8608–8617).

Wang, Y., Tan, D. J., Navab, N., & Tombari, F. (2020b). Softpoolnet: Shape descriptor for point cloud completion and classification. In A. Vedaldi, H. Bischof, T. Brox, & J. M. Frahm (Eds.), *Computer vision—ECCV 2020* (pp. 70–85). Springer.

Wen, X., Han, Z., Cao, Y. P., Wan, P., Zheng, W., & Liu, Y. S. (2021). Cycle4completion: Unpaired point cloud completion using cycle transformation with missing region coding. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 13080–13089).

Wen, X., Li, T., Han, Z., & Liu, Y. S. (2020a). Point cloud completion by skip-attention network with hierarchical folding. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*.

Wen, X., Xiang, P., Han, Z., Cao, Y. P., Wan, P., Zheng, W., & Liu, Y. S. (2020b). *PMP-NET: Point cloud completion by learning multi-step point moving paths*. arXiv preprint arXiv:2012.03408

Wu, J., Zhang, C., Xue, T., Freeman, W. T., & Tenenbaum, J. B. (2016). Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Proceedings of the 30th international conference on neural information processing systems* (pp. 82–90).

Xiang, P., Wen, X., Liu, Y. S., Cao, Y. P., Wan, P., Zheng, W., & Han, Z. (2021). Snowflakenet: Point cloud completion by snowflake point deconvolution with skip-transformer. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 5499–5509).

Xie, H., Yao, H., Sun, X., Zhou, S., & Zhang, S. (2019). Pix2vox: Context-aware 3d reconstruction from single and multi-view images. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 2690–2698).

Xie, H., Yao, H., Zhang, S., Zhou, S., & Sun, W. (2020a). Pix2vox++: Multi-scale context-aware 3d object reconstruction from single and multiple images. *International Journal of Computer Vision, 128*(12), 2919–2935.

Xie, H., Yao, H., Zhou, S., Mao, J., Zhang, S., & Sun, W. (2020b). GRNET: Gridding residual network for dense point cloud completion. In A. Vedaldi, H. Bischof, T. Brox, & J. M. Frahm (Eds.), *Computer vision—ECCV 2020* (pp. 365–381). Springer.

Xu, X., & Lee, G. H. (2020). Weakly supervised semantic point cloud segmentation: Towards 10x fewer labels. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 13706–13715).

Yang, B., Rosa, S., Markham, A., Trigoni, N., & Wen, H. (2018a). Dense 3d object reconstruction from a single depth view. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 41*, 2820–2834.

Yang, B., Wang, S., Markham, A., & Trigoni, N. (2020). Robust attentional aggregation of deep feature sets for multi-view 3d reconstruction. *International Journal of Computer Vision, 128*(1), 53–73.

Yang, B., Wen, H., Wang, S., Clark, R., Markham, A., & Trigoni, N. (2017). 3d object reconstruction from a single depth view with adversarial learning. In *Proceedings of the IEEE international conference on computer vision workshops* (pp. 679–688).

Yang, F., Sun, Q., Jin, H., & Zhou, Z. (2020). Superpixel segmentation with fully convolutional networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 13964–13973)

Yang, Y., Feng, C., Shen, Y., & Tian, D. (2018b). Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 206–215).

Yang, Z., Sun, Y., Liu, S., Shen, X., & Jia, J. (2019). STD: Sparse-to-dense 3d object detector for point cloud. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 1951–1960).

Yu, X., Rao, Y., Wang, Z., Liu, Z., Lu, J., & Zhou, J. (2021). Pointr: Diverse point cloud completion with geometry-aware transformers. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 12498–12507).

Yuan, W., Khot, T., Held, D., Mertz, C., & Hebert, M. (2018). PCN: Point completion network. In *2018 international conference on 3D vision (3DV)* (pp. 728–737). IEEE.

Zhang, J., Chen, W., Wang, Y., Vasudevan, R., & Johnson-Roberson, M. (2020a). *Point set voting for partial point cloud analysis*. arXiv preprint arXiv:2007.04537

Zhang, W., Yan, Q., & Xiao, C. (2020b). *Detail preserved point cloud completion via separated feature aggregation*. arXiv preprint arXiv:2007.02374

Zhirong W., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., & Xiao, J. (2015). 3d shapenets: A deep representation for volumetric shapes. In *2015 IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 1912–1920).

Zhou, L., Du, Y., & Wu, J. (2021). *3d shape generation and completion through point-voxel diffusion*. arXiv preprint arXiv:2104.03670

5.2.3 Learning Local Displacements for Point Cloud Completion (Conference on Computer Vision and Pattern Recognition 2022)

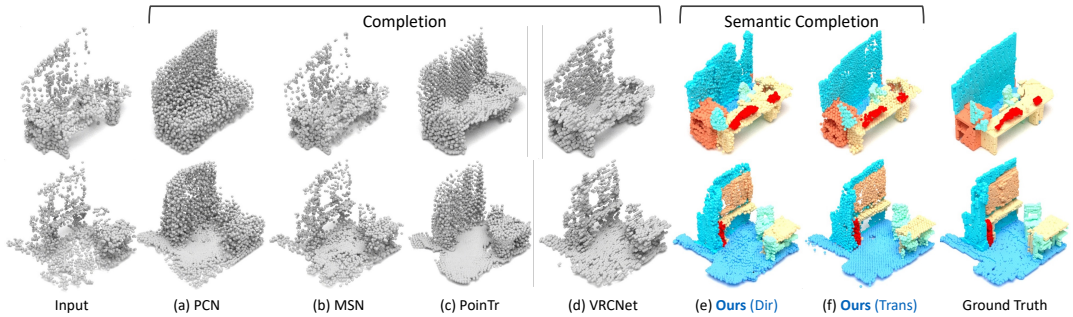


Figure 5.5 The proposed Disp3D [4] is the first few works which can semantically reconstruct a scene in the form of a point cloud with an end-to-end model.

As described in previous works of ours, SoftPoolNet [6] and SoftPool++ [5] can keep local geometries where adjacent samples in a latent feature could be spatially related in the original 3D space. The proposed methods improved the point cloud completion with the help of convolutions in the decoder compared to using the PointNet feature. The fact is that the basic operations in both encoders of SoftPoolNet and SoftPool++ are still multi-layer perceptron (MLP) which cannot progressively process points in the local neighborhood.

In this work, we propose an approach using local displacement vectors to process local features progressively in encoder-decoder structures. Our proposed learnable displacement vectors perform feature extraction by matching the point features to extract the information from the local point sets with a similar shape. While the point cloud completion usually depends on down-sampling in the encoder and up-sampling in the decoder, we design our operator to be used in both the encoder and decoder by applying a neighbor-pooling and making a standalone up-sampling operator. Considering that the input point cloud should be part of the completed output, we enforce a constraint that the order in the output point cloud moves from the observed to the occluded so that a particular part of the output could be trained to be specifically close to the input partial scan. By evaluating our proposed operator in a transformer architecture, e.g. PoinTr [46] on both object and indoor scene completion tasks, we achieve state-of-the-art in point cloud completion for both objects and scenes.

Contributions

Yida Wang investigated the advantages and disadvantages of existing local operators for point cloud on the topic of both completion and segmentation. He then proposed *local displacement* and the graph pooling accordingly. Next, he conducted the experiment comparing precision between volumetric data and point cloud for semantic completion.

David Tan generalized the selection of points' local subset with KNN to other methods.

Federico Tombari helped revise the manuscript.

Nassir Navab financed this work on behalf of the leader of TUM CAMP.

Learning Local Displacements for Point Cloud Completion

Yida Wang[1], David Joseph Tan[2], Nassir Navab[1], Federico Tombari[1,2]

[1] Technical University of Munich
[2] Google

Published version: https://doi.org/10.1109/CVPR52688.2022.00162

Learning Local Displacements for Point Cloud Completion

Yida Wang[1], David Joseph Tan[2], Nassir Navab[1], Federico Tombari[1,2]
[1]Technische Universität München [2]Google Inc.

Abstract

We propose a novel approach aimed at object and semantic scene completion from a partial scan represented as a 3D point cloud. Our architecture relies on three novel layers that are used successively within an encoder-decoder structure and specifically developed for the task at hand. The first one carries out feature extraction by matching the point features to a set of pre-trained local descriptors. Then, to avoid losing individual descriptors as part of standard operations such as max-pooling, we propose an alternative neighbor-pooling operation that relies on adopting the feature vectors with the highest activations. Finally, upsampling in the decoder modifies our feature extraction in order to increase the output dimension. While this model is already able to achieve competitive results with the state of the art, we further propose a way to increase the versatility of our approach to process point clouds. To this aim, we introduce a second model that assembles our layers within a transformer architecture. We evaluate both architectures on object and indoor scene completion tasks, achieving state-of-the-art performance.

1. Introduction

Understanding the entire 3D space is essential for both humans and machines to understand how to safely navigate an environment or how to interact with the objects around them. However, when we capture the 3D structure of an object or scene from a certain viewpoint, a large portion of the whole geometry is typically missing due to self-occlusion and/or occlusion from its surrounding. To solve this problem, geometric completion of scenes [2, 27, 32] and objects [16, 20, 39, 44, 45] has emerged as a task that takes on a 2.5D/3D observation and fills out the occluded regions, as illustrated in Fig. 1.

There are multiple ways to represent 3D shapes. Point cloud [3, 6], volumetric grid [8, 27], mesh [11] and implicit surfaces [18, 21, 40] are among the most common data formats. These representations are used for most 3D-related computer vision tasks such as segmentation, classification and completion. For what concerns geometric completion,

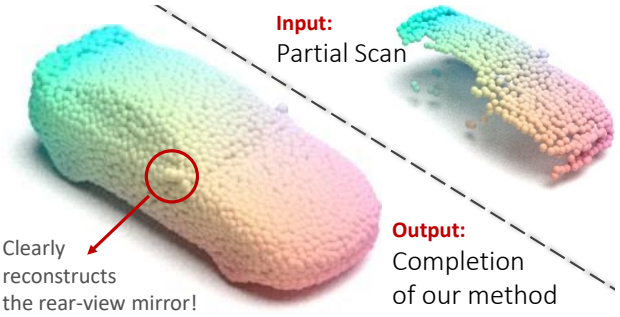


Figure 1. From the input partial scan to our object completion, we visualize the amount of detail in our reconstruction.

most works are focused on either point cloud or volumetric data. Among them, the characteristic of having an explicitly defined local neighbourhood makes volumetric data easier to process with 3D convolutions [7, 41, 42]. One drawback introduced by the predefined local neighborhood is the inaccuracy due to the constant resolution of the voxels, meaning that one voxel can represent several small structures.

On the other hand, point clouds have the advantage of not limiting the local resolution, although they come with their own sets of drawbacks. Mainly, there are two problems in processing point clouds: the undefined local neighborhood and unorganized feature map. Aiming at solving these issues, PointNet++ [23], PMP-Net [35], PointConv [37] and PointCNN [13] employ k-nearest neighbor search to define a local neighborhood, while PointNet [22] and SoftPoolNet [33] adopt the pooling operation to achieve permutation invariant features. Notably, point cloud segmentation and classification were further improved by involving k-nearest neighbor search to form local features in PointNet++ [23] compared to global features in PointNet [22]. Several variations of PointNet [22] also succeeded in improving point cloud completion as demonstrated in FoldingNet [43], PCN [45], MSN [16]. Other methods such as SoftPoolNet [33] and GRNet [39] explicitly present local neighbourhood in sorted feature map and voxel space, respectively.

This paper investigates grouping local features to improve the point cloud completion of objects and scenes. We apply these operation in encoder-decoder architectures

which iteratively uses a feature extraction operation with the help of a set of displacement vectors as part of our parametric model. In addition, we also introduce a new pooling mechanism called neighbor-pooling, aimed at downsampling the data in the encoder while, at the same time, preserving individual feature descriptors. Finally, we propose a new loss function that gradually reconstructs the target from the observable to the occluded regions. The proposed approach is evaluated on both object completion dataset with ShapeNet [3], and semantic scene completion on NYU [25] and CompleteScanNet [36], attaining significant improvements producing high resolutions reconstruction with fine-grained details.

2. Related works

This section focuses on the three most related fields – point cloud completion, point cloud features and semantic scene completion.

Point cloud completion. Given the partial scan of an object similar to Fig. 1, 3D completion aims at estimating the missing shape. In most cases, the missing region is due to self-occlusion since the partial scan is captured from a single view of the object. Particularly for point cloud, FoldingNet [43] and AtlasNet [11] are among the first works to propose an object completion based on PointNet [22] features by deforming one or more 2D grids into the desired shape. Then, PCN [45] extended their work by deforming a collection of much smaller 2D grids in order to reconstruct finer structures.

Through encoder-decoder architectures, ASFM-Net [38] and VRCNet [20] match the encoded latent feature with a completion shape prior, which produce good coarse completion results. To preserve the observed geometry from the partial scan for the fine reconstruction, MSN [16] and VRCNet [20] bypass the observed geometries by using either the minimum density sampling (MDS) or the farthest point sampling (FPS) from the observed surface and building skip connections. By embedding a volumetric sub-architecture, GRNet [39] preserves the discretized input geometries with the volumetric U-connection without sampling in the point cloud space. In more recent works, PMP-Net [35] gradually reconstructs the entire object from the observed to the nearest occluded regions. Also focusing on only predicting the occluded geometries, PoinTr [44] is among the first few transformer methods targeted on point cloud completion by translating the partial scan proxies into a set of occluded proxies to further refine the reconstruction.

Point cloud features. Notably, a large amount of work in object completion [11,16,33,35,39,43,45] rely on PointNet features [22]. The main advantage of [22] is its capacity to be permutation invariant through max-pooling. This is a crucial characteristic for the input point cloud because its data is unstructured.

However, the max-pooling operation disassembles the point-wise features and ignores the local neighborhood in 3D space. This motivated SoftPoolNet [33] to solve this problem by sorting the feature vectors based on the activation instead of taking the maximum values for each element. In effect, they were able to concatenate the features to form a 2D matrix so that a traditional 2D convolution from CNN can be applied.

Apart from building feature representation through pooling operations, PointNet++ [23] samples the local subset of points with the farthest point sampling (FPS) then feeds it into PointNet [22]. Based on this feature, SA-Net [34] then groups the features in different resolutions with KNN for further processing, while PMP-Net [35] uses PointNet++ features to identify the direction to which the object should be reconstructed. PoinTr [44] also solves the permutational invariant problem without pooling by adding the positional coding of the input points into a transformer.

Semantic scene completion. All the point cloud completion are designed to reconstruct a single object. Extending these methods from objects to scenes is difficult because of the difference in size and content. When we tried to train these methods for objects, we noticed that the level of noise is significantly increased such that most objects in the scene are unrecognizable. Evidently, for semantic scene completion, the objective is not only to build the full reconstruction of the scene but also to semantically label each component.

On the other hand, there have been a number of methods for semantic scene completion based on voxel grids that was initiated by SSCNet [27]. Using a similar volumetric data with 3D convolutions [7, 41, 42], VVNet [12] convolves on the 3D volumes which are back-projected from the depth images, revealing the camera view instead of a TSDF volume. Later works such as 3D-RecGAN [42] and ForkNet [32] use discriminators to optimize the convolutional encoder and decoder during training. Since 3D convolutions are heavy in terms of memory consumption especially when the input is presented in high resolution, SketchSSC [4] learns the 3D boundary of all objects in the scene to quickly estimate the resolution of the invariant features.

Although there are quite many methods targeting on volumetric semantic scene completion, there are still no related works proposed explicitly for point cloud semantic scene completion which we achieved in this paper.

3. Operators

Whether reconstructing objects or scenes from a single depth image, the objective is to process the given point

cloud of the partial scan $\mathcal{P}_{\text{in}}$ to reconstruct the complete structure $\mathcal{P}_{\text{out}}$. Most deep learning solutions [16, 20, 33, 43, 45] solve this problem by building an encoder-decoder architecture. The encoder takes the input point cloud to iteratively *down*-sample it into its latent feature. Then, the decoder iteratively *up*-sample the latent feature to reconstruct the object or scene. In this section, we illustrate our novel down-sampling and up-sampling operations that cater to point cloud completion. Thereafter, in the following sections, we use our operators as building blocks to assemble two different encoder-decoder architectures that perform object completion and semantic scene completion. We also discuss the associated loss functions.

3.1. Down-sampling operation

To formalize the down-sampling operation, we denote the input as the set of feature vectors $\mathcal{F}_{\text{in}} = \{\mathbf{f}_i\}_{i=1}^{|\mathcal{F}_{\text{in}}|}$ where $\mathbf{f}_i$ is a feature vector and $|\cdot|$ is the number of elements in the set. Note that, in the first layer of the encoder, $\mathcal{F}_{\text{in}}$ is then set to the coordinates of the input point cloud. We introduce a novel down-sampling operation inspired from the Iterative Closest Point (ICP) algorithm [1, 5]. Taking an arbitrary anchor $\mathbf{f}$ from $\mathcal{F}_{\text{in}}$, we start by defining a vector $\delta \in \mathbb{R}^{D_{\text{in}}}$. From the trainable variable δ, we find the feature closest to $\mathbf{f}+\delta$ and compute the distance. This is formally formulated as a function

$$d(\mathbf{f}, \delta) = \min_{\forall \tilde{\mathbf{f}} \in \mathcal{F}_{\text{in}}} \|(\mathbf{f}+\delta) - \tilde{\mathbf{f}}\| \tag{1}$$

where δ represents a displacement vector from $\mathbf{f}$. Multiple displacement vectors are used to describe the local geometry, each with a weight $\sigma \in \mathbb{R}$. We then assign the set as $\{(\delta_i, \sigma_i)\}_{i=1}^{s}$ and aggregate them with the weighted function

$$g(\mathbf{f}) = \sum_{i=0}^{s} \sigma_i \tanh \frac{\alpha}{d(\mathbf{f}, \delta_i) + \beta} \tag{2}$$

where the constants α and β are added for numerical stability. Here, the hyperbolic tangent in $g(\mathbf{f})$ produces values closer to 1 when the distance $d(\cdot)$ is small and closer to 0 when the distance is large. In practice, we can speed-up (1) with the k-nearest neighbor search for each anchor. A simple example of this operation is depicted in Fig. 2. This illustrates the operation in the first layer where we process the point cloud so that we can geometrically plot a feature in $\mathcal{F}_{\text{in}}$ with respect to $\{(\delta_i, \sigma_i)\}_{i=1}^{s}$.

Furthermore, to enforce the influence of the anchor in this operation, we also introduce the function

$$h(\mathbf{f}) = \rho \cdot \mathbf{f} \tag{3}$$

that projects $\mathbf{f}$ on $\rho \in \mathbb{R}^{D_{\text{in}}}$, which is a trainable parameter. Note that both functions $g(\cdot)$ and $h(\cdot)$ produce a scalar value.

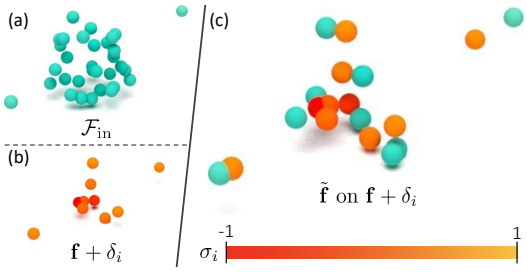


Figure 2. (a) k-nearest neighbor in reference to an anchor $\mathbf{f}$; (b) displacement vectors around the anchor $\mathbf{f}+\delta_i$ and the corresponding weight σ_i; and, (c) closest features $\tilde{\mathbf{f}}$ to $\mathbf{f}+\delta_i$ for all i.

Thus, if we aim at building a set of output feature vectors, each with a dimension of D_{out}, we construct the set as

$$\mathcal{F}_{\text{out}} = \left\{ [g_b(\mathbf{f}_a) + h(\mathbf{f}_a)]_{b=1}^{D_{\text{out}}} \right\}_{a=1}^{|\mathcal{F}_{\text{in}}|} \tag{4}$$

where different sets of trainable parameters $\{(\delta_i, \sigma_i)\}_{i=1}^{s}$ are assigned to each element, while different ρ for each output vector. Moreover, the variables s in (2) and D_{out} in (4) are the hyper-parameters. We label this operation as the *feature extraction*.

It is noteworthy to mention that the proposed down-sampling operation is different from 3D-GCN [15], which only takes the cosine similarity. While still being scale-invariant, hence suitable for object classification and segmentation, they ignore the metric structure of the local 3D geometry; consequently, making completion difficult because the original scale of the local geometry is missing.

Neighbor pooling. The final step in our down-sampling operation is to reduce the size of $\mathcal{F}_{\text{out}}$ with pooling. However, unlike Graph Max-Pooling (GMP) [15], that takes the element-wise maximum value of the feature across all the vectors, we select the subset of feature vectors with the highest activations. Therefore, while GMP disassembles their features as part of their pooling operation, we preserve the feature descriptors from $\mathcal{F}_{\text{out}}$. From the definition of $\mathcal{F}_{\text{out}}$ in (4), we base our activation for each vector $\mathbf{f}_a$

$$\mathcal{A}_a = \sum_{b=1}^{D_{\text{out}}} \tanh |g_b(\mathbf{f}_a)| \tag{5}$$

on the results of $g(\cdot)$ from (2). Thereafter, we only take the $\frac{1}{\tau}$ of the number of feature vectors with the highest activations.

3.2. Up-sampling operation

The up-sampling and pooling operations in the encoder reduce the point cloud to a latent vector. In this case, if we directly use the operation in (4), the first layer in the decoder ends up with one vector since $|\mathcal{F}_{\text{in}}|$ is one. Subsequently, all

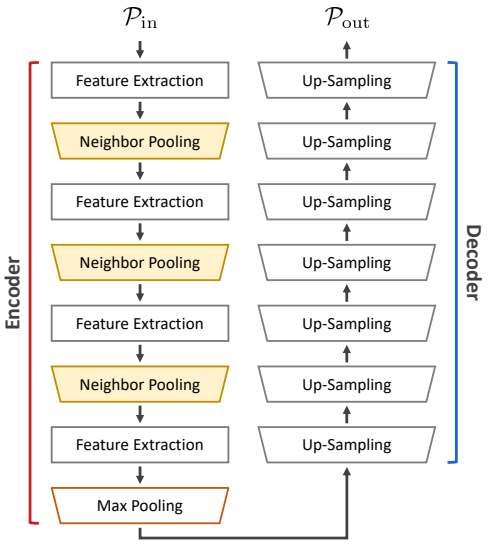


Figure 3. This architecture is composed of the proposed operators to build its encoder and decoder.

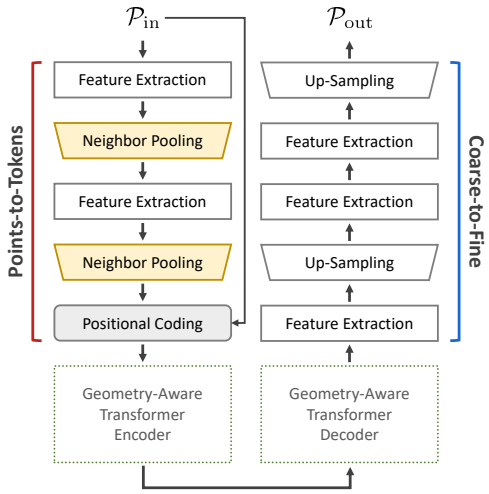


Figure 4. This architecture is derived from the transformers backbone, where we use the proposed operators to convert the input 3D points to tokens and to perform the coarse-to-fine strategy.

the other layers in the decoder result in a single vector. To solve this issue, our up-sampling iteratively runs (4) so that, denoting $\mathcal{F}_{\text{in}}$ as the input to the layer, we build the set of output feature vectors as

$$\begin{aligned}\mathcal{F}_{\text{up}} &= \{\mathcal{F}_{\text{out}}^{u}\}_{u=1}^{N_{\text{up}}} \\ &= \left\{[g_b^u(\mathbf{f}_a) + h_b^u(\mathbf{f}_a)]_{b=1}^{D_{\text{out}}}\right\}_{a=1,u=1}^{a=|\mathcal{F}_{\text{in}}|,u=N_{\text{up}}}\end{aligned} \tag{6}$$

which increases the number of vectors by N_{up}. As a result, $\mathcal{F}_{\text{up}}$ is a set of $N_u \cdot |\mathcal{F}_{\text{in}}|$ feature vectors. In addition to the list of hyper-parameters in Sec. 3.1, our up-sampling operation also takes N_{up} as a hyper-parameter.

4. Encoder-decoder architectures

In order to uncover the strengths of our operators in Sec. 3 (*i.e.* feature extraction, neighbor pooling and up-sampling), we used them as building blocks to construct two different architectures. The first directly implements our operators to build an encoder-decoder while the second takes advantage of our operators to improve the transformers derived from PoinTr [44]. We refer the readers to the Supplementary Materials for the detailed parameters of the architectures.

4.1. Direct application

The objective of the first architecture is to establish that building it solely from the proposed operators (with the additional max-pooling) can already be competitive in point cloud completion. We then propose an encoder-decoder architecture based on our operators alone as shown in Fig. 3.

The encoder is composed of four alternating layers of feature extraction and neighbor pooling. As the number of points from the input is reduced by 128 times, we use a max-pooling operator to extract a vector as our latent feature. Taking the latent feature from the encoder, the decoder is then constructed from a series of up-sampling operators, resulting in a fine completion of 16,384 points.

4.2. Transformers

The second architecture aims at showing the diversity of the operators to improve the state-of-the-art from PoinTr [44] that uses transformers. We therefore propose a transformer-based architecture that is derived from [44] and our operators as summarized in Fig. 4.

Before computing the attention mechanisms in the transformer, the partial scan are subsampled due to the memory constraint of the GPU. PoinTr [44] implements the Farthest Point Sampling (FPS) to reduce the number of points and MLP to convert the points to features. Conversely, our architecture applies the proposed operators. Similar to Sec. 4.1, this involves alternating the features extraction and neighbor pooling. Since the Fourier feature [28] and SIRENs [26] have proven that the sinusoidal activation is helpful in presenting complex signals and their derivatives in layer-by-layer structures, a positional coding based on the 3D coordinates is then added to the features. In Fig. 4, we refer this block as *points-to-token*. Thereafter, we use the geometry-aware transformers from [44] which produces a coarse point cloud.

From the coarse point cloud, we then replace their coarse-to-fine strategy with our operators. This includes a series of alternating feature extraction and up-sampling operators as shown in Fig. 4.

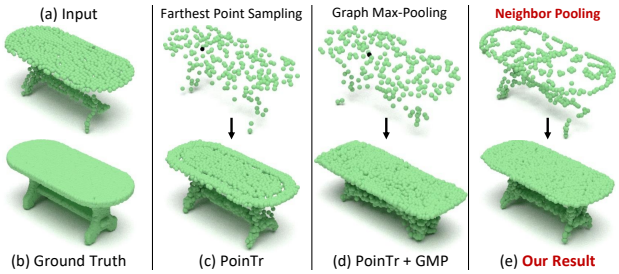


Figure 5. The first row compares the point tokens chosen by Farthest Point Sampling (FPS) in PointTr [44], Graph Max-Pooling (GMP) [15] in PointTr [44] and our proposed neighbor pooling in our transformer architecture. These tokens are then fed to the transformer and the coarse-to-fine strategy to produce the reconstruction shown in the second row.

It it noteworthy to emphasize the difference between our architecture from PoinTr [44] and to understand the implication of the changes. The contributions of points-to-tokens and coarse-to-fine to the overall architecture is illustrated in Fig. 5. We can observe from this figure that the FPS from PoinTr [44] only finds the distant points while the results of our neighbor pooling sketches the contours of the input point cloud to capture the meaningful structures of the object. Notably, by looking at our sketch, we can already identify the that the object is a table. This is contrary to the random points from PoinTr [44]. Moreover, our coarse-to-fine strategy uniformly reconstructs the planar region on the table as well as its base. Later, in Sec. 7, we numerically evaluate these advantages in order to show that the individual components has their own merits.

Since we previously discussed in Sec. 3.1 the difference of our down-sampling operation against 3D-GMP [15], we became curious to see the reconstruction in Fig. 5 if we replace the FPS in PoinTr [44] with the cosine similarity and GMP of [15]. Similar to PoinTr, the new combination selects distant points as its tokens while the table in their final reconstruction increased in size. In contrast, our tokens are more meaningful and the final results are more accurate.

5. Loss functions

Given the input point cloud $\mathcal{P}_{\text{in}}$ (*e.g.* from a depth image), the objective of completion is to build the set of points $\mathcal{P}_{\text{out}}$ that fills up the missing regions in our input data. Since we train our architecture in a supervised manner, we denote $\mathcal{P}_{\text{gt}}$ as the ground truth.

Completion. To evaluate the predicted point cloud, we impose the Earth-moving distance [9]. Comparing the output points to the ground truth and vice-versa, we end up

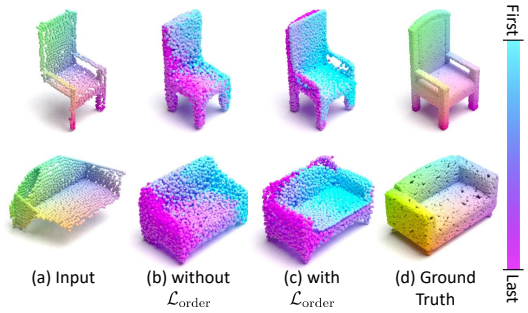


Figure 6. Compares the order of the point clouds reconstructed in the object completion with and without $\mathcal{L}_{\text{order}}$

with

$$\mathcal{L}_{\text{out}\to\text{gt}} = \sum_{p\in\mathcal{P}_{\text{out}}} \|p - \phi_{\text{gt}}(p)\|_2 \tag{7}$$

$$\mathcal{L}_{\text{gt}\to\text{out}} = \sum_{p\in\mathcal{P}_{\text{gt}}} \|p - \phi_{\text{out}}(p)\|_2 \tag{8}$$

where $\phi_i(p)$ is a bijective function that finds the closest point in the point cloud $\mathcal{P}_i$ to p.

Order of points in $\mathcal{P}_{\text{out}}$. After training with (7) and (8), we noticed that the points in the output reconstruction are ordered from left to right as shown in Fig. 6(b). We want to take advantage of this organization and investigate this behavior further. Assuming the idea that, among the points in $\mathcal{P}_{\text{out}}$, we are confident that the input point cloud must be part of it, we introduce a loss function that enforces that the first subset in $\mathcal{P}_{\text{out}}$ is similar to $\mathcal{P}_{\text{in}}$. We formally write this loss function as

$$\mathcal{L}_{\text{order}} = \sum_{p\in\mathcal{P}_{\text{in}}} \mathcal{S}(\theta_{\text{out}}(p)) \cdot \|p - \phi_{\text{out}}(p)\|_2 \tag{9}$$

where $\theta_{\text{out}}(p)$ is the index of the closest point in $\mathcal{P}_{\text{out}}$ based on $\phi_{\text{out}}(p)$ while

$$\mathcal{S}(\theta) = \begin{cases} 1, & \text{if } \theta \leq |\mathcal{P}_{\text{in}}| \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

is a step function that returns one if the index is within the first $|\mathcal{P}_{\text{in}}|$ points.

When we plot the results with $\mathcal{L}_{\text{order}}$ in Fig. 6(c), we noticed that the order in $\mathcal{P}_{\text{out}}$ moves from the observed to the occluded. In addition, fine-grained geometrical details such as the armrest of the chair are visible when training with $\mathcal{L}_{\text{order}}$; thus, improving the overall reconstruction.

Semantic scene completion. In addition to the architecture in Sec. 4 and the loss functions in (7), (8) and (9) for completion, a semantic label is added to each point in the predicted cloud $\mathcal{P}_{\text{out}}$. Given N_c categories, we denote the

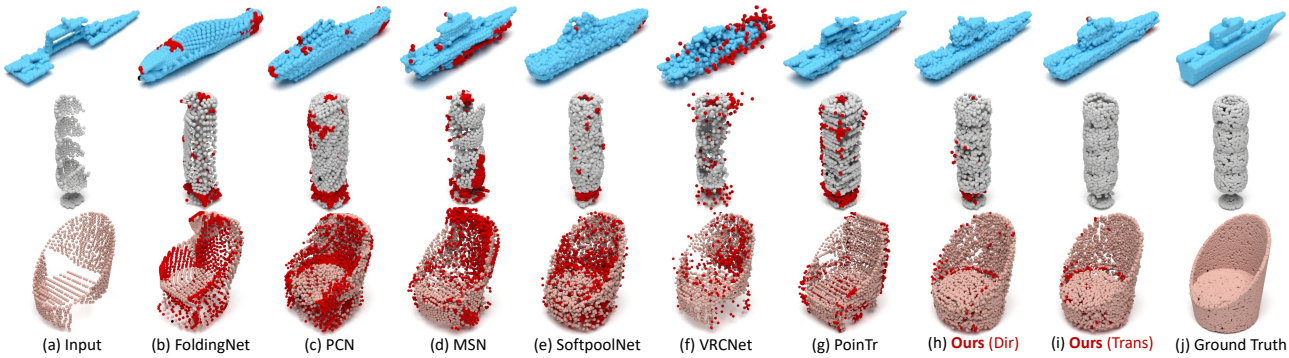


Figure 7. Object completion results where we highlight the errors in red points.

label for each point as a one-hot code $\mathbf{l}_i = [l_{i,c}]_{c=1}^{n_c}$ for the i-th point in $\mathcal{P}_{\text{out}}$ and the c-th category. Since training is supervised, the ground truth point clouds are also labeled with the semantic category.

After establishing the correspondence between the predicted point cloud to the ground truth in (7) in training, we also extract the ground truth semantic label $\hat{\mathbf{l}}_i$. It then follows that the binary cross-entropy of the i-th point is computed

$$\epsilon_i = -\frac{1}{N_c}\sum_{c=i}^{N_s} \hat{l}_{i,c}\log l_{i,c} + (1-\hat{l}_{i,c})(1-\log l_{i,c}) \quad (11)$$

and formulate the semantic loss function as

$$\mathcal{L}_{\text{semantic}} = \frac{\gamma}{|\mathcal{P}_{\text{in}}|}\sum_{i=i}^{|\mathcal{P}_{\text{in}}|}\epsilon_i \quad (12)$$

where the weight

$$\gamma = \frac{0.01}{\mathcal{L}_{\text{out}\to\text{gt}} + \mathcal{L}_{\text{gt}\to\text{out}}} \quad (13)$$

triggers to increase the influence of the $\mathcal{L}_{\text{semantic}}$ in training as the completion starts to converge. Note that γ is an important factor, since the output point cloud is erratic in the initial iterations, which means that it can abruptly change from one iteration to the next before the completion starts converging.

6. Experiments

To highlight the strengths of the proposed method, this section focuses on two experiments – object completion and semantic scene completion.

6.1. Object completion

We evaluate the geometric completion of a single object on the ShapeNet [3] database where they have the point clouds of the partial scans as input and their corresponding ground truth completed shape. The input scans are composed of 2,048 points while the database provides a low resolution output of 2,048 points and a high resolution of 16,384 points. We follow the standard evaluation on 8 categories where all objects are roughly normalized into the same scale with point coordinates ranging between -1 to 1.

Numerical results. We conduct our experiments based on three evaluation strategies from Completion3D [29], PCN [45] and MVP [20]. Evaluating on 8 objects (*plane*, *cabinet*, *car*, *chair*, *lamp*, *sofa*, *table*, *vessel*), they measure the predicted reconstruction through the L2-Chamfer distance, L1-Chamfer distance and the F-Score@1%, respectively. Note that, in this paper, we also follow the standard protocol where the value presented for the Chamfer distance is multiplied by 10^3. Although Table 1 only shows the average results across all categories, we refer the readers to the supplementary materials for the more detailed comparison.

One of the key observations in this table is the capacity of our direct architecture to surpass most of the other methods' results. Among 11 approaches, our Chamfer distance is only worse than 3 methods while our F-Score@1% is better than all of them. This therefore establishes the strength of our operators since our first architecture is solely composed of it. Moreover, our second architecture, which combines our operators with the transformer, reduces the error by 3-5% on the Chamfer distance and increases the accuracy by 4.5% on the F-Score@1%.

The table also examines the effects of $\mathcal{L}_{\text{order}}$ to our reconstruction. Training with $\mathcal{L}_{\text{order}}$ improves our results by 0.12-0.13 in Chamfer distance and 0.013-0.021 in F-Score@1%, validating our observations in Fig. 6.

Qualitative results. We compare our object completion results in Fig. 7 with the recently proposed methods: FoldingNet [43], PCN [45], MSN [16], SoftPoolNet [33], VRCNet [20] and PoinTr [44]. The red points in the figure highlight the errors in the reconstruction. All the approaches reconstructs a point cloud with 16,384 points with the excep-

Method	Completion3D *L2-Chamfer*	PCN *L1-Chamfer*	MVP *F-Score@1%*
FoldingNet [43]	19.07	14.31	–
SoftPoolNet [33]	11.07	9.20	0.666
TopNet [29]	14.25	12.15	0.576
PCN [45]	18.22	9.64	0.614
MSN [16]	–	9.97	0.690
GRNet [39]	10.64	8.83	0.677
ECG [19]	–	–	0.736
NSFA [47]	–	–	0.770
CRN [30]	9.21	8.51	0.724
SCRN [31]	9.13	8.29	–
VRCNet [20]	8.12	–	0.781
PoinTr [44]	9.22	8.38	0.741
ASFM-Net [38]	6.68	–	–
Ours (Direct)	8.35	8.46	0.801
–*without* $\mathcal{L}_{\text{order}}$	8.47	8.59	0.788
–*input* $\mathcal{P}_{\text{gt}}$	5.11	5.37	0.923
Ours (Transformer)	**6.64**	**7.96**	**0.816**
–*without* $\mathcal{L}_{\text{order}}$	6.74	8.09	0.795
–*input* $\mathcal{P}_{\text{gt}}$	4.46	4.95	0.962

Table 1. Evaluation on Completion3D [29], PCN [45] and MVP [20] datasets with their corresponding metrics for the object completion task.

tion for FoldingNet with 2,048 points and MSN with 8,192.

Since FoldingNet and PCN take advantage of their mathematical assumption where they rely on deforming one or more planar grids, they tend to over-smooth their reconstruction where finer details such as the boat is flattened. In contrast, our method can perform better on the smooth regions as well as the finer structures. Nevertheless, the more recent approaches like [16,20,33,44] can also produce more descriptive reconstruction on the boat. However, they produce more errors which is highlighted in the unconventional lamp or chair. Overall, our reconstructions are closer to the ground truth.

Failure cases. In addition to the qualitative results, we also examine the failure cases in Fig. 8. Most of them are objects with unusual structures like the car without the wheels. Another issue is when there is an insufficient amount of input point cloud to describe the object such as

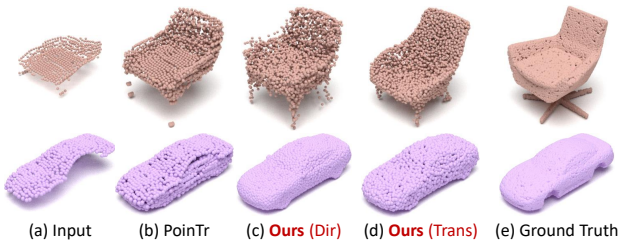


Figure 8. Examples of the failure cases in object completion.

Method	Resolution	*Average IoU*
Lin *et al.* [14]	60	12.0
Geiger and Wang [10]	60	19.6
SSCNet [27]	60	30.5
VVNet [12]	60	32.9
SaTNet [17]	60	34.4
ForkNet [32]	80	37.1
CCPNet [46]	240	38.5
SketchSSC [4]	60	41.1
SISNet [2]	60	**52.4**
Ours (Direct)	60	40.0
–*with* $\gamma = 1$ *in* $\mathcal{L}_{\text{semantic}}$	60	37.2
Ours (Transformer)	60	42.4
–*with* $\gamma = 1$ *in* $\mathcal{L}_{\text{semantic}}$	60	38.9

Table 2. Semantic scene completion on NYU [25] dataset. The value in resolution (x) is the output volumetric resolution which is $x \times 0.6x \times x$.

the chair. Notably, compared to the state-of-the-art, our reconstructions are still better in these situations.

6.2. Semantic scene completion

This evaluation aims at reconstructing the scene from a single depth image through a point cloud or an SDF volume where each point or voxel is categorized with a semantic class. Originally introduced for 2.5D semantic segmentation, NYU [25] and ScanNet [6], which were later annotated for semantic completion by [27, 36], are among the most relevant benchmark datasets in this field. These datasets include pairs of depth image and the corresponding semantically labeled 3D reconstruction.

Semantic scene completion with voxels. NYU are provided with real scans for indoor scenes which are acquired with a Kinect depth sensor. Following SSCNet [27], the semantic categories include 12 classes of varying shapes and sizes: *empty space*, *ceiling*, *floor*, *wall*, *window*, *chair*, *bed*, *sofa*, *table*, *tvs*, *furniture* and *other objects*.

Since the other point cloud completion do not handle semantic segmentation, we start our evaluation by comparing with the voxel-based approaches which perform the both the completion and the semantic segmentation such as [2, 4, 10, 12, 14, 17, 27, 32, 46]. Considering that the volumetric data evaluates through the IoU, we need to convert our point clouds to voxel grids to make the comparison.

One of the significant advantage of point clouds over voxels is that we are not constrained to a specific resolution. Since most method evaluate on $60 \times 36 \times 60$, we converted our point cloud to this resolution. Our approach achieves competitive average IoU of 42.4% which is better than all the other methods except for SISNet [2]. However, it is noteworthy to mention that our method faces additional errors associated to the conversion from point cloud to vox-

Method	CompleteScanNet	NYU
FoldingNet [43]	11.25	14.66
AtlasNet [11]	8.92	10.12
PCN [45]	8.19	9.98
MSN [16]	7.28	8.65
SoftPoolNet [33]	8.27	9.29
GRNet [39]	4.56	5.80
VRCNet [20]	4.29	5.45
PoinTr [44]	5.08	5.92
Ours (Direct)	3.17	4.72
Ours (Transformer)	**3.04**	**4.38**

Table 3. Evaluation on CompleteScanNet [36] and NYU [25] dataset for scene completion, measuring the average Chamfer distance trained with L2 distance (multiplied by 10^3) with the output resolution of 16,384.

els. In addition, the ground truth voxels for the furnitures in the NYU dataset is a solid volume which is not a plausible format for point cloud approaches which focuses more on the surface reconstruction. This in effect decreases the IoU of our method.

Moreover, Table 2 includes a small ablation study to verify the contribution of γ from (13) in $\mathcal{L}_{\text{semantic}}$. If we discard (13) by setting γ to one, the IoU for our models decrease by 7.5-9%; thus, proving the advantage in adaptively weighing the semantic loss function.

Point cloud scene completion. Another relevant dataset is from ScanNet [6] which was supplemented with the ground truth semantic completion by CompleteScanNet [36]. This include a total of 45,451 paired partial scan and semantic completion for training. Our evaluation in Table 3 takes 2,048 points as input and reconstructs the scene with 16,384 points. Since there is no previous work that focused on point cloud scene completion, we compare against methods that were designed for a single object completion such as PCN [45], MSN [16], SoftPoolNet [33] and GRNet [39]. Based on our evaluation in Table 3, both versions of our architectures attain the best results. Notably, we also compared these methods on the NYU dataset in Table 3. Similarly, the proposed architectures also achieve the state-of-the-art in point cloud scene completion.

7. Ablation study

This section focuses on the strengths of our operator in our transformer architecture. Although we adapt the transformer from PoinTr [44], we argue that every component we added is significant to the overall performance. To evaluate this, we disentangle the points-to-tokens and coarse-to-fine blocks. In practice, we separate the backbone, which takes points in the partial scan as input and outputs a coarse point cloud, from the coarse-to-fine strategy. Evidently, in our approach, the points-to-tokens block is part of the backbone.

Since most methods can also be separated in this manner, we then compose Table 4 to mix-and-match different backbones with different coarse-to-fine methods for object and scene completion. In both tables, we classified the other coarse-to-fine methods as: (1) *deform* which includes the operation in deforming 3D grids; (2) *deconv* which processes with MLP, 1D or 2D deconvolutions; and, (3) Edge-aware Feature Expansion (*EFE*) [19]. We then highlight the originally proposed architectures in yellow.

For any given backbone in every row, our coarse-to-fine method produces the best results. Moreover, for any given coarse-to-fine strategy in every column, our backbone performs the best. Therefore, this study essentially proves that each of the proposed components in our transformer architecture has a significant role in the overall performance.

8. Conclusion

We propose three novel operators for point cloud processing. To bring out the value of these operators, we apply them on two novel architectures that are designed for object completion and semantic scene completion. The first assembles together the proposed operators in an encoder-decoder fashion, while the second incorporates them in the context of transformers. Notably, both architectures produce highly competitive results, with the latter achieving the state of the art in point cloud completion for both objects and scenes.

OBJECT COMPLETION

Backbone	Coarse-to-Fine			
	deform	deconv	EFE	**Ours**
MSN [16]	7.28	9.34	7.15	6.91
PoinTr [44]	5.48	5.71	4.91	3.76
SoftPoolNet [33]	10.08	8.27	7.65	7.63
GRNet [39]	9.25	5.61	5.26	4.90
VRCNet [20]	8.09	8.88	5.08	4.21
Ours	4.93	4.99	4.12	**3.04**

SCENE COMPLETION

Backbone	Coarse-to-Fine			
	deform	deconv	EFE	**Ours**
MSN [16]	9.97	12.31	9.26	9.08
PoinTr [44]	8.38	8.49	8.31	8.13
TreeGAN [24]	14.26	9.72	9.12	9.05
SoftPoolNet [33]	11.73	9.20	8.75	8.64
GRNet [39]	9.12	8.83	8.73	8.51
VRCNet [20]	10.03	10.20	8.52	8.26
Ours	8.19	8.30	8.07	**7.96**

Table 4. Mix-and-match evaluation on different backbone attached to different coarse-to-fine methods for object and scene completion. The originally proposed combinations are marked in yellow.

References

[1] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, volume 1611, pages 586–606. International Society for Optics and Photonics, 1992. 3

[2] Yingjie Cai, Xuesong Chen, Chao Zhang, Kwan-Yee Lin, Xiaogang Wang, and Hongsheng Li. Semantic scene completion via integrating instances and scene in-the-loop. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 324–333, 2021. 1, 7

[3] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 1, 2, 6

[4] Xiaokang Chen, Kwan-Yee Lin, Chen Qian, Gang Zeng, and Hongsheng Li. 3d sketch-aware semantic scene completion via semi-supervised structure prior. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4193–4202, 2020. 2, 7

[5] Yang Chen and Gérard Medioni. Object modelling by registration of multiple range images. *Image and vision computing*, 10(3):145–155, 1992. 3

[6] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5828–5839, 2017. 1, 7, 8

[7] Angela Dai, Charles Ruizhongtai Qi, and Matthias Nießner. Shape completion using 3d-encoder-predictor cnns and shape synthesis. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume 3, 2017. 1, 2

[8] Angela Dai, Daniel Ritchie, Martin Bokeloh, Scott Reed, Jürgen Sturm, and Matthias Nießner. Scancomplete: Large-scale scene completion and semantic segmentation for 3d scans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4578–4587, 2018. 1

[9] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 605–613, 2017. 5

[10] Andreas Geiger and Chaohui Wang. Joint 3d object and layout inference from a single rgb-d image. In *German Conference on Pattern Recognition*, pages 183–195. Springer, 2015. 7

[11] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. A papier-mâché approach to learning 3d surface generation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 1, 2, 8

[12] Yuxiao Guo and Xin Tong. View-volume network for semantic scene completion from a single depth image. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. AAAI Press, 2018. 2, 7

[13] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. In *Advances in Neural Information Processing Systems*, pages 820–830, 2018. 1

[14] Dahua Lin, Sanja Fidler, and Raquel Urtasun. Holistic scene understanding for 3d object detection with rgbd cameras. In *Proceedings of the IEEE international conference on computer vision*, pages 1417–1424, 2013. 7

[15] Zhi-Hao Lin, Sheng-Yu Huang, and Yu-Chiang Frank Wang. Convolution in the cloud: Learning deformable kernels in 3d graph convolution networks for point cloud analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1800–1809, 2020. 3, 5

[16] Minghua Liu, Lu Sheng, Sheng Yang, Jing Shao, and Shi-Min Hu. Morphing and sampling network for dense point cloud completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 11596–11603, 2020. 1, 2, 3, 6, 7, 8

[17] Shice Liu, YU HU, Yiming Zeng, Qiankun Tang, Beibei Jin, Yinhe Han, and Xiaowei Li. See and think: Disentangling semantic scene completion. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 263–274. Curran Associates, Inc., 2018. 7

[18] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1

[19] Liang Pan. Ecg: Edge-aware point cloud completion with graph convolution. *IEEE Robotics and Automation Letters*, 5(3):4392–4398, 2020. 7, 8

[20] Liang Pan, Xinyi Chen, Zhongang Cai, Junzhe Zhang, Haiyu Zhao, Shuai Yi, and Ziwei Liu. Variational relational point completion network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8524–8533, 2021. 1, 2, 3, 6, 7, 8

[21] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2019. 1

[22] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660, 2017. 1, 2

[23] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems (NIPS)*, 2017. 1, 2

[24] Dong Wook Shu, Sung Woo Park, and Junseok Kwon. 3d point cloud generative adversarial network based on tree structured graph convolutions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3859–3868, 2019. 8

[25] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *European conference on computer vision*, pages 746–760. Springer, 2012. 2, 7, 8

[26] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33, 2020. 4

[27] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017. 1, 2, 7

[28] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *NeurIPS*, 2020. 4

[29] Lyne P Tchapmi, Vineet Kosaraju, Hamid Rezatofighi, Ian Reid, and Silvio Savarese. Topnet: Structural point cloud decoder. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 383–392, 2019. 6, 7

[30] Xiaogang Wang, Marcelo H. Ang Jr. , and Gim Hee Lee. Cascaded refinement network for point cloud completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 7

[31] Xiaogang Wang, Marcelo H Ang Jr, and Gim Hee Lee. A self-supervised cascaded refinement network for point cloud completion. *arXiv preprint arXiv:2010.08719*, 2020. 7

[32] Yida Wang, David Joseph Tan, Nassir Navab, and Federico Tombari. Forknet: Multi-branch volumetric semantic completion from a single depth image. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 8608–8617, 2019. 1, 2, 7

[33] Yida Wang, David Joseph Tan, Nassir Navab, and Federico Tombari. Softpoolnet: Shape descriptor for point cloud completion and classification. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 70–85, Cham, 2020. Springer International Publishing. 1, 2, 3, 6, 7, 8

[34] Xin Wen, Tianyang Li, Zhizhong Han, and Yu-Shen Liu. Point cloud completion by skip-attention network with hierarchical folding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2

[35] Xin Wen, Peng Xiang, Zhizhong Han, Yan-Pei Cao, Pengfei Wan, Wen Zheng, and Yu-Shen Liu. Pmp-net: Point cloud completion by learning multi-step point moving paths. *arXiv preprint arXiv:2012.03408*, 2020. 1, 2

[36] Shun-Cheng Wu, Keisuke Tateno, Nassir Navab, and Federico Tombari. Scfusion: Real-time incremental scene reconstruction with semantic completion. *arXiv preprint arXiv:2010.13662*, 2020. 2, 7, 8

[37] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9621–9630, 2019. 1

[38] Yaqi Xia, Yan Xia, Wei Li, Rui Song, Kailang Cao, and Uwe Stilla. Asfm-net: Asymmetrical siamese feature matching network for point completion. *arXiv preprint arXiv:2104.09587*, 2021. 2, 7

[39] Haozhe Xie, Hongxun Yao, Shangchen Zhou, Jiageng Mao, Shengping Zhang, and Wenxiu Sun. Grnet: Gridding residual network for dense point cloud completion. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 365–381, Cham, 2020. Springer International Publishing. 1, 2, 7, 8

[40] Qiangeng Xu, Weiyue Wang, Duygu Ceylan, Radomir Mech, and Ulrich Neumann. Disn: Deep implicit surface network for high-quality single-view 3d reconstruction. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 492–502. Curran Associates, Inc., 2019. 1

[41] Bo Yang, Stefano Rosa, Andrew Markham, Niki Trigoni, and Hongkai Wen. Dense 3d object reconstruction from a single depth view. *IEEE transactions on pattern analysis and machine intelligence*, 2018. 1, 2

[42] Bo Yang, Hongkai Wen, Sen Wang, Ronald Clark, Andrew Markham, and Niki Trigoni. 3d object reconstruction from a single depth view with adversarial learning. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 679–688, 2017. 1, 2

[43] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 206–215, 2018. 1, 2, 3, 6, 7, 8

[44] Xumin Yu, Yongming Rao, Ziyi Wang, Zuyan Liu, Jiwen Lu, and Jie Zhou. Pointr: Diverse point cloud completion with geometry-aware transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12498–12507, 2021. 1, 2, 4, 5, 6, 7, 8

[45] Wentao Yuan, Tejas Khot, David Held, Christoph Mertz, and Martial Hebert. Pcn: Point completion network. In *2018 International Conference on 3D Vision (3DV)*, pages 728–737. IEEE, 2018. 1, 2, 3, 6, 7, 8

[46] Pingping Zhang, Wei Liu, Yinjie Lei, Huchuan Lu, and Xiaoyun Yang. Cascaded context pyramid for full-resolution 3d semantic scene completion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7801–7810, 2019. 7

[47] Wenxiao Zhang, Qingan Yan, and Chunxia Xiao. Detail preserved point cloud completion via separated feature aggregation. *arXiv preprint arXiv:2007.02374*, 2020. 7

Learning Local Displacements for Point Cloud Completion
Supplementary

Yida Wang[1], David Joseph Tan[2], Nassir Navab[1], Federico Tombari[1,2]
[1]Technische Universität München [2]Google Inc.

1. Supplementary materials

As we discussed in the paper, this document aims at showing the detailed parameters of our architectures and more comprehensive results for both object completion and semantic scene completion. It also includes additional qualitative results that compares different methods against the proposed.

1.1. Parameters in architectures

This work introduces two architectures to highlight the benefits of the proposed layers. We list the parameters set in every layer of our direct architecture in Table 1 and our transformer architecture in Table 2.

1.2. Object completion

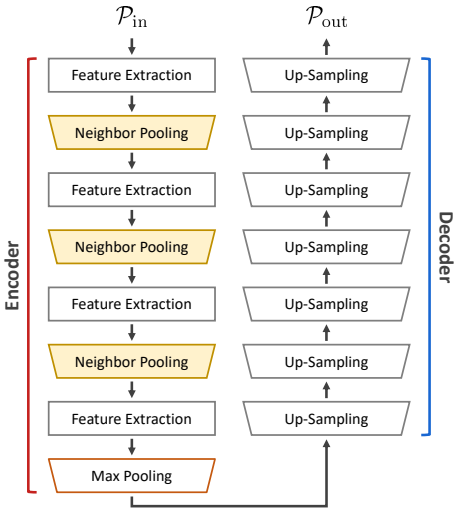
We exhibit a more detailed comparison on the object completion evaluation in Table 3, Table 4 and Table 5 for the Completion3D [14], PCN [26] and MVP [11] datasets, respectively. While we only show the average results in the paper, these tables show the per-category evaluation. Based on these results, our architectures are better in most categories when evaluating the Chamfer distance in Table 3 and Table 4; while, better in all categories when evaluating the F-Score in Table 5.

1.3. Semantic scene completion with voxels

Since most of the point cloud approaches only perform completion, we compared our semantic scene completion results to the voxel-based approaches in Table 6. In order to do this, we converted our high resolution point cloud to a lower resolution $60 \times 36 \times 60$ voxels. Table 6 shows the per-category comparison against the voxel-based approaches. Notably, although downsizing our point cloud introduces errors and difference (*e.g.* the objects in the point cloud are hollow while in the voxels are solid), we still achieve competitive IoU results.

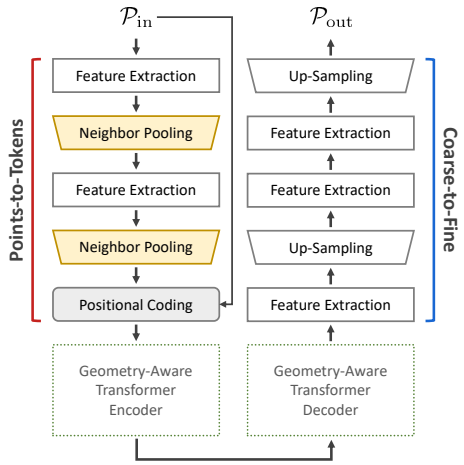
1.4. Semantic scene completion with point clouds

We illustrate the semantic scene completion results in Fig. 1, evaluated on CompleteScanNet [21]. Since there is no other point cloud completion approach that explicitly claim that they can reconstruct scenes, we utilize the architectures that were designed for object completion: PCN [26], MSN [8], PoinTr [25] and VRCNet [11]. Due to this, in Fig. 1, we perform the more complicated semantic completion while the other methods carry out the simpler



Layers	Parameters		
Feature Extraction	$s = 10$	$D_{\text{out}} = 16$	
Neighbor Pooling	$\tau = 8$		
Feature Extraction	$s = 10$	$D_{\text{out}} = 64$	
Neighbor Pooling	$\tau = 4$		
Feature Extraction	$s = 10$	$D_{\text{out}} = 64$	
Neighbor Pooling	$\tau = 4$		
Feature Extraction	$s = 10$	$D_{\text{out}} = 64$	
Max Pooling	–		
Up-Sampling	$s = 5$	$N_{\text{up}} = 2$	$D_{\text{out}} = 256$
Up-Sampling	$s = 10$	$N_{\text{up}} = 8$	$D_{\text{out}} = 64$
Up-Sampling	$s = 10$	$N_{\text{up}} = 4$	$D_{\text{out}} = 64$
Up-Sampling	$s = 10$	$N_{\text{up}} = 4$	$D_{\text{out}} = 32$
Up-Sampling	$s = 10$	$N_{\text{up}} = 8$	$D_{\text{out}} = 3$
Up-Sampling	$s = 10$	$N_{\text{up}} = 1$	$D_{\text{out}} = 3$
Up-Sampling	$s = 10$	$N_{\text{up}} = 8$	$D_{\text{out}} = 3$

Table 1. Parameters in each layer of our *direct* architecture.



Layers	Parameters
Feature Extraction	$s = 10$ $D_{\text{out}} = 16$
Neighbor Pooling	$\tau = 4$
Feature Extraction	$s = 10$ $D_{\text{out}} = 64$
Neighbor Pooling	$\tau = 4$
Positional Coding	–
Transformer	*Similar to [25]*
Feature Extraction	$s = 10$ $D_{\text{out}} = 64$
Up-Sampling	$s = 10$ $N_{\text{up}} = 8$ $D_{\text{out}} = 3$
Feature Extraction	$s = 10$ $D_{\text{out}} = 64$
Feature Extraction	$s = 10$ $D_{\text{out}} = 64$
Up-Sampling	$s = 10$ $N_{\text{up}} = 8$ $D_{\text{out}} = 3$

Table 2. Parameters in each layer of our *transformer* architecture.

completion task.

We observe from the other methods [8, 11, 25, 26] that their results show a high level of noise such that the objects in the scenes are no longer comprehensible. In comparison, our results have significantly less noise and produce reconstructions that are very similar to the ground truth. Moreover, a particular attention is given to PoinTr [25] since we derived our transformer architecture from them. Comparing our results against [25], our reconstructions are significantly more accurate. This in effect demonstrate the important contribution of our proposed layers to our transformer architecture.

References

[1] Yingjie Cai, Xuesong Chen, Chao Zhang, Kwan-Yee Lin, Xiaogang Wang, and Hongsheng Li. Semantic scene completion via integrating instances and scene in-the-loop. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 324–333, 2021. 5

[2] Xiaokang Chen, Kwan-Yee Lin, Chen Qian, Gang Zeng, and Hongsheng Li. 3d sketch-aware semantic scene completion via semi-supervised structure prior. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4193–4202, 2020. 5

[3] Angela Dai, Charles Ruizhongtai Qi, and Matthias Nießner. Shape completion using 3d-encoder-predictor cnns and shape synthesis. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume 3, 2017. 4

[4] Andreas Geiger and Chaohui Wang. Joint 3d object and layout inference from a single rgb-d image. In *German Conference on Pattern Recognition*, pages 183–195. Springer, 2015. 5

[5] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. A papier-mâché approach to learning 3d surface generation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 3, 4

[6] Yuxiao Guo and Xin Tong. View-volume network for semantic scene completion from a single depth image. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. AAAI Press, 2018. 5

[7] Dahua Lin, Sanja Fidler, and Raquel Urtasun. Holistic scene understanding for 3d object detection with rgbd cameras. In *Proceedings of the IEEE international conference on computer vision*, pages 1417–1424, 2013. 5

[8] Minghua Liu, Lu Sheng, Sheng Yang, Jing Shao, and Shi-Min Hu. Morphing and sampling network for dense point cloud completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 11596–11603, 2020. 1, 2, 4

[9] Shice Liu, YU HU, Yiming Zeng, Qiankun Tang, Beibei Jin, Yinhe Han, and Xiaowei Li. See and think: Disentangling semantic scene completion. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 263–274. Curran Associates, Inc., 2018. 5

[10] Liang Pan. Ecg: Edge-aware point cloud completion with graph convolution. *IEEE Robotics and Automation Letters*, 5(3):4392–4398, 2020. 4

[11] Liang Pan, Xinyi Chen, Zhongang Cai, Junzhe Zhang, Haiyu Zhao, Shuai Yi, and Ziwei Liu. Variational relational point completion network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8524–8533, 2021. 1, 2, 3, 4

[12] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems (NIPS)*, 2017. 4

[13] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene comple-

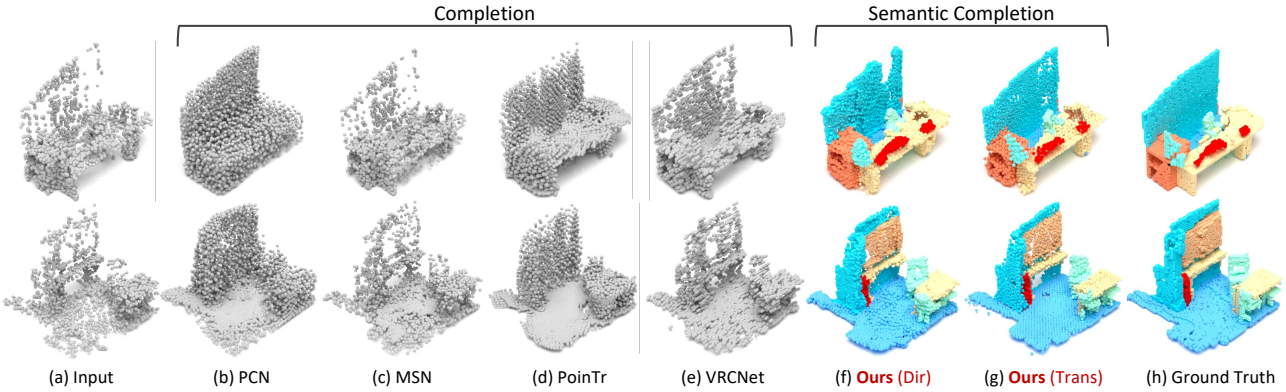


Figure 1. Semantic scene completion results on the CompleteScanNet [21] dataset

Output Resolution = 2,048, L2 metric, Completion3D [14] benchmark

Method	plane	cabinet	car	chair	lamp	sofa	table	vessel	*Avg.*
FoldingNet [24]	12.83	23.01	14.88	25.69	21.79	21.31	20.71	11.51	19.07
PointSetVoting [27]	6.88	21.18	15.78	22.54	18.78	28.39	19.96	11.16	18.18
AtlasNet [5]	10.36	23.40	13.40	24.16	20.24	20.82	17.52	11.62	17.77
PCN [26]	9.79	22.70	12.43	25.14	22.72	20.26	20.27	11.73	18.22
TopNet [14]	7.32	18.77	12.88	19.82	14.60	16.29	14.89	8.82	14.25
SA-Net [19]	5.27	14.45	7.78	13.67	13.53	14.22	11.75	8.84	11.22
SoftPoolNet [18]	6.39	17.26	8.72	13.16	10.78	14.95	11.01	6.26	11.07
GRNet [23]	6.13	16.90	8.27	12.23	10.22	14.93	10.08	5.86	10.64
PMP-Net [20]	3.99	14.70	8.55	10.21	9.27	12.43	8.51	5.77	9.23
CRN [15]	3.38	13.17	8.31	10.62	10.00	12.86	9.16	5.80	9.21
SCRN [16]	3.35	12.81	7.78	9.88	10.12	12.95	9.77	6.10	9.13
VRCNet [11]	3.94	10.93	6.44	9.32	8.32	11.35	8.60	5.78	8.12
ASFM-Net [22]	**2.38**	9.68	5.84	**7.47**	7.11	**9.65**	**6.25**	4.84	6.68
Ours (direct)	3.52	12.72	7.37	9.21	8.57	11.66	8.77	4.97	8.35
–without $\mathcal{L}_{\text{order}}$	3.64	12.83	7.48	9.34	8.70	11.79	8.88	5.07	8.47
Ours (transformer)	2.41	**9.54**	**4.99**	7.89	**6.89**	9.92	7.20	**4.29**	**6.64**
–without $\mathcal{L}_{\text{order}}$	2.48	9.62	5.10	7.99	7.01	10.04	7.29	4.39	6.74

Table 3. Evaluation on the object completion on Completion3D [14] benchmark based on the Chamfer distance trained with L2 distance (multiplied by 10^4) with the output resolution of 2,048.

tion from a single depth image. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017. 5

[14] Lyne P Tchapmi, Vineet Kosaraju, Hamid Rezatofighi, Ian Reid, and Silvio Savarese. Topnet: Structural point cloud decoder. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 383–392, 2019. 1, 3, 4

[15] Xiaogang Wang, Marcelo H. Ang Jr. , and Gim Hee Lee. Cascaded refinement network for point cloud completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 3, 4

[16] Xiaogang Wang, Marcelo H Ang Jr, and Gim Hee Lee. A self-supervised cascaded refinement network for point cloud completion. *arXiv preprint arXiv:2010.08719*, 2020. 3, 4

[17] Yida Wang, David Joseph Tan, Nassir Navab, and Federico Tombari. Forknet: Multi-branch volumetric semantic completion from a single depth image. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 8608–8617, 2019. 4, 5

[18] Yida Wang, David Joseph Tan, Nassir Navab, and Federico Tombari. Softpoolnet: Shape descriptor for point cloud completion and classification. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 70–85, Cham, 2020. Springer International Publishing. 3, 4

[19] Xin Wen, Tianyang Li, Zhizhong Han, and Yu-Shen Liu. Point cloud completion by skip-attention network with hierarchical folding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*,

Output Resolution = 16,384, L1 metric, PCN [26] dataset

Method	plane	cabinet	car	chair	lamp	sofa	table	vessel	*Avg.*
3D-EPN [3]	13.16	21.80	20.31	18.81	25.75	21.09	21.72	18.54	20.15
ForkNet [17]	9.08	14.22	11.65	12.18	17.24	14.22	11.51	12.66	12.85
PointNet++ [12]	10.30	14.74	12.19	15.78	17.62	16.18	11.68	13.52	14.00
FoldingNet [24]	9.49	15.80	12.61	15.55	16.41	15.97	13.65	14.99	14.31
AtlasNet [5]	6.37	11.94	10.11	12.06	12.37	12.99	10.33	10.61	10.85
TopNet [14]	7.61	13.31	10.90	13.82	14.44	14.78	11.22	11.12	12.15
PCN [26]	5.50	10.63	8.70	11.00	11.34	11.68	8.59	9.67	9.64
MSN [8]	5.60	11.96	10.78	10.62	10.71	11.90	8.70	9.49	9.97
SoftPoolNet [18]	6.93	10.91	9.78	9.56	8.59	11.22	8.51	8.14	9.20
GRNet [23]	6.45	10.37	9.45	9.41	7.96	10.51	8.44	8.04	8.83
PMP-Net [20]	5.65	11.24	9.64	9.51	**6.95**	10.83	8.72	7.25	8.73
CRN [15]	4.79	9.97	8.31	9.49	8.94	10.69	7.81	8.05	8.51
SCRN [16]	4.80	9.94	9.31	8.78	8.66	9.74	7.20	7.91	8.29
PoinTr [25]	4.75	10.47	8.68	9.39	7.75	10.93	7.78	7.29	8.38
Ours (direct)	5.34	**9.20**	**8.26**	8.96	9.40	**10.46**	7.54	8.56	8.47
–*without* $\mathcal{L}_{\text{order}}$	5.47	9.34	8.37	9.09	9.54	10.59	7.69	8.66	8.59
Ours (transformer)	**4.43**	10.03	8.28	**8.96**	7.29	10.55	**7.31**	**6.85**	**7.96**
–*without* $\mathcal{L}_{\text{order}}$	4.56	10.17	8.42	9.10	7.41	10.66	7.41	6.96	8.09

Table 4. Evaluation on the object completion on PCN [26] dataset based on the Chamfer distance trained with L1 distance (multiplied by 10^3) with the output resolution of 16,384.

Output Resolution = 16,384, F-Score@1%, MVP [11] dataset

Method	plane	cabinet	car	chair	lamp	sofa	table	vessel	*Avg.*
TopNet [14]	0.789	0.621	0.612	0.443	0.387	0.506	0.639	0.609	0.576
PCN [26]	0.816	0.614	0.686	0.517	0.455	0.552	0.646	0.628	0.614
MSN [8]	0.879	0.692	0.693	0.599	0.604	0.627	0.730	0.696	0.690
SoftPoolNet [18]	0.843	0.568	0.636	0.623	0.698	0.568	0.680	0.71	0.666
GRNet [23]	0.853	0.578	0.646	0.635	0.710	0.580	0.690	0.723	0.677
ECG [10]	0.906	0.680	0.716	0.683	0.734	0.651	0.766	0.753	0.736
NSFA [29]	0.903	0.694	0.721	0.737	0.783	0.705	0.817	0.799	0.770
CRN [15]	0.898	0.688	0.725	0.670	0.681	0.641	0.748	0.742	0.724
VRCNet [11]	0.928	0.721	0.756	0.743	0.789	0.696	0.813	0.800	0.781
PoinTr [25]	0.888	0.681	0.716	0.703	0.749	0.656	0.773	0.760	0.741
Ours (direct)	0.926	0.738	0.766	0.783	0.837	0.709	0.829	0.821	0.801
–without $\mathcal{L}_{\text{order}}$	0.910	0.750	0.741	0.734	0.835	0.715	0.839	0.783	0.788
Ours (transformer)	**0.942**	**0.753**	**0.780**	**0.799**	**0.851**	**0.725**	**0.844**	**0.836**	**0.816**
–without $\mathcal{L}_{\text{order}}$	0.922	0.731	0.759	0.776	0.831	0.703	0.824	0.813	0.795

Table 5. Evaluation on the object completion on MVP [11] dataset based on the F-Score@1% trained with L2 Chamfer distance and the output resolution of 16,384.

June 2020. 3

[20] Xin Wen, Peng Xiang, Zhizhong Han, Yan-Pei Cao, Pengfei Wan, Wen Zheng, and Yu-Shen Liu. Pmp-net: Point cloud completion by learning multi-step point moving paths. *arXiv preprint arXiv:2012.03408*, 2020. 3, 4

[21] Shun-Cheng Wu, Keisuke Tateno, Nassir Navab, and Federico Tombari. Scfusion: Real-time incremental scene reconstruction with semantic completion. *arXiv preprint arXiv:2010.13662*, 2020. 1, 3

[22] Yaqi Xia, Yan Xia, Wei Li, Rui Song, Kailang Cao, and Uwe Stilla. Asfm-net: Asymmetrical siamese feature matching network for point completion. *arXiv preprint arXiv:2104.09587*, 2021. 3

[23] Haozhe Xie, Hongxun Yao, Shangchen Zhou, Jiageng Mao,

Method	res.	whole	ceil.	floor	wall	win.	chair	bed	sofa	table	tvs	furn.	objs	*Avg.*
Lin *et al*. [7]	60	36.4	0.0	11.7	13.3	14.1	9.4	29.0	24.0	6.0	7.0	16.2	1.1	12.0
Geiger and Wang [4]	60	44.4	10.2	62.5	19.1	5.8	8.5	40.6	27.7	7.0	6.0	22.6	5.9	19.6
SSCNet [13]	60	55.1	15.1	94.6	24.7	10.8	17.3	53.2	45.9	15.9	13.9	31.1	12.6	30.5
VVNet [6]	60	61.1	19.3	94.8	28.0	12.2	19.6	57.0	50.5	17.6	11.9	35.6	15.3	32.9
SaTNet [9]	60	60.6	17.3	92.1	28.0	16.6	19.3	57.5	53.8	17.7	18.5	38.4	18.9	34.4
ForkNet [17]	80	37.1	36.2	93.8	29.2	18.9	17.7	61.6	52.9	23.3	19.5	45.4	20.0	37.1
CCPNet [28]	240	63.5	23.5	96.3	35.7	20.2	25.8	61.4	56.1	18.1	28.1	37.8	20.1	38.5
SketchSSC [2]	60	71.3	43.1	93.6	40.5	24.3	30.0	57.1	49.3	29.2	14.3	42.5	28.6	41.1
SISNet [1]	60	**78.2**	**54.7**	93.8	**53.2**	**41.9**	**43.6**	**66.2**	**61.4**	**38.1**	**29.8**	**53.9**	**40.3**	**52.4**
Ours (direct)	60	63.7	38.1	97.1	37.0	15.5	18.7	55.2	54.9	29.6	21.4	49.2	23.7	40.0
–*with* $\gamma = 1$ *in* $\mathcal{L}_{\text{semantic}}$	60	58.2	35.1	94.3	34.0	12.7	15.8	52.3	52.0	26.7	18.4	46.3	20.9	37.2
Ours (transformer)	60	66.1	40.4	**98.6**	39.6	18.1	21.2	57.5	57.0	31.9	23.5	51.3	26.4	42.4
–*with* $\gamma = 1$ *in* $\mathcal{L}_{\text{semantic}}$	60	63.4	36.6	95.0	36.6	14.8	18.1	53.9	53.4	28.8	20.1	47.8	22.5	38.9

Table 6. Semantic completion on NYU dataset. The value in res. (x) is the output volumetric resolution which is $x \times 0.6x \times x$.

Shengping Zhang, and Wenxiu Sun. Grnet: Gridding residual network for dense point cloud completion. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 365–381, Cham, 2020. Springer International Publishing. 3, 4

[24] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 206–215, 2018. 3, 4

[25] Xumin Yu, Yongming Rao, Ziyi Wang, Zuyan Liu, Jiwen Lu, and Jie Zhou. Pointr: Diverse point cloud completion with geometry-aware transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12498–12507, 2021. 1, 2, 4

[26] Wentao Yuan, Tejas Khot, David Held, Christoph Mertz, and Martial Hebert. Pcn: Point completion network. In *2018 International Conference on 3D Vision (3DV)*, pages 728–737. IEEE, 2018. 1, 2, 3, 4

[27] Junming Zhang, Weijia Chen, Yuping Wang, Ram Vasudevan, and Matthew Johnson-Roberson. Point set voting for partial point cloud analysis. *arXiv preprint arXiv:2007.04537*, 2020. 3

[28] Pingping Zhang, Wei Liu, Yinjie Lei, Huchuan Lu, and Xiaoyun Yang. Cascaded context pyramid for full-resolution 3d semantic scene completion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7801–7810, 2019. 5

[29] Wenxiao Zhang, Qingan Yan, and Chunxia Xiao. Detail preserved point cloud completion via separated feature aggregation. *arXiv preprint arXiv:2007.02374*, 2020. 4

5.3 Generalizing Feature Learning for Other Tasks

Finally, inspired by the improved performance on 3D completion by feature learning, especially variational inference used in Section 5.1, we want to generalize feature optimization methods in this section for other 1D, 2D, and 3D tasks, including language translation, planar reconstruction, hand pose estimation, etc. Using a similar 2D encoder that produces variational latent features as introduced in Section 5.1.1, Section 5.3.1 proposes a variational architecture to estimate 3D key points on the hand from a single RGB image. Moreover, using variational inference for feature learning, we introduce a *nebula anchors* in Section 5.3.2 to form latent clusters, such that categorical information is learned in an implicit way which potentially benefits many tasks, including 3D completion as well.

5.3.1 Variational Object-aware 3D Hand Pose from a Single RGB Image (IEEE Robotics and Automation Letters 2019)

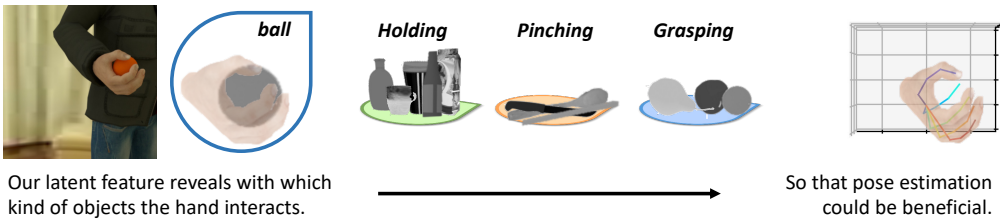


Figure 5.6 By understanding the type of the grasped object, our proposed model [9] can produce realistic hand gestures from a single RGB image even when the hand is partially occluded by the object.

Apart from 3D completion, as introduced in previous sections, there are other 3D tasks, including hand pose estimation which we focus on in this work. Considering that we have exploited approaches to modify the distribution of the latent features e.g. in AdversarialSSC [10] and ForkNet [8] to improve the performance, some feature learning techniques such as metric learning and variational inference are investigated in this work for 3D hand pose estimation from a single RGB image. Given the task of estimating 3D hand key points from a single RGB image, it is likely to be with occlusion because of the interacted objects. Here, we embed variational latent features to reconstruct the occluded parts by the decoders. Since the latent space is a compact representation of the input domain, numerous unrealistic components are removed during inference, which increases the accuracy of the outcome.

Intuitively, hand pose estimation could benefit from knowing the categories of the grasped object because the same category of objects has a similar scale and shape layout. We introduce additional features in the latent space that specialize in the shape and size of the grasped object to influence the final hand pose regression. Consequently, the deep architecture can internally infer the category of the grasped object to enhance the 3D reconstruction. When the object labels are available, metric learning can be adopted as well to form clusters in the latent space to reveal the object categories during inference. Then in the unsupervised case, we adopt an unsupervised clustering approach that could be optimized with SGD [165] aiming at forming clusters similar to those obtained via triplet training.

Existed 3D hand-object manipulation benchmarks with joint annotations are missing, so we propose a synthetic dataset with realistic images, hand masks, and 3D joint coordinates.

Contributions

Yafei Gao implemented baseline architectures and made the conducted synthetic hand-object dataset rendered with Blender.

Yida Wang implemented the clustering, variational inference, and metric learning.

Pietro Falco and Federico Tombari regulated the proposed synthetic dataset. They guided to demonstrate the real-world demos as well.

Nassir Navab financed this work on behalf of the leader of TUM CAMP.

Variational Object-Aware 3-D Hand Pose From a Single RGB Image

Yafei Gao[1,*], Yida Wang[1,*], Pietro Falco[3], Nassir Navab[1], Federico Tombari[1,2]

[1] Technical University of Munich
[2] Google
[3] ABB Corporate Research
[*] Equal contribution

Published version: https://doi.org/10.1109/LRA.2019.2930425

Variational Object-aware 3D Hand Pose from a Single RGB Image

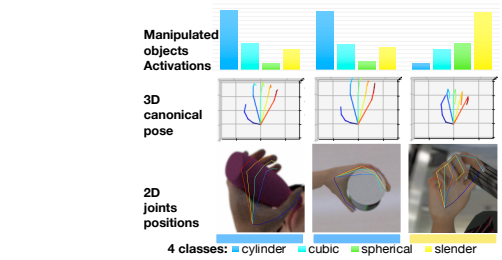
Yafei Gao[1,*], Yida Wang[1,*], Pietro Falco[2], Nassir Navab[1] and Federico Tombari[1,3]

***Abstract*—We propose an approach to estimate the 3D pose of a human hand while grasping objects from a single RGB image. Our approach is based on a probabilistic model implemented with deep architectures, which is used for regressing, respectively, the 2D hand joints heat maps and the 3D hand joints coordinates. We train our networks so to make our approach robust to large object- and self-occlusions, as commonly occurring with the task at hand. Using specialized latent variables, the deep architecture internally infers the category of the grasped object so to enhance the 3D reconstruction, based on the underlying assumption that objects of a similar category, i.e. with similar shape and size, are grasped in a similar way. Moreover, given the scarcity of 3D hand-object manipulation benchmarks with joint annotations, we propose a new annotated synthetic dataset with realistic images, hand masks, joint masks and 3D joints coordinates. Our approach is flexible as it does not require depth information, sensor calibration, data gloves, or finger markers. We quantitatively evaluate it on synthetic datasets achieving state-of-the-art accuracy, as well as qualitatively on real sequences.**

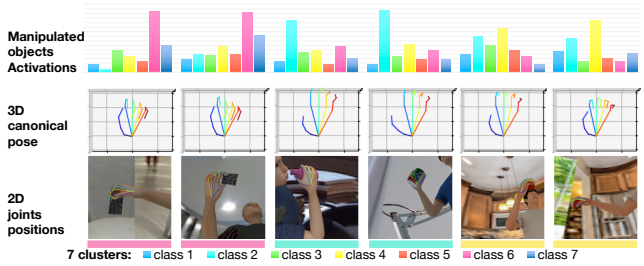
***Index Terms*—variational inference, triplet**

I. Introduction

HAND pose estimation is now a required technology for many emerging consumer applications such as virtual and augmented reality (VR, AR), robotics, gaming, and human-machine interface. Concerning robotics, a key problem in the scientific community is to program both stationary robots and modern mobile manipulators without strong technical background. The classical programming techniques can be optimal in industrial production lines where the environment is completely structured. However, in applications that require human-robot collaboration and in new areas of service robotics such as logistics, healthcare, and house automation, robotic systems have to be reprogrammed in an intuitive and easy way by nonexpert users. An effective way to instruct robots in an intuitive fashion is programming by demonstration, where the robot observes humans performing a task and learns to reproduce it. A key bottleneck, especially for manipulation tasks, is the observation of human hands while grasping and manipulating objects, as this presents the challenge of occlusion, since parts of the hand are dynamically occluded by the grasped object. Notably, also the other aforementioned applications such as AR, VR and gaming need to deal with hand-object interaction and occlusion.



(a) training with object labels



(b) training without object labels

Fig. 1: 3D hand pose estimation from monocular under heavy occlusion. The canonical hand joint configurations for objects of the same category are similar. Hence, we leverage specialized latent features to regress accurate hand pose based on the detected object's information.

In this paper, we address the problem of estimating the hand pose in 3D space while grasping objects using a single RGB camera. 3D hand pose means the 3D positions of the hand joints with respect to a frame fixed to the wrist. As it consumes data only from a monocular camera, our system does not require any depth sensor, sensor calibration, data gloves as in [1], or finger markers [2]. To the best of our knowledge, this is the first work that addresses discriminative hand pose estimation using monocular image as input while the hand interacts with an object. We propose a method based on two stages, each carried out via a deep architecture which are released here [1]. Given an RGB image, the first step is to filter the hand out of the environment and to generate a heat map that highlights the joints' location. To deal with the frequent occlusions of the hand caused by the grasped objects, a deep

Manuscript received: February, 24, 2019; Revised: May, 31, 2019; Accepted: July, 2, 2019.

This paper was recommended for publication by Editor Dongheui Lee upon evaluation of the Associate Editor and Reviewers' comments.

[1]CAMP, Technische Universität München, Boltzmannstr. 3 85748 Garching yafei.gao@tum.de, yida.wang@tum.de, navab@cs.tum.edu, tombari@in.tum.de

[2]Deptôf Automation Solutions, ABB Corporate Research, Forskargränd 7, 72178 Västerås, Sweden pietro.falco@se.abb.com

[3]Google Zurich

* The first two authors contributed equally to this work.

Digital Object Identifier (DOI): see top of this page.

[1]*https://github.com/wangyida/VO-handpose*

architecture with variational latent feature is purposely trained to reconstruct the occluded parts. Then, the 3D coordinates of the joints are estimated based on the generated heat map. In order to enhance the 3D pose estimation procedure, we introduce additional features in the latent space that specialize to the shape and size of the grasped object to influence the final hand pose regression. The underlying assumption for this is that, for grasping similar objects in terms of shape and size (e.g., from the same category), the joint (knuckle) configurations of the two grasping poses are similar to each other (see Fig. 1). In addition, given the current lack in literature of 3D hand-object manipulation benchmarks with joint annotations, we have generated a new annotated synthetic dataset that includes realistic images, hand masks, joint masks and 3D joints coordinates.

To summarize, the main contributions of this work are: i) a novel variational deep architecture to reconstruct a 2D hand mask in presence of large occlusions and to identify a heat-map of hand joints' locations using a monocular image; ii) a deep network that estimates 3D hand pose from the 2D heat map, leveraging latent features that identify the object category to accurately estimate the hand pose; iii) a novel synthetic dataset of hands grasping objects with rich annotations.

II. Related Works

A. 3D Bare-Hand Pose Estimation

Approaches for 3D bare hand pose estimation can be sorted into two categories: discriminative (appearance-based) and generative (model-based). While generative approaches classify input X with expected output Y by learning a model of joint probability $P(X, Y)$ and using Bayes rule to compute $P(Y|X)$, discriminative methods classify X by learning a direct map between X and Y or directly estimating the posterior probability.

Most generative approaches for hand pose try to fit a parametric model to the data by generating model hypotheses and evaluating them on the observed data [3], [4], [5]. Instead, discriminative approaches directly learn a forward-pass function from training data and establish a mapping from image to hand pose [6], [7]. In this case, a critical factor is represented by finding a suitable learning model which has the ability to handle a large amount of features and possible hand poses [8], [9]. As deep architectures handle more complex tasks, convolutional neural networks (CNNs) fully utilise stacked convolutional operations to predict 2D joint locations for real-time continuous pose recovery from a single depth image [10]. Depth images alse help estimate 3D poses [11], [12], [13]. With input of RGB images, 3D models are also used to produce 2D samples to learn 3D Orientation of objects [14]. Recently, Zimmermann et al. [15] propose a concatenated architecture to estimate hand segmentation, joints position and 3D poses sequentially.

B. Hand Pose Estimation with Object Interaction

Due to substantially increased occlusion caused by the objects, related work in this field primarily falls within the generative category and either assumes simplified working conditions (e.g., empty background) or employs additional input modalities (e.g., multi-view or depth data). A differentiable objective function for pose estimation is proposed in [16] where edges, optical flow, salient points and collisions are used to capture the motion of two hands interacting with an object on an empty background. Kyriazis [17] suggests an ensemble of collaborative trackers to handle multi-object scenarios based on RGB-D data as input. As for discriminative approaches, Romero proposes a hand pose retrieval approach for RGB images, where nearest neighbors from a large hand pose database are retrieved based on object shape information [18]. A discriminative top-down approach is proposed in [19] using CNNs, able to estimate the hand joints and object locations from a depth camera. First object pixels are segmented out, then a two-channel image containing both the input depth map and the masked depth map are used to regress the 3D joint position. However, experiments are proposed where only a tennis ball is used as interacting object, which exhibits a similar silhouette from diverse observation perspectives, consequently only simplified occlusion cases are taken into consideration. Another discriminative approach based on depth data is proposed by Choi [20], which uses parallel deep architectures and embeds object shape information into latent features. Two networks share intermediate observations produced from different perspectives to create a more informed representation. Instead of processing low-level data to detect or remove occluded regions, it exploits a CNN-based framework to extract grasp estimates from those regions. Interestingly, Choi's approach and our work share the assumption that there is a strong correlation between the object category and the grasping pose. Differently from Choi, we aim to solve the task using only monocular data.

C. Hand-Object Datasets

To the best of our knowledge, there exist the following fully-annotated hand-object datasets: Hand-Sphere Dataset [19], SynthHands Dataset [21], GANerated Hands Dataset [22], First-Person Hand Action [23], and Stereo Dataset [24]. Hand-Sphere [19] captures hands grasping spherical objects using a Kinect sensor and providing both hand segmentation and pose estimation. For segmentation task, paired depth maps and RGB images are provided with 5635 samples for training and 1042 for testing, while the pose estimation dataset consists of 3986 samples for training and 745 for testing. Hand-Sphere [19] lacks diversity of manipulated objects since the object has a similar silhouette from different viewpoints, hence provides limited types of occlusion. Also, this characteristic does not allow to evaluate the assumption that hand poses are correlated to the shape and category of the grasped object. SynthHands [21] is a synthetic dataset for hand pose estimation from depth and color data, with and without object interaction. It uses a merged reality approach to capture and synthesize large amounts of annotated data of natural hand interaction in cluttered scenes. Occluded hand and interacting objects are directly observed. However, to implement grasping hand pose estimation task, many interactions within this dataset are physically invalid. Recently, GANerated Hands Dataset [22] was

proposed, containing more than 330,000 color images of hands with 2D and 3D annotations of 21 keypoints on a synthetic hand model. The use of GANs help increase the realism of the dataset, nevertheless physically invalid joint configurations still exist. In this dataset, the hand pixels are already segmented and extracted. Stereo Dataset [24] is composed by 18000 stereo image pairs and 18000 depth images captured from different scenarios and the ground-truth 3D positions of palm and finger joints obtained from the manual label.

Our dataset differs in two main aspects. First, it includes a variety of objects with labels. Second, it is designed to realistically simulate hand-object interaction, hence facilitating the use of parametric models trained on our dataset on real data. A dataset with object labels is important as it allows us to test approaches based on object category information.

III. Methodology

This section describes the proposed approach for 3D hand pose estimation from a single image, devised to deal with large hand occlusion. Inspired by [15], we first segment the complete hand and regress joint locations as heatmaps, then use such joint locations to guide 3D pose estimation by regressing the 3D canonical and relative hand poses. As a reminder, 3D canonical poses are a set of absolute joint coordinates, while 3D relative poses depend from a specific viewpoint. In [15], three sub-architectures are used sequentially, namely HandSegNet, PoseNet and PosePrior. Since this approach estimates the pose of bare hands, it is not designed to deal with occlusions that occurr during hand-object interaction. To overcome this limitation, as depicted in Fig. 2, we propose to replace the discriminative model in [15] with a generative one based on two encoding-decoding networks and two learned latent spaces (z_h and z_j in Fig. 2), so to regress more robustly joint configurations and hand poses. Since the latent space is a compact representation of the input domain, we believe this approach enforces a large number of unrealistic poses to be dropped during inference, increasing the accuracy of the outcome.

In particular, first we replace HandSegNet (i.e., the network in [15] for hand segmentation) with a variational convolutional architecture based on an encoder-decoder pair and trained to regress hand masks and joint location heatmaps. This network is concatenated with another decoder, i.e. PoseNet in [15]. Then, we replace PosePrior, which estimates in parallel the canonical pose and the rotation matrix and obtains the relative pose by multiplying the canonical coordinate by the rotation matrix, with another variational auto-encoder, which is trained via triplet learning to regress 3D canonical and relative joint coordinates from the inferred joint heatmaps. Note that, similarly to [15], hand side is also necessary for pose inference. To describe more formally our pipeline in the following, let X be the input RGB image, while the outputs are a hand segmentation mask (Y_h), a multi-channel joint heatmap Y_j and a set of estimated 3D joint coordinates Y_p. Two latent features z_h and z_j, which are extracted from X and Y_j, are used to regress 2D maps Y_h, Y_j and 3D poses Y_p. We apply, on both sub-architectures, variational inference and triplet training.

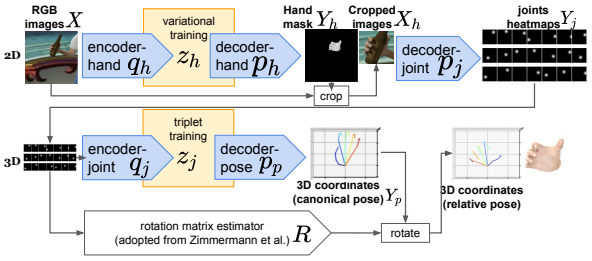


Fig. 2: Proposed architecture for 3D hand pose estimation.

A. 2D Hand Mask and Joint Heatmap

Supervised CNNs for segmentation such as DCN [25] and stacked CNN [26] can effectively perform hand segmentation only on those parts of the hand that are visible, so HandSegNet [15] can not segment well the hand in presence of occlusion even if trained with complete hand masks, this leading to errors nearby hand joints. As shown in Fig. 5, the segmented hand region generated by [15] mostly contains visible hand parts, resulting in disconnected components in presence of occluding objects. To solve this problem, we apply variational inference on a latent feature z_h of input X and split the 2D estimator into an encoder-decoder pair. Instead of using loss functions targeting hand segmentation directly, we set two constraints to enhance the ability to obtain the 2D hand mask in presence of occlusion:

- the decoder $p_h(z_h)$ always generates a complete hand from latent feature z_h;
- the encoder $q_h(X)$ generates a latent feature $z_h^{'}$ which is likely to produce a complete hand.

First, the input image X is encoded into the latent features z_h by encoder q_h, then used to generate a hand mask Y_h at 256×256 resolution via decoder p_h. Here the grasped object is removed once the hand mask is generated. Then, for the aim of extracting hand joint information in form of pixel-wise heat maps, Y_j are generated from a decoder $p_j()$ with a masked hand X_h as input. This means that p_j is concatenated after p_h, so we can also generate p_j directly from the latent features z_h via a combined decoder $p_j(p_h)$.

p_j is an extended convolutional architecture with 24 layers, that outputs a 21-channel heatmap of size $32 \times 32 \times 21$, each channel associated to one of the 21 joints. Hence, we combine the encoder and the 2 decoders together as $p_j(p_h(q_h))$, generating the 2D pose as the heat map Y_j from the image X. Both the encoder $q_h(X)$ and the decoder $p_h(z_h)$ are simultaneously optimised with a variational constraint [27] for latent variables. The Kullback-Leibler (KL) divergence $D[Q_h(z|X)||P_h(z|Y)]$ between the posterior $P_h(z|Y)$ and a likelihood $Q_h(z|X)$ is used to evaluate the capability of the encoder to generate latent features which are likely to produce the expected target Y:

$$E_{z\sim Q}[\log Q_h(z|X) - \log P_h(Y|z) - \log P(z)] + \log P(Y) \ . \quad (1)$$

Since the likelihood term $Q_h(z|X)$ is hardly tractable, variational inference [27] solves this problem by redefining a specific encoding function $q_h(\cdot)$ with latent features $z = q_h(X)$ following a Gaussian distribution $N(0, I)$, such that the

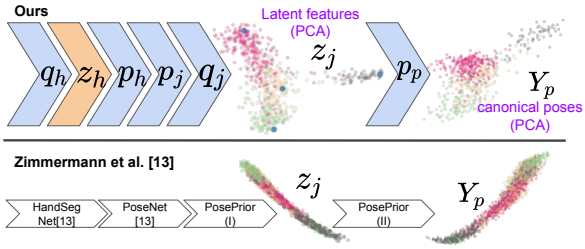


Fig. 3: Difference of data distribution after triplet training is applied for 3D pose estimation. Different colors represent different categories.

associated probability density function $Q_h(z|X)$ is expected to fit the posterior $P_h(z|Y)$. We modify equation (1) as follows to obtain the cost function for our encoder and decoder:

$$\begin{aligned}\mathcal{L}_{enc-hand} &= D[Q_h(z|X)||P(z)] \\ \mathcal{L}_{gen-hand} &= -E_{z\sim Q}[\log P_h(Y_h|z)] \ .\end{aligned} \tag{2}$$

The loss function in (2) can be interpreted as: the conditional log likelihood $\log P_h(X|z)$ used in variational auto-encoder [28] is replaced with $\log P_h(Y|z)$ where the expected network output is different from the input, so that the decoding process can be carried out for different targets Y.

Once the completed hand segmentation network is trained with equation (2), we use the generated hand mask to crop the original image X. In this case, the hand is centered in the image while all fingers are included for further processing. We can define a loss function for the decoder simply by replacing $p_h(z_h)$ with $p_j(p_h(z_h), X)$, i.e. the likelihood $P_h(Y_h|z)$ to generate hand masks is changed to generate joints maps $P_j(Y_j|z, X)$; thus the final loss function used to regress the 2D joint heatmap is

$$\mathcal{L}_{gen-joint} = -E_{z\sim Q}[\log P_j(Y_j|z, X)] \ . \tag{3}$$

Finally, we train hand segmentation and joint estimation together with the following loss:

$$\mathcal{L}_{2d} = \mathcal{L}_{enc-hand} + \mathcal{L}_{gen-hand} + \mathcal{L}_{enc-joint} + \mathcal{L}_{gen-joint} \ . \tag{4}$$

According to the architectural design, to get those latent features we apply 3 fully-connected layers with dimensions $[256, 128, 128]$ respectively, followed by ReLU [29] to process the output of the 18th convolutional layer in HandSegNet [15] and the output of the first fully-connected layer in PosePrior [15]. The decoder of our 2D hand joint estimator relies on the convolutional architecture used for people detection in [30], with which is concatenated with the decoder for hand segmentation.

B. Object-aware 3D Hand Pose Estimation

As the last step of our model, we estimate the canonical and real 3D hand poses based on the images cropped by the generated 2D joint heatmaps Y_j. The canonical 3D hand pose [31] defines a 3D pose which is rotation invariant and independent of the camera view, so that a set of 2D hand joints can be projected into the same 3D canonical pose even if their real poses are different. Here we also apply variational inference on the latent features to make the predicted 3D pose more stable. Since humans grasp objects with a strategy that depends on the size and shape of the object, the 3D canonical hand poses should be similar to each other when grasping similar objects. Fig. 3 shows that the distribution of 3D hand poses (trained without using any explicit object label) becomes correlated to object categories. We then conclude that 3D hand pose estimation in presence of objects can benefit from knowing the object category, should such information be available in advance. However, at test time we want to predict the hand pose just from an RGB image, without any additional information. Therefore, instead of using object labels as an additional input, we implicitly add category information to the learned latent feature.

When estimating the 3D hand pose from the heatmap Y_j using the combined encoder-decoder architecture q_j, p_p, we propose to use triplet training to optimize the latent features z_{pose} produced by q_j, so that they form clusters driven by object categories. Since most datasets do not provide object labels for training, we introduce an approach to learn object-related latent clusters, by introducing additional latent variables which are used as cluster centers. Our unsupervised clustering could be used within stochastic gradient descent (SGD) [32] to train a deep network, which would not be possible for other clustering methods such as K-means [33].

Object-driven Latent Features via Triplet Training In case the training dataset contains annotations regarding the category of the manipulated object (such as for the proposed HOP dataset described in Sec. IV), we want to learn latent features that are similar when the category of the grasped object is the same. To push latent features to cluster together driven by object categories, we use metric learning. Metric learning optimises feature distributions with relative information between training samples: the distribution of latent features changes so that the features which are expected to produce similar outputs are also close to each other. Triplet training [34] was initially applied to face recognition [35]. We apply a triplet cost function inspired from [36] together with a pairwise term. The triplet loss function is computed from triplets, i.e. three instances of the same feed-forward network with shared weights [34]. Each triplet is composed of a reference sample, a positive sample and a negative sample. We use z^{ref} to denote the feature of reference input X^{ref} which is processed by function f.

In our case, f is the concatenated architecture of joint estimator $p_j(p_h(q_h))$ and encoder q_p. $f(X^{pos})$ and $f(X^{neg})$ denote, respectively, the positive (same label as X^{ref}) and negative (different label as X^{ref}) anchors of the triplet. As metric for training we use the Euclidean distance. As an example, imagine a triplet with 3 hands holding, respectively, a bottle, a mug and a tomato. The reference sample is grasping a bottle. As hands have a similar configuration when grasping a bottle and a mug, and a significantly different one when holding a tomato, the hand with mug is labeled as positive sample while the hand holding the tomato is regarded as a negative sample.

During training, positive features z^{pos} are those belonging to

the same category as the reference feature z^{ref}, while negative features z^{neg} are those belonging to different ones. We set a loss function to make squared Euclidean metric $||z^{ref}-z^{neg}||_2^2$ larger than $||z^{ref}-z^{pos}||_2^2$:

$$\mathcal{L}_{triplet} = \sum \ln(\max(1, 2 - \frac{||z^{ref}-z^{neg}||_2^2}{||z^{ref}-z^{pos}||_2^2+m})) + \sum ||z^{ref}-z^{pos}||_2^2 \ , \quad (5)$$

where m is the margin for triplet. A pair-wise term $|z^{ref}-z^{pos}|^2$ is used for moving latent features on the boundaries of two categories, since the triplet term does not influence a lot for those samples. The dimensionality of both of the latent features z_h and z_p is 256. We choose it empirically through a grid search approach from 2^4 to 2^{10} to balance the fitting capacity for data and time efficiency.

Unsupervised Latent Feature Optimisation In case the training data does not provide object category labels, we adopt an unsupervised clustering approach which could be used with SGD [32] aiming at forming clusters similar to those obtained via triplet training. Although K-means [33] clustering is simple enough to form clusters without using additional information, it can only be applied on a set of features without changing values. Since latent features are continuously updated during training, K-means would not converge easily. To solve this problem, we introduce N additional latent variables $\{c_1, c_2, \ldots, c_N\}$ acting as cluster centers, having the same dimension as the latent features. Suppose that batch size is M, latent variables $\{z_1, z_2, \ldots, z_M\}$ are labeled with the index of its nearest center $c_m \in \{c_1, c_2, \ldots, c_N\}$ during training. The loss function $\mathcal{L}_{cluster}$ enforces those variables holding the same label to stay close to their cluster center. At the same time, the center itself moves towards the region of space where variables with the same label are present. The clustering loss function $\mathcal{L}_{cluster}$ is determined as below:

$$\mathcal{L}_{cluster} = \sum_{m=1}^{M} ||z_m - c_n||_2^2 \ . \quad (6)$$

The additional variables $\{c_1, c_2, \ldots, c_N\}$ are randomly initialized from a Gaussian distribution. Once this unsupervised clustering method is applied, triplet training still works when the indices of latent variables are almost fixed. Thus, the supervised and unsupervised loss functions can be used together for metric learning:

$$\mathcal{L}_{metric} = \mathcal{L}_{triplet} + \mathcal{L}_{cluster} \quad (7)$$

We use the squared Euclidean distance $\mathcal{L}_{pose} = ||Y_p - p_p(z_j)||_2^2$ between the expected canonical 3D pose and the output Y_p of decoder p_p as regression loss for the pose estimation. The overall loss function hence becomes:

$$\mathcal{L}_{3d} = \mathcal{L}_{triplet} + \mathcal{L}_{cluster} + \mathcal{L}_{pose} \quad (8)$$

Another advantage of enforcing such cluster centers is that the network can be trained with categorical supervision as cluster center of every category. It is then possible to exploit these centers at test time to infer the category of the grasped object, e.g. by comparing the distances between the latent feature and each cluster center. Since the predicted canonical poses are camera-view independent, we apply a rotation matrix R to change Y_p into a camera related pose, and we apply fully-connected layers with linear activation based on input Y_j. We adopt the architecture proposed by [15] by adding loss function $||R_{gt} - R||_2^2$ to $\mathcal{L}_{3d}$. In the end, the relative 3D poses are optimised with

$$\mathcal{L}_{3d_r} = \mathcal{L}_{pose} + ||R_{gt} - R||_2^2 + \mathcal{L}_{cluster} + \mathcal{L}_{triplet} \quad (9)$$

where the $\mathcal{L}_{triplet}$ term is optional and used only when object labels are available.

(a) Scenes (b) 21 joint (c) Objects (4 categories)

Fig. 4: Characteristics of the proposed HOP dataset

IV. Hand-Object Dataset for 3D Pose Estimation (HOP)

The datasets described in Sec. II-C are not sufficient for training an object-oriented hand pose estimation network due to the lack of the necessary variations of objects and shapes. We propose a new dataset dubbed Hand-Object Pose (HOP), which contains 11,820 pairs of RGB images and masks at 320×320 resolution, with 800 samples to be used for testing. Hand poses include 21 3D joints, which are manually created according to physical grasping pose and degrees of freedom (DoF) of each joint. Then they are used to precisely annotating CAD models. Images encompass 5 female and 5 male subjects who grasp 30 different objects with 600 randomly rendered background images. We assign category labels for each image based on the characteristics of the object present therein. The dataset includes 4 object categories: i) cylindrical objects (e.g., bottle, can, milk carton), ii) bars and sticks (e.g., pencil, fork, chopsticks), iii) cubical objects (e.g., book, smart phone, cutting board), iv) spherical objects (e.g., orange, ball, tomato).

Dataset Generation: With the help of MakeHuman [2], an open source computer graphics software for prototyping of photo realistic 3D avatars, we obtained 3D models with skeleton in different body shapes, skin colors and ages. 3D object models are obtained from *TurboSquid*[3] and human activities dataset [37]. We imitate physical human hand movement and manually create series of interaction animations between a human and an object using Blender[4]. Hand animations were captured from different viewpoints. Two point lamps randomly placed in each scene ensure the diversity of illumination conditions. After fixing the location of camera and light sources, background images, which exclude people or animals

[2] *http://www.makehumancommunity.org*
[3] *https://www.turbosquid.com*
[4] *https://www.blender.org*

in the scene, are selected from www.pexels.com. Mask images of both isolated hand and isolated object for each scene are generated by Blender as well, which may be useful for future research in object segmentation. We used Cycles Renderer[5], a physical-based unbiased path tracing engine designed for animations, to produce photo-realistic renders.

Annotations: As we placed a standing human model in the origin, 3D hand joint coordinates and object locations are automatically obtained according to the relative position of the center of the base of support (BoS). Object categories are manually defined. The dataset also provides intrinsic camera matrix and 3D keypoint positions in camera coordinate system as well. The synthetic dataset and the corresponding source code (python-blender) will be publicly released.

V. Experiments

We evaluate our work on both bare hand datasets and our object manipulation dataset, via quantitative and qualitative results. For ablation purposes, we first independently test the 2D joint estimation and 3D pose estimation stages. When testing the 3D pose estimation stage alone, we feed ground-truth heatmaps as input. In addition, we also test the whole architecture composed of all proposed stages.

A. 2D Hand Segmentation

As we are analysing our 2D processing architectures, only $\{q_h, p_h, p_j\}$ are optimised and tested. We compare performance in terms of completed hand segmentation on synthetic data using HandSegNet in [15] and our variational 2D hand estimator. We use 30000 samples from the rendered hand dataset (RHD) [15] and 10220 samples from the proposed HOP dataset together to train the 2D hand segmentation network. The network is randomly initialized and trained using ADAM [38] optimiser for 120,000 iterations. The learning rate is 1×10^{-5} for the first 60,000 iterations, 1×10^{-6} for the following 30,000 iterations and 1×10^{-7} until the end. We quantitatively evaluate the performance of our methods on completed hand segmentation and joint estimation compared to HandSegNet and PoseNet in [15]. We show results in Fig. 5 under different challenging factors: (1) huge occlusion, (2) small occlusion, (3) complex background, (4) skin interference and (5) data migration from synthetic to real scenes.

The estimated 2D joints, hand masks, and estimated 3D joints (zoomed in) are shown in Fig. 5. Our approach overcomes occlusions from grasped objects better than [15]. One common problem of completed hand segmentation networks is skin interference, which means that such networks tends to segment other body parts out of the background instead of the hand. Fig. 5 also shows that our method performs much better even when face and hand overlaps. Fig. 5, bottom shows that our method works for real scenarios even when the parametric model is trained on synthetic data only, thanks to the realism of the proposed dataset (Sec. IV), as well as the capability of our inference model to handle the domain shift from synthetic to real data. The accuracy of the

[5]*https://www.cycles-renderer.org/*

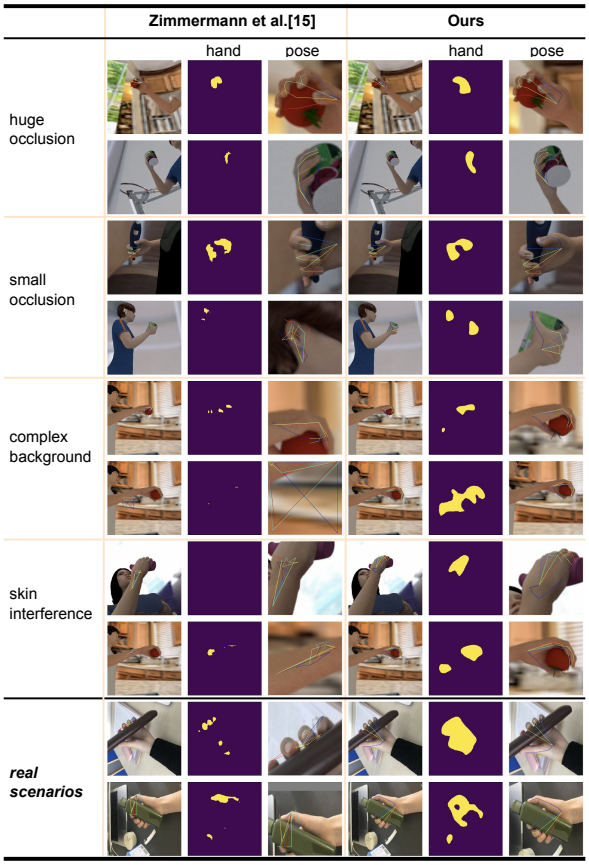


Fig. 5: Qualitative results of hand segmentation masks and poses of cropped hands using masks on HOP dataset.

estimated hand masks proves that our variational training for the encoder-decoder architecture is effective, also avoiding to split the hand into several components around the object. One limitation highlighted by these results are the recurring blurred boundaries of the hand mask, which sometimes affects the segmentation of fingers.

TABLE I shows quantitative results for 2D joint estimation on the RHD [15] and the HOP datasets. We report the area under the curve (AUC) on the percentage of correct keypoints (PCK) with 20 pixels as threshold, the median value of endpoint error (EPE median) in pixels and the average endpoint error (EPE mean) in pixels. The Table shows how *Our-PoseNet* is effective on both HOP and RHD [15] datasets, achieving the best results with an AUC of, respectively, 0.775 and 0.704. The purpose of testing on RHD is to show that we also have good performance on bare hand pose estimation. In this case, triplet training is not adopted as there are no object labels, though we can still use the proposed variational 2D joint estimation architecture and the 3D pose estimator with the cluster loss function.

Compared to HS-PoseNet [15], our variational embedding yields a 0.032 improvement on AUC. If we use Hourglass [39], i.e. a deep architecture initially applied for human joint estimation, as a replacement for PoseNet in [15], the AUC increases to 0.741. Accordingly, we replaced PoseNet with

data	method	AUC	EPE median	EPE mean
HOP (ours)	HS-PoseNet [15]	0.722	4.285	13.392
	HS-Hourglass [39]	0.741	5.159	10.709
	Our-PoseNet	0.754	4.024	10.707
	Our-Hourglass	**0.775**	**3.791**	**8.496**
RHD [15]	HS-PoseNet [15]	0.635	6.745	18.741
	Our-PoseNet	**0.704**	**4.215**	**17.520**

TABLE I: 2D joint estimation by HS-PoseNet (i.e., HandSegNet + PoseNet as proposed in [15]) and by our approach (i.e., the proposed architecture $q_h - p_h$.) EPE are in pixels.

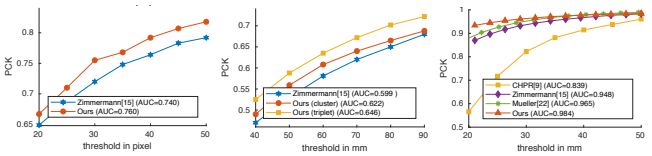
method	cylinder	slender	cubical	sphere	*Avg.*
HS-PoseNet [15]	0.694	0.773	0.767	0.701	0.734
Ours-PoseNet	**0.743**	**0.812**	**0.828**	**0.721**	**0.776**

TABLE II: Categorical joint estimation results on HOP dataset. HS-PoseNet is the joint HandSegNet + PoseNet architecture in [15]. Ours-PoseNet is the combination of q_h, p_h architecture and PoseNet (p_j). The AUC is calculated over the error range in 0 to 20mm.

Hourglass in our architecture as well (i.e., *Our-Hourglass*), obtaining an increased accuracy of 0.775, hence proving once again the effectiveness of our variational approach. Although RHD does not include object occlusion, our method still performs better than HandSegNet [15], since there are still self-occlusions in the free hand case. Fig. 6(a) illustrates the AUC on PCK for which the error thresholds range from 20 pixels to 50 pixels. For the evaluation on the HOP dataset, we tested the performance on each of the 4 categories separately. Both HandSegNet [15] and our work are trained under the same conditions from scratch. Our approach performs better than HandSegNet [15] in each category, with a category-wise average of AUC 0.776, which is 0.032 higher than HandSegNet.

B. 3D Pose Estimation

To compare the performance with a focus on 3D pose estimation only, we optimise only $\{q_j, p_p\}$ and feed ground-truth heatmaps as input, in order to unbias the comparison from the other stages of the pipeline. Given the correlation between the grasping pose and the object shape, first we train the proposed architecture q_j and p_p based on our HOP dataset with object labels. TABLE III shows AUC, median, and mean EPE for the RHD [15], GANeratedHands [22], Stereo [24] and our HOP datasets. For completeness, we tested both triplet training and unsupervised clustering. Since most available datasets lack object labels, we trained this variational model on HOP without labels. By learning the latent variables in an unsupervised way, we force the network to create hyper-clusters in the latent space. The number of clusters is not constant and is adjusted for each dataset. Specifically, we choose a number of clusters in the range of 4 - 10, eventually picking 5 for HOP, 7 for RHD [15], 10 for GANeratedHands [22] and Stereo [24]. Generally speaking, a high number of clusters tends to reduce the advantages brought in by clustering and metric learning, while quite small number of clusters tend to make the network a variational



(a) HOP (2D) (b) HOP (3D) (c) Stereo [24] (3D)

Fig. 6: Accuracy for 2D and 3D joint estimation as PCK over a threshold (pixels for 2D and mm for 3D) on the joint (RHD+HOP) dataset.

data	method	AUC	EPE median	EPE mean
RHD [15]	PosePrior [15]	0.555	18.932	28.804
	Ours (cluster)	**0.587**	**16.301**	**27.569**
GANeratedHands [22]	PosePrior [15]	0.977	7.665	8.790
	Ours (cluster)	**0.981**	**7.197**	**8.243**
Stereo [24]	CHPR [9]	0.839	-	-
	PosePrior [15]	0.948	9.543	11.064
	GANeratedHands [22]	0.965	-	-
	Ours (cluster)	**0.984**	**7.606**	**8.943**
HOP (ours)	PosePrior [15]	0.534	19.728	30.860
	Ours (triplet)	**0.597**	**15.901**	**27.326**
	Ours (cluster)	0.583	16.741	28.018

TABLE III: Results on RHD [15], GANeratedHands [22], Stereo [24] and our HOP with triplet training (triplet) and unsupervised clustering (cluster). EPE are in millimeters.

inference model. TABLE III shows the result of unsupervised training latent variables on those datasets respectively. The AUC ranges from 0.981 to 0.984 when we choose clusters between 4 and 10 in Stereo [24] dataset. Fig. 6(b) compares the PCK curve among PosePrior, our approach with clustering loss, and our approach with Triplet training, all trained on the HOP dataset. Combining the results shown in these tables, we can observe that (1) the proposed network efficiently improves the performance with respect to the original one, and (2) triplet training on latent variables using label information leads to better results than hyper clustering on latent space when labels are not available.

C. Comparison on the entire pipeline

Finally, we compare the performance of the whole pipeline based on all parametric models $\{q_h, p_h, p_j, q_j, p_p\}$. We evaluate the results by training on HOP. We report the AUC on the PCK for 20 millimeters threshold, the EPE median in millimeters, and the EPE mean in millimeters. TABLE IV shows that the proposed approach with variational inference and metric learning has a good AUC of 0.580 which is 0.016 higher than previous work [15]. When we are training the whole architecture and only apply unsupervised clustering on the latent features from q_j, the AUC improves to 0.669, which is more than 0.100 higher than [15]. Note that both supervised and unsupervised learning exploit object categories. In supervised learning categories are given by the user as labels, while in the unsupervised approach, they are clustered automatically. If we have labels and the generated hand mask is accurate enough, supervised learning performs better. If the generated hand mask has low quality, it is better to use clustering to automatically infer object categories.

method	AUC	EPE median	EPE mean
Zimmermann et al. [15]	0.543	32.003	45.581
Ours w/o variational	0.564	30.193	44.466
Ours (cluster)	**0.669**	**23.280**	**36.052**
Ours (triplet)	0.580	29.485	41.012

TABLE IV: Evaluation on joint training with 2D (q_h,p_h,p_j) and 3D (q_j,p_p) processing with triplet training (triplet) and unsupervised clustering (cluster). AUC is calculated over error range from $0mm$ to $50mm$.

VI. Conclusion and Future Work

Our proposed variational network with metric learning estimates the 3D hand pose while grasping an object from a single RGB image. We leverage the correlation between the hand pose and the category of the grasped object to design an effective architecture that does not require input 3D data. Since available datasets often do not include object category labels, a clustering method is introduced to group objects in an unsupervised fashion. Notably, in our approach the object category is the only information used for the objects. Both for supervised and unsupervised training, its validity is based on the assumption that hand poses for objects of the same category are similar. If a user grasps the same object in different and less natural ways, the performance of our architecture would decrease as this would invalidate such assumption. However, we believe that in many robotic applications (i.e., programming by demonstration) this assumption holds, since users typically train robots by grasping similar objects with similar hand configurations. An interesting future direction regards the use of contact information to correct the estimated 3D hand pose, guaranteeing consistency between pose and contact [40].

References

[1] J.-H. Kim, N. D. Thang, and T.-S. Kim, "3-d hand motion tracking and gesture recognition using a data glove," in *Int. Symp. Industrial Electronics (ISIE)*. IEEE, 2009.

[2] P. Falco, G. De Maria, C. Natale, and S. Pirozzi, "Data fusion based on optical technology for observation of human manipulation," *Int. J. Optomechatronics*, vol. 6, no. 1, pp. 37–70, 2012.

[3] I. Oikonomidis, N. Kyriazis, and A. A. Argyros, "Efficient model-based 3d tracking of hand articulations using kinect." in *BMVC*, vol. 1, no. 2, 2011, p. 3.

[4] C. Qian, X. Sun, Y. Wei, X. Tang, and J. Sun, "Realtime and robust hand tracking from depth," in *CVPR*, 2014.

[5] A. Tagliasacchi, M. Schröder, A. Tkach, S. Bouaziz, M. Botsch, and M. Pauly, "Robust articulated-icp for real-time hand tracking," in *Computer Graphics Forum*, 2015.

[6] M. Oberweger, P. Wohlhart, and V. Lepetit, "Training a feedback loop for hand pose estimation," in *ICCV*, 2015.

[7] Y. Cai, L. Ge, J. Cai, and J. Yuan, "Weakly-supervised 3d hand pose estimation from monocular rgb images," in *ECCV*, 2018.

[8] D. Tang, T.-H. Yu, and T.-K. Kim, "Real-time articulated hand pose estimation using semi-supervised transductive regression forests," in *ICCV*, 2013.

[9] X. Sun, Y. Wei, S. Liang, X. Tang, and J. Sun, "Cascaded hand pose regression," in *CVPR*, 2015.

[10] J. Tompson, M. Stein, Y. Lecun, and K. Perlin, "Real-time continuous pose recovery of human hands using convolutional networks," *TOG*, vol. 33, no. 5, p. 169, 2014.

[11] S. Yuan, G. Garcia-Hernando, B. Stenger, G. Moon, J. Yong Chang, K. Mu Lee, P. Molchanov, J. Kautz, S. Honari, L. Ge, *et al.*, "Depth-based 3d hand pose estimation: From current achievements to future goals," in *CVPR*, 2018, pp. 2636–2645.

[12] S. Li and D. Lee, "Point-to-pose voting based hand pose estimation using residual permutation equivariant layer," in *CVPR*, 2019, pp. 11 927–11 936.

[13] G. Moon, J. Yong Chang, and K. Mu Lee, "V2v-posenet: Voxel-to-voxel prediction network for accurate 3d hand and human pose estimation from a single depth map," in *CVPR*, 2018.

[14] M. Sundermeyer, Z.-C. Marton, M. Durner, M. Brucker, and R. Triebel, "Implicit 3d orientation learning for 6d object detection from rgb images," in *ECCV*, 2018.

[15] C. Zimmermann and T. Brox, "Learning to estimate 3d hand pose from single rgb images," in *ICCV*, 2017, pp. 4903–4911.

[16] L. Ballan, A. Taneja, J. Gall, L. Van Gool, and M. Pollefeys, "Motion capture of hands in action using discriminative salient points," in *ECCV*. Springer, 2012.

[17] N. Kyriazis and A. Argyros, "Scalable 3d tracking of multiple interacting objects," in *CVPR*, 2014.

[18] J. Romero, H. Kjellström, C. H. Ek, and D. Kragic, "Non-parametric hand pose estimation with object context," *Image and Vision Computing*, vol. 31, no. 8, pp. 555–564, 2013.

[19] D. Goudie and A. Galata, "3d hand-object pose estimation from depth with convolutional neural networks," in *FG*. IEEE, 2017.

[20] C. Choi, S. H. Yoon, C.-N. Chen, and K. Ramani, "Robust hand pose estimation during the interaction with an unknown object," in *CVPR*, 2017.

[21] F. Mueller, D. Mehta, O. Sotnychenko, S. Sridhar, D. Casas, and C. Theobalt, "Real-time hand tracking under occlusion from an egocentric rgb-d sensor," in *ICCV*, 2017.

[22] F. Mueller, F. Bernard, O. Sotnychenko, D. Mehta, S. Sridhar, D. Casas, and C. Theobalt, "Ganerated hands for real-time 3d hand tracking from monocular rgb," in *CVPR*, 2018.

[23] G. Garcia-Hernando, S. Yuan, S. Baek, and T.-K. Kim, "First-person hand action benchmark with rgb-d videos and 3d hand pose annotations," in *CVPR*, 2018.

[24] J. Zhang, J. Jiao, M. Chen, L. Qu, X. Xu, and Q. Yang, "3d hand pose tracking and estimation using stereo matching," *arXiv preprint arXiv:1610.07214*, 2016.

[25] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *ICCV*, 2015.

[26] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction," *ICANN*, pp. 52–59, 2011.

[27] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[28] M.-N. Tran, D. J. Nott, and R. Kohn, "Variational bayes with intractable likelihood," *JCGS*, vol. 26, no. 4, pp. 873–882, 2017.

[29] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[30] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, "Convolutional pose machines," in *CVPR*, 2016.

[31] A. Fossati, M. Dimitrijevic, V. Lepetit, and P. Fua, "From canonical poses to 3d motion capture using a single camera," *PAMI*, vol. 32, no. 7, pp. 1165–1181, 2010.

[32] M. Zinkevich, M. Weimer, L. Li, and A. J. Smola, "Parallelized stochastic gradient descent," in *NIPS*, 2010.

[33] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithm: analysis and implementation," *TPAMI*, vol. 24, no. 7, pp. 881–892, Jul 2002.

[34] E. Hoffer and N. Ailon, "Deep metric learning using triplet network," in *International Workshop on Similarity-Based Pattern Recognition*. Springer, 2015, pp. 84–92.

[35] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, "A discriminative feature learning approach for deep face recognition," in *ECCV*, 2016.

[36] P. Wohlhart and V. Lepetit, "Learning descriptors for object recognition and 3d pose estimation," in *CVPR*, 2015.

[37] A. Pieropan, G. Salvi, K. Pauwels, and H. Kjellström, "A dataset of human manipulation actions," in *ICRA Workshop on Autonomous Grasping and Manipulation*, 2014.

[38] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[39] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *ECCV*. Springer, 2016.

[40] P. Falco, C. Natale, and R. Dillmann, "Ensuring kinetostatic consistency in observation of human manipulation," *RAS*, vol. 61, no. 5, pp. 545 – 553, 2013.

5.3.2 Self-supervised Latent Space Optimization with Nebula Variational Coding (IEEE Transactions on Pattern Analysis and Machine Intelligence 2022)

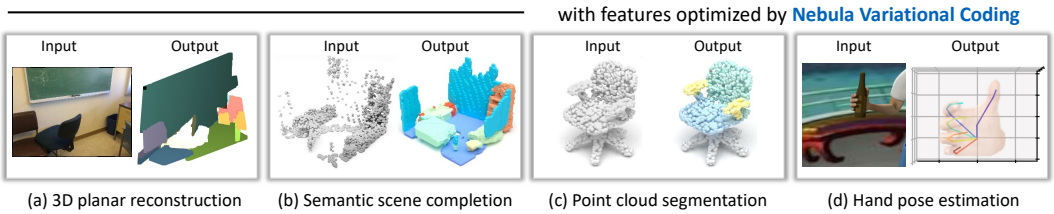


Figure 5.7 The proposed nebula variational coder [3] (NVC) on different architectures that are designed to solve problems in different tasks.

Further exploiting the latent clusters with the help of learnable cluster centers introduced in Section 5.3.1, we propose a new set of parameters called *nebula anchors* to parameterize the cluster centers. Each vector in the set of the nebula anchors has the same dimension as the latent features. In terms of architectural design, our proposed methods can be generalized to different tasks; we propose a variational inference model with latent space optimized by anchors, which can be applied to different architectures and used in a variety of supervised and unsupervised training including classification, regression, and recurrent prediction.

Such a variational inference model has its advantage over traditional variational models such as VAE. The latent space of VAE often includes unused regions accountable for generating samples that do not belong to the actual distribution. Specifically for VAEs, the Gaussian assumption with the fixed expected density does not always match the encoder's likelihood with the decoder's true posterior, leading, e.g. , to blurry reconstructed contours in perceptual VAE. The formed clusters in our proposed variational coder adapt to the generated semantics of the training data. We demonstrate experimentally that it can be used within different architectures designed to solve different problems.

During training, to match features to anchors, the latent variables are labeled with the ID of the nearest nebula anchor so that the variables with the same label are expected to stay close to the associated anchor. At the same time, the anchor itself moves towards regions with a higher density of variables and the same label. Regarding the optimization procedure of our proposed nebula anchors, we propose the Nebula loss, inspired by Newton's law of universal gravitation, where the force is proportional to the two masses and inversely proportional to their squared distance. Since each anchor is the cluster center of the latent features, the mass value is related to the distance of the features to the anchor and the number of features in the anchor. By design, our nebula anchors can be applied to the latent space of the VAEs and other deep networks such as Convolutional Neural Networks (CNN), Recurrent Neural Networks, and adversarial generative models. We have tested the proposed nebula anchor for optimizing many different types of tasks, including NLP and 2D and 3D computer vision, with noticeable improvements.

Contributions

Yida Wang implemented and proposed the prototype by using additional trainable parameters in latent space to form clusters in a variational inference framework, which eventually forms the proposed *nebula anchors*. He conducted the experiments to validate the proposed method on different tasks.

David Tan and Yida Wang further related the initial idea concerning gravity law so that the amount of samples assigned for each cluster center is conditioned for optimizing the centers.

Federico Tombari clarified that there are corner cases about generalizing such proposed *nebula anchors* to other tasks. He contributed to publication-related expenses on behalf of Google Research Zurich.

Nassir Navab financed on behalf of the leader of TUM CAMP together with Federico.

Self-supervised Latent Space Optimization with Nebula Variational Coding

Yida Wang[1], David Joseph Tan[2], Nassir Navab[1], Federico Tombari[1,2]

[1] Technical University of Munich
[2] Google

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.
Published version: https://doi.org/10.1109/TPAMI.2022.3160539

Self-supervised Latent Space Optimization with Nebula Variational Coding

Yida Wang, David Joseph Tan, Nassir Navab, and Federico Tombari

Abstract—Deep learning approaches process data in a layer-by-layer way with intermediate (or latent) features. We aim at designing a general solution to optimize the latent manifolds to improve the performance on classification, segmentation, completion and/or reconstruction through probabilistic models. This paper proposes a variational inference model which leads to a clustered embedding. We introduce additional variables in the latent space, called *nebula anchors*, that guide the latent variables to form clusters during training. To prevent the anchors from clustering among themselves, we employ the variational constraint that enforces the latent features within an anchor to form a Gaussian distribution, resulting in a generative model we refer as Nebula Variational Coding (NVC). Since each latent feature can be labeled with the closest anchor, we also propose to apply metric learning in a self-supervised way to make the separation between clusters more explicit. As a consequence, the latent variables of our variational coder form clusters which adapt to the generated semantic of the training data, e.g. the categorical labels of each sample. We demonstrate experimentally that it can be used within different architectures designed to solve different problems including text sequence, images, 3D point clouds and volumetric data, validating the advantage of our proposed method.

Index Terms—nebula anchor, variational inference, self-supervised learning, metric learning

1 Introduction

In machine learning, we aim at learning relationships of different kinds between the input and the output signals. For instance, solutions in language translation [1], [2] and 3D understanding [3], [4], [5] aim at completely transforming the input data to a different form as the probabilities in classification and segmentation. They rely on compressing the input to its latent representations while identifying its discriminative information. On the other hand, solutions in image processing [6], [7], [8], [9] and 3D understanding [8], [10], [11], [12], [13], [14] aim at preserving the information from the input data as part of the output, formulating additional skip connection [8], [9], [10], [12] and fusion [11] between them.

Although there are traditional methods that can reduce the dimensionality of the input in an unsupervised way, such as principal component analysis (PCA) [15] and independent component analysis (ICA) [16], as well as in a supervised way, e.g. linear discriminant analysis (LDA) [17], they can hardly be applied when modeling complex data.

Moving towards deep learning, such compression can also be modeled with an encoder-decoder architecture. Auto-encoders (AE) [18], [19] are among the simplest, yet effective, architectures able to extract the latent features in an unsupervised fashion. For these reasons, they are commonly employed in a variety of tasks such as image denoising [20], [21] and retrieval [22]. Moreover, more complicated tasks such as pose estimation [23], [24], [25] or 3D completion [5], [10] utilize a more generalized encoder-decoder architecture.

Looking at the bigger picture, there are three ways to improve the architecture, namely, encoder design [26], decoder design [27] and latent feature optimization [26]. In this work, we focus on proposing a novel latent feature optimization approach.

One of the popular approaches to optimize the latent feature is through the variational inference [28], [29]. It matches the likelihood of the encoder with the true posterior of the decoder by setting a distribution constraint on the latent features such as Gaussian or Bernoulli. Since they impose a distribution in the latent space, it becomes feasible to generate reasonable output, with a trained decoder, directly from random sampling of the latent feature, transforming the deep architecture to a generative model such as variational auto-encoder (VAE) [28], [29].

However, there are two main problems in assuming a simple distribution like the Gaussian in VAEs. First, the latent space often includes unused regions which are accountable for generating samples that do not belong to the real distribution. Specifically for VAEs, the Gaussian assumption with the fixed expected density does not always match the likelihood of the encoder with the true posterior of the decoder, leading, e.g., to blurry reconstructed contours in perceptual VAE [30]. Such problem is mostly influenced by the decoder. With respect to the encoder, another problem is the uncertainty described in [31], where the latent variables are not expected to be continuous in some situations.

To solve these issues, conditional information [29] and additional parameters [5], [10] out of the encoder-decoder inference model are used to improve their results. Nevertheless, these solutions carry their own disadvantages. In the former, the conditional information is still required during inference; while, in the latter, the additional architecture makes training less efficient.

We propose a new variational inference model which can be applied to different architectures, and used in a variety of supervised and unsupervised training including classification, regression and recurrent prediction. Our approach relies on the idea of having additional latent variables used during training, called *nebula anchors*, that are explicitly embedded in training the model so to help

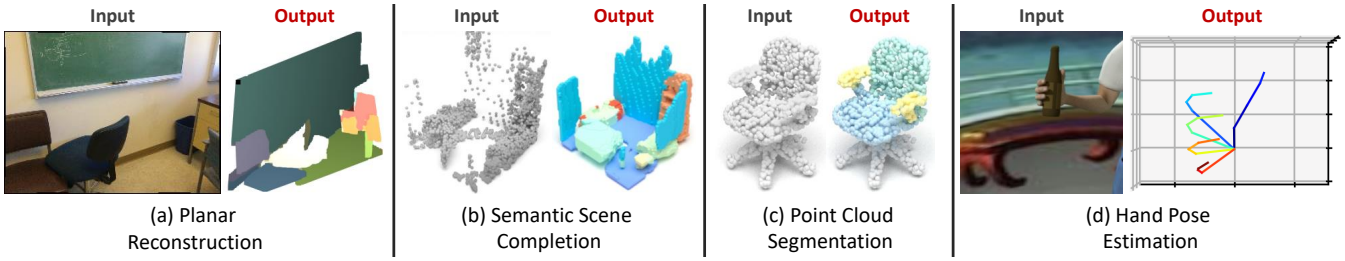


Fig. 1: Example applications of the proposed nebula variational coder (NVC) on different architectures that are designed to solve problems in (a) planar reconstruction, (b) semantic scene completion, (c) point cloud segmentation and (d) hand pose estimation. Note that (a) and (b) take real images captured from real scenes as input while (c) and (d) take synthetic images rendered from realistic scenes.

the formation of hyper-clusters in the latent space. One of its main advantages is the ability to exploit the semantic meaning such as the categorical information in the latent space without any human supervision. As an example on the MNIST dataset [32], we illustrate that our generative model automatically forms 10 clusters, each representing a different digit (see Fig. 4 (b)).

The limitation of our anchor-based solution is the need to pre-define one key hyper-parameter which is the number of anchors. To ease this, we also introduce a self-supervised metric learning derived from the Siamese [33] and Triplet [34], [35], [36] losses, tailored for the nebula anchors. We apply the metric learning efficiently using the labels produced by the nebula anchors based on the nearest neighbour classifier to separate the clusters. Notably, unlike other generative models such as CVAE [29] that use labels as conditional variables to improve inference, our method exploits the additional information only during training, which implies that no additional information other than the input samples is required during inference.

By design, our nebula anchors can be applied on the latent space of not just the VAEs, but also other deep networks such as Convolutional Neural Networks (CNN) [37], [38], Recurrent Neural Networks (RNNs) [39], [40] as well as adversarial generative models such as GANs [5], [41], [42]. Motivated by the range of applicability, we evaluate the proposed approach on various datasets that include image reconstruction on language translation on WMT16 [43], image reconstruction on MNIST [32], 3D volumetric completion on objects from ShapeNet [10], 3D point cloud segmentation on PointNet [44], 3D hand pose estimation on HOP [25] and Stereo [45], 3D planar reconstruction on NYUv2 [46] and ScanNet [47], and 3D semantic completion on ScanNet [47]. Some of these results are shown in Fig. 1. Strikingly, even if our experiments evaluate on varying datasets with distinct objectives and dimensionalities such as sentences, images and 3D point clouds, we demonstrate that our model is beneficial in the encoder-decoder architectures.

2 Related Works

An encoder-decoder architecture is designed to extract the latent features from the input which are then mapped to the output. Auto-encoders (AEs) [48], [49] are among the simplest encoder-decoder architectures. Their features are used to solve a large variety of problems in image compression [48], video compression [50], anomaly detection [51], saliency detection [49] and 3D segmentation [27]. By making the parametric model deeper such as stacked convolutional AE [52], wider such as BAE-Net [27] or nested such as AE^2-Nets [53], they improve ability of AEs to fit the training data.

While AEs are trained in an unsupervised way, there are also architectures that use supervised information to help train the latent features. Those works show obvious advantages against traditional approaches [54]. Tackling a more complex problem like reconstructing 3D structures from a single 2D image, Deep Supervision [55] utilizes the additional skeleton labels to optimize the output of the hidden layers without using the variational methods to estimate the occluded area in the 2D image. Recently, there are more works in self-supervised learning for image classification [56], point cloud retrieval [57] and 3D Axon Segmentation [58].

Ranging from a simple auto-encoder to a more complicated encoder-decoder architecture, the proposed nebula anchors can be integrated in all of them. Particularly, this work proposes a general optimization approach to improve the latent space in an encoder-decoder architecture which can be applied to different problems. It is noteworthy to mention that our experiments in Sec. 4 evaluates on language translation, image reconstruction, and 3D reconstruction and pose estimation, where we demonstrate the superiority of incorporating the nebula anchors in the existing architectures.

The related work on the latent space optimization can be categorize into four main fields, namely, variational embedding, clustering, metric learning and adversarial learning. The following sections discuss them in more detail. Compared to these fields, we propose a variational approach that forms clusters in the latent space while incorporating metric learning as an optional optimization approach to further improve the performance.

2.1 Variational embedding

Given an encoder-decoder architecture, the variational inference use the encoder to approximate the posterior of the latent features that are conditioned on the expected network output. This is governed by a simple distribution of the data in latent space. These methods are derived from VAE, where [28] proposes an assumption that the latent feature of VAE behaves like PCA components. This, in effect, makes the convergence in training more efficient.

To solve the problem caused by the simplicity of the latent space of VAE, categorical labels are integrated in conditional VAE (CVAE), which is used in generation [29] and prediction [31]. With a better performance than the standard VAE, they prove that such information holds the potential to improve the generative models.

Using variational inference in clustering, GMVAE [59] and its graph embedding version [60] demonstrate that multi-modal Gaussian can reveal categorical information better than VAE. Another notable method is from VQ-VAE [61] which aims at solving the "posterior collapse" by discretizing the trained features into a table. Moreover, InfoVAE [62] improves the evidence lower bound (ELBO) objective since they observed that the ELBO favors in optimizing the distribution over the inference.

Going beyond the assumption of simple distributions in the latent space, Auxiliary Deep Generative Models [42] adds auxiliary latent variables. But the disadvantage of such work is that the auxiliary latent variables are necessary not just during training but also at inference time.

Apart from dealing with the visual data, VAE can also be used for natural language processing (NLP) such as machine translation. VAE-LSTM [63] composes both the encoder and the decoder with LSTM operations while VAE-CNN [64] uses LSTM in the encoder and the dilated CNN to form the decoder. In these methods, to solve the latent variable collapse problem [63], [65] of VAEs, HR-VAE [65] imposes regularization for all the hidden states of the LSTM encoder.

2.2 Clustering the latent features

The main technique in the proposed method is the formation of clusters in the latent space through our nebula anchors. However, there are many related works focusing on building clusters in the latent space.

As a simple deep clustering approach, DEC [66] is among the first few works that clusters the latent space in AEs. Aiming at including the discrete class labels into the latent space, K-means [67] is another method to cluster the feature [67], [68]. For instance, DCN [69] uses the K-means clusters to learn a compact feature within an AE architecture.

A notable work is from Gumbel-softmax [70] where they train discrete latent variables by utilizing additional labels in a way similar to our work. But, in contrast, [70] interpolates between the discrete and continuous distributions while our loss function directly operate on the distance metric between features and clusters. Moreover, PrototypicalNet [71] builds an embedding table for the latent features which are then used to represent a feature by applying the softmax activations over distances of the feature to each vector in the table. Moreover, the hierarchical clustering such as HG [72] can exploit hierarchically organized auxiliary labels to learn the clusters in the embedding space; while, IMSAT [73] and IIC [74] learn the latent clusters by maximizing the sample-wise categorical information.

In some cases, more than one labels are given for a single sample. For instance, the object in an image is labeled with the illumination and its styles. Targeted on this situation, the latent space are then disentangled so that the latent clusters reveal specific attributes such as in CDD [75] and Y-AE [26]. Among all the disentanglement approaches, two [26], [76], [77] or three [75] attributes are commonly investigated.

2.3 Metric learning

There have been several methods that apply metric learning in the latent space optimization [35], [78], [79]. Assuming a supervised learning, these methods optimizes the distance among the samples so that they reflect their ground truth semantic similarity. They formulate the pairwise distance metrics, which include: the triplet loss and its derivatives [35], [80], [81], [82], the contrastive loss and its derivatives [78], [83], and the Neighborhood Component Analysis and its derivatives [36], [79], [84]. Among all those losses, Triplet learning [34], [35], [80], [85], [86] is one of the typical learning strategy where the pairwise distances are further labeled as positive or negative based on the pair-wise relationships, resulting in clusters in the latent space.

In this method, we also employ the triplet loss. But, differently from them, our method can exploit the auxiliary labels even if they do not have explicit labels, i.e. unsupervised learning.

2.4 Adversarial feature learning

In some tasks, the training data is different from test data which causes a data migration problem. This implies that the feature domains extracted by the same encoder are different. The objective then is to make both domains similar to each others so that the network trained from one type of data could be applied to the test data. Such tasks could be referred as domain adaptation [87], [88], [89] or domain generalization [90], and could be solved by discriminative training [91], [92].

For instance, the domain discriminator in DANN [93] is used to distinguish the source domain from the target domain using a binary code, while ADDA [94] uses two different discriminators for the source feature and the target features. If additional category labels are available, SymNets [95] can further improve the domain adaptation by using the two-level domain confusion losses from the domain level to category level. By making the latent feature extracted by the encoder in the same dimension as the input, GVB [96] uses discriminative training to make the latent features from the two different domains to be in the same sub-space.

3 Methodology

Given the task of estimating the expected output Y from the input X, the architecture of generative models such as VAE [97] and CVAE [29] constitute two parts – the encoder $\mathcal{E}(X)$ that compresses the input X into the latent variable z, and the generator or decoder $\mathcal{G}(z)$, which maps z into the output Y.

During training, the parameters in the architecture are optimized through the loss function where its definition depends on the problem at hand. For instance, regression tasks such as image reconstruction [32] and language translation [98] commonly define the loss function by means of the Euclidean distance

$$\mathcal{L}_{\text{Euclidean}} = \|\tilde{Y} - \mathcal{G}(\mathcal{E}(X))\|^2 \tag{1}$$

where we differentiate the predicted output (Y) from the ground truth ($\tilde{Y}$). The solutions for the completion of partially scanned 3D volumetric objects [10] or point cloud segmentation [44] predict a one-hot encoded vector that allows training by means of a binary cross entropy loss function

$$\mathcal{L}_{\text{Entropy}} = -\tilde{Y}\log(\mathcal{G}(\mathcal{E}(X))) - (1-\tilde{Y})\log(1-\mathcal{G}(\mathcal{E}(X)))\ . \tag{2}$$

Furthermore, the point cloud reconstruction of the objects [4] evaluates whether the reconstructed point cloud $\mathcal{G}(\mathcal{E}(X))$ matches the given ground truth $\tilde{Y}$ through the Chamfer distance

$$\mathcal{L}_{\text{Chamfer}} = \text{Chamfer}((\mathcal{G}(\mathcal{E}(X)), \tilde{Y})\ . \tag{3}$$

Later in Sec. 4, we use in our experiments the same loss functions as in (1), (2) and (3) for the respective problems.

In most works [3], [44], the role of the latent feature is simplified to be the output of the encoder and the input of the generator. However, since the latent features extract the most useful information from X, we aim at capturing the contextual information from the latent feature (and indirectly from X) by formulating clusters in the latent space in order to easily build the relation between the latent feature and the output.

3.1 Learning with Nebula Anchors

Inspired by K-means [67] where the sampled features form clusters by iteratively updating their centers, we introduce the concept of *nebula anchors* to parameterize the cluster centers.

We define the set of the m nebula anchors as $\mathcal{A} = \{a_1, a_2, \ldots, a_m\}$, each represented by a vector with the same dimension m as that of the latent variable. In order to match them, the latent variables are labeled with the ID of the nearest nebula anchor so that the variable with the same label are expected to stay close to the associated anchor, while the anchor itself moves towards regions with higher density of variables and the same label. We can then define $\mathcal{Z}_{a_i}$ as the set of latent features that are close to a_i. It follows that a_i is the cluster center of the latent features in $\mathcal{Z}_{a_i}$.

3.1.1 Nebula loss

Our Nebula loss is inspired by the concept of Newton's law of universal gravitation, where the force is proportional to the two masses and inversely proportional to their squared distance. In this context, we interpret the mass

$$M(a_i) = 1 + \sum_{\mathcal{E}(X)\in z_{a_i}} \|\mathcal{E}(X) - a_i\|^2 \tag{4}$$

as the sum of distances from each feature attracted to its anchor. Since the anchor a_i is the cluster center of the latent features in $\mathcal{Z}_{a_i}$, the value of the mass is related to the distance of the features to the anchor and the number of features in the anchor.

Instead of dividing by the squared distance, we use the negative logarithm of the squared distance

$$D^{-2}(a_i, a_j) = -\log\|a_j - a_i\|^2 \ \propto\ \frac{1}{\|a_j - a_i\|^2}\ . \tag{5}$$

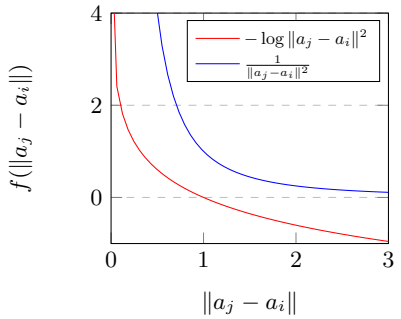


Fig. 2: Comparison of $-\log\|a_j - a_i\|^2$ and $\frac{1}{\|a_j - a_i\|^2}$.

Notably, the negative logarithm function is proportional to the inverse of the distance as shown in Fig. 2. Considering that we have the assumption of a variational latent space (see Sec. 3.1.3), the distance between a pair of anchors $\|a_j - a_i\|^2$ has a stable range roughly between 0 to 1. The main difference between the two is the capacity of the logarithmic function to enforce the optimized distance between the two anchors to be 1 instead of infinity. This, in effect, makes the optimization more stable.

Now that we have defined the mass of an anchor and inverse distance between two anchors, we build the formula for the gravitational force as

$$F(a_i, a_j) = M(a_i)\cdot M(a_j)\cdot D^{-2}(a_i, a_j)\ . \tag{6}$$

By minimizing the force, we also minimize the distance from the feature to the anchors which makes them more compact and, at the same time, maximize the distance between the anchors. Therefore, to build the final loss, we sum up the gravitational forces from all pairs of anchors. The nebula loss then is

$$\begin{aligned}\mathcal{L}_{\text{nebula}} &= \sum_{i=1}^{m-1}\sum_{j=1+1}^{m} F(a_i, a_j) \\ &= \sum_{i=1}^{m-1}\sum_{j=1+1}^{m} M(a_i)\cdot M(a_j)\cdot D^{-2}(a_i, a_j) \\ &= \sum_{i=1}^{m-1} M(a_i)\cdot \sum_{j=1+1}^{m} M(a_j)\cdot D^{-2}(a_i, a_j)\end{aligned} \tag{7}$$

which ensures two criteria: (1) the latent feature belongs to the closest anchor; and, (2) the anchors are separated from each other.

3.1.2 Distinction from K-means

As K-means [67] has been widely used for feature clustering [67], [68], we want to point out the differences from the proposed nebula loss. If the nebula anchors are optimized by K-means [67], the cluster centers are then updated directly with

$$a_i = \frac{1}{|\mathcal{Z}_{a_i}|}\sum_{z_j\in\mathcal{Z}_{a_i}} z_j \tag{8}$$

where $|\mathcal{Z}_{a_i}|$ is the number of elements in $\mathcal{Z}_{a_i}$. The loss function is therefore written as

$$\mathcal{L}_{K\text{-means}} = \sum_{i=1}^{m} \sum_{z_j \in \mathcal{Z}_{a_i}} \|z_j - a_i\|^2 = \sum_{i=1}^{m} |\mathcal{Z}_{a_i}| \cdot \mathrm{Var}(\mathcal{Z}_{a_i}) \tag{9}$$

where $\mathrm{Var}(\cdot)$ is the variance of the set. However, since K-means is evaluated on all the samples to update the centers of every cluster, it could not be applied in the latent features when the network is being optimized by the Stochastic Gradient Descent (SGD) [99]. One problem is that $\mathcal{L}_{K\text{-means}}$ depends on a large amount of samples, which is not accessible when we optimize the parametric model with samples in the mini-batches.

Although the Robbins-Monro stochastic approximation [100] can perform batch-wise optimization similar to SGD using

$$a_i^{\text{new}} = a_i^{\text{old}} + l_r \sum_{\mathcal{E}(X) \in z_{a_i}} (\mathcal{E}(X) - a_i^{\text{old}}) \tag{10}$$

with the learning rate l_r, this equation requires each subset $\mathcal{Z}_{a_i}$ to come from a fixed set of latent features with a constant total variance. Prior to convergence, the parameters in the encoder constantly change which, in effect, changes the values of the latent features across all the mini-batches during training.

Differently from the cluster centers in K-means, our nebula anchors could be optimized via SGD. To train our loss batch-wise, we make the variational assumption on the distribution of latent features and initialize the nebula anchors in a Gaussian distribution accordingly. Hence, this function is evaluated under unsupervised training and forces the network to create nebula anchor-driven clusters in the latent space.

3.1.3 *Variational constraint*

The variational inference model imposes a pre-defined distribution on the latent feature to optimize the encoder-decoder architecture. Contrary to other methods, we implement the variation constraint through the nebula anchors.

We first adopt the variation translation model from GNMT [98] where the expected Y is different from input X. This implies that we can denote a given problem through the probabilistic model $P(Y|X)$.

Given the encoder $\mathcal{E}(X)$ which produces the latent feature z and the generator $\mathcal{G}(z)$, the objective is to make the expectation $\mathbb{E}_{z\sim Q}P(Y|z)$ of the likelihood $P(Y|z)$ to be close to the true probability $P(Y)$, where the probability $Q(z)$ is determined by $\mathcal{E}(X)$ while $P(Y|z)$ is determined by $\mathcal{G}(z)$. We then use the Kullback-Leibler (KL) divergence $\mathbb{D}$ from posterior $P(z|Y)$ to $Q(z|X,\mathcal{A})$, written as

$$\mathbb{D}_{\text{KL}}[Q(z|X,\mathcal{A})||P(z|Y)] = \mathbb{E}_{z\sim Q}\left[\log\left(\frac{Q(z|X,\mathcal{A})}{P(z|Y)}\right)\right] \tag{11}$$

to measure the difference between those two distributions. Thus, by minimizing the KL divergence, we evaluate the capacity of the encoder to generate latent variables that are likely to produce the expected target.

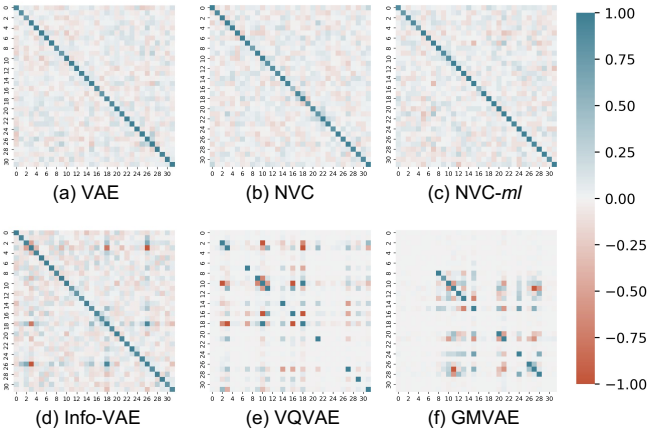


Fig. 3: Comparison of covariance matrices computed from the latent features that relies on variational constraint.

Since $P(z|Y)$ is intractable, similar to VAE [97], [101], we rewrite the KL-divergence from (11) as

$$\begin{aligned}
&\mathbb{D}_{\text{KL}}[Q(z|X,\mathcal{A})||P(z|Y)] \\
&= \mathbb{E}_{z\sim Q}\left[\log\left(\frac{Q(z|X,\mathcal{A})}{P(Y|z)\cdot P(z)}\right)\right] + \log P(Y) \\
&= \log P(Y) - \left(-\mathbb{E}_{z\sim Q}\left[\log\left(\frac{Q(z|X,\mathcal{A})}{P(Y|z)\cdot P(z)}\right)\right]\right)
\end{aligned} \tag{12}$$

where the second term is called evidence lower bound (ELBO) of $\log P(Y)$. Considering that the first term is independent of $Q(z|X,\mathcal{A})$, the optimization then focuses on

$$\begin{aligned}
\text{ELBO} &= -\mathbb{E}_{z\sim Q}\left[\log\left(\frac{Q(z|X,\mathcal{A})}{P(Y|z)\cdot P(z)}\right)\right] \\
&= \mathbb{E}_{z\sim Q}\left[\log P(Y|z)\right] - \mathbb{E}_{z\sim Q}\left[\log\left(\frac{Q(z|X,\mathcal{A})}{P(z)}\right)\right] \\
&= \mathbb{E}_{z\sim Q}\left[\log P(Y|z)\right] - \mathbb{D}_{\text{KL}}[Q(z|X,\mathcal{A})||P(z)]\,.
\end{aligned} \tag{13}$$

Finally, we define the loss function of the generative model as $\mathcal{L}_{\text{enc-gen}} = -\text{ELBO}$. The final form of the loss function is written as

$$\mathcal{L}_{\text{enc-gen}} = \underbrace{\mathbb{D}[Q(z|X,\mathcal{A}))||P(z)]}_{\mathcal{L}_{\text{enc}}} \underbrace{-\mathbb{E}_{z\sim Q}[\log P(Y|z)]}_{\mathcal{L}_{\text{gen}}} \tag{14}$$

where the first term ($\mathcal{L}_{\text{enc}}$) enforces the encoder to produce latent features which satisfy a Gaussian distribution while the second term ($\mathcal{L}_{\text{gen}}$) enforces the predicted output from the latent feature fits the expected ground truth.

Although the encoder is optimized by $\mathcal{L}_{\text{nebula}}$ which enforces clusters and $\mathcal{L}_{\text{enc}}$ which enforces a Gaussian distribution, it is noteworthy to mention that there is no conflict in optimizing both losses. In training, our latent feature converges to a single Gaussian distribution while the Nebula loss focuses on forming clusters within this distribution. We illustrate this occurrence in Fig. 3 by plotting the covariance matrix of the latent features. This figure show that our convariance matrix is almost identity which behaves similar to VAE [97]. Conversely, other variational clustering approaches like Info-VAE [62], VQ-VAE [61] and GMVAE [59], which rely on additional loss functions, discretization or multiple Gaussian distributions, deviate from an identity matrix.

3.2 Self-supervised metric learning from the anchors

One interesting characteristic of nebula anchors is the capacity to form clusters even under unsupervised or self-supervised settings. To separate the clusters in the latent space further, we employ the losses involving metric learning.

Based on these clusters, we label all samples based on the closest nebula anchor. The labels allow us to apply the self-supervised metric learning on the samples with the Siamese term [33]

$$\mathcal{L}_{\text{pair}}(X_i, X_p) = |\mathcal{E}(X_i) - \mathcal{E}(X_p)|^2 \tag{15}$$

so that distances between the samples from the same category become smaller while the distances between the samples from different categories become larger; and, the loss function for triplet learning [34]

$$\mathcal{L}_{\text{triplet}}(X_i, X_p, X_n) = \ln\left(\max\left(1, 2 - \frac{|\mathcal{E}(X_i) - \mathcal{E}(X_n)|^2}{|\mathcal{E}(X_i) - \mathcal{E}(X_p)|^2 + 0.01}\right)\right) \tag{16}$$

so that samples from the same cluster, i.e. the positive pairs X_i and X_p, are closer than samples from distinct clusters, i.e. negative pairs X_i and X_n. We apply these loss functions on all the possible permutations in the batch.

Consequently, although summing the two losses to $\mathcal{L}_{\text{metric}}$ does not significantly improve our results, the improvements are consistent throughout all our experiments in the evaluation section. This makes us conclude that they are optional but, at the same time, also valuable to the overall performance.

3.3 Model optimization

Based on the previous sections, the final loss function is thus defined as

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{enc-gen}} + \mathcal{L}_{\text{nebula}} + \mathcal{L}_{\text{metric}} \tag{17}$$

where the self-supervised metric learning loss term ($\mathcal{L}_{\text{metric}}$) is optional. Similar to the optimization in VAE [97], the probability density function of the encoder $Q(z|X, a)$ is defined as the Gaussian distribution $N(z|\mu,\Sigma)$, where $\mu(X;\theta)$ and $\Sigma(X;\theta)$ are arbitrary deterministic functions. In addition, we use the re-parameterization approach from VAE, explained in [97], to optimize the basic generative loss $\mathcal{L}_{\text{enc-gen}}$.

4 Experiments

To assess the proposed NVC model, we conduct an extensive evaluation of the proposed method on various applications. From 1D to 3D information, we dive into the following datasets:

1) **WMT16** [43] for 1D language translation of the text sequences;
2) **MNIST** [32] for 2D image reconstruction;
3) **ShapeNet** [102] for 3D semantic completion;
4) **PointNet** [44] for 3D point cloud segmentation;
5) **Stereo** [45] and **HOP** [25] for 3D hand pose estimation;
6) **NYUv2** [103] and **ScanNet** [47] for 3D planar reconstruction; and,
7) **ScanNet** [47] for 3D semantic scene completion.

Method	WMT13	WMT14	WMT15	Variance
NMT [98] (greedy)	27.1	-	27.6	**0.19**
NMT [98] (beam=10)	28.0	-	28.9	0.23
NMT-GNMT [98]	29.0	-	29.9	0.30
– *with* GMVAE [104]	29.8	-	30.4	-
– *with* **NVC**	31.2	-	32.4	0.25
– *with* **NVC-ML**	**33.4**	-	**34.9**	0.31
FusedBert [105]	-	30.8	-	-
– *with* GMVAE [104]	-	31.2	-	-
– *with* **NVC**	-	31.6	-	0.19
– *with* **NVC-ML**	-	32.1	-	0.26
Transformer-Rep [106]	-	33.9	-	-
– *with* GMVAE [104]	-	33.9	-	-
– *with* **NVC**	-	34.1	-	0.31
– *with* **NVC-ML**	-	**34.2**	-	0.33

TABLE 1: Evaluation on the WMT German-English translation [43], performance is reported in BLEU score [106].

The objective is to evaluate the advantage of imposing the nebula anchors on the latent space. Thus, in the following experiments, we assess the difference of with and without NVC where the latent space and nebula anchors are trained in an unsupervised fashion. Moreover, we distinguish NVC from NVC-ML which includes the optional metric learning from Sec. 3.2.

4.1 1D language translation (WMT16)

Neural machine translation system usually rely on sequence-to-sequence models such as [39], [40], [107]. These models embed 1D input sentences by means of an encoder; then, a recurrent model – typically a Long-Short Term Memory (LSTM) [108] network – operates on the latent space; and finally, a decoder processes the embedded representation to obtain an output translation and to capture the long-range dependencies in the sentences.

We propose an experiment based on the WMT German-English dataset [43]. One of the baseline architectures is a 4-layer Neural Machine Translation (NMT) model [98] with LSTM units. We apply our NVC model on the 1024-dimensional embedding of the last GNMT layer. NVC is applied with and without metric learning, i.e. only with nebula anchors, to train the latent variables in a fully unsupervised way.

Due to the popularity of transformers, we also investigate using FusedBert [105] and TransformerRep [106] as our baseline architectures. For the transformers, we apply NVC on the fused latent feature of the BERT-encoder attention and the self-attention.

Table 1 shows the advantages of our NVC approach, demonstrating its capacity to generalize even when applied on recurrent models like NMT [98], FusedBert [105] and TransformerRep [106]. The results from the other methods are taken from [109].

Utilizing the same baseline architectures, we also compare the our NVC against GMVAE [59] in order to quantify the difference between the two clustering techniques. Table 1 demonstrates that GMVAE [59] marginally improves the machine translation models while the proposed method reveals a clear improvement.

	Method	rel	δ_1	δ_2	δ_3
Unsupervised	Auto-encoder	0.191	78.6%	85.3%	91.9%
	DVAE [21]	0.188	80.2%	86.8%	92.8%
	– *with* **NVC**	0.162	82.8%	89.2%	95.0%
	– *with* **NVC-ML**	0.147	84.5%	92.7%	96.2%
	VAE [97]	0.169	82.4%	91.3%	95.2%
	– *with* **NVC**	0.138	84.3%	92.6%	96.1%
	– *with* **NVC-ML**	**0.133**	**85.3%**	**93.8%**	**97.2%**
Supervised	DVAE *with* **NVC-ML**	0.142	85.3%	92.8%	96.9%
	VAE *with* **NVC-ML**	0.141	85.1%	92.5%	96.3%
	CVAE [29]	0.165	82.9%	91.7%	95.4%
	– *with* **NVC-ML**	**0.116**	**87.6%**	**94.3%**	**98.6%**

TABLE 2: Evaluation of the reconstruction on MNIST [32] for different variational models. We use 10 anchors in the latent space. The term rel is the absolute relative error while the threshold accuracy δ_i is described in Adabins [111].

4.2 2D image reconstruction (MNIST)

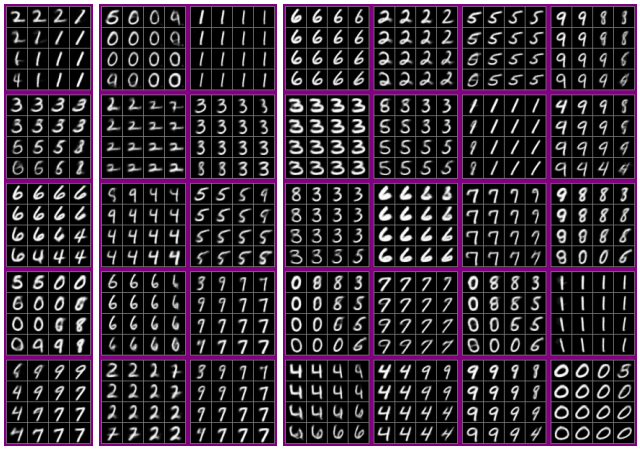
We also evaluate the MNIST [32] dataset to demonstrate that training with NVC performs as theoretically expected, especially in terms of the learned digit-aware sub-manifolds in the latent space centered on the anchors. The MNIST dataset includes hand written digits characters from 10 categories which are evaluated based on the reconstruction precision of the auto-encoder.

We adopt an auto-encoder as the basic architecture with three fully-connected layers for each of the encoder and the decoder. Notably, we train the NVC with self-supervised metric learning without the categorical label of the samples. To compare against the variational inference models without the latent anchors, one hidden layer is added to produce the latent feature to train a VAE, a DVAE and a CVAE.

In Table 2, we compare a series of variational inference models such as VAE [97], DVAE [21] and CVAE [29] with and without NVC. Using the reconstruction metrics proposed in [110], this table demonstrates the effectiveness of the proposed variational coder in improving the models. We also include the experiments where the anchors are trained by assigning the ground truth categorical labels during the metric learning which is referred as *Supervised* in Table 2. It shows that our proposed self-supervised learning, resulting in having digit-related embeddings, is helpful to improve the baseline methods.

For this experiment, it is noteworthy to mention that the simplicity of the auto-encoder helps us generate insights on the characteristics inferred by the learned nebula anchors. Fig. 4 shows that different numbers of nebula anchors perform well in learning meaningful manifolds according to the hidden information such as the digit labels. When using five nebula anchors, the five manifolds already include all 10 categories ranging from 0 to 9. Increasing to 10 anchors reveal that every anchor is surrounded by each of the 10 digits. If we increase further to 20 anchors, they are not only separate by the digits but also by the font styles. Later in Sec. 5, we numerically evaluate the number of anchors from different datasets.

In addition, we investigate the behavior of the latent space during training. By comparing VAE with and without the proposed methods, i.e. NVC and NVC-M, Fig. 5 (a) plots the entropy of the latent variables at each training



(a) 5 (b) 10 (c) 20

Fig. 4: Visualization of the manifold centered on the nebula anchors learned with different numbers of anchors.

step. Based on this figure, the entropy with NVC is larger

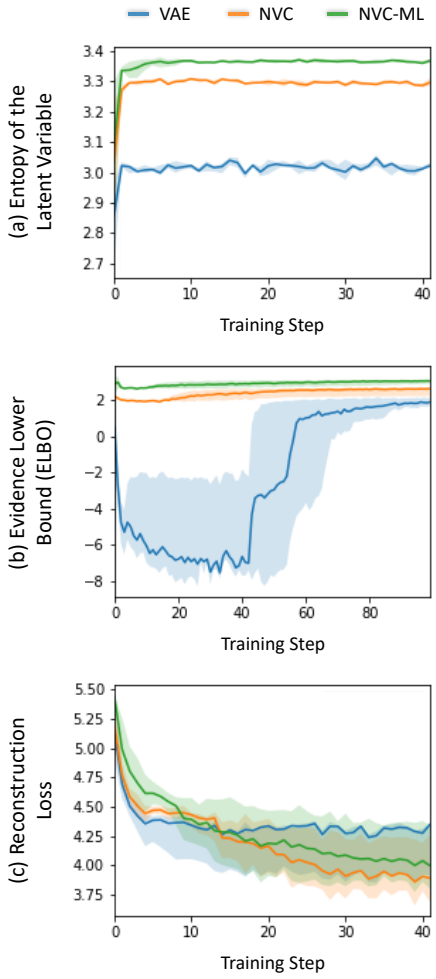


Fig. 5: Influence of the nebula anchors on (a) the entropy of the latent variables, (b) the evidence lower bound (ELBO) and (c) the reconstruction loss during training.

Methods	10	16	20	30	120
GMVAE [59]	88.54	91.26	96.92	93.22	89.70
VQ-VAE [61]	72.32	81.45	82.60	94.22	96.89
NVC	**98.21**	**97.79**	**97.51**	**97.19**	**97.03**

TABLE 3: Compares different number of anchors (or clusters) on the classification accuracy evaluated on MNIST [32].

while adding the self-supervised metric learning improves it further. Note that the higher entropy implies that clusters formed are more separated. Hence, the encoder optimized with the nebula anchors produces a more informative embedding. Although this also implies that the higher entropy deviates from the Gaussian distribution, we visualize in Fig. 3 that the optimized distribution remains similar to Gaussian.

To validate the optimization target $\mathcal{L}_{\text{enc}}$ in (14), we also visualize the evidence lower bound (ELBO) at each training step in Fig. 5 (b). Here, it becomes evident that the ELBO with the proposed method is raised compared to standard VAE so that its gap from the true posterior (which is always larger than ELBO) becomes smaller. Consequently, the reconstruction loss converges to a smaller error with optimized with the proposed Nebula anchors as illustrated in Fig. 5 (c).

Moreover, we also compare different clustering approaches, varying the number of anchors (or clusters). Table 3 illustrates the adaptability of the proposed method to acquire consistent results across different anchor sizes. This is in contrast to the other methods [59], [61] where they peak at only specific number of classes.

4.3 3D volumetric completion (ShapeNet)

Adapting the evaluation strategy from 3D-RecGAN [10], we use ShapeNet [102] to generate the training and test data for 3D object completion, wherein each reconstructed object surface is paired with a corresponding ground truth voxelized shape with a size of $64 \times 64 \times 64$. The dataset comprises of four object classes: *bench*, *chair*, *couch* and *table*. Note that [10] prepared an evaluation for both synthetic and real input data.

4.3.1 Synthetic data

We perform two evaluations in Table 4. The first is a single category test [10] such that each category is trained and tested separately while the second considers the categories in order to label the voxels. We compare our results against [3], [5], [10], [112], [113] by applying the nebula anchors while learning the latent feature. The results are reported in IoU with a 3D resolution of $64 \times 64 \times 64$. By introducing the anchors, the IoU of 3D-AE [10], 3D-RecGAN [10] and ForkNet [5] are improved from 76.3%, 77.5% and 84.1% to 77.5%, 81.6% and 85.1%, respectively. With metric learning, the result obtained from ForkNet [5] achieves the best performance of 86.2% among all the approaches.

4.3.2 Real data

Since both 3D-RecGAN [10] and ForkNet [5] evaluated the real scans [10] captured by Kinect, we also evaluate them with and without the proposed method.

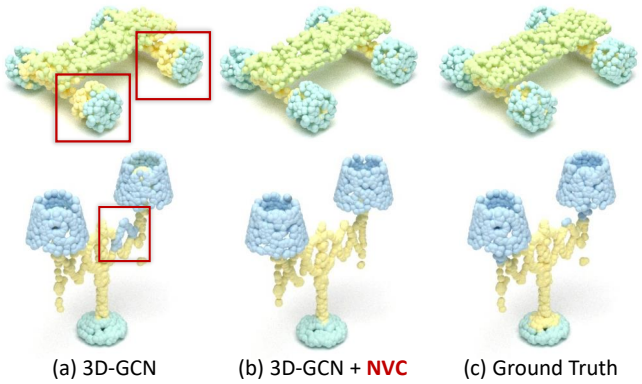


Fig. 6: Comparison of part segmentation from 3D-GCN [120] with and without the proposed NVC optimization.

The single category test in Table 4 suggested by 3D-RecGAN [10] shows that, by simply optimizing the latent features using our approach (i.e. nebula anchor with metric learning), the completion IoU is improved by 4% on average while 2.5% on ForkNet [5]. In addition, the results in IoU also show that, even without the metric learning, the nebula anchors help improve 3D-RecGAN by 3.1% and ForkNet by 1.1%.

4.4 3D point cloud segmentation (PointNet)

We evaluated the proposed method in a 3D semantic segmentation task for point clouds. In particular, we focus on the dataset proposed by PointNet [44], which is a subset of ShapeNet including 16 categories and a total of 16,881 point clouds. In this case, we applied our NVC model to the network architecture proposed in PointNet, and again

	Method	bench	chair	couch	table	*Avg.*
Synthetic [102]	Varley et.al [112]	65.3	61.9	81.8	67.8	69.2
	3D-EPN [3]	75.8	73.9	83.4	77.2	77.6
	Han et.al [113]	54.4	46.9	48.3	56.0	51.4
	3D-AE [10]	73.3	73.6	83.2	75.0	76.3
	– *with* **NVC**	74.9	73.9	84.8	76.4	77.5
	– *with* **NVC-ML**	76.5	74.4	85.6	77.6	78.6
	3D-RecGAN [10]	74.5	74.1	84.4	77.0	77.5
	– *with* **NVC**	76.7	78.4	90.0	81.4	81.6
	– *with* **NVC-ML**	78.2	79.1	92.7	82.8	83.2
	ForkNet [5]	79.1	80.6	92.4	84.0	84.1
	– *with* **NVC**	80.2	82.1	92.9	85.3	85.1
	– *with* **NVC-ML**	**81.9**	**83.5**	**93.4**	**86.0**	**86.2**
Real [10]	Han et.al [113]	18.4	14.8	10.1	12.6	14.0
	3D-AE [10]	23.1	17.8	10.7	14.8	16.6
	– *with* **NVC**	23.6	18.1	10.7	16.1	17.1
	– *with* **NVC-ML**	25.0	19.2	12.5	16.9	18.4
	3D-RecGAN [10]	23.0	17.4	10.9	14.6	16.5
	– *with* **NVC**	27.9	19.1	13.6	17.8	19.6
	– *with* **NVC-ML**	29.2	19.8	14.6	18.4	20.5
	ForkNet [5]	32.7	24.1	15.9	22.5	23.8
	– *with* **NVC**	34.1	25.0	16.4	24.2	24.9
	– *with* **NVC-ML**	**34.9**	**26.8**	**17.8**	**25.5**	**26.3**

TABLE 4: Evaluation of the object completion in terms of IoU (in %) on ShapeNet [102]. The resolution of Varley et.al [112] and 3D-EPN [3] is $32 \times 32 \times 32$ while $64 \times 64 \times 64$ for the others.

Method	aero	bag	cap	car	chair	earpod	guitar	knife	lamp	laptop	motor	mug	pistol	rocket	skateboard	table	*Avg.*
Wu et al. [114]	63.2	-	-	-	73.5	-	-	-	74.4	-	-	-	-	-	-	74.8	-
Yi et al. [115]	81.0	78.4	77.7	75.7	87.6	61.9	**92.0**	85.4	**82.5**	**95.7**	**70.6**	91.9	**85.9**	53.1	69.8	75.3	81.4
3DCNN [116]	75.1	72.8	73.3	70.0	87.2	63.5	88.4	79.6	74.4	93.9	58.7	91.8	76.4	51.2	65.3	77.1	79.4
SGPN [117]	80.4	78.6	78.8	71.5	88.6	78.0	90.9	83.0	78.8	95.8	77.8	93.8	87.4	60.1	92.3	89.4	85.8
SSCNN [118]	81.6	81.7	81.9	75.2	90.2	74.9	93.0	86.1	84.7	95.6	66.7	92.7	81.6	60.6	82.9	82.1	84.7
– *with* **NVC-ML**	77.8	75.6	77.0	72.1	89.5	64.2	91.9	82.3	76.2	94.1	60.5	94.2	78.0	54.4	67.9	80.0	77.2
PointNet++ [119]	82.4	79.0	87.7	77.3	90.8	71.8	91.0	85.9	83.7	95.3	71.6	94.1	81.3	58.7	76.4	82.6	85.1
PointNet [44]	83.4	78.7	82.5	74.9	89.6	**73.0**	91.5	85.9	80.8	95.3	65.2	**93.0**	81.2	57.9	72.8	80.6	83.7
– *with* **NVC-ML**	84.1	81.6	82.7	76.5	89.8	71.6	91.8	86.1	81.7	95.7	66.3	92.2	82.3	61.0	73.2	81.7	84.4
3D-GCN [120]	83.1	**84.0**	86.6	77.5	90.3	**74.1**	90.9	86.4	83.8	95.6	66.8	94.8	81.3	59.6	75.7	82.8	85.1
– *with* **NVC-ML**	**83.5**	83.1	**91.4**	**79.6**	**92.9**	70.3	**97.4**	**87.1**	**87.4**	**97.0**	**78.0**	**97.2**	**84.4**	**61.6**	**82.3**	**87.9**	**86.3**

TABLE 5: Evaluation of the semantic point cloud segmentation on ShapeNet [102]. The results are reported in terms of mIoU as global average as well as for each of the 16 classes of the dataset.

trained the self-supervised metric learning using the class labels of each point cloud.

Table 5 shows the semantic segmentation results of NVC against the state of the art in terms of mean-Intersection-over-Union (mIoU), i.e. the standard metric for this dataset [44]. The table shows that NVC is beneficial to improve the performance of PointNet, and achieve the best overall result and on most individual categories.

In addition, we show some qualitative results in Fig. 6, comparing the segmentation results from 3D-GCN [120] with those reported by our method. From the figure, we can see that NVC yields a more accurate segmentation, allowing to better distinguish the borders between different object parts.

4.5 3D hand pose estimation (Stereo/HOP)

To compare our work with VO-Hand [125] which also uses trainable anchors to define the cluster centers, we tried to apply 5 anchors on the HOP [125] dataset, 7 on RHD [24], and 10 on GANeratedHands [23] and Stereo [45]. In [125], a high number of clusters tends to reduce its advantages while a small number of clusters brings the network back to the standard variational inference model.

Table 6 shows the result of the unsupervised training of the latent variables on those datasets, where the proposed nebula anchors improves the performance of VO-hand [25] on the HOP [25] dataset where the hands are occluded by grasped objects from 0.597 to 0.623 in terms of AUC. In the Stereo [45] dataset where hands are not occluded, we also improve the performance in AUC from 0.984 to 0.986. Note that the pose estimation in Stereo [45] is easier than HOP [25] because the former does not include any hand occlusion; thus, the results are saturated. This explains why the improvement between VO-hand [25] and our work is marginal in Stereo [45].

	Method	AUC	EPE Median	EPE Mean
Stereo [45]	CHPR [121]	0.839	-	-
	GANeratedHands [23]	0.965	-	-
	PosePrior [24]	0.948	9.543	11.064
	VO-Hand [25] (cluster)	0.984	7.606	8.943
	– *with* **NVC**	**0.986**	**7.511**	**8.897**
HOP [25]	PosePrior [24]	0.534	19.728	30.860
	VO-Hand [25] (triplet)	0.597	15.901	27.326
	– *with* **NVC**	**0.623**	**13.935**	**26.405**
	VO-Hand [25] (cluster)	0.583	16.741	28.018
	– *with* **NVC**	0.599	16.063	27.729

TABLE 6: Evaluation on Stereo [45] and HOP [25] with the baseline approach from VO-Hand [25] trained with unsupervised clustering (cluster) [25] or triplet learning (triplet). The endpoint errors (EPE) are in millimeters.

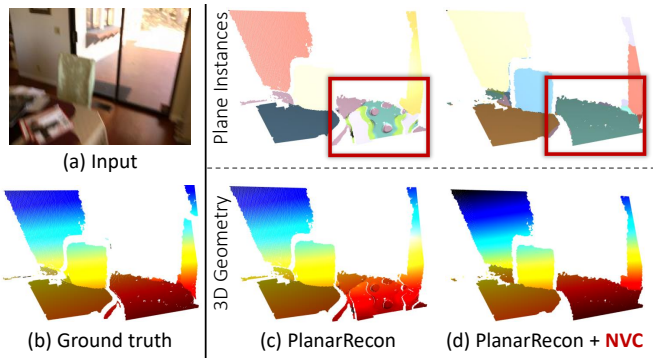


Fig. 7: Comparison of planar reconstruction from PlanarRecon [126] with and without the proposed NVC optimization.

4.6 3D planar reconstruction (ScanNet)

The ScanNet dataset [47] provides real RGB images and their corresponding depth captured by depth cameras. Aiming at simplifying the 3D reconstruction using large planes, PlanarRecon [126] further fits 3D planes to the consolidated 3D point cloud which is back-projected from the depth images using the camera's intrinsic parameters. While constructing the 3D geometries in instance-level planes, PlanarRecon [126] also incorporates the semantic annotations from ScanNet. The resulting dataset contains 50,000 training and 760 testing images with a resolution of 256×192.

We evaluate our approach by optimizing the latent feature of PlanarRecon [46] using nebula anchors on both ScanNet [47] and NYUv2 [103] dataset. Pixel and plane recalls are reported in Table 7, where the introduced nebula anchor improves the performance on all threshold of depth difference. By calculating with pixel-wise absolute difference (rel) in Table 9 on NYUv2 [103] dataset, the performance of PlanarRecon [46] is improved from 0.134 to 0.126 with the help of adding anchors in latent space. This improvement is illustrated in Fig. 7 where the blurry RGB input makes PlanarRecon [46] hard to reconstruct the floor as a single plane, while the added nebula anchor helps to reconstruct the floor as a whole piece. Additionally, the best

	Method	0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.45	0.50	0.55	0.60
Pixel	MWS [122]	2.40	8.02	13.70	18.06	22.42	26.22	28.65	31.13	32.99	35.14	36.82	38.09
	NYU-Toolbox [123]	3.97	11.56	16.66	21.33	24.54	26.82	28.53	29.45	30.36	31.46	31.96	32.34
	PlaneNet [124]	22.79	42.19	52.71	58.92	62.29	64.31	65.20	66.10	66.71	66.96	67.11	67.14
	PlanarRecon [46]	30.59	51.88	62.83	68.54	72.13	74.28	75.38	76.57	77.08	77.35	77.54	77.86
	– *with* **NVC**	**33.45**	**54.91**	**65.74**	**71.05**	**75.43**	**77.61**	**78.10**	**79.91**	**80.12**	**80.67**	**81.66**	**81.92**
Plane	MWS [122]	1.69	5.32	8.84	11.67	14.40	16.97	18.71	20.47	21.68	23.06	24.09	25.13
	NYU-Toolbox [123]	3.14	9.21	13.26	16.93	19.63	21.41	22.69	23.48	24.18	25.04	25.50	25.85
	PlaneNet [124]	15.78	29.15	37.48	42.34	45.09	46.91	47.77	48.54	49.02	49.33	49.53	49.59
	PlanarRecon [46]	22.93	40.17	49.40	54.58	57.75	59.72	60.92	61.84	62.23	62.56	62.76	62.93
	– *with* **NVC**	**26.98**	**44.22**	**53.12**	**58.65**	**62.30**	**64.63**	**65.07**	**66.18**	**67.11**	**67.68**	**67.91**	**68.11**

TABLE 7: Evaluation of the depth estimation on ScanNet [47] incorporating 3D plane estimations for the indoor scenes. The pixel and plane recalls are reported according to threshold of depth difference.

performance is also achieved as 0.125 using our proposed metric learning.

4.7 3D semantic completion (CompleteScanNet)

We also evaluate on the 3D semantic completion from the CompleteScanNet [128] dataset which is built from the ScanNet dataset [47]. Here, we apply the proposed nebula anchor in the latent space of ForkNet [5] where the variational constraint is already used. The results are compared to original ForkNet [5], ScanComplete [127] and SCFusion [128] in terms of IoU on 11 categories.

By incorporating NVC, Table 8 demonstrates a significant improvement on ForkNet increasing the IoU from 9.3% to 14.1%. We illustrate an example of these improvements in Fig. 8 where we can observe that our NVC helped ForkNet [5] classify the table correctly. Furthermore, this consequently reduces its gap between ForkNet [5], and the state-of-the art methods from ScanComplete [127] and SCFusion [128].

5 Understanding the Number of Anchors

This section focuses on the hyperparameter of the proposed method which is the number of anchors. Fig. 9 and Table 9 provide the results with different numbers of nebula anchors. Since the number of anchors is a crucial hyperparameter, we evaluate how the number of anchors influence the performance of inference model on three datasets – image reconstruction (MNIST) [32], 3D planar reconstruction (ScanNet) [47] and 3D object completion (ShapeNet) [102].

When the number of anchors is set to zero, the performance of the inference model is the same as original model. Although training with less than 5 anchors starts to improve the performance of the original architectures, the significant improvements are more evident in the range of 5 to 20 anchors depending on the task.

The goal of this section is to conduct an ablation study to investigate the optimum performance in relation to number of anchors. Ideally, the hyperparameter must have a stable range of optimal values so that we can easily set its value prior to training. Therefore, this study highlight the influence of M_a in Sec. 5.1 as well as the influence of the metric learning in Sec. 5.2 in order to stabilize the optimal range.

5.1 Influence of M_a

From Sec. 3.1, the mass M_a acts as the weight of different clusters centered around the anchors, which depends on how many samples are assign to the anchors. Thus, anchors with a small number of samples back-propagate much smaller gradients. By weighing the loss with the mass, the important clusters are focused during training.

An interesting observation in Fig. 9 is the effects of M_a to the range of the optimal number of anchors. We notice that training without M_a makes the performance of the inference model sensitive to the number of anchors, reaching the best results only at a certain peak. In this case, the mass-based term in $\mathcal{L}_{\text{nebula}}$ is not used but instead we use the Euclidean distance between the latent samples and their closest anchor to optimize the network. After investigating Fig. 9, the plot from ShapeNet demonstrate the worst case scenario where there is only one optimal value for the number of anchors (i.e. 5); otherwise, its performance drops significantly. With the help of M_a, the results becomes more stable such that a range of anchor sizes achieve good results instead of one.

5.2 Influence of the metric learning

As described in Sec. 3.2, we can optionally apply metric learning such as Siamese and triplet training on the clusters. This becomes more evident in Fig. 9 where NVC with metric learning (NVC-ML) further improves the results. Looking more closely at the plots, we notice that the metric learning also stabilizes the range of optimal number of anchors especially for ScanNet [47] in Fig. 9.

6 Ablation Study on the Loss Function

Using the experiments from the object completion in Table 4 and semantic completion in Table 8, we perform an ablation study on the loss introduced in this paper. The first comparison in Table 10 shows the evaluation when simplify $\mathcal{L}_{\text{nebula}}$ to a Euclidean distance (labelled as without M_{a}). This change based on ForkNet [5] produces a noticeable reduction in performance by 2.8% in terms of IoU for object completion and 2.6% for semantic scene completion.

The second comparison shows the advantage of having the components $\mathcal{L}_{\text{pair}}$ and $\mathcal{L}_{\text{triplet}}$ in metric learning. Table 10 validates that without either of the loss, the IoU is reduced by up to 2.4% for object completion with ForkNet [5] and 2.5% with ScanComplete. Note that the results indicate that training without $\mathcal{L}_{\text{triplet}}$ introduce larger performance deduction compared to training without $\mathcal{L}_{\text{pair}}$.

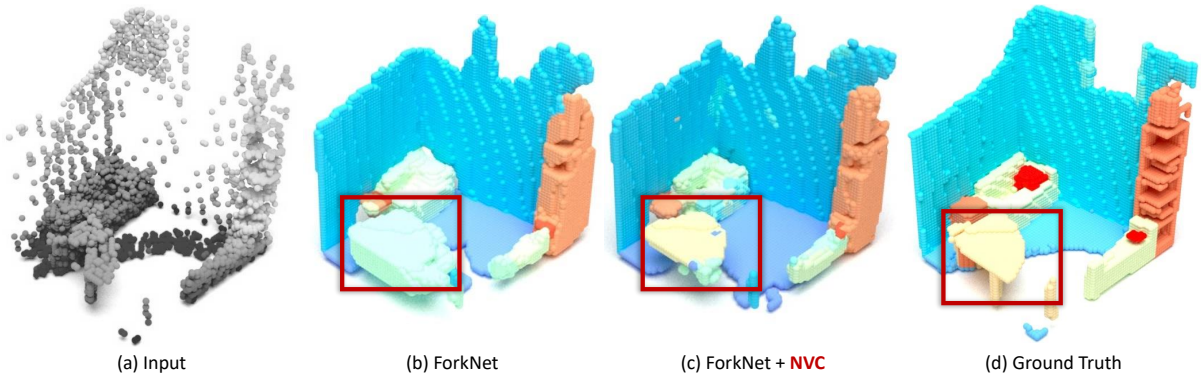


Fig. 8: Comparison of semantic scene completion from ForkNet [5] with and without the proposed NVC optimization.

Method	ceil.	floor	wall	win.	chair	bed	sofa	table	tvs	furn.	objs	*Avg.*
ForkNet [5]	5.2	22.5	12.3	0.0	9.2	5.7	18.2	14.9	0.1	12.6	1.8	9.3
ForkNet [5]+NVC	10.2	29.3	18.2	3.3	14.5	11.9	25.1	17.9	4.7	16.0	3.9	14.1
ScanComplete [127]	16.4	39.3	35.0	1.8	20.4	3.8	11.2	27.7	0.6	13.2	7.8	16.1
SCFusion [128]	12.8	32.9	26.5	9.6	22.5	20.7	26.4	21.0	7.4	19.2	8.6	18.9

TABLE 8: Evaluation of the semantic completion on CompleteScanNet [128]. The results are reported in terms of IoU at a resolution of $64 \times 64 \times 64$.

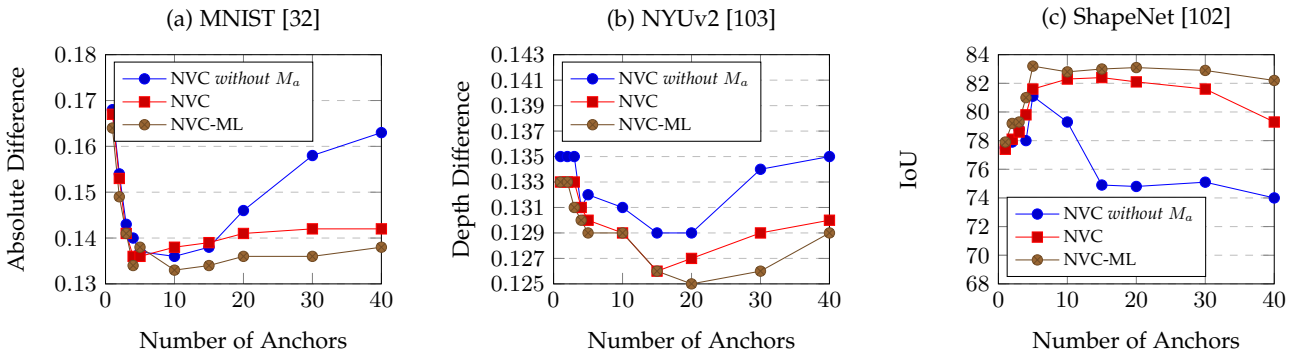


Fig. 9: Plots the performance on the image reconstruction (MNIST) [32], 3D planar reconstruction (NYUv2) [103] and 3D object completion (ShapeNet) [102] with different amount of nebula anchor.

Dataset	Method	$\mathcal{G}(\mathcal{E}())$	1	2	3	4	5	10	15	20	30	40
MNIST [32]: rel with VO-Hand [25]	NVC *without* M_a	0.169	0.168	0.154	0.143	0.140	0.137	**0.136**	0.138	0.146	0.158	0.163
	NVC	0.169	0.167	0.153	0.141	**0.136**	0.136	0.138	0.139	0.141	0.142	0.142
	NVC-ML	0.169	0.164	0.149	0.141	0.134	0.138	**0.133**	0.134	0.136	0.136	0.138
NYUv2 [103]: rel with PlanarRecon [46]	NVC *without* M_a	0.134	0.135	0.135	0.135	0.131	0.132	0.131	**0.129**	0.129	0.134	0.135
	NVC	0.134	0.133	0.133	0.133	0.131	0.130	0.129	**0.126**	0.127	0.129	0.130
	NVC-ML	0.134	0.133	0.133	0.131	0.130	0.129	0.129	0.126	**0.125**	0.126	0.129
ShapeNet [102]: IoU (in %) with 3D-RecGAN [10]	NVC *without* M_a	77.5	77.5	77.9	78.8	78.0	**81.1**	79.3	74.9	74.8	75.1	74.0
	NVC	77.5	77.4	78.1	78.6	79.8	81.6	82.3	**82.4**	82.1	81.6	79.3
	NVC-ML	77.5	77.9	79.2	79.3	81.0	**83.2**	82.8	83.0	83.1	82.9	82.2

TABLE 9: Comparison of different amount of nebula anchors a with and without the self-supervised metric learning, evaluated for the 2D image reconstruction on MNIST [32] and the depth estimation on NYUv2 [103] reported in absolute difference (rel) and 3D object completion on ShapeNet [102] reported in Intersection over Union (IoU).

7 Conclusion

Focused on self-supervised latent space optimization, we present a novel nebula variational coder based on two main contributions: (i) forming clusters in the latent space through the additional variables, called nebula anchors,

Dataset	Method	Baseline	*with* **NVC**	*without* M_a	*with* **ML**	*without* $\mathcal{L}_{pair}$	*without* $\mathcal{L}_{triplet}$
Object Completion on Real [10]	3D-AE [10]	16.6	17.1	16.7	**18.4**	17.9	17.2
	3D-RecGAN [10]	16.5	19.6	17.0	**20.5**	18.1	17.9
	ForkNet [5]	23.8	24.9	22.1	**26.3**	24.5	23.9
Semantic Completion on CompleteScanNet [128]	ForkNet [5]	9.3	14.1	11.5	**16.2**	14.9	14.2
	ScanComplete [127]	16.1	18.4	16.9	**19.8**	17.2	17.3
	SCFusion [128]	18.9	20.0	19.1	**22.6**	20.7	20.5

TABLE 10: Ablation study on loss functions, comparing the nebula loss against a Euclidean loss (i.e. without M_a) and comparing the contribution of the losses metric learning. The results are reported in the average IoU (%) across different categories for object completion [10] in Table 4 and semantic scene completion [128] in Table 8.

trained in an unsupervised way; and, (ii) by labeling features with the assigned anchors, employing metric learning to further separate the clusters in a self-supervised way. The proposed approach showed, on one side, the performance gains when tested on supervised and unsupervised tasks and, on the other, good generalization capabilities to deal with different network architectures and data dimensionality.

REFERENCES

[1] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," *arXiv preprint arXiv:1910.13461*, 2019.

[2] G. Lample and A. Conneau, "Cross-lingual language model pretraining," *arXiv preprint arXiv:1901.07291*, 2019.

[3] A. Dai, C. R. Qi, and M. Nießner, "Shape completion using 3d-encoder-predictor cnns and shape synthesis," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, vol. 3, 2017.

[4] W. Yuan, T. Khot, D. Held, C. Mertz, and M. Hebert, "Pcn: Point completion network," in *3D Vision (3DV), 2018 International Conference on*, 2018.

[5] Y. Wang, D. J. Tan, N. Navab, and F. Tombari, "Forknet: Multi-branch volumetric semantic completion from a single depth image," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 8608–8617.

[6] Y. Yuan, W. Su, and D. Ma, "Efficient dynamic scene deblurring using spatially variant deconvolution network with optical flow guided training," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 3555–3564.

[7] L. Bao, Z. Yang, S. Wang, D. Bai, and J. Lee, "Real image denoising based on multi-scale residual dense block and cascaded u-net with block-connection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 448–449.

[8] V. F. Abrevaya, A. Boukhayma, P. H. Torr, and E. Boyer, "Cross-modal deep face normals with deactivable skip connections," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 4979–4989.

[9] R. Azad, M. Asadi-Aghbolaghi, M. Fathy, and S. Escalera, "Bi-directional convlstm u-net with densley connected convolutions," in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2019, pp. 0–0.

[10] B. Yang, S. Rosa, A. Markham, N. Trigoni, and H. Wen, "Dense 3d object reconstruction from a single depth view," *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[11] M. Liu, L. Sheng, S. Yang, J. Shao, and S.-M. Hu, "Morphing and sampling network for dense point cloud completion," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 11 596–11 603.

[12] H. Xie, H. Yao, S. Zhou, J. Mao, S. Zhang, and W. Sun, "Grnet: Gridding residual network for dense point cloud completion," in *Computer Vision – ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds. Cham: Springer International Publishing, 2020, pp. 365–381.

[13] X. Wen, T. Li, Z. Han, and Y.-S. Liu, "Point cloud completion by skip-attention network with hierarchical folding," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[14] P.-S. Wang, Y. Liu, and X. Tong, "Deep octree-based cnns with output-guided skip connections for 3d shape and scene completion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 266–267.

[15] A. J. Bell and T. J. Sejnowski, "An information-maximization approach to blind separation and blind deconvolution," *Neural computation*, vol. 7, no. 6, pp. 1129–1159, 1995.

[16] A. Hyvärinen, "The fixed-point algorithm and maximum likelihood estimation for independent component analysis," *Neural Processing Letters*, vol. 10, no. 1, pp. 1–5, 1999.

[17] D. L. Swets and J. J. Weng, "Using discriminant eigenfeatures for image retrieval," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 18, no. 8, pp. 831–836, 1996.

[18] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of Machine Learning Research*, vol. 11, no. Dec, pp. 3371–3408, 2010.

[19] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial autoencoders," *arXiv preprint arXiv:1511.05644*, 2015.

[20] Y. Bengio, L. Yao, G. Alain, and P. Vincent, "Generalized denoising auto-encoders as generative models," in *Advances in Neural Information Processing Systems*, 2013, pp. 899–907.

[21] D. J. Im, S. Ahn, R. Memisevic, Y. Bengio *et al.*, "Denoising criterion for variational auto-encoding framework." in *AAAI*, 2017, pp. 2059–2065.

[22] H. Yun, Y. Kim, T. Kang, and K. Jung, "Pairwise context similarity for image retrieval system using variational auto-encoder," *IEEE Access*, 2021.

[23] F. Mueller, F. Bernard, O. Sotnychenko, D. Mehta, S. Sridhar, D. Casas, and C. Theobalt, "Ganerated hands for real-time 3d hand tracking from monocular rgb," in *CVPR*, 2018.

[24] C. Zimmermann and T. Brox, "Learning to estimate 3d hand pose from single rgb images," in *ICCV*, 2017, pp. 4903–4911.

[25] Y. Gao, Y. Wang, P. Falco, N. Navab, and F. Tombari, "Variational object-aware 3d hand pose from a single rgb image," *IEEE Robotics and Automation Letters*, 2019.

[26] M. Patacchiola, P. Fox-Roberts, and E. Rosten, "Y-autoencoders: Disentangling latent representations via sequential encoding," *Pattern Recognition Letters*, vol. 140, pp. 59–65, 2020.

[27] Z. Chen, K. Yin, M. Fisher, S. Chaudhuri, and H. Zhang, "Bae-net: Branched autoencoder for shape co-segmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 8490–8499.

[28] M. Rolinek, D. Zietlow, and G. Martius, "Variational autoencoders pursue pca directions (by accident)," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 406–12 415.

[29] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 3483–3491.

[30] A. Dosovitskiy and T. Brox, "Generating images with perceptual similarity metrics based on deep networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 658–666.

[31] J. Walker, C. Doersch, A. Gupta, and M. Hebert, "An uncertain future: Forecasting from static images using variational autoen-

coders," in *European Conference on Computer Vision*. Springer, 2016, pp. 835–851.

[32] Y. LeCun, "The mnist database of handwritten digits," *http://yann. lecun. com/exdb/mnist/*, 1998.

[33] J. Lu, X. Zhou, Y.-P. Tan, Y. Shang, and J. Zhou, "Neighborhood repulsed metric learning for kinship verification," *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 2, pp. 331–345, 2014.

[34] P. Wohlhart and V. Lepetit, "Learning descriptors for object recognition and 3d pose estimation," in *Proc. IEEE CVPR*, 2015.

[35] B. Kumar, G. Carneiro, I. Reid *et al.*, "Learning local image descriptors with deep siamese and triplet convolutional networks by minimising global loss functions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5385–5394.

[36] Y. Movshovitz-Attias, A. Toshev, T. K. Leung, S. Ioffe, and S. Singh, "No fuss distance metric learning using proxies," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 360–368.

[37] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*. Curran Associates, Inc., 2012, pp. 1097–1105.

[38] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[39] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[40] D. Britz, A. Goldie, M.-T. Luong, and Q. Le, "Massive exploration of neural machine translation architectures," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 1442–1451.

[41] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 2672–2680.

[42] L. Maaløe, C. K. Sønderby, S. K. Sønderby, and O. Winther, "Auxiliary deep generative models," *arXiv preprint arXiv:1602.05473*, 2016.

[43] O. Bojar, R. Chatterjee, C. Federmann, Y. Graham, B. Haddow, M. Huck, A. J. Yepes, P. Koehn, V. Logacheva, C. Monz *et al.*, "Findings of the 2016 conference on machine translation," in *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, 2016, pp. 131–198.

[44] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[45] J. Zhang, J. Jiao, M. Chen, L. Qu, X. Xu, and Q. Yang, "3d hand pose tracking and estimation using stereo matching," *arXiv preprint arXiv:1610.07214*, 2016.

[46] Z. Yu, J. Zheng, D. Lian, Z. Zhou, and S. Gao, "Single-image piece-wise planar 3d reconstruction via associative embedding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1029–1037.

[47] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "Scannet: Richly-annotated 3d reconstructions of indoor scenes," in *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017.

[48] Y. Choi, M. El-Khamy, and J. Lee, "Variable rate deep image compression with a conditional autoencoder," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 3146–3154.

[49] J. Zhang, D.-P. Fan, Y. Dai, S. Anwar, F. S. Saleh, T. Zhang, and N. Barnes, "Uc-net: Uncertainty inspired rgb-d saliency detection via conditional variational autoencoders," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 8582–8591.

[50] A. Habibian, T. v. Rozendaal, J. M. Tomczak, and T. S. Cohen, "Video compression with rate-distortion autoencoders," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 7033–7042.

[51] D. Gong, L. Liu, V. Le, B. Saha, M. R. Mansour, S. Venkatesh, and A. v. d. Hengel, "Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1705–1714.

[52] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction," *Artificial Neural Networks and Machine Learning–ICANN 2011*, pp. 52–59, 2011.

[53] C. Zhang, Y. Liu, and H. Fu, "Ae2-nets: Autoencoder in autoencoder networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2577–2585.

[54] T.-H. Chan, K. Jia, S. Gao, J. Lu, Z. Zeng, and Y. Ma, "Pcanet: A simple deep learning baseline for image classification?" *IEEE transactions on image processing*, vol. 24, no. 12, pp. 5017–5032, 2015.

[55] C. Li, M. Zeeshan Zia, Q.-H. Tran, X. Yu, G. D. Hager, and M. Chandraker, "Deep supervision with shape concepts for occlusion-aware 3d object parsing," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5465–5474.

[56] S. Gidaris, A. Bursuc, N. Komodakis, P. Pérez, and M. Cord, "Boosting few-shot visual learning with self-supervision," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 8059–8068.

[57] H. Wang, L. Yang, X. Rong, J. Feng, and Y. Tian, "Self-supervised 4d spatio-temporal feature learning via order prediction of sequential point cloud clips," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021, pp. 3762–3771.

[58] T. Klinghoffer, P. Morales, Y.-G. Park, N. Evans, K. Chung, and L. J. Brattain, "Self-supervised feature extraction for 3d axon segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 978–979.

[59] N. Dilokthanakul, P. A. Mediano, M. Garnelo, M. C. Lee, H. Salimbeni, K. Arulkumaran, and M. Shanahan, "Deep unsupervised clustering with gaussian mixture variational autoencoders," *arXiv preprint arXiv:1611.02648*, 2016.

[60] L. Yang, N.-M. Cheung, J. Li, and J. Fang, "Deep clustering by gaussian mixture variational autoencoders with graph embedding," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 6440–6449.

[61] A. van den Oord, O. Vinyals, and K. Kavukcuoglu, "Neural discrete representation learning," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 6309–6318.

[62] S. Zhao, J. Song, and S. Ermon, "Infovae: Balancing learning and inference in variational autoencoders," in *Proceedings of the aaai conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 5885–5892.

[63] S. R. Bowman, L. Vilnis, O. Vinyals, A. Dai, R. Jozefowicz, and S. Bengio, "Generating sentences from a continuous space," in *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 10–21. [Online]. Available: https://www.aclweb.org/anthology/K16-1002

[64] Z. Yang, Z. Hu, R. Salakhutdinov, and T. Berg-Kirkpatrick, "Improved variational autoencoders for text modeling using dilated convolutions," in *International conference on machine learning*. PMLR, 2017, pp. 3881–3890.

[65] R. Li, X. Li, C. Lin, M. Collinson, and R. Mao, "A stable variational autoencoder for text modelling," in *Proceedings of the 12th International Conference on Natural Language Generation*. Tokyo, Japan: Association for Computational Linguistics, Oct.–Nov. 2019, pp. 594–599. [Online]. Available: https://www.aclweb.org/anthology/W19-8673

[66] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *International conference on machine learning*. PMLR, 2016, pp. 478–487.

[67] M. Tang, I. B. Ayed, D. Marin, and Y. Boykov, "Secrets of grabcut and kernel k-means," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1555–1563.

[68] M. Norouzi and D. J. Fleet, "Cartesian k-means," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013.

[69] B. Yang, X. Fu, N. D. Sidiropoulos, and M. Hong, "Towards k-means-friendly spaces: Simultaneous deep learning and cluster-

ing," in *international conference on machine learning*. PMLR, 2017, pp. 3861–3870.

[70] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[71] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," *Advances in Neural Information Processing Systems*, vol. 30, pp. 4077–4087, 2017.

[72] J. H. W. Jr., "Hierarchical grouping to optimize an objective function," *Journal of the American Statistical Association*, vol. 58, no. 301, pp. 236–244, 1963.

[73] J. Chang, L. Wang, G. Meng, S. Xiang, and C. Pan, "Deep adaptive image clustering," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5879–5887.

[74] X. Ji, J. F. Henriques, and A. Vedaldi, "Invariant information clustering for unsupervised image classification and segmentation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 9865–9874.

[75] A. Gonzalez-Garcia, J. Van De Weijer, and Y. Bengio, "Image-to-image translation for cross-domain disentanglement," *Advances in neural information processing systems*, vol. 31, pp. 1287–1298, 2018.

[76] N. Hadad, L. Wolf, and M. Shahar, "A two-step disentanglement method," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 772–780.

[77] Z. Zheng and L. Sun, "Disentangling latent space for vae by label relevant/irrelevant dimensions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 192–12 201.

[78] X. Wang, X. Han, W. Huang, D. Dong, and M. R. Scott, "Multi-similarity loss with general pair weighting for deep metric learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5022–5030.

[79] Z. Wu, A. A. Efros, and S. X. Yu, "Improving generalization via scalable neighborhood component analysis," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 685–701.

[80] E. Hoffer and N. Ailon, "Deep metric learning using triplet network," in *International Workshop on Similarity-Based Pattern Recognition*. Springer, 2015, pp. 84–92.

[81] K. Sohn, "Improved deep metric learning with multi-class n-pair loss objective," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, 2016, pp. 1857–1865.

[82] H. Oh Song, Y. Xiang, S. Jegelka, and S. Savarese, "Deep metric learning via lifted structured feature embedding," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4004–4012.

[83] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2. IEEE, 2006, pp. 1735–1742.

[84] J. Goldberger, G. E. Hinton, S. Roweis, and R. R. Salakhutdinov, "Neighbourhood components analysis," *Advances in neural information processing systems*, vol. 17, pp. 513–520, 2004.

[85] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu, "Learning fine-grained image similarity with deep ranking," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1386–1393.

[86] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 815–823.

[87] L. Hu, M. Kan, S. Shan, and X. Chen, "Unsupervised domain adaptation with hierarchical gradient synchronization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 4043–4052.

[88] H. Tang, K. Chen, and K. Jia, "Unsupervised domain adaptation via structurally regularized deep clustering," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 8725–8735.

[89] C. Yang and S.-N. Lim, "One-shot domain adaptation for face generation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 5921–5930.

[90] R. Gong, W. Li, Y. Chen, and L. V. Gool, "Dlow: Domain flow for adaptation and generalization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2477–2486.

[91] E. Schonfeld, B. Schiele, and A. Khoreva, "A u-net based discriminator for generative adversarial networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 8207–8216.

[92] R. Chen, W. Huang, B. Huang, F. Sun, and B. Fang, "Reusing discriminators for encoding: Towards unsupervised image-to-image translation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 8168–8177.

[93] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *The journal of machine learning research*, vol. 17, no. 1, pp. 2096–2030, 2016.

[94] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7167–7176.

[95] Y. Zhang, H. Tang, K. Jia, and M. Tan, "Domain-symmetric networks for adversarial domain adaptation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5031–5040.

[96] S. Cui, S. Wang, J. Zhuo, C. Su, Q. Huang, and Q. Tian, "Gradually vanishing bridge for adversarial domain adaptation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 12 455–12 464.

[97] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2014.

[98] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," in *arXiv:1609.08144*, 2016.

[99] M. Zinkevich, M. Weimer, L. Li, and A. J. Smola, "Parallelized stochastic gradient descent," in *Advances in neural information processing systems*, 2010, pp. 2595–2603.

[100] H. Robbins and S. Monro, "A stochastic approximation method," *Ann. Math. Statist.*, vol. 22, no. 3, pp. 400–407, 09 1951.

[101] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *Journal of the American Statistical Association*, no. just-accepted, 2017.

[102] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, "ShapeNet: An Information-Rich 3D Model Repository," Stanford University — Princeton University — Toyota Technological Institute at Chicago, Tech. Rep. arXiv:1512.03012 [cs.GR], 2015.

[103] C. Couprie, C. Farabet, L. Najman, and Y. LeCun, "Indoor semantic segmentation using depth information," *arXiv preprint arXiv:1301.3572*, 2013.

[104] A. C. Aguilera, P. M. Olmos, A. Artés-Rodríguez, and F. Pérez-Cruz, "Regularizing transformers with deep probabilistic layers," *arXiv preprint arXiv:2108.10764*, 2021.

[105] J. Zhu, Y. Xia, L. Wu, D. He, T. Qin, W. Zhou, H. Li, and T. Liu, "Incorporating bert into neural machine translation," in *International Conference on Learning Representations*, 2019.

[106] S. Takase and S. Kiyono, "Rethinking perturbations in encoder-decoders for fast training," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2021, pp. 5767–5780.

[107] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[108] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[109] M. Luong, E. Brevdo, and R. Zhao, "Neural machine translation (seq2seq) tutorial," *https://github.com/tensorflow/nmt*, 2017.

[110] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2650–2658.

[111] S. F. Bhat, I. Alhashim, and P. Wonka, "Adabins: Depth estimation using adaptive bins," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4009–4018.

[112] J. Varley, C. DeChant, A. Richardson, J. Ruales, and P. Allen, "Shape completion enabled robotic grasping," in *Intelligent Robots*

and Systems (IROS), 2017 IEEE/RSJ International Conference on. IEEE, 2017, pp. 2442–2447.
[113] X. Han, Z. Li, H. Huang, E. Kalogerakis, and Y. Yu, "High-resolution shape completion using deep neural networks for global structure and local geometry inference," in *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2017.
[114] Z. Wu, R. Shou, Y. Wang, and X. Liu, "Interactive shape co-segmentation via label propagation," *Computers & Graphics*, vol. 38, pp. 248–254, 2014.
[115] L. Yi, V. G. Kim, D. Ceylan, I. Shen, M. Yan, H. Su, A. Lu, Q. Huang, A. Sheffer, L. Guibas *et al.*, "A scalable active framework for region annotation in 3d shape collections," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 6, p. 210, 2016.
[116] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas, "Volumetric and multi-view cnns for object classification on 3d data," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5648–5656.
[117] W. Wang, R. Yu, Q. Huang, and U. Neumann, "Sgpn: Similarity group proposal network for 3d point cloud instance segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2569–2578.
[118] L. Yi, H. Su, X. Guo, and L. J. Guibas, "Syncspeccnn: Synchronized spectral cnn for 3d shape segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2282–2290.
[119] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++ deep hierarchical feature learning on point sets in a metric space," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 5105–5114.
[120] Z.-H. Lin, S.-Y. Huang, and Y.-C. F. Wang, "Convolution in the cloud: Learning deformable kernels in 3d graph convolution networks for point cloud analysis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1800–1809.
[121] X. Sun, Y. Wei, S. Liang, X. Tang, and J. Sun, "Cascaded hand pose regression," in *CVPR*, 2015.
[122] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski, "Manhattan-world stereo," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2009, pp. 1422–1429.
[123] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *European conference on computer vision*. Springer, 2012, pp. 746–760.
[124] C. Liu, J. Yang, D. Ceylan, E. Yumer, and Y. Furukawa, "Planenet: Piece-wise planar reconstruction from a single rgb image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2579–2588.
[125] Y. Gao, Y. Wang, P. Falco, N. Navab, and F. Tombari, "Variational object-aware 3-d hand pose from a single rgb image," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4239–4246, 2019.
[126] Z. Yu, J. Zheng, D. Lian, Z. Zhou, and S. Gao, "Single-image piece-wise planar 3d reconstruction via associative embedding," in *CVPR*, 2019, pp. 1029–1037.
[127] A. Dai, D. Ritchie, M. Bokeloh, S. Reed, J. Sturm, and M. Nießner, "Scancomplete: Large-scale scene completion and semantic segmentation for 3d scans," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4578–4587.
[128] S.-C. Wu, K. Tateno, N. Navab, and F. Tombari, "Scfusion: Real-time incremental scene reconstruction with semantic completion," *arXiv preprint arXiv:2010.13662*, 2020.

Yida Wang is a Ph.D. candidate from Technische Universität München (TUM) focusing on 3D understanding and reconstruction based on deep architectures and statistical approaches. His research activities have been published on prestigious conferences such as CVPR, ICCV, ECCV, 3DV, IROS, ICRA and journals such as IJCV, TIP and RA-L. One of his work targeting on 3D point cloud completion is presented with an oral presentation in ECCV 2020.

David Joseph Tan is a Research Scientist at Google. He got his Ph.D. from the Technische Universität München (TUM). In 2017, the culmination of his Ph.D. research on the ultra-fast 6D pose estimation garnered him the Best Demo Award at ISMAR, EXIST-Gründerstipendium from the German government and the Promotionspreise from TUM. Expanding his research in computer vision and machine learning, he continuously publishes in top-tier conferences and journals.

Nassir Navab is a full professor and director of the Laboratories for Computer Aided Medical Procedures at Technical University of Munich (TUM) and Johns Hopkins University with secondary faculty appointments at both Medical Schools. He is also the director of Medical Augmented Reality summer school series at Balgrist Hospital in Zurich. He received the prestigious Siemens Inventor of the Year Award in 2001. He also received the SMIT Technology Award in 2010 and IEEE ISMAR 10 Years Lasting Impact Award in 2015. He had received his PhD from INRIA and University of Paris XI in France and enjoyed two years postdoctoral fellowship at MIT Media Laboratory before joining SCR in 1994. He is Fellow of the MICCAI Society and asked on its board of directors from 2007 to 2012 and from 2014 to 2017. He has been serving on the Steering Committee of the IEEE Symposium on Mixed and Augmented Reality since 2001. He is the author of hundreds of peer reviewed scientific papers and 50 granted US and over 80 international patents. He served as General Chair for MICCAI 2015, ISMAR 2001, 2005 and 2014. He is a funding board member of IPCAI 2010-2021 and Area Chair for ICCV 2022 and ECCV 2020. He is on the editorial board of many international journals including IEEE TMI and MedIA. As of March 2022, his papers have received over 50000 citations and enjoy an h-index of 100.

Federico Tombari is a Research Scientist and manager at Google where he leads an applied research team in computer vision and machine learning. He is also a Lecturer (PrivatDozent) at the Technical University of Munich (TUM). He has 200+ peer-reviewed publications in the field of 3D computer vision and machine learning and their applications to robotics, autonomous driving, healthcare and augmented reality. He got his PhD in 2009 from the University of Bologna, where he was Assistant Professor from 2013 to 2016, and his Habilitation from TUM in 2018. In 2018-19, he was co-founder and managing director of a Munich-based startup on 3D perception for AR and robotics. He regularly serves as Chair and Associate Editor for international conferences and journals (RA-L, ECCV18, IROS20, ICRA20, IROS21, 3DV19, 3DV20, 3DV21 among others). He was the recipient of two Google Faculty Research Awards, one Amazon Research Award, two CVPR Outstanding Reviewer Awards. He has been a research partner of private and academic institutions including Google, Toyota, BMW, Audi, Amazon, Univ. Stanford, ETH and JHU.

Part II

Conclusion and Future Work

Conclusion

6

This dissertation focuses on 3D completion from a single view with learning-based approaches. It mainly investigates two trending 3D data formats, including volumetric data and point cloud as introduced in Section 3.2. In terms of the contributed approaches, Section 5.1 demonstrates our contributions in volumetric space by leveraging 3D convolutions, achieving higher efficiency and accuracy compared to related volumetric methods. Next, our proposed point cloud methods are demonstrated in Section 5.2 to reconstruct with adaptive local resolutions to incorporate the finer geometric details. Apart from 3D completion, we also investigate optimizing latent features in Section 5.3 to solve other tasks such as semantic segmentation and natural language processing.

Processing local features on volumetric data is popular because gridded 3D data is similar to well-organized pixels on an image. Since the local neighborhood is already well-structured, designing encoder-decoder architectures constructed by convolutions to process local and global information is easy. Although given such processing simplicity, volumetric completion has a slow inference speed due to the 3D convolutions. Since the volumetric data back-projected from the depth image is filled with numerous empty voxels, we proposed using the 2D depth image directly as input for the encoder. Consequently, shifting from 3D convolutions to 2D in the encoder reduces the training and inference time. However, in end-to-end training, a 3D decoder could not infer directly on top of the reshaped 2D features. We solve this problem by using discriminative training on latent space [10] so that the 2D encoder can generate latent features in the same domain of 3D VAE latent features.

Another issue is the need for many real 3D annotations for training in real scenarios. In addition, the existing real training data, e.g. NYU dataset, contains wrong semantic annotations, which misleads the optimization while training the parametric model. To solve such a problem, we proposed ForkNet [8], which automatically generates paired realistic data as supervision during training. Such ability is achieved by applying a proposed SDF-Semantic consistency on the graph model. To make the model robust to the incorrect semantics during training, one of ForkNet's decoders focuses on revealing geometries alone.

Overall, the reconstruction from volumetric data is generally constrained by the fixed resolution of the 3D grid. Motivated to increase the resolution, our research is then shifted to point cloud. The computational resources of point cloud methods are focused on reconstructing actual geometries alone, i.e. without the empty spaces. However, the challenge of handling point clouds is their unorganized structure.

Given the unordered point cloud map, our proposed SoftPoolNet [6] automatically sorts every point, ensuring that the latent feature is identical no matter how the input map is

permuted. Based on the 2D latent feature, a regional convolution network involving *soft-pool* operators is structured to reform the 3D shape. SoftPoolNet outperforms other point cloud approaches in numerical evaluations and shows visual advantages.

Due to the compression in the encoder, a typical limitation of SoftPoolNet is that it tends to lose sharp geometries of the input shape, which should be preserved. Standard encoder-decoder architecture preserves input geometry in volumetric data with the help of skip connections between encoder and decoder, which are formulated with element-wise summation or feature concatenation. Such skip-connection cannot be applied in the point cloud because the two subsets of point-wise features from the encoder and decoder are not explicitly indexed with each other in the feature map. Our proposed SoftPool++ [5] overcomes this limitation by using skip connections specifically devised for point clouds, with the help of a transformation matrix that projects the features from the encoder to the decoder and vice versa. It eliminates the domain gap between encoder features and decoder features. Structuring multiple *SoftPool++* modules in an encoder-decoder structure, this work becomes the first to propose a point-wise skip connection with feature transformation. Such a design then overcomes the loss of information in the encoder.

After a careful survey on point cloud completion, we noticed that the related methods could only complete 3D objects but fail when trained and tested on 3D scene completion or 3D semantic scene completion. In contrast to volumetric data, local neighborhoods are not explicitly defined in point clouds, so extracting local structural features for complex point cloud reconstruction, such as scenes, is hard. Previous local descriptors using point cloud convolutions are proposed to extract local features for segmentation and classification. However, a general local descriptor is missing for completion. Therefore, we proposed Disp3D [4] to represent the local structures by matching local sets of points to displacement vectors. The proposed displacement vector-based operator processes the point cloud from a local perspective, which could be used to form both encoders to down-sample point cloud and decoders to up-sample point cloud. Notably, our approach is the first to successfully infer 3D semantic scene completion from a single view through point clouds.

Moreover, we also aimed at generalizing the improvements we achieved in the completion task to other tasks. In particular, to investigate latent feature optimization, we propose optimizing with additional parameters in the latent space. Our proposed variational approach [9] optimizes the latent feature with learnable cluster centers so that the 3D decoder can easily infer the grasping gestures from a single RGB image. Then, inspired by the gravity law, we further improve the performance of previous works by considering the number of samples getting assigned to parameterized cluster centers, which we call *nebula anchors* [3]. The advantage of such anchors extends to multiple tasks, including NLP, image reconstruction, planar reconstruction, point cloud segmentation, hand pose estimation, etc.

Furthermore, our other collaborated works such as structural-SLAM [7] for mapping and localization, Lidar upsampling [2] with BMW, and SecNet [1] at Facebook Reality Labs for eye tracking are all inspired or back-boned by the mentioned works in this dissertation. These proposed methods could be widely adapted to other applications.

Future Works

7

Based on the research conducted to complete this dissertation and other collaborated works, we recommend the following future directions, including (a) efficient volumetric processing, (b) completion without categorical priors, (c) 3D contact detection, and (d) geometries in radiance field.

Efficient volumetric processing. While we have tried to present completion in point cloud instead of volumetric data to increase the local resolution of the completion results, we notice that some volumetric solutions are also worth a try aiming to overcome the resource consumption problem so that output resolution can be practically higher with similar computational resources, e.g. using sparse convolution [100] to process volumetric data, thus allowing a direct comparison with approaches that are targeting on the large scene such as ScanComplete [146].

Completion without categorical priors. As for point cloud completion, completing a scene with enormous occlusions still needs to be qualitatively satisfying with the current approaches, possibly because every scene does not have a strong categorical global shape prior. First, We want to investigate whether point cloud completion on complex scene datasets can be implemented without learning a strong categorical global shape prior [166]. After that, we want to develop solutions to either present such shape prior with the help of some segmentation supervision or propose pipelines that handle scene completion without using a latent shape prior. One potential solution is completing from a local-to-global perspective because single-object completion is always easier with centered targets.

3D contact detection. Considering that our proposed VO-Hand targets hand pose estimation from a single RGB image where the hand interacts with an object. An interesting future direction is using the contact information [167, 168] to correct the estimated 3D hand pose, guaranteeing consistency between pose and contact. Such estimation depends on involving mesh models between both the hand and the interacted object on hand.

Geometries in radiance field. Inspired by the excellent smoothness of meshes extracted from a well-constructed signed distance field, 3D surfaces extracted from an implicit field may have much better geometric precision. Nevertheless, due to the lack of precise 3D mesh supervision, learning such implicit fields directly in 3D space takes much work. Given a set of calibrated multi-view images towards a static scene, there are already works [50, 169, 51, 170] investigating jointly learning the surface and the appearance of a static scene using only the appearance supervision presented in 2D space with the help of a learned radiance field. We plan to investigate recent works about optimizing the plausible implicit field by training with radiance supervision.

Part III

Appendix

Bibliography

[1] Y. Wang, Y. Shen, D. J. Tan, F. Tombari, and S. S. Talathi. "SecNet: Semantic Eye Completion in Implicit Field". In: *Annual Conference on Neural Information Processing Systems*. PMLR. 2023, pp. 241–256 (see pp. 1, 180).

[2] A. Savkin, Y. Wang, S. Wirkert, N. Navab, and F. Tombari. "Lidar Upsampling With Sliced Wasserstein Distance". In: *IEEE Robotics and Automation Letters* 8.1 (2022), pp. 392–399 (see pp. 1, 180).

[3] Y. Wang, D. J. Tan, N. Navab, and F. Tombari. "Self-supervised Latent Space Optimization with Nebula Variational Coding". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022) (see pp. 1, 157, 180).

[4] Y. Wang, D. J. Tan, N. Navab, and F. Tombari. "Learning Local Displacements for Point Cloud Completion". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2022 (see pp. 1, 125, 180).

[5] Y. Wang, D. J. Tan, N. Navab, and F. Tombari. "SoftPool++: An Encoder–Decoder Network for Point Cloud Completion". In: *International Journal of Computer Vision* (2022), pp. 1–20 (see pp. 1, 29, 37–40, 101, 125, 180).

[6] Y. Wang, D. J. Tan, N. Navab, and F. Tombari. "Softpoolnet: Shape descriptor for point cloud completion and classification". In: *European Conference on Computer Vision*. Springer. 2020, pp. 70–85 (see pp. 1, 14, 28, 32, 37–40, 74, 101, 125, 179).

[7] Y. Li, N. Brasch, Y. Wang, N. Navab, and F. Tombari. "Structure-slam: Low-drift monocular slam in indoor environments". In: *IEEE Robotics and Automation Letters* 5.4 (2020), pp. 6583–6590 (see pp. 1, 180).

[8] Y. Wang, D. J. Tan, N. Navab, and F. Tombari. "ForkNet: Multi-branch Volumetric Semantic Completion from a Single Depth Image". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 8608–8617 (see pp. 1, 26, 31, 35, 38, 55, 74, 146, 179).

[9] Y. Gao, Y. Wang, P. Falco, N. Navab, and F. Tombari. "Variational Object-aware 3D Hand Pose from a Single RGB Image". In: *IEEE Robotics and Automation Letters* (2019) (see pp. 2, 146, 180).

[10] Y. Wang, D. J. Tan, N. Navab, and F. Tombari. "Adversarial Semantic Scene Completion from a Single Depth Image". In: *2018 International Conference on 3D Vision (3DV)*. 2018 (see pp. 2, 31, 35, 42, 146, 179).

[11] D. Neven, B. De Brabandere, S. Georgoulis, M. Proesmans, and L. Van Gool. "Towards end-to-end lane detection: an instance segmentation approach". In: *2018 IEEE intelligent vehicles symposium (IV)*. IEEE. 2018, pp. 286–291 (see p. 5).

[12] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. "End-to-end object detection with transformers". In: *European conference on computer vision*. Springer. 2020, pp. 213–229 (see p. 5).

[13] K. He, X. Zhang, S. Ren, and J. Sun. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778 (see pp. 5, 15).

[14] Y. Wang, X. Zhang, T. Yang, and J. Sun. "Anchor detr: Query design for transformer-based detector". In: 36.3 (2022), pp. 2567–2575 (see p. 6).

[15] Y. Liu, T. Wang, X. Zhang, and J. Sun. "Petr: Position embedding transformation for multi-view 3d object detection". In: (2022), pp. 531–548 (see p. 6).

[16] F. Lu and E. Milios. "Globally consistent range scan alignment for environment mapping". In: *Autonomous robots* 4.4 (1997), pp. 333–349 (see p. 6).

[17] W. Yuan, T. Khot, D. Held, C. Mertz, and M. Hebert. "Pcn: Point completion network". In: *2018 International Conference on 3D Vision (3DV)*. IEEE. 2018, pp. 728–737 (see pp. 6, 11, 28, 30, 32, 36–40).

[18] M. Liu, L. Sheng, S. Yang, J. Shao, and S.-M. Hu. "Morphing and sampling network for dense point cloud completion". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 2020, pp. 11596–11603 (see pp. 6, 28, 29, 36, 38–40, 101).

[19] S. Tulsiani, S. Gupta, D. F. Fouhey, A. A. Efros, and J. Malik. "Factoring shape, pose, and layout from the 2d image of a 3d scene". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 302–310 (see p. 6).

[20] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 4040–4048 (see p. 6).

[21] R. Girdhar, D. F. Fouhey, M. Rodriguez, and A. Gupta. "Learning a predictable and generative vector representation for objects". In: *European Conference on Computer Vision*. Springer. 2016, pp. 484–499 (see p. 6).

[22] C. Godard, O. Mac Aodha, and G. J. Brostow. "Unsupervised monocular depth estimation with left-right consistency". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 270–279 (see p. 6).

[23] H. Jung, Y. Kim, D. Min, C. Oh, and K. Sohn. "Depth prediction from a single image with conditional adversarial networks". In: *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2017, pp. 1717–1721 (see p. 6).

[24] Z. Li and N. Snavely. "Megadepth: Learning single-view depth prediction from internet photos". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 2041–2050 (see p. 6).

[25] Z. Zhang. "Microsoft kinect sensor and its effect". In: *IEEE multimedia* 19.2 (2012), pp. 4–10 (see p. 6).

[26] L. Keselman, J. Iselin Woodfill, A. Grunnet-Jepsen, and A. Bhowmik. "Intel realsense stereoscopic depth cameras". In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 2017, pp. 1–10 (see p. 6).

[27] W. Wu, Z. Qi, and L. Fuxin. "Pointconv: Deep convolutional networks on 3d point clouds". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 9621–9630 (see p. 11).

[28] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen. "Pointcnn: Convolution on x-transformed points". In: *Advances in Neural Information Processing Systems*. 2018, pp. 820–830 (see p. 11).

[29] J. Kim, J. K. Lee, and K. M. Lee. "Deeply-recursive convolutional network for image super-resolution". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 1637–1645 (see p. 11).

[30] H. Noh, S. Hong, and B. Han. "Learning deconvolution network for semantic segmentation". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 1520–1528 (see pp. 11, 34).

[31] Y. Yang, C. Feng, Y. Shen, and D. Tian. "Foldingnet: Point cloud auto-encoder via deep grid deformation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 206–215 (see pp. 11, 28, 36–40).

[32] Z.-H. Lin, S.-Y. Huang, and Y.-C. F. Wang. "Convolution in the cloud: Learning deformable kernels in 3D graph convolution networks for point cloud analysis". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 1800–1809 (see pp. 11, 28, 37).

[33] Y. Li and Y. Yuan. "Convergence analysis of two-layer neural networks with relu activation". In: *Advances in neural information processing systems* 30 (2017) (see p. 12).

[34] L.-J. Deng, G. Vivone, W. Guo, M. Dalla Mura, and J. Chanussot. "A variational pansharpening approach based on reproducible kernel Hilbert space and heaviside function". In: *IEEE Transactions on Image Processing* 27.9 (2018), pp. 4330–4344 (see p. 12).

[35] A. Kawamoto, T. Matsumori, S. Yamasaki, T. Nomura, T. Kondoh, and S. Nishiwaki. "Heaviside projection based topology optimization by a PDE-filtered scalar function". In: *Structural and Multidisciplinary Optimization* 44.1 (2011), pp. 19–24 (see p. 12).

[36] N. A. Kudryashov. "Logistic function as solution of many nonlinear differential equations". In: *Applied Mathematical Modelling* 39.18 (2015), pp. 5733–5742 (see p. 12).

[37] J. Berkson. "Application of the logistic function to bio-assay". In: *Journal of the American statistical association* 39.227 (1944), pp. 357–365 (see p. 12).

[38] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. "Attention is all you need". In: *Advances in neural information processing systems*. 2017, pp. 5998–6008 (see pp. 12, 16).

[39] Y. LeCun et al. "LeNet-5, convolutional neural networks". In: *URL: http://yann. lecun. com/exdb/lenet* 20.5 (2015), p. 14 (see p. 14).

[40] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. "Pointnet: Deep learning on point sets for 3d classification and segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 652–660 (see pp. 14, 27, 36, 37, 40, 74, 101).

[41] L. Bao, Z. Yang, S. Wang, D. Bai, and J. Lee. "Real image denoising based on multi-scale residual dense block and cascaded U-Net with block-connection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2020, pp. 448–449 (see p. 15).

[42] Y. Yu, X. Si, C. Hu, and J. Zhang. "A review of recurrent neural networks: LSTM cells and network architectures". In: *Neural computation* 31.7 (2019), pp. 1235–1270 (see p. 15).

[43] M. Ravanelli, P. Brakel, M. Omologo, and Y. Bengio. "Light gated recurrent units for speech recognition". In: *IEEE Transactions on Emerging Topics in Computational Intelligence* 2.2 (2018), pp. 92–102 (see p. 15).

[44] S. Hochreiter and J. Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780 (see p. 15).

[45] A. C. Aguilera, P. M. Olmos, A. Artés-Rodríguez, and F. Pérez-Cruz. "Regularizing Transformers with Deep Probabilistic Layers". In: *Neural Netw.* 161.C (2023), 565–574 (see p. 16).

[46] X. Yu, Y. Rao, Z. Wang, Z. Liu, J. Lu, and J. Zhou. "Pointr: Diverse point cloud completion with geometry-aware transformers". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 12498–12507 (see pp. 16, 28, 36, 40, 125).

[47] L. Floridi and M. Chiriatti. "GPT-3: Its nature, scope, limits, and consequences". In: *Minds and Machines* 30 (2020), pp. 681–694 (see p. 16).

[48] J. Zhu, Y. Xia, L. Wu, D. He, T. Qin, W. Zhou, H. Li, and T. Liu. "Incorporating BERT into Neural Machine Translation". In: *International Conference on Learning Representations*. 2019 (see p. 16).

[49] L. P. Kaelbling, M. L. Littman, and A. W. Moore. "Reinforcement learning: A survey". In: *Journal of artificial intelligence research* 4 (1996), pp. 237–285 (see p. 17).

[50] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. "Nerf: Representing scenes as neural radiance fields for view synthesis". In: *Communications of the ACM* 65.1 (2021), pp. 99–106 (see pp. 17, 181).

[51] Y. Wang, Q. Han, M. Habermann, K. Daniilidis, C. Theobalt, and L. Liu. "NeuS2: Fast Learning of Neural Implicit Surfaces for Multi-view Reconstruction". In: *arXiv preprint arXiv:2212.05231* (2022) (see pp. 17, 181).

[52] L. Yue, H. Shen, Q. Yuan, and L. Zhang. "A locally adaptive L1- L2 norm for multi-frame super-resolution of images with mixed noise and outliers". In: *Signal Processing* 105 (2014), pp. 156–174 (see p. 18).

[53] A. Lydia and S. Francis. "Adagrad—an optimizer for stochastic gradient descent". In: *Int. J. Inf. Comput. Sci* 6.5 (2019), pp. 566–568 (see p. 18).

[54] F. Zou, L. Shen, Z. Jie, W. Zhang, and W. Liu. "A sufficient condition for convergences of adam and rmsprop". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 11127–11135 (see p. 18).

[55] M. C. Mukkamala and M. Hein. "Variants of rmsprop and adagrad with logarithmic regret bounds". In: *International conference on machine learning*. PMLR. 2017, pp. 2545–2553 (see p. 18).

[56] T. Kurbiel and S. Khaleghian. "Training of deep neural networks based on distance measures using RMSProp". In: *arXiv preprint arXiv:1708.01911* (2017) (see p. 18).

[57] R. V. K. Reddy, B. S. Rao, and K. P. Raju. "Handwritten Hindi digits recognition using convolutional neural network with RMSprop optimization". In: *2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS)*. IEEE. 2018, pp. 45–51 (see p. 18).

[58] L. Hu, M. Kan, S. Shan, and X. Chen. "Unsupervised domain adaptation with hierarchical gradient synchronization". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 4043–4052 (see p. 19).

[59] H. Tang, K. Chen, and K. Jia. "Unsupervised domain adaptation via structurally regularized deep clustering". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 8725–8735 (see p. 19).

[60] C. Yang and S.-N. Lim. "One-Shot Domain Adaptation For Face Generation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 5921–5930 (see p. 19).

[61] R. Gong, W. Li, Y. Chen, and L. V. Gool. "Dlow: Domain flow for adaptation and generalization". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 2477–2486 (see p. 19).

[62] E. Schonfeld, B. Schiele, and A. Khoreva. "A u-net based discriminator for generative adversarial networks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 8207–8216 (see p. 19).

[63] R. Chen, W. Huang, B. Huang, F. Sun, and B. Fang. "Reusing discriminators for encoding: Towards unsupervised image-to-image translation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 8168–8177 (see p. 19).

[64] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. "Domain-adversarial training of neural networks". In: *The journal of machine learning research* 17.1 (2016), pp. 2096–2030 (see p. 19).

[65] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell. "Adversarial discriminative domain adaptation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 7167–7176 (see p. 19).

[66] Y. Zhang, H. Tang, K. Jia, and M. Tan. "Domain-symmetric networks for adversarial domain adaptation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 5031–5040 (see p. 19).

[67] S. Cui, S. Wang, J. Zhuo, C. Su, Q. Huang, and Q. Tian. "Gradually vanishing bridge for adversarial domain adaptation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 12455–12464 (see p. 19).

[68] B. Kumar, G. Carneiro, I. Reid, et al. "Learning local image descriptors with deep siamese and triplet convolutional networks by minimising global loss functions". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 5385–5394 (see p. 20).

[69] X. Wang, X. Han, W. Huang, D. Dong, and M. R. Scott. "Multi-similarity loss with general pair weighting for deep metric learning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 5022–5030 (see p. 20).

[70] Z. Wu, A. A. Efros, and S. X. Yu. "Improving generalization via scalable neighborhood component analysis". In: *Proceedings of the European Conference on Computer Vision*. 2018, pp. 685–701 (see p. 20).

[71] E. Hoffer and N. Ailon. "Deep metric learning using triplet network". In: *International Workshop on Similarity-Based Pattern Recognition*. Springer. 2015, pp. 84–92 (see p. 20).

[72] K. Sohn. "Improved deep metric learning with multi-class n-pair loss objective". In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*. 2016, pp. 1857–1865 (see p. 20).

[73] H. Oh Song, Y. Xiang, S. Jegelka, and S. Savarese. "Deep metric learning via lifted structured feature embedding". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 4004–4012 (see p. 20).

[74] R. Hadsell, S. Chopra, and Y. LeCun. "Dimensionality reduction by learning an invariant mapping". In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognitionb*. Vol. 2. IEEE. 2006, pp. 1735–1742 (see p. 20).

[75] J. Goldberger, G. E. Hinton, S. Roweis, and R. R. Salakhutdinov. "Neighbourhood components analysis". In: *Advances in neural information processing systems* 17 (2004), pp. 513–520 (see p. 20).

[76] Y. Movshovitz-Attias, A. Toshev, T. K. Leung, S. Ioffe, and S. Singh. "No fuss distance metric learning using proxies". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 360–368 (see p. 20).

[77] P. Wohlhart and V. Lepetit. "Learning descriptors for object recognition and 3d pose estimation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3109–3118 (see p. 20).

[78] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu. "Learning fine-grained image similarity with deep ranking". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 1386–1393 (see p. 20).

[79] F. Schroff, D. Kalenichenko, and J. Philbin. "Facenet: A unified embedding for face recognition and clustering". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 815–823 (see p. 20).

[80] J. Lu, X. Zhou, Y.-P. Tan, Y. Shang, and J. Zhou. "Neighborhood repulsed metric learning for kinship verification". In: *IEEE transactions on pattern analysis and machine intelligence* 36.2 (2014), pp. 331–345 (see p. 20).

[81] M. Rolinek, D. Zietlow, and G. Martius. "Variational autoencoders pursue pca directions (by accident)". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 12406–12415 (see p. 20).

[82] K. Sohn, H. Lee, and X. Yan. "Learning structured output representation using deep conditional generative models". In: *Advances in neural information processing systems* 28 (2015) (see p. 20).

[83] J. Walker, C. Doersch, A. Gupta, and M. Hebert. "An uncertain future: Forecasting from static images using variational autoencoders". In: *European Conference on Computer Vision*. Springer. 2016, pp. 835–851 (see p. 20).

[84] N. Dilokthanakul, P. A. Mediano, M. Garnelo, M. C. Lee, H. Salimbeni, K. Arulkumaran, and M. Shanahan. "Deep unsupervised clustering with gaussian mixture variational autoencoders". In: *arXiv preprint arXiv:1611.02648* (2016) (see p. 21).

[85] L. Yang, N.-M. Cheung, J. Li, and J. Fang. "Deep clustering by gaussian mixture variational autoencoders with graph embedding". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 6440–6449 (see p. 21).

[86] A. van den Oord, O. Vinyals, and K. Kavukcuoglu. "Neural discrete representation learning". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 2017, pp. 6309–6318 (see p. 21).

[87] S. Zhao, J. Song, and S. Ermon. "Infovae: Balancing learning and inference in variational autoencoders". In: *Proceedings of the aaai conference on artificial intelligence*. Vol. 33. 01. 2019, pp. 5885–5892 (see p. 21).

[88] L. Maaløe, C. K. Sønderby, S. K. Sønderby, and O. Winther. "Auxiliary deep generative models". In: *International conference on machine learning*. PMLR. 2016, pp. 1445–1453 (see p. 21).

[89] S. R. Bowman, L. Vilnis, O. Vinyals, A. Dai, R. Jozefowicz, and S. Bengio. "Generating Sentences from a Continuous Space". In: *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 10–21 (see p. 21).

[90] Z. Yang, Z. Hu, R. Salakhutdinov, and T. Berg-Kirkpatrick. "Improved variational autoencoders for text modeling using dilated convolutions". In: *International conference on machine learning*. PMLR. 2017, pp. 3881–3890 (see p. 21).

[91] R. Li, X. Li, C. Lin, M. Collinson, and R. Mao. "A Stable Variational Autoencoder for Text Modelling". In: *Proceedings of the 12th International Conference on Natural Language Generation*. Tokyo, Japan: Association for Computational Linguistics, 2019, pp. 594–599 (see p. 21).

[92] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, et al. "Google's neural machine translation system: Bridging the gap between human and machine translation". In: *arXiv:1609.08144*. 2016 (see p. 22).

[93] D. P. Kingma and M. Welling. "Auto-Encoding Variational Bayes". In: *2nd International Conference on Learning Representations, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. Ed. by Y. Bengio and Y. LeCun. 2014 (see p. 22).

[94] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. "Variational inference: A review for statisticians". In: *Journal of the American Statistical Association* just-accepted (2017) (see p. 22).

[95] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese. "3d-r2n2: A unified approach for single and multi-view 3d object reconstruction". In: *European conference on computer vision*. Springer. 2016, pp. 628–644 (see pp. 23, 33).

[96] H. Xie, H. Yao, X. Sun, S. Zhou, and S. Zhang. "Pix2vox: Context-aware 3d reconstruction from single and multi-view images". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 2690–2698 (see pp. 24, 33).

[97] H. Xie, H. Yao, S. Zhang, S. Zhou, and W. Sun. "Pix2Vox++: multi-scale context-aware 3D object reconstruction from single and multiple images". In: *International Journal of Computer Vision* 128.12 (2020), pp. 2919–2935 (see pp. 24, 33).

[98] B. Yang, S. Wang, A. Markham, and N. Trigoni. "Robust attentional aggregation of deep feature sets for multi-view 3D reconstruction". In: *International Journal of Computer Vision* 128.1 (2020), pp. 53–73 (see pp. 24, 33).

[99] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser. "Semantic scene completion from a single depth image". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2017 (see pp. 24, 26, 31, 34, 35, 55).

[100] B. Liu, M. Wang, H. Foroosh, M. Tappen, and M. Pensky. "Sparse convolutional neural networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 806–814 (see pp. 25, 181).

[101] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. "PointNet++ deep hierarchical feature learning on point sets in a metric space". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 2017, pp. 5105–5114 (see pp. 25, 28, 29, 36–38, 40).

[102] R. Azad, M. Asadi-Aghbolaghi, M. Fathy, and S. Escalera. "Bi-directional convlstm u-net with densley connected convolutions". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*. 2019, pp. 0–0 (see pp. 26, 34).

[103] A. Kirillov, Y. Wu, K. He, and R. Girshick. "Pointrend: Image segmentation as rendering". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 9799–9808 (see pp. 26, 34).

[104] F. Yang, Q. Sun, H. Jin, and Z. Zhou. "Superpixel segmentation with fully convolutional networks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 13964–13973 (see pp. 26, 34).

[105] A. Dai, C. Ruizhongtai Qi, and M. Nießner. "Shape completion using 3d-encoder-predictor cnns and shape synthesis". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 5868–5877 (see pp. 26, 34, 38).

[106] B. Yang, H. Wen, S. Wang, R. Clark, A. Markham, and N. Trigoni. "3d object reconstruction from a single depth view with adversarial learning". In: *Proceedings of the IEEE International Conference on Computer Vision Workshops*. 2017, pp. 679–688 (see pp. 26, 34).

[107] B. Yang, S. Rosa, A. Markham, N. Trigoni, and H. Wen. "Dense 3D object reconstruction from a single depth view". In: *IEEE transactions on pattern analysis and machine intelligence* (2018) (see pp. 26, 31, 34, 35).

[108] Y. Guo and X. Tong. "View-volume Network for Semantic Scene Completion from a Single Depth Image". In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. Stockholm, Sweden: AAAI Press, 2018 (see pp. 26, 31, 34, 35).

[109] X. Chen, K.-Y. Lin, C. Qian, G. Zeng, and H. Li. "3d sketch-aware semantic scene completion via semi-supervised structure prior". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 4193–4202 (see pp. 26, 35).

[110] D. P. Kingma and J. Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014) (see p. 27).

[111] T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. Aubry. "A Papier-Mâché Approach to Learning 3D Surface Generation". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018 (see pp. 28, 36–40).

[112] X. Wen, Z. Han, Y.-P. Cao, P. Wan, W. Zheng, and Y.-S. Liu. "Cycle4completion: Unpaired point cloud completion using cycle transformation with missing region coding". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 13080–13089 (see pp. 28, 36).

[113] T. Huang, H. Zou, J. Cui, X. Yang, M. Wang, X. Zhao, J. Zhang, Y. Yuan, Y. Xu, and Y. Liu. "Rfnet: Recurrent forward network for dense point cloud completion". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 12508–12517 (see pp. 28, 36).

[114] Y. Shen, C. Feng, Y. Yang, and D. Tian. "Mining point cloud local structures by kernel correlation and graph pooling". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 4548–4557 (see pp. 28, 37).

[115] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas. "Kpconv: Flexible and deformable convolution for point clouds". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 6411–6420 (see pp. 28, 37).

[116] X. Wen, P. Xiang, Z. Han, Y.-P. Cao, P. Wan, W. Zheng, and Y.-S. Liu. "Pmp-net: Point cloud completion by learning multi-step point moving paths". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 7443–7452 (see pp. 28, 36, 38).

[117] P. Xiang, X. Wen, Y.-S. Liu, Y.-P. Cao, P. Wan, W. Zheng, and Z. Han. "Snowflakenet: Point cloud completion by snowflake point deconvolution with skip-transformer". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 5499–5509 (see pp. 28, 36).

[118] L. Zhou, Y. Du, and J. Wu. "3d shape generation and completion through point-voxel diffusion". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 5826–5835 (see pp. 28, 36).

[119] H. Xie, H. Yao, S. Zhou, J. Mao, S. Zhang, and W. Sun. "Grnet: Gridding residual network for dense point cloud completion". In: *European Conference on Computer Vision*. Springer. 2020, pp. 365–381 (see pp. 28, 32, 36–40).

[120] X. Wang, M. H. Ang, and G. H. Lee. "Voxel-based Network for Shape Completion by Leveraging Edge Generation". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 13189–13198 (see pp. 28, 29, 36, 37).

[121] I. Lim, M. Ibing, and L. Kobbelt. "A convolutional decoder for point clouds using adaptive instance normalization". In: *Computer Graphics Forum*. Vol. 38. 5. Wiley Online Library. 2019, pp. 99–108 (see pp. 29, 37).

[122] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove. "DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 165–174 (see p. 29).

[123] J. Chibane, T. Alldieck, and G. Pons-Moll. "Implicit functions in feature space for 3d shape reconstruction and completion". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 6970–6981 (see p. 29).

[124] P. Erler, P. Guerrero, S. Ohrhallinger, N. J. Mitra, and M. Wimmer. "Points2surf learning implicit surfaces from point clouds". In: *European Conference on Computer Vision*. Springer. 2020, pp. 108–124 (see p. 29).

[125] P. Guerrero, Y. Kleiman, M. Ovsjanikov, and N. J. Mitra. "Pcpnet learning local shape properties from raw point clouds". In: *Computer Graphics Forum*. Vol. 37. 2. Wiley Online Library. 2018, pp. 75–85 (see p. 29).

[126] H. Oleynikova, A. Millane, Z. Taylor, E. Galceran, J. Nieto, and R. Siegwart. "Signed distance fields: A natural representation for both mapping and planning". In: *RSS 2016 Workshop: Geometry and Beyond-Representations, Physics, and Scene Understanding for Robotics*. University of Michigan. 2016 (see p. 29).

[127] J. Zhang, Y. Yao, and L. Quan. "Learning signed distance field for multi-view surface reconstruction". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 6525–6534 (see p. 29).

[128] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, et al. "Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera". In: *Proceedings of the 24th annual ACM symposium on User interface software and technology*. 2011, pp. 559–568 (see p. 29).

[129] J. Chibane, T. Alldieck, and G. Pons-Moll. "Implicit Functions in Feature Space for 3D Shape Reconstruction and Completion". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2020 (see p. 29).

[130] M. Kazhdan and H. Hoppe. "Screened poisson surface reconstruction". In: *ACM Transactions on Graphics (ToG)* 32.3 (2013), pp. 1–13 (see p. 30).

[131] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al. "Shapenet: An information-rich 3d model repository". In: *arXiv preprint arXiv:1512.03012* (2015) (see p. 30).

[132] L. P. Tchapmi, V. Kosaraju, H. Rezatofighi, I. Reid, and S. Savarese. "TopNet: Structural Point Cloud Decoder". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 383–392 (see pp. 30, 37–40).

[133] L. Pan, X. Chen, Z. Cai, J. Zhang, H. Zhao, S. Yi, and Z. Liu. "Variational Relational Point Completion Network". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 8524–8533 (see pp. 30, 40).

[134] S. Koch, A. Matveev, Z. Jiang, F. Williams, A. Artemov, E. Burnaev, M. Alexa, D. Zorin, and D. Panozzo. "Abc: A big cad model dataset for geometric deep learning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 9601–9611 (see p. 30).

[135] P. K. Nathan Silberman Derek Hoiem and R. Fergus. "Indoor Segmentation and Support Inference from RGBD Images". In: *European Conference on Computer Vision (ECCV)*. 2012 (see pp. 31, 35, 55).

[136] A. Geiger and C. Wang. "Joint 3d object and layout inference from a single rgb-d image". In: *German Conference on Pattern Recognition (GCPR)*. Springer. 2015 (see p. 31).

[137] D. Lin, S. Fidler, and R. Urtasun. "Holistic scene understanding for 3d object detection with rgbd cameras". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2013 (see p. 31).

[138] S. Liu, Y. HU, Y. Zeng, Q. Tang, B. Jin, Y. Han, and X. Li. "See and Think: Disentangling Semantic Scene Completion". In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Curran Associates, Inc., 2018, pp. 263–274 (see pp. 31, 34, 35).

[139] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. "Scannet: Richly-annotated 3d reconstructions of indoor scenes". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 5828–5839 (see p. 31).

[140] S.-C. Wu, K. Tateno, N. Navab, and F. Tombari. "Scfusion: Real-time incremental scene reconstruction with semantic completion". In: (2020), pp. 801–810 (see p. 31).

[141] M. Tatarchenko, S. R. Richter, R. Ranftl, Z. Li, V. Koltun, and T. Brox. "What do single-view 3d reconstruction networks learn?" In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 3405–3414 (see p. 32).

[142] C. Niu, J. Li, and K. Xu. "Im2struct: Recovering 3d shape structure from a single rgb image". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 4521–4529 (see p. 33).

[143] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. "Going deeper with convolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9 (see p. 34).

[144] K. He, X. Zhang, S. Ren, and J. Sun. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778 (see p. 34).

[145] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon. "KinectFusion: Real-time dense surface mapping and tracking". In: *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*. IEEE. 2011, pp. 127–136 (see p. 34).

[146] A. Dai, D. Ritchie, M. Bokeloh, S. Reed, J. Sturm, and M. Nießner. "Scancomplete: Large-scale scene completion and semantic segmentation for 3d scans". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 4578–4587 (see pp. 34, 181).

[147] S. Reed, A. van den Oord, N. Kalchbrenner, S. G. Colmenarejo, Z. Wang, Y. Chen, D. Belov, and N. de Freitas. "Parallel multiscale autoregressive density estimation". In: *Proceedings of the 34th International Conference on Machine Learning (ICML)*. JMLR. org. 2017 (see p. 34).

[148] B. Yang, S. Rosa, A. Markham, N. Trigoni, and H. Wen. "3D Object Dense Reconstruction from a Single Depth View". In: *arXiv preprint arXiv:1802.00411* (2018) (see p. 34).

[149] M. Gadelha, S. Maji, and R. Wang. "3d shape induction from 2d views of multiple objects". In: *International Conference on 3D Vision (3DV)*. 2017 (see p. 35).

[150] P. Zhang, W. Liu, Y. Lei, H. Lu, and X. Yang. "Cascaded context pyramid for full-resolution 3D semantic scene completion". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 7801–7810 (see p. 35).

[151] Y. Cai, X. Chen, C. Zhang, K.-Y. Lin, X. Wang, and H. Li. "Semantic scene completion via integrating instances and scene in-the-loop". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 324–333 (see p. 35).

[152] B. Gong, Y. Nie, Y. Lin, X. Han, and Y. Yu. "ME-PCN: Point Completion Conditioned on Mask Emptiness". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 12488–12497 (see p. 36).

[153] X. Wen, T. Li, Z. Han, and Y.-S. Liu. "Point Cloud Completion by Skip-Attention Network With Hierarchical Folding". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020 (see pp. 37, 38).

[154] J. Zhang, W. Chen, Y. Wang, R. Vasudevan, and M. Johnson-Roberson. "Point set voting for partial point cloud analysis". In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 596–603 (see p. 38).

[155] X. Wang, M. H. A. J., and G. H. Lee. "Cascaded Refinement Network for Point Cloud Completion". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020 (see pp. 38–40).

[156] W. Zhang, Q. Yan, and C. Xiao. "Detail preserved point cloud completion via separated feature aggregation". In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXV 16*. Springer. 2020, pp. 512–528 (see pp. 39, 40).

[157] Z. Huang, Y. Yu, J. Xu, F. Ni, and X. Le. "PF-Net: Point fractal network for 3D point cloud completion". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 7662–7670 (see p. 39).

[158] L. Pan. "Ecg: Edge-aware point cloud completion with graph convolution". In: *IEEE Robotics and Automation Letters* 5.3 (2020), pp. 4392–4398 (see p. 40).

[159] L. Yariv, J. Gu, Y. Kasten, and Y. Lipman. "Volume rendering of neural implicit surfaces". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 4805–4815 (see p. 39).

[160] P. Wang, L. Liu, Y. Liu, C. Theobalt, T. Komura, and W. Wang. "Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction". In: *arXiv preprint arXiv:2106.10689* (2021) (see p. 39).

[161] F. Darmon, B. Bascle, J.-C. Devaux, P. Monasse, and M. Aubry. "Improving neural implicit surfaces geometry with patch warping". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 6260–6269 (see p. 39).

[162] X. Long, C. Lin, P. Wang, T. Komura, and W. Wang. "Sparseneus: Fast generalizable neural surface reconstruction from sparse views". In: *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXII*. Springer. 2022, pp. 210–227 (see p. 39).

[163] H. Wang, Q. Liu, X. Yue, J. Lasenby, and M. J. Kusner. "Unsupervised point cloud pre-training via occlusion completion". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 9782–9792 (see p. 40).

[164] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. "ShapeNet: An Information-Rich 3D Model Repository". In: *CoRR* abs/1512.03012 (2015) (see p. 55).

[165] M. Zinkevich, M. Weimer, L. Li, and A. J. Smola. "Parallelized stochastic gradient descent". In: *Advances in neural information processing systems*. 2010, pp. 2595–2603 (see p. 146).

[166] R. Hanocka, G. Metzer, R. Giryes, and D. Cohen-Or. "Point2Mesh: A Self-Prior for Deformable Meshes". In: *ACM Trans. Graph.* 39.4 (2020) (see p. 181).

[167] D. Tzionas and J. Gall. "3d object reconstruction from hand-object interactions". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 729–737 (see p. 181).

[168] B. Tekin, F. Bogo, and M. Pollefeys. "H+ o: Unified egocentric recognition of 3d hand-object poses and interactions". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 4511–4520 (see p. 181).

[169] A. Pumarola, E. Corona, G. Pons-Moll, and F. Moreno-Noguer. "D-nerf: Neural radiance fields for dynamic scenes". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 10318–10327 (see p. 181).

[170] Z. Li, T. Müller, A. Evans, R. H. Taylor, M. Unberath, M.-Y. Liu, and C.-H. Lin. "Neuralangelo: High-Fidelity Neural Surface Reconstruction". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 8456–8465 (see p. 181).

List of Figures

List of Tables