Fakultät für Informatik
Technische Universität München

Learning Priors From RGB-D Data for 3D Scene Understanding

Ji Hou

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Die Dissertation wurde am 24.06.2021 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 28.10.2021 angenommen.

Acknowledgement

Studying Ph.D is a long-term journey on how to do research and more importantly on how to manage the time and mentality. During this process, I have experienced endeavour and success, but also met challenges. I am indebted to many people that helped me along the road and I would like to take this opportunity to acknowledge the many people whose help and support made this thesis possible.

First and foremost, I would like to express my sincere gratitude to my supervisor Prof. Dr. Matthias Nießner for giving me the opportunity to join his group and for his guidance on the research and presentation skills throughout my doctoral study. Furthermore, I would like to thank Prof. Dr. Angela Dai for teaching me step by step in the beginning and all the help in my projects. I owe big thanks to Dr. Justus Thies and Dr. Kiwon Um for sharing their experiences on many things. I also want to thank the defense committee members: Prof. Dr. Stefan Leutenegger and Prof. Niloy J. Mitra for the help and valuable discussions for my thesis.

Moreover, I would like to thank all my colleagues. Many thanks to Aljaz Bozic and Armen Avetisyan for sharing the knowledge unreservedly. I want to thank Andreas Roessler for meaningful discussion about life and research. Dejan Azinović and Yawar Siddiqui deserve mention too, thank for reading the drafts of my submissions and providing thoughtful and invaluable feedback. I also want to thank Manuel Dahnert for bringing us so much fun by making all the memes. I am also grateful to have accompany from my co-workers Norman Mueller, Zhenyu Chen, Hao Yu, Guy Gafni, Shivangi Aneja, Andrei Burov, Christian Diller, Pablo Palafox, Alexey Bokhovkin, as well as generous helps from Susanne Weitz, Sebastian Wohner and Assia Franzmann. Many Thanks for Mengyu Chu, Wei He, You Xie and Junpeng Wang from Thuerey's group for the interesting and entertaining chat in my spare time. Thanks to all the great interns, Yang Li, Shijie Li, Xiaochen Fan, Jingwei Huang, Vassilis Choutas, Hassan Abu Alhaija, Yizhak Ben-Shabat and Cheng Lin, from all over the world to bring us so much happiness in the summer time. During my Ph.D, I have had unforgettable internship at Facebook AI Research. I want to thank my mentors at Facebook, Saining Xie and Benjamin Graham for the guidance during my internship.

Saving the best for last, I want to thank my parents, Xiaoliang Hou and Sumin Wang for their unconditionally support thoughrout my Ph.D. Finally, I want to acknowledge my wife, Ting Wang, for her endless love and support. Thanks for your understanding and company during the hardest time in my Ph.D journey. Thanks for your encouragement when I am upset especially during paper rejection. I am so lucky to have you and looking forward to our future. To all my family, mentors, and colleagues: thank you for being a part of my life!

Abstract

Understanding 3D environments is a long-term research topic since many years in computer vision. It is fundamental to understand 3D surroundings in many real-world computer vision applications, e.g. interactivity in robotics or AR/VR. With recent breakthroughs in deep learning, the computer vision community has made tremendous progress on perception in images. However, research in 3D perception has not been fully explored. Thanks to the commodity 3D sensors, such as the Kinect series, a variety of RGB-D datasets have been collected to enhance the 3D perception research. In this dissertation, we aim to investigate possible deep-learning-based solutions for 3D perception based on RGB-D data.

First, we propose 3D-SIS, a novel 3D semantic instance segmentation algorithm leveraging not only geometry but also color information. Our framework uses an anchor mechanism to regress 3D bounding box locations, classify semantic labels, and segment the 3D shape inside the bounding box. We further show a performance boost by fusing the color features learned from 2D CNNs with geometric features learned from 3D CNNs. Instance segmentation is the most complex and important task in perception as it combines detection, classifications and segmentation. Our work shows the possibility of 3D semantic instance segmentation.

Furthermore, we propose RevealNet, the first 3D semantic instance completion framework. In this work, we aim to "seeing behind objects", namely completing the missing geometry of each object in RGB-D Scans additional to the output of instance semantic segmentation. We leverage 3D-SIS framework and add the completion branch network to complete 3D object. Since there are no ground truth in 3D reconstruction in terms of perfect object geometry, we use CAD model aligned to the 3D scan as ground-truth. Intuitively, better 3D reconstruction quality leads to better performance of downstream tasks. Our experiments also observe a performance boost for detection and instance segmentation tasks with our completion branch.

Data is critical for learning, especially 3D data that is even harder to acquire and annotate. We explore data-efficient scenarios for 3D perceptual tasks, namely learning with limited annotated data. We propose a data-efficient benchmark with limited annotations (LA) and limited reconstructions (LR) for three popular 3D perceptual tasks. For example, we propose to use e.g. only 1% of annotated points for training semantic segmentation task in LA. We also propose an unsupervised pre-training algorithm leveraging point-wise and spatial contexts contrastive learning. Our experiments show a significant improvement with our pre-training algorithms on several popular benchmarks and downstream tasks. We show better performance in both data-efficient scenarios and on all currently available data.

Consequently, we discuss the limitations and potential future directions of our research.

Zusammenfassung

Das Verständnis von 3D-Umgebungen ist seit vielen Jahren ein langfristiges Forschungsthema in der Computer-Vision. Es ist von grundlegender Bedeutung, die 3D-Umgebung in vielen realen Computer-Vision-Anwendungen zu verstehen, z.B. Interaktivität in der Robotik oder AR/VR. Mit den Durchbrüchen beim Deep Learning hat die Computer-Vision-Community enorme Fortschritte bei der Wahrnehmung in Bildern erzielt. Die Forschung zur 3D-Wahrnehmung wurde jedoch noch nicht vollständig erforscht. Dank der Standard-3D-Sensoren wie der Kinect-Serie wurde eine Vielzahl von RGB-D Datasets gesammelt, um die 3D-Wahrnehmungsforschung zu verbessern. In dieser Dissertation wollen wir mögliche Deep Learning basierte Lösungen für die 3D-Wahrnehmung untersuchen, die auf RGB-D Daten basieren.

Zunächst schlagen wir 3D-SIS vor, einen neuartigen Algorithmus zur Segmentierung semantischer 3D-Instanzen, der nicht nur die Geometrie, sondern auch die Farbinformationen wirksam nutzt. Unser Framework verwendet einen Ankermechanismus, um die Positionen von 3D Begrenzungsrahmen zu regressieren, semantische Beschriftungen zu klassifizieren und die 3D-Form innerhalb des Begrenzungsrahmens zu segmentieren. Wir zeigen weiterhin eine Leistungssteigerung, indem wir die aus 2D CNNs gelernten Farbmerkmale mit den aus 3D CNNs gelernten geometrischen Merkmalen verschmelzen. Die Instanzsegmentierung ist die komplexeste und wichtigste Aufgabe in der Wahrnehmung, da sie Erkennung, Klassifizierung und Segmentierung kombiniert. Unsere Arbeit zeigt die Möglichkeit der Segmentierung semantischer 3D-Instanzen.

Darüber hinaus schlagen wir RevealNet vor, das erste Rahmenwerk zur 3D Semantic Instance Completion. In dieser Arbeit wollen wir "hinter Objekte sehen", indem wir die fehlende Geometrie jedes Objekts in RGB-D Scans zusätzlich zur Ausgabe der Instanzsemantiksegmentierung vervollständigen. Wir nutzen das 3D-SIS Framework und fügen das Netzwerk für die Vervollständigung des 3D Objekts hinzu. Da es bei der 3D Rekonstruktion keine Grundwahrheit in Bezug auf eine perfekte Objektgeometrie gibt, verwenden wir ein auf den 3D Scans ausgerichtetes CAD Modell als Grundwahrheit. Intuitiv führt eine bessere 3D Rekonstruktionsqualität zu einer besseren Leistung von nachgeschalteten Aufgaben. Unsere Experimente beobachten auch eine Leistungssteigerung für Erkennungs- und Instanzsegmentierungsaufgaben mit unsere Vervollständigung.

Daten sind für das Lernen von entscheidender Bedeutung, insbesondere 3D Daten, die noch schwieriger zu bekommen und zu annotatieren sind. Wir untersuchen daten-effiziente Szenarien für 3D Wahrnehmungsaufgaben, nämlich Lernen mit begrenzten annotatierten Daten. Wir schlagen einen daten-effizienten Benchmark mit begrenzten Annotationen (LA) und begrenzten Rekonstruktionen (LR) für drei beliebte 3D Perception Aufgaben vor. Zum Beispiel schlagen wir vor, z.B. nur 1% der mit Anmerkungen versehenen Punkte für das Training der semantischen Segmentierungsaufgabe in LA. Wir schlagen auch einen unbeaufsichtigten Algorithmus vor dem Training vor, der punktuelles und räumliches Contrastive Loss nutzt. Unsere Experimente zeigen eine signifikante Verbesserung mit unseren Algorithmen vor dem Training bei mehreren gängigen Benchmarks und nachgelagerten Aufgaben. Wir zeigen eine bessere Leistung sowohl in daten-effizienten Szenarien als auch in allen derzeit verfügbaren Daten.

Infolgedessen diskutieren wir die Grenzen und möglichen zukünftigen Richtungen unserer Forschung.

Contents

Part I

Introduction

1 Introduction

3D Computer Vision has been a very popular research topic since last century, which mainly researches on two counterparts, 3D reconstruction and 3D understanding. 3D reconstruction tries to digitize the real world from images. Its applications range from online shopping, 3D printing to computer video games. Classic image-based 3D reconstruction algorithms, such as Structure from Motion (SfM) [1] and Multi-View Stereo (MVS)[2], leverage motion and stereo correspondence as their main cue to estimate camera parameters and dense 3D reconstruction. Recently due to the hardware development, commodity 3D sensors become readily available (see Figure 1.1. With the help of inertial measurement unit (IMU) and depth camera, it is much easier to get 3D reconstructions of much better quality, e.g. BundleFusion [3]. Taking the advantage of developed 3D sensors, large-scale RGB-D data, such as ScanNet [4], has been collected to enhance the 3D understanding research.

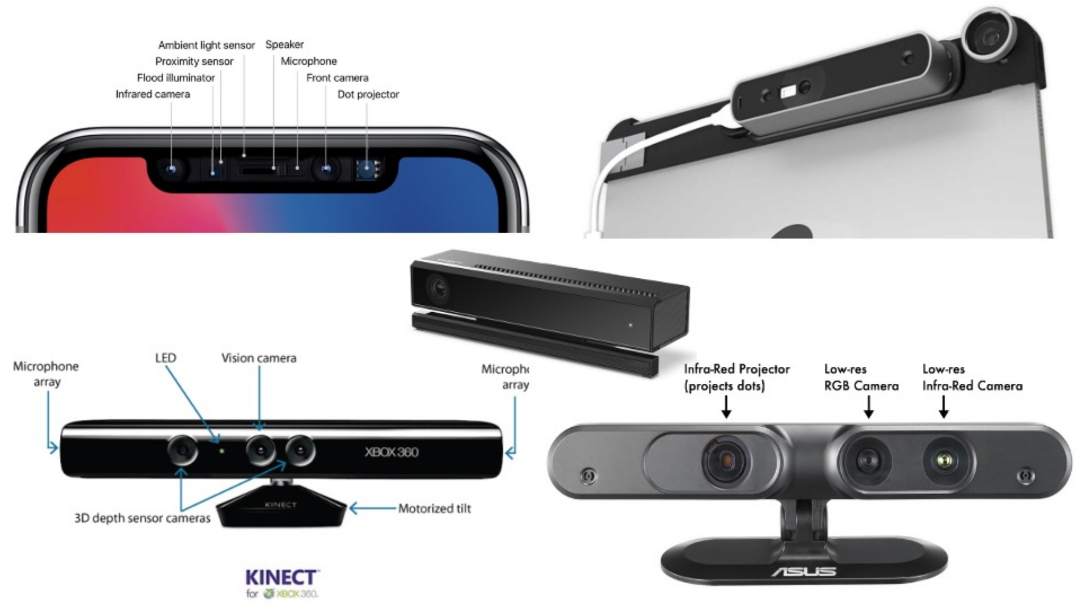


Figure 1.1: Depth Sensors on the Market. Commodity 3D Sensors are now readily available on the market from a variety of companies. Portable sensors are easily mounted on mobile devices and carried around for collecting data.

3D Understanding or 3D perception aims to understand the surrounding 3D environments, e.g. semantic segmentation or object detection. Its application ranges from robotics interactivity to autonomous driving. As prerequisite of understanding 3D environments, reconstructing better quality geometry is a must. As large scale and high quality 3D reconstruction is readily available as mentioned above, developing 3D understanding algorithms is emerging

The computer vision community has witness a tremendous progress in semantically understanding images since decades. From traditional machine learning techniques and hand-crafted feature descriptors, such as Support Vector Machine (SVM) [5] and Scale-Invariant Feature Transform (SIFT) [6], to Deep Learning era, the metrics on benchmarks, such as ImageNet [7] and COCO [8], has been progressively improved. In this dissertation, we aim to extend the deep learning techniques to 3D understanding domain by leveraging the large scale RGB-D data and advanced deep learning techniques.

We first show the possibility of 3D understanding on RGB-D data with deep learning techniques. We introduce 3D-SIS, a very first 3D semantic instance segmentation algorithm. Inspired by R-CNN series in 2D perception work, we introduce the anchor mechanism into 3D perception. We deploy anchors averagely in the 3D feature space with which we also fuse the color features learned from 2D CNNs. The anchors spreading in the space will be regressed to their closed objects that have most similar sizes as the anchor. This stage is called bounding boxes regression. Afterwards, we crop the regions of features defined by the regressed anchors. Using those cropped features, we further predict each object's semantic class label as well as their 3D shapes. Our experiments show a significant improvement over methods that operate on single frame. We also show using color features learned from 2D CNNs outperforms simple RGB color features. In this work, we show a deep learning based algorithm jointly learning features from RGB and geometry using multi-view RGB-D input recorded with commodity RGB-D sensors on a very important 3D understanding task.

As discussed previously, good quality of 3D reconstruction is critical for 3D scene understanding in RGB-D scans. However, 3D scans have holes and missing geometry due to missing view angles in the reconstruction process. To this end, we introduced the new task of semantic instance completion. We propose to combine the completion task (to learn complete missing geometry) together with 3D instance segmentation task. In this newly proposed task, we tackle the problem of "seeing behind objects" by predicting the missing geometry of individually segmented objects in RGB-D scans. This enables a variety of applications in terms of complex interactions with objects in 3D, such as robotic grasping. Even without seeing behind the objects, robot arm needs to know the object's full 3D shape. Furthermore, we introduce RevealNet, the first 3D semantic instance completion approach to jointly detect objects and predict their complete geometry. Our proposed 3D CNNs learn from both color and geometry features to detect and classify objects, then predicts the voxel occupancy for the complete geometry of the object in an end-to-end fashion, which can be run on a full 3D scan in a single forward pass. On both real and synthetic scan data, we significantly outperform state-of-the-art approaches that do instance segmentation and completion separately. To verify our point that good quality geometry leads to better 3D understanding, we also show experiments that using completion loss significantly improves the perception performance, such as 3D detection and instance segmentation.

Previously introduced algorithms highly rely on data, not only on quality of 3D reconstruction but also on richness of collected and annotated data. In general, data is critical for data-driven approaches. Deep learning is known for requiring huge amount of annotated data for training. To this end, 3D data is even harder and more complex to acquire and annotate. In the following, we focus on how to train a 3D perceptual model with fewest annotated data. Therefore, we focus on data-efficient 3D scene understanding through a novel unsupervised pre-training algorithm that integrates the scene contexts in the point-wise contrastive learning framework. We first define the problem by proposing a data-efficient benchmark with two common scenarios, namely limited annotations (LA) and limited reconstructions (LR). For each case, we propose to evaluate on three popular 3D perceptual tasks, i.e. 3D semantic segmentation, instance segmentation and detection. For example, we propose to use e.g. only 1% of annotated points for training semantic segmentation task in 3D semantic segmentation task or only one annotated bounding box per scene for training a 3D detection model. Then, we propose our unsupervised pre-training algorithm leveraging point-wise and spatial contexts contrastive learning. Our experiments show a significant improvement with our pre-training algorithms on several popular benchmarks and downstream tasks. We show better performance in both data-efficient scenarios and on all currently available data. As conclusion, we show the possibility of using extremely few data or annotations to achieve competitive performance leveraging representation learning. Our results and findings are very encouraging and can potentially open up new opportunities in 3D (interactive) data collection, unsupervised 3D representation learning, and large-scale 3D scene understanding.

We start with a specific and very important 3D scene understanding task, namely 3D Semantic Instance Segmentation, and introduce our algorithm 3D SIS, where we show the possibility to work on RGB-D data with deep learning. Following, we introduce RevealNet, the first 3D Semantic Instance Completion algorithm to show the benefits of better object geometry to scene understanding. Finally, we focus on data-efficient scenarios for 3D scene understanding. As a summary, we contribute in the following three aspects in 3D scene understanding.

- We introduced 3D-SIS, a very first 3D Semantic Instance Segmentation Algorithm, that jointly learn from geometry and color features.

- We introduced the new task 3D Semantic Instance Completion task which is important for many real-world robotics applications. We also propose RevealNet, the first 3D Semantic Instance Completion framework and show a significantly improvement on other scene understanding tasks by completing instances.

- We explored data-efficient learning on 3D scene understanding task. By proposing Contrastive Scene Contexts, a novel unsupervised pre-training algorithm, we show the possibility of of using extremely few data or annotations to achieve competitive performance.

1.1 Dissertation Overview

This thesis is structured in 7 chapters that are grouped into three parts as following:

- **Part I:** Introduction (Chapters 1–2)
  - Chapter 1 (Introduction) introduces the history and recent development of 3D scene understanding and our contributions to the community.
  - Chapter 2 (Theoretical Fundamentals) explains basic concepts on 3D scene understanding to assist understanding the thesis.
- **Part II:** 3D Scene Understanding and Data-Efficient Learning (Chapters 3–5)
  - Chapter 3 introduces our work 3D-SIS on 3D Semantic Instance Segmentation that is a fundamental task towards understanding 3D environments.
  - Chapter 4 introduces our work RevealNet that further improves to better understand 3D environments in the scene by completing the missing geometry of the scan.
  - Chapter 5 introduces our work on efficiently training 3D scene understanding models to better understand 3D scenes with extremely few annotated 3D data.
- **Part III:** Conclusion & Outlook (Chapters 6–7)
  - Chapter 6 (Conclusion) summarizes our proposed methods and concludes our contributions.
  - Chapter 7 (Outlook) discusses the existing problems in our proposed methods and hints the potential direction.

1.2 Contributions

This thesis addresses the existing problems in 3D Scene Understanding with learned 3D priors, including specific tasks such as 3D Instance Segmentation and Completion, as well as the performance of mentioned tasks in data-efficient scenarios. For 3D Instance Segmentation, we propose 3D-SIS [9], a top-down anchor-based method to segment each 3D instance in 3D scenes. 3D-SIS jointly leverages geometry and color information to achieve accurate instance segmentation results. Instance Segmentation has holes and incompleteness on the geometry surface, and therefore we propose RevealNet [10] to further complete the missing geometry on the shape surface. RevealNet further indicates that completed geometry can benefit the segmentation performance. Besides specific tasks, data is a critical factor for learning 3D priors. To solve this problem, we propose contrastive scene contexts [11], a novel pre-training algorithm based on contrastive learning framework. In this paper, we first propose the important benchmark on data-efficent 3D scene understanding, and further show the effectiveness of the proposed pre-training method in different scene understanding tasks in data-efficient scenarios. More specifically, structured by publications, this thesis is built from the following contributions:

- We introduce our proposed method 3D-SIS for 3D Semantic Instance Segmentation. 3D-SIS is among the very first approaches targeting this important problem. We first introduce and adapt the anchor mechanism into this 3D task and fuse the 2D color information to achieve accurate predictions both on bounding box level as well as surface geometry. 3D-SIS is the first paper to make the 3D instance segmentation possible in a top-down fashion. With the help of color and geometry information, 3D-SIS achieved the state-of-the-art segmentation results and experimentally indicates the effectiveness of fusing 2D color into 3D scene understanding tasks. The method development and implementation was done by the first author. Data generation and baselines were done with the help of Angela Dai. Discussions with the co-authors led to the final paper [9].

- We introduce RevealNet for 3D Instance Completion. RevealNet is the first approach targeting the task of 3D Semantic Instance Completion. Compared to 3D-SIS, RevealNet not only segment each instance but also complete them in 3D scans. RevealNet leverages an autoencoder architecture, and therefore is able to get rid of the second backbone (not like 3D-SIS that have two backbones). Our approach first does instance segmentation and then further completes each object's surface geometry that are segmented out from first stage. RevealNet achieved state-of-the-art results compared to its alternative baselines. Our approach also indicates that better geometry can help semantic understanding. The method development and implementation was done by the first author. Alternative baselines were provided by Angela Dai. Discussions with the co-authors led to the final paper [10].

- To solve the data huger problem of training deep learning models, we explore the data-efficient 3D scene understanding with contrastive scene contexts. It is widely known that deep learning methods can consume huge amount of data in order to train a state-of-the-art model. This problem becomes even more critical in 3D Scene Understanding, since the annotation and collection of 3D data are more expensive compared to 2D. Therefore, we propose the data-efficient 3D scene understanding benchmark to systematically research on this problem. Meanwhile, we also propose our solution, which is a self-supervised pre-training method leveraging contrastive learning on point-wise loss and scene contexts. With our proposed approach, we can train competitive deep learning models with only 0.1% annotations that achieve 96% of the baseline performance that uses full annotations. Our data-efficient benchmark on 3D scene understanding attracts wide attentions pointing a meaningful and promising research direction, and already has a number of competitive submissions. The method development and implementation was done by the first author. Discussions with the co-authors led to the final paper [11]

1.3 List of Publications

J. Hou, A. Dai, and M. Nießner, “3D-SIS: 3D Semantic Instance Segmentation of RGB-D Scans,” in *Proceedings of Computer Vision and Pattern Recognition (CVPR), IEEE*, 2019

J. Hou, A. Dai, and M. Nießner, “RevealNet: Seeing Behind Objects in RGB-D Scans,” in *Proceedings of Computer Vision and Pattern Recognition (CVPR), IEEE*, 2020

J. Hou, B. Graham, M. Nießner, and S. Xie, “Exploring Data-Efficient 3D Scene Understanding with Contrastive Scene Contexts,” in *Proceedings of Computer Vision and Pattern Recognition (CVPR), IEEE*, 2021

2 Fundamentals and Methods

3D Computer Vision has been developing very fast equipped with deep learning techniques. In this dissertation, we discuss about our algorithms proposed for 3D scene understanding and its efficient learning. As deep learning is the fundamental technique used in our proposed algorithms, some fundamental concepts and basic algorithms of deep learning is described in Appendix for better understanding the thesis.

3D Deep Learning mainly researches on training neural network on 3D data, e.g. point cloud or voxel grids. In Section 2.1, we introduce different representations of 3D data that can be used for training a 3D neural network. In Section 2.2, we introduce a variety of 3D neural network architectures designed for consuming different types of 3D data. At last in Section 2.3, we introduce several popular 3D datasets used for training 3D neural networks and popular tasks commonly evaluated in 3D scene understanding.

2.1 3D Representation

Deep learning has achieved great success in 2D domain, where neural networks learn primarily from images. In 3D computer vision, we can still leverage the power of deep learning techniques, such as neural network architectures, optimizations as well as the regularization and so on. In 3D domain, however, the network has to learn from a totally different data representations from 2D scenarios. In the following sections, we will give a brief introduction on common 3D data representations used in 3D computer vision.

2.1.1 Point Cloud

Point cloud is the simplest way to represent 3D geometry in computer vision. A point cloud is a set of data points in space. Each point is represented by its set of X, Y and Z coordinate, in some cases also with a list of rgb values to represent color for this point. The xyz coordinates represent 3D geometry in an unstructual way. More specifically, points can appear differently in the space even for representing the same shape. Point cloud is also permute invariant, i.e. the order of points should not matter in the representation. We show a illustration of point clouds representing different 3D objects in Figure 2.1.

There are multiple ways to generate point clouds. LiDAR is the most common way to acquire point cloud for outdoor scenes. LiDAR is a method for determining ranges with a laser and measuring the time for the reflected light to return to the receiver. LiDAR is an acronym of ”light detection and ranging” or ”laser imaging, detection, and ranging”. LiDAR has some important factors, such as range and lines. Range denotes

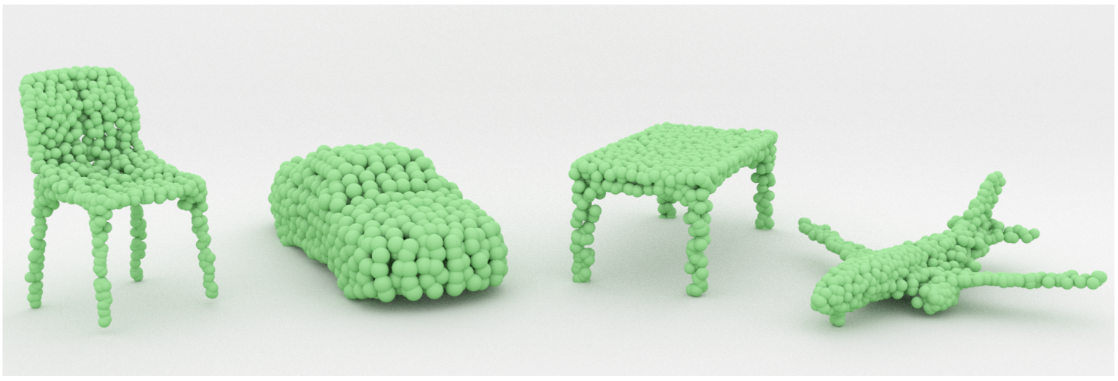


Figure 2.1: Point Cloud Representation. Point cloud can be used to represent 3D geometry of different shapes [15], such as table, chair etc.

the maximum distance it can detect and lines tell how sparse the point cloud is. For instance, an 120-m and 64-line LiDAR can detect objects as far as 120 meters and rotates 64 lasers and measured the time of flight to calculate distance of surrounding objects. In general, LiDAR produces very sparse point cloud. We show classic point clouds produced by LiDAR sensor in Figure 2.2. Using LiDAR to detect 3D environments on the streets is a common practice for autonomous driving.

Different from autonomous driving, robotics application normally runs in indoor environments, where RGB-D cameras are often used. RGB-D cameras not only capture the color information like normal cameras, but also detect a dense depth map. Leveraging depth values, pixel coordinates and intrinsic matrix of the camera, we can easily generate a point cloud. To this end, point clouds produced by RGB-D camera are generally much denser than LiDAR.

2.1.2 Voxel Grid

Compared to point cloud, voxel grid is structured data, i.e. voxel grid is identical for representing the same geometry. Voxel grid is normally represented by a 3D tensor and a voxel size. Voxel size indicates its resolution, such as 2cm or 3cm. Smaller voxel size can represent finer details. Depending on what to put in the voxel, there are several types of voxel grids. As the voxel grid is the most similar representation to 2D pixels, it attracts a lot of attention in the beginning. But it has the limitation on the resolutions, such as huge memory consumption for finer details.

Occupancy Grid is a binary representation. More specifically, the content in each voxel is either 0 or 1. Zero represents empty or free space, and one means occupied. Occupancy grid is not a continues representation, so that it can not extract surface inside one voxel.

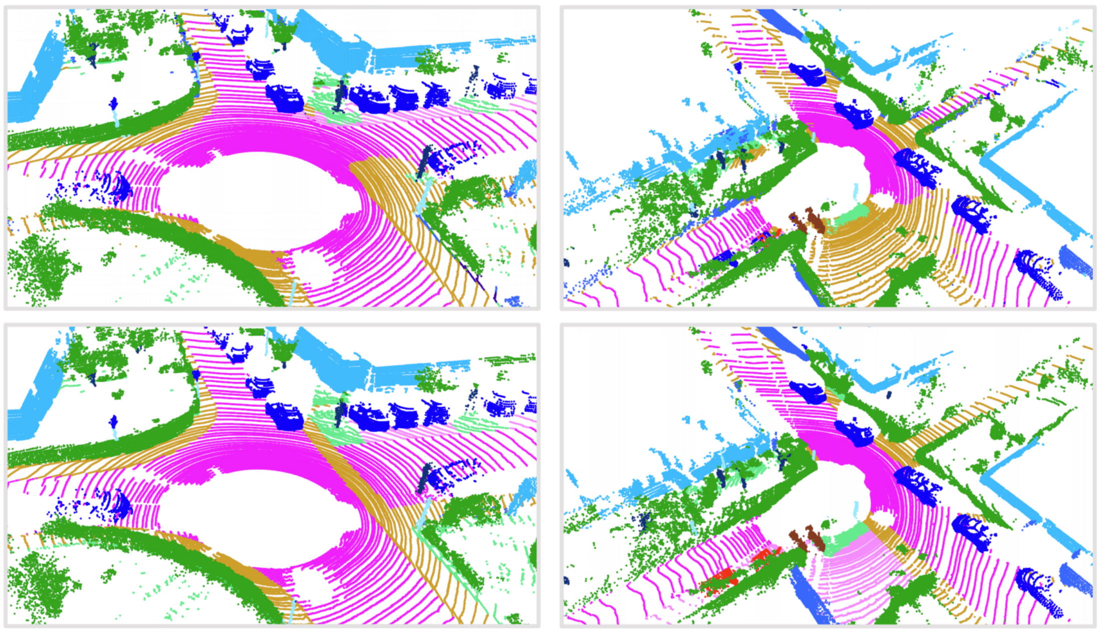


Figure 2.2: Point Cloud From LiDAR Sensors [16]. Number of circles (like the water ripple) indicates how many lasers are rotated by the LiDAR sensor. More circles represent denser point cloud.

SDF Grid puts the signed distance field value (SDF) in each voxel. Signed distance field measures the distance from current voxel to its closest surface. Positive sign denotes outside of the objects, whereas negative sign denotes inside the surface. Compared to occupancy grid, it can extract surface within a voxel. As the values in SDF voxels are continuous, surface can be extracted within a voxel by Marching Cubes, where it interpolates 0 values as surface, as presented in Figure 2.4. In practice, Truncated SDF (TSDF) is often used. TSDF truncates the sdf values by a certain threshold, as the free space is not interesting when it is too far from the surface. There are multiple ways to generate SDF voxel grid; the most popular one is volumetric fusion [3], in which multiple RGB-D frames are fused into the voxel grid according to their camera poses and camera parameters.

2.1.3 Mesh

Mesh is composed by vertices that can be seen as point clouds as well as faces. Faces define how to connect vertices. Normally triangles are used as faces, so that one face includes indices of three vertices. Compared to voxel grid, mesh only exists on surface, and do not model the free space.

We show a mesh of human face in Figure 2.5, in which each surface contains four vertices called quads as showed on the right. If hiding the connectivity, we can see

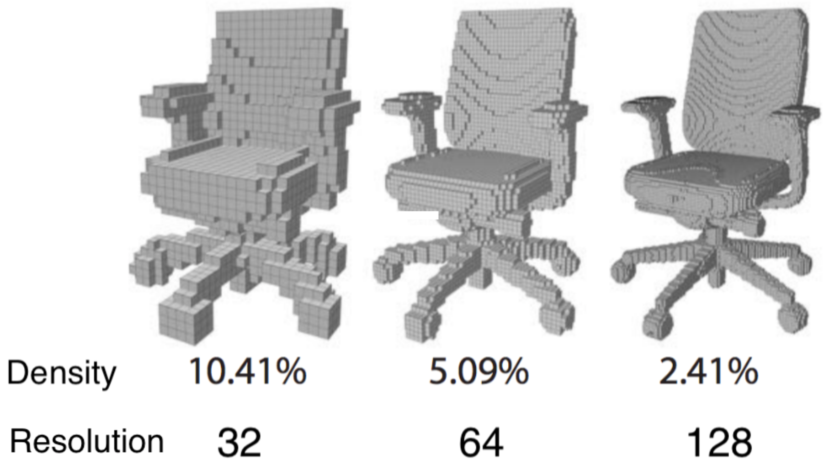


Figure 2.3: Voxel Grid Representation. Higher resolution can represent finer details, however, the number of voxels increases exponentially. Density denotes how many voxels are occupied divided by the total number of voxels. Lower density means more free space. The higher resolutions the voxel grid has, the more voxels are in free space.

a smoothed human face on the left in the figure. Generally, mesh produces the most visually pleasing graphics. There are several ways to generate mesh. For instance, we can produce mesh surface from SDF voxel grid by Marching Cubes as introduced in Section 2.1.2 or poisson mesh reconstruction from point cloud.

2.1.4 Implicit Neural Representation

Compared to traditional geometry representations, such as volumetric grids or point clouds, implicit neural representation aims to parameterize geometry with learned neural network features. More specifically, implicit neural representation models implicit functions, such as signed distance function (SDF). For example, DeepSDF [18] uses MLPs to represent implicit SDF functions, and it takes xyz coordinates as input, and outputs the SDF values of given xyz locations. Similarly, Occupancy Network [19] outputs a binary value to indicate if the current location is on the surface given xyz coordinates. To this end, implicit neural representation gets rid of the limitations of resolutions. By sampling more and more xyz inputs, it can generate infinite finer resolutions.

Implicit functions. Mathematically, an explicit function is the function, in which the dependent variable is given in terms of the independent variables, e.g. $y = x + 1$. In this function, the dependent variable y is determined by the independent variable x. Implicit functions, on the other hand, are usually given in a relation of the form $R(x_1, \ldots, x_n) = 0$, where R is a function of several variables, such as $x^2 + y^2 = 0$. To this regard, signed distance function (SDF) is a classic implicit function. SDF outputs

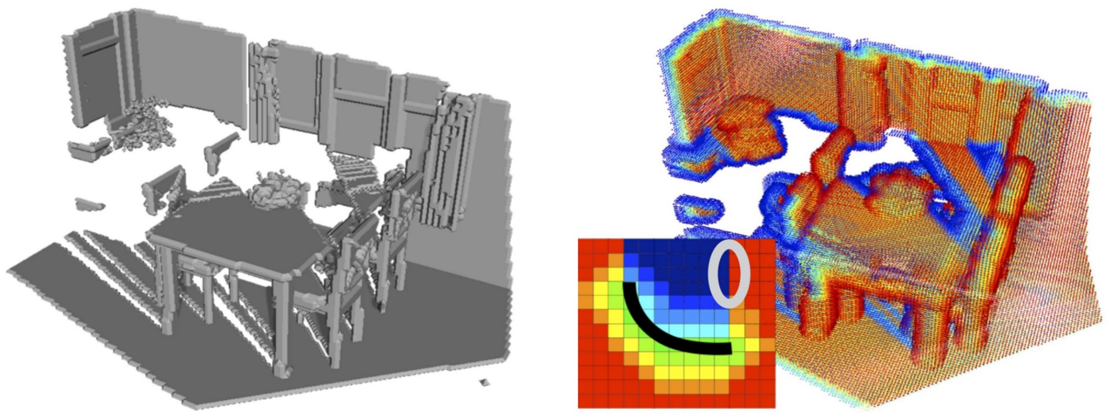


Figure 2.4: Isosurface at 0 in SDF. Surface is extracted at value 0, which is interpolated within voxels [17]

the distance value from current location to its closest surface, given xyz coordinates as the input variables to the function.

2.2 3D Neural Network

In previous section, we introduced commonly used 3D representations. To this end, we need to leverage neural network to learn objectives by consuming the data. In this section, we introduction several popular types of 3D neural networks and what are their preferred data representations.

2.2.1 3D Convolutional Neural Network

3D Convolutional Neural Network (3D CNNs) is very similar to its counter part 2D CNNs. The only difference is that it has one more dimension in kernel size. As illustrated in Figure 2.6, it slides the kernel in the 3D tensor inputs.

As voxels are very similar to pixels, 3D CNNs are usually used to learn from voxel grid data. For 3D Neural Network composed of 3D convolutions, we also need activation functions, pooling layers etc. They are also very similar to 2D, but with one more dimension.

2.2.2 SparseConv

3D CNNs has the drawback that it slides the kernels also on free space. However, free space is not interesting for many tasks, such as semantic segmentation. Because we only care the labels on the surface. As more than 90% voxels are in free space. 3D CNNs consume huge memory and are trained very slowly for dealing with those voxels in free space.

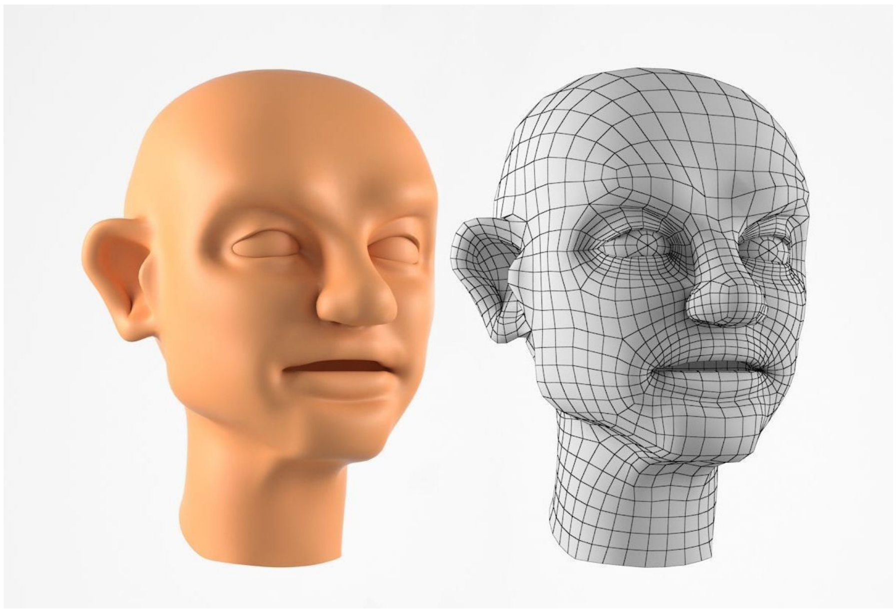


Figure 2.5: Human Face Mesh. The connectivity is presented on the right, defined by quadrilaterals (quads).

To conquer this drawback, SparseConv is invented [20]. The core idea is the conv kernel only convolves with the values on the surface, and the voxels in free space are ignored. We show a illustration in Figure 2.7. Since sparseconv only convolves on surface values, the global features summarised by disconnected surfaces can be aggregated by building a deeper network. As SparseConv can save huge memory and training speed by ignoring free space voxels, a very deep 3D neural network is easy to build. In summary, SparseConv is a very powerful backbone for 3D data as it saves memoery and training speed from free space thus can build very deep network for extracting meaningful features.

2.2.3 PointNet

As the name suggests, PointNet [21] are designed for raw point cloud inputs, which enables several downstream tasks presented in Figure 2.8. To process point cloud, PointNet has several properties.

- Permutation (Order) Invariance: given the unstructured nature of point cloud data, a scan made up of N points has N! permutations. The subsequent data processing must be invariant to the different points orders.
- Transformation Invariance: it must be invariant to certain geometric transformations, such as rotation. For instance, an apple rotated by 90 degree should still be classified as an apple.

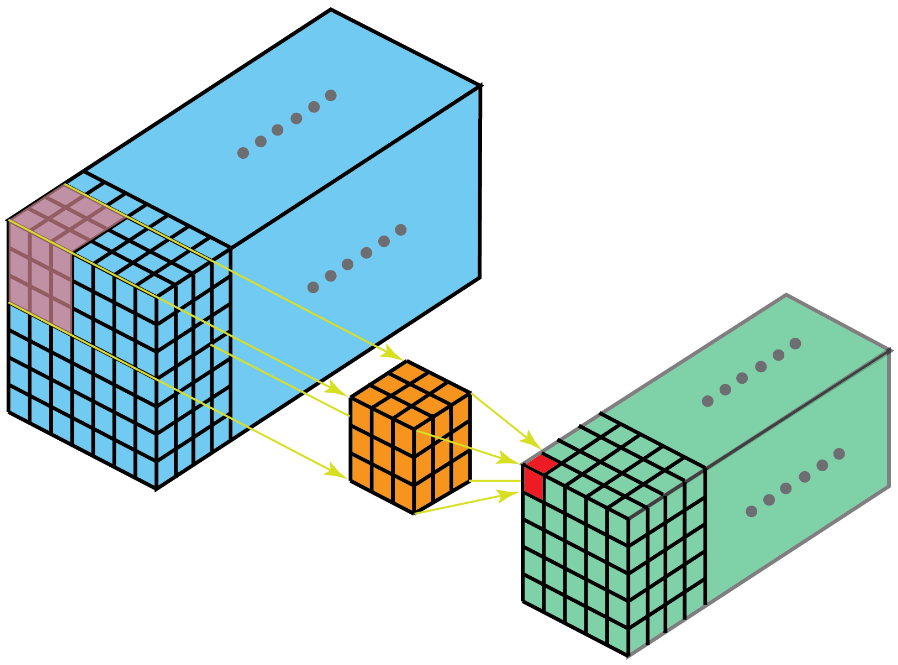


Figure 2.6: 3D Convolution Operation. Kernel Tensor is convoluted with Volume Tensor in a sliding window fashion.

- Point Interactions: neighboring points often carries useful information, and a single point should not be treated in isolation.

PointNet architecture is surprisingly simple and quite intuitive. It uses a shared multi-layer perceptron (MLP) with several designed modules to extract meaning features and satisfy the properties mentioned before at the same time. PointNet implements the symmetric function with max pooling to extract feature for a local region. To this end, it is invariant to the orders of points in this region (Permutation Invariance).

Motivated by Spatial Transformer Networks (STNs) [22], PointNet introduces Spatial Transformer (ST) to satisfy Transformation Invariance. For a given input point cloud, ST applies an appropriate rigid or affine transformation estimated from T-Net to achieve pose normalization. T-Net is a regression network predicting an input-dependent transformation matrix. Pose normalization, leveraging the estimated transformation matrix from T-Net, aims to normalize the input point cloud to a canonical pose.

2.3 RGB-D Datasets and Related Work

In previous sections, we introduced various representations of 3D data and powerful neural network backbones. All of those make 3D scene understanding tasks possible. In this section, we walk through the mainstream tasks and data used in 3D Scene

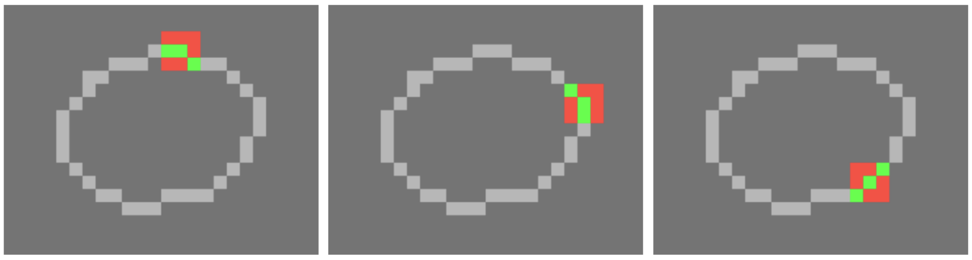


Figure 2.7: SparsConv Convolves Only With Surface Values. Greens represent activated voxels that are convolved with kernels; red voxel is not activated, thus has no influence on kernel weights [20].

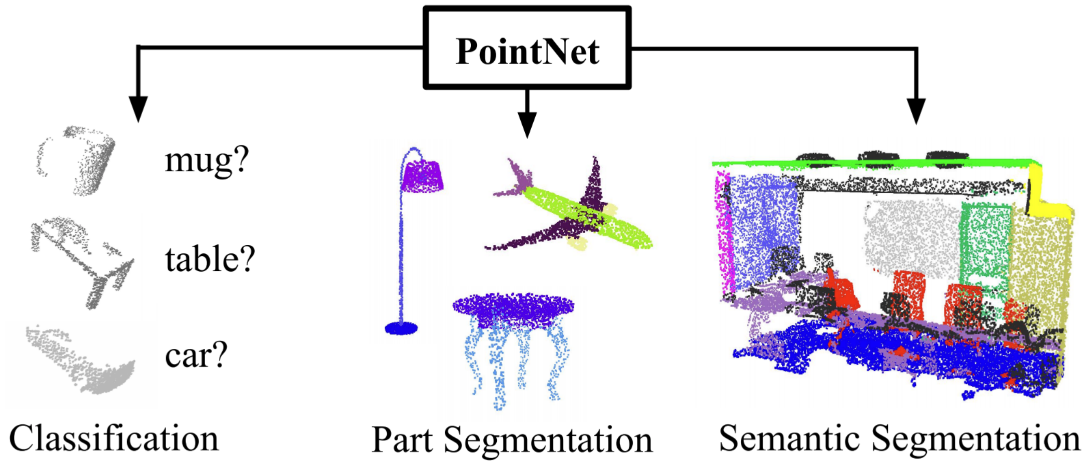


Figure 2.8: PointNet Backbone. PointNet enables several downstream tasks as it can extract features from unordered point cloud data [21].

Understanding. In Section 2.3.1, we introduce the most popular RGB-D datasets that facilities the research. In Section 2.3.2, we introduce the mainstreams tasks included in 3D scene understanding areas and their state-of-the-art methods. In Section 2.3.3, we introduce the state-of-the-art methods for training 3D Scene Understanding tasks with extremely few annotated data.

2.3.1 RGB-D Datasets

RGB-D data refers to pairs of frames including depth frame and color frame. Each pair of frames are associated with a set of camera parameters such as intrinsic and extrinsic. With the help of those information, a 3D reconstruction can be obtained in either mesh or point cloud. In the following, we introduce several very popular RGB-D datasets that boost the research of 3D scene understanding recently.

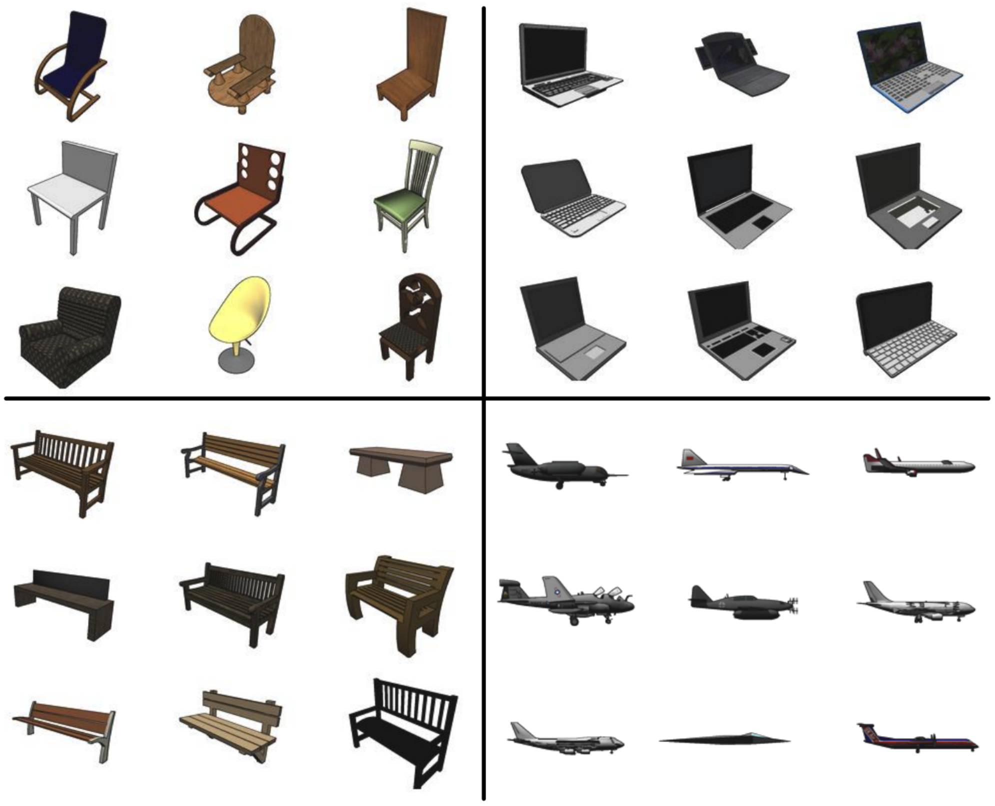


Figure 2.9: Aligned Models in ShapeNet [23]. Examples of categories of chair, laptop, bench, and airplane.

ShapeNet [24] is a richly-annotated, large-scale repository of shapes represented by 3D CAD models of objects. ShapeNet contains 3D models from a multitude of semantic categories, such as chair, laptop and bench. ShapeNet has indexed more than 3,000,000 models out of 3,135 categories. Each object is also attached with textures. It is a collection of many semantic annotations for each 3D model, such as consistent rigid alignments, parts and bilateral symmetry planes, as well as physical sizes. ShapeNet promotes data-driven geometric analysis, and provides a large-scale quantitative benchmark for research in computer graphics and vision. Furthermore, it also provides renderings from different view angles centered in each object, which enables a lot of research of reconstructing objects from single or multiple RGB images. We show examples of some categories in Figure 2.9.

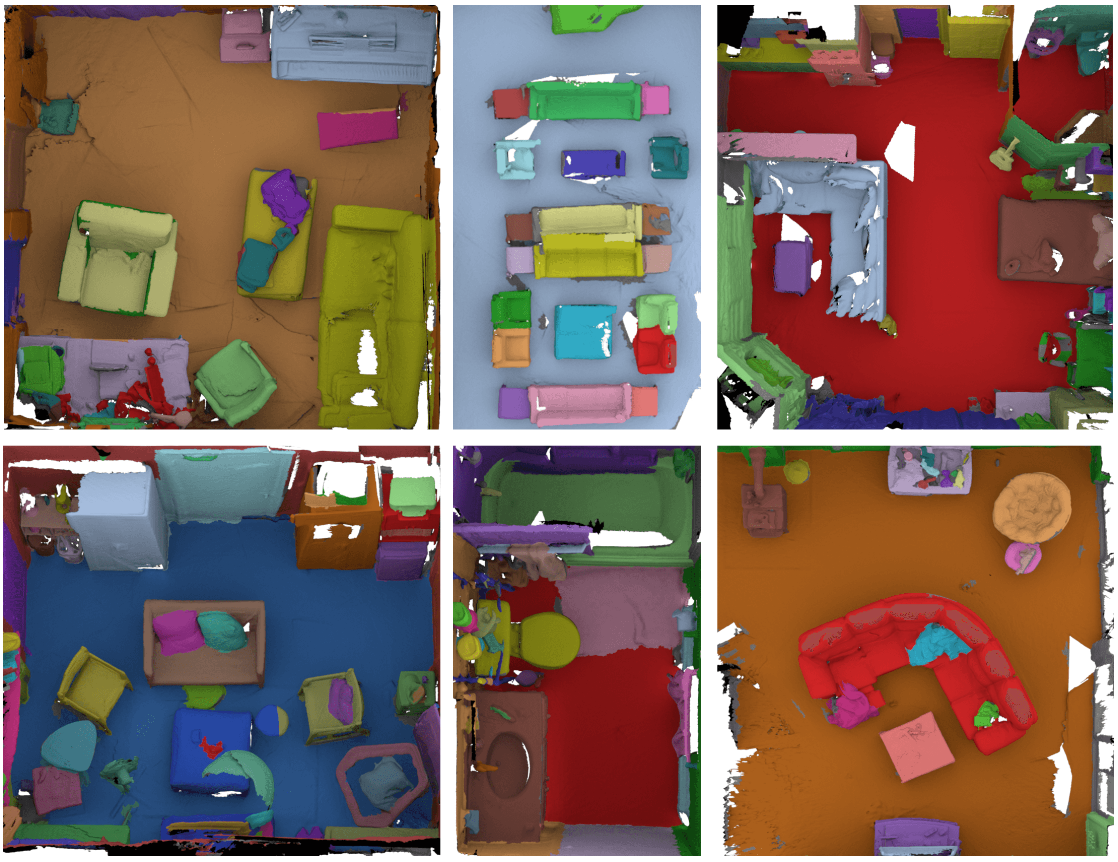


Figure 2.10: Reconstruction and Annotations in ScanNet [4]. ScanNet provides instance-level annotations in 3D. Different colors represent different instances.

ScanNet [4] is a richly annotated large-scale RGB-D dataset of indoor scenes, including 1213 indoor scenes for training, 321 scenes for validation and 100 scenes as test data. It mostly records the hotel rooms, offices in the universities as well as rooms in houses. For each scene, it provides the raw RGB frame and depth frames. For each depth frame and color frame, it provides the camera intrinsic matrix. ScanNet also provides camera pose for each depth frame that is globally optimized by BundleFusion [3]. Additionally, ScanNet includes the 3D reconstruction from volumetric fusion. For each scene, it provides precise instance-level annotation as showed in Figure 2.10. Furthermore, the authors also back project the 3D annotations back to 2D color frames based on the camera poses and intrinsic matrix, which enables the research of geometry-color cross-modality. ScanNet has achieved great success and push the 3D scene understanding research forward as it sets up the major benchmark in 3D scene understanding tasks, including semantic segmentation, as it provides per-point semantic label annotations on 3D mesh, instance segmentation, as it has the instance-level annotations, as well as object detection, as it provides tight axis-aligned bounding boxes estimated from instance annotation. A variety of methods have been developed and compared on this benchmark.

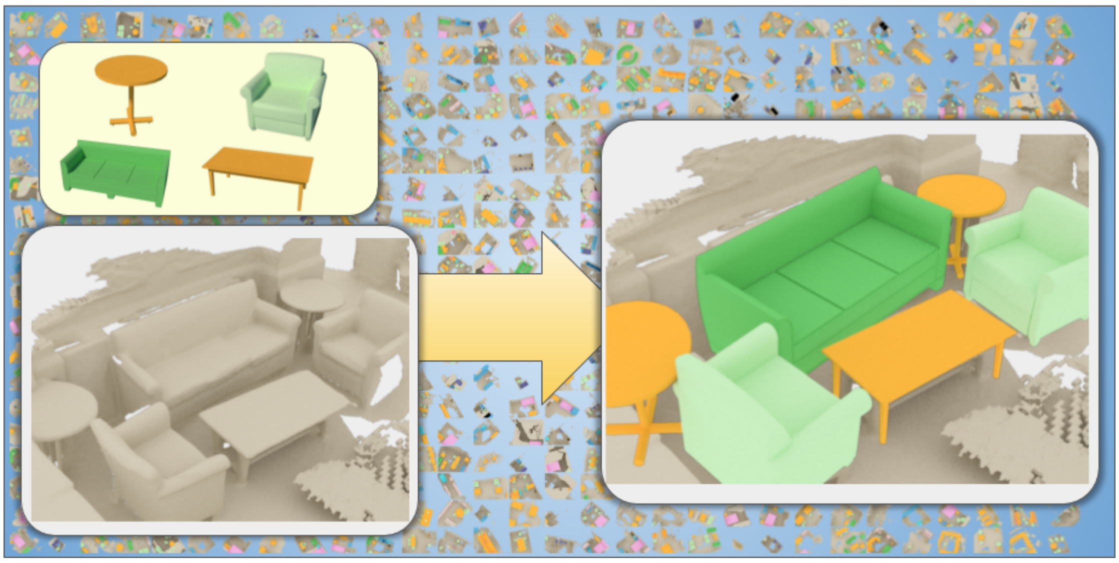


Figure 2.11: Scan2CAD [25] Alignments. Scan2CAD aligns instance-level objects with complete geometry (CAD model) to their incomplete counter parts in 3D scans from ScanNet.

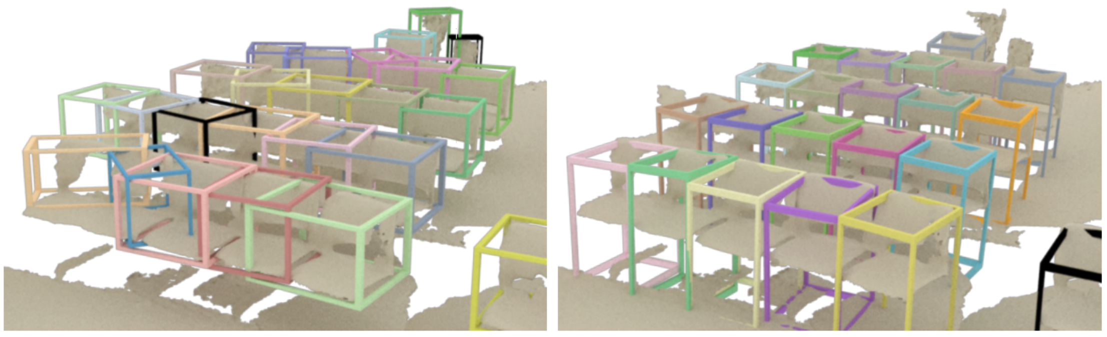


Figure 2.12: Amodal Bounding Box Annotations in Scan2CAD [25]. Compared to ScanNet annotations, Scan2CAD provides amodel (full scale) 3D bounding boxes computed from complete geometry (CAD model). Left: bounding box annotations from ScanNet; right: bounding box annotations from Scan2CAD.

Scan2CAD [25] aligns clean 3D CAD models from a 3D object database to the noisy and incomplete geometry of RGB-D scans in ScanNet. Scan2CAD provides the alignment based on 1506 ScanNet scans and 14225 CAD models from ShapeNet. As the alignments between the CAD model and scans are based on the key-point matching, thus Scan2CAD annotated 97607 key-point pairs between the CAD models and their counter parts in scans. A alignment example between CAD models and a specific scan is presented in Figure 2.11. Scan2CAD enables a variety of research in 3D scene understanding, such as object detection as it provides amodal 3D bounding box annotations computed from aligned CAD model with complete geometry, whereas ScanNet computes tight bounding boxes from incomplete object geometry illustrated in Figure 2.12. As Scan2CAD gives complete geometry of objects in incomplete scans, it enables the a new research direction, semantic instance completion introduced in 4.

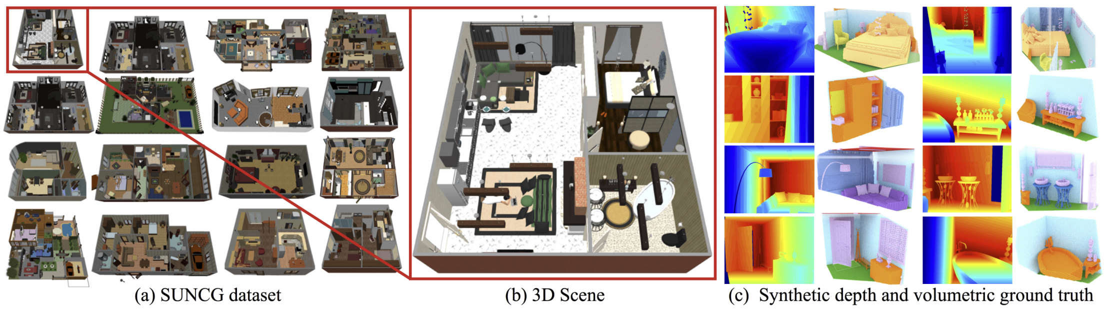


Figure 2.13: SUNCG Dataset. SUNCG [17] creates synthetic 3D scenes with complete geometry. It also contains rendered depth images and volumetric semantic ground truth (instance level).

SUNCG [17] is a manually created large-scale dataset of synthetic 3D scenes with realistic room and furniture layouts. Each scene has dense volumetric annotations as well as instance information. SUNCG provides 49,884 valid floors that contain 404,058 rooms and 5,697,217 object instances from 2644 unique object meshes covering 84 categories. The authors manually labeled all the objects in the library to assign category labels. In Figure 2.13, we show a set of top view renderings of each floor in SUNCG. As a synthetic RGB-D dataset, SUNCG contains even a lot more scenes than the largest real-world RBG-D dataset (ScanNet). It provides huge 3D data that can easily be manipulated and enables the research of semantic scene completion that will be introduced in Section 2.3.2.

SUN-RGBD is captured by four different sensors and contains 10,335 RGB-D images. The whole dataset is densely annotated and includes 146,617 2D polygons and 64,595 amodal 3D bounding boxes[1] with accurate object orientations, as well as a 3D room

[1]Amodal perception is the perception of the whole of a physical structure when only parts of it affect the sensory receptors

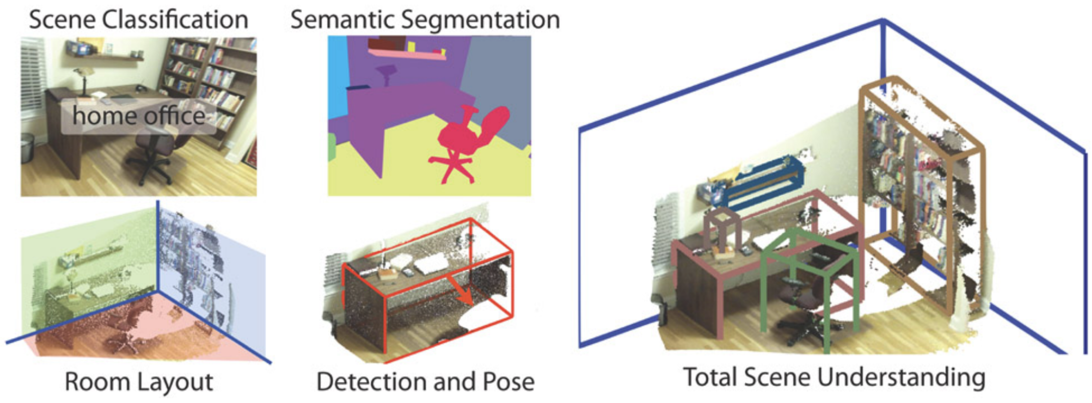


Figure 2.14: SUN-RGBD Dataset. SUN-RGBD [26] contains various annotations, such as room layout, scene classification and semantic segmentation etc, but it is mostly known for 3D object detection (amodal) from single RBG-D image.

layout and scene category for each image. This dataset is a very popular benchmark on object detection from single RGB-D image. We show some examples of the data in Figure 2.15.

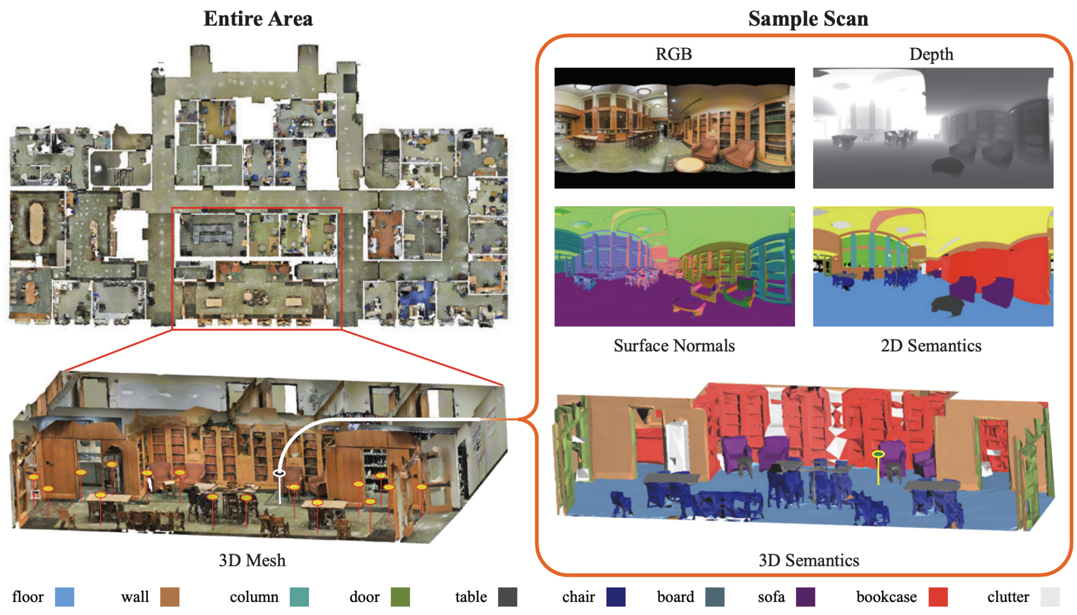


Figure 2.15: S3DIS Dataset. S3DIS Dataset [27] is very similar to ScanNet, including 2D and 3D semantics. S3DIS is recorded to contain the whole floor (entire area), and the rooms can be cropped from the floor.

S3DIS covers indoor scenes over 6,000 m^2 and contains over 70,000 RGB-D images, along with surface normals, semantic annotations, as well as camera information. S3DIS also includes reconstructed raw and semantically annotated 3D meshes and point clouds. The dataset provides not only per-point label but also instance-level annotations, which enables development of 3D semantic segmentation and instance segmentation.

2.3.2 3D Scene Understanding Tasks

RGB-D datasets have equipped and driven forward the research of 3D scene understanding. In general, 3D scene understanding aims to understand the 3D environments of an indoor scene, such as identifying the objects' locations and their semantic labels as well as segmentation of the scene. In this section, we introduce the popular tasks counted as 3D scene understanding.

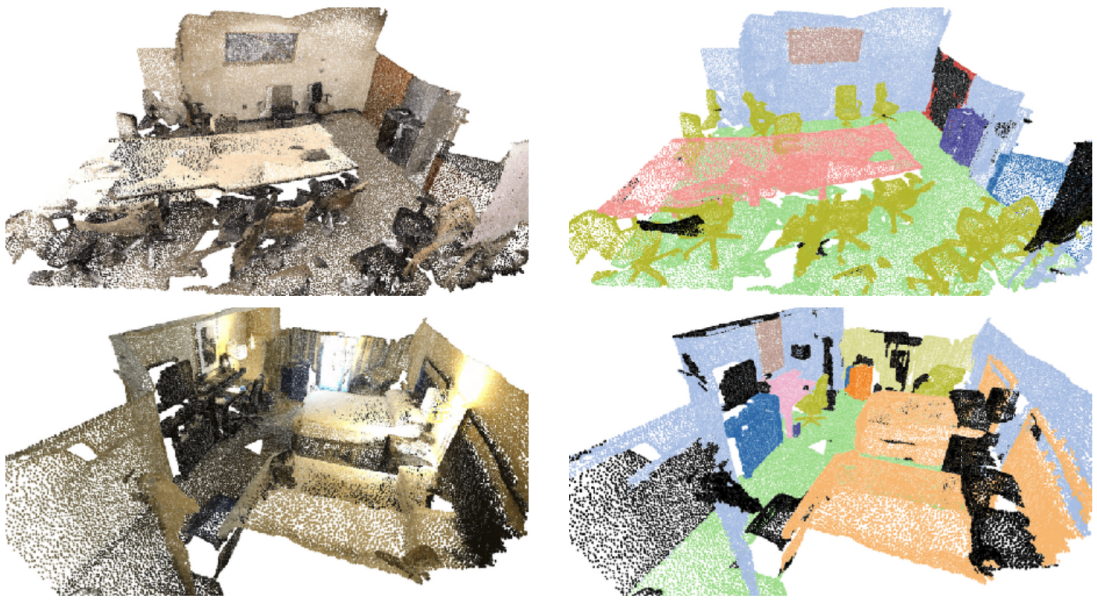


Figure 2.16: Semantic Segmentation in ScanNet [28] Left: input point cloud; right: semantic segmentation prediction. Different colors represent different semantic classes.

2.3.2.1 3D Semantic Segmentation

Semantic segmentation aims to assign a semantic label to each point for point cloud or voxel for voxelgrid input as illustrated in Figure 2.16. In recent years, 3D semantic segmentation has achieved great success starting from PointNet [21] to SparseConv [20], [28]. The main metric in this task is Intersection-Over-Union (IoU). IoU is easy to compute as showed in Equation 2.1. This task has one-to-one mapping from input to output and don't require additional techniques such as clustering algorithms, thus can be used as simplest downstream task to verify the capability of a backbone. Besides pure geometric input, Dai et al. [29] introduces a cross-modality learning method (3DMV) for this task. 3DMV uses TSDF (see Section 2.1.2 as input for 3D part, and RGB color images that look at the same region as input for 2D part. As illustrated in Figure 2.17, 3D CNNs are used to summarize 3D geometric features, and 2D CNNs are used to learn 2D color features. They are fused together by back-projection layer to predict final results. Back-projection layer projects per-pixel features to voxels based on camera parameters, such as intrinsic and extrinsic matrix. The introduction of 2D CNNs rather than direct usage of raw RGB values aims to reduce the mismatch degree of the pixel and voxel resolution, as one voxel can contain multiple pixels.

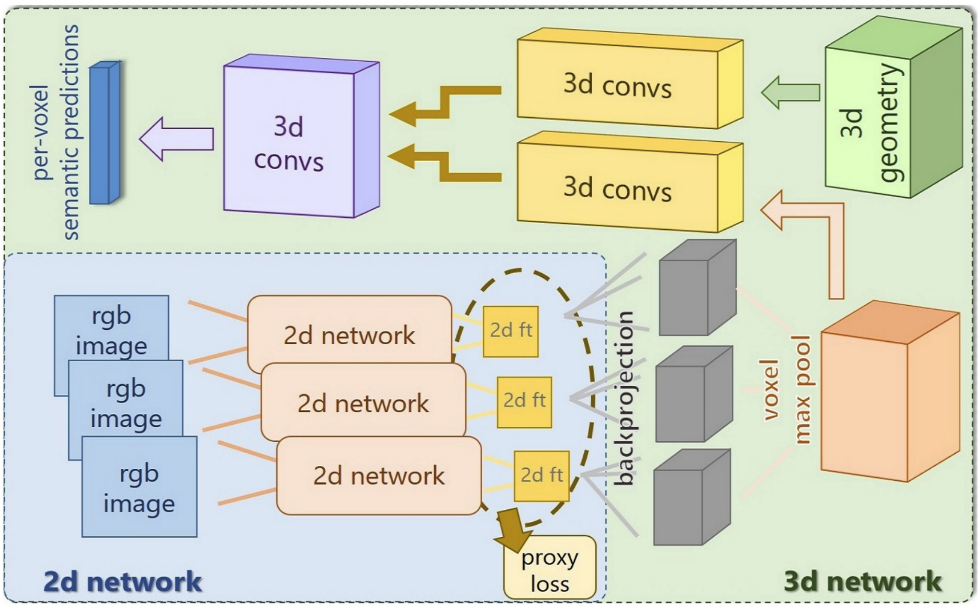


Figure 2.17: Illustration of 3DMV [29]. Color features are fused with geometric features through back-projection layer to predict per-voxel semantic labels.

$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}} \tag{2.1}$$

2.3.2.2 3D Semantic Instance Segmentation

In this section, we introduce the state-of-the-art methods in 3D semantic instance segmentation. The input of instance segmentation is the same as semantic segmentation, e.g. a point cloud or voxelgrid. The output, however, is not a one-to-one mapping. As the goal is to find individual object shape in the scan, the output is of an arbitrary number depending on how many objects in the scene as illustrated. The most popular metric used in this task is mean Average Precision at N (mAP@N). N means the IoU overlap as defined in Equation 2.1. This metric ranks the the predictions with their scores or confidences. Starting from the prediction with top scores, it computes true positives (TP), false positives (FP) and false negatives (FN) in an accumulative way. True positive refers to the prediction that has at least N% overlap with any of the ground-truth object, and False positive means the predictions that satisfy overlap but predict wrong semantic labels. False Negative refers to the missing detection from ground truth. Accordingly, precision and recall can be computed by Equation 2.2 and 2.3. To this end, we can draw a precision and recall curve. Average Precision is the integral over the curve, and mAP is the mean value of APs over semantic classes.

There are two streams of methods developed in instance segmentation, namely top-down and bottom-up. Top-down first detect objects and then segment their shape. Bottom-up method first do semantic segmentation and then cluster the points into instances. In this section, we introduce two bottom-up methods PointGroup [30] and SGPN [31]. The top-down method is introduced in Chapter 3.

$$Precision = \frac{TP}{TP + FP} \tag{2.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.3}$$

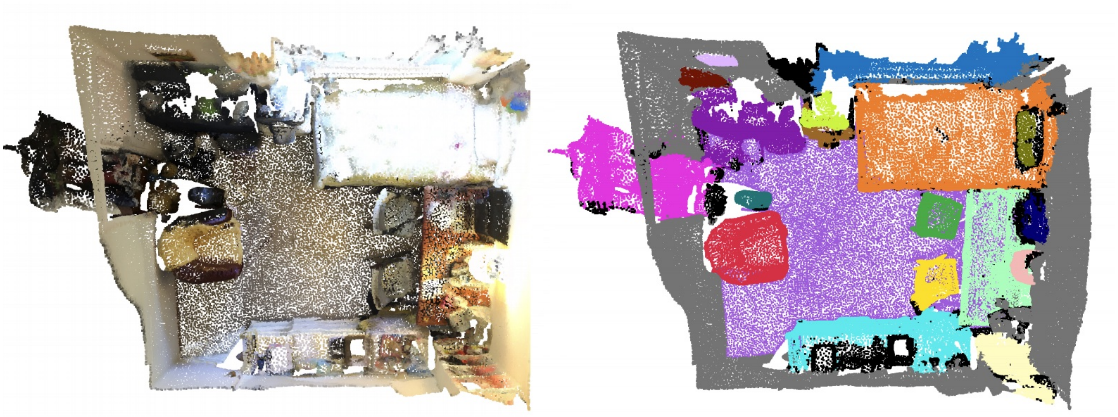


Figure 2.18: Instance Segmentation in ScanNetV2. Left: point cloud input; right: instance segmentation prediction from PointGroup [30]. Different colors represent different instances.

SGPN is the first method towards 3D instance segmentation in indoor scene. It process the input point cloud with a PointNet backbone. SGPN obtains per-point features, and input these features into three branches, namely a similarity matrix, a confidence map and a semantic prediction branch. Similarity groups the points into instance proposals. Confidence map predictis the score for each proposal. Semantic prediction branch predicts the semantic class for each instance. As it computes distance between each pair of points, SGPN consumes huge memory. To solve this issue, it process scene block by block. Finally, blocks are merged into the prediction of the whole scene. This pipeline is described in Figure 2.19.

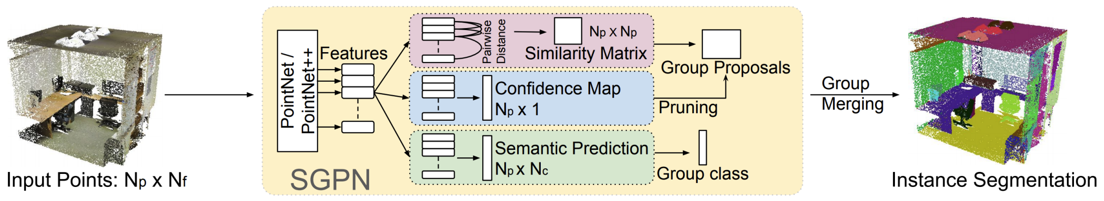


Figure 2.19: SGPN Pipeline [31]. SGPN consumes point cloud with a PointNet backbone, and propose instances by a similarity matrix that indicates the similarity between each pair of points in embedded feature space.

PointGroup first predict semantic segmentation with a SparseConv backbone. Additionally, it enforces a voting-center loss, i.e. points belonging to the same object are moved to the object center. Different from SGPN where the clustering happens in feature space, PointGroup clusters in spatial space. The points that are closed to each other in euclidean space are grouped together into one instance. PointGroup proposes to cluster in two euclidean spaces called dual clustering, one with center voting loss that points are moved to the object centers, one without this loss. In this way, each point can be assigned to more than one instance. Thus, they use Non Maximum Suppression (NMS) to remove duplicates based on the scores estimated by a small network called ScoreNet. The whole pipeline is showed in Figure 2.21.

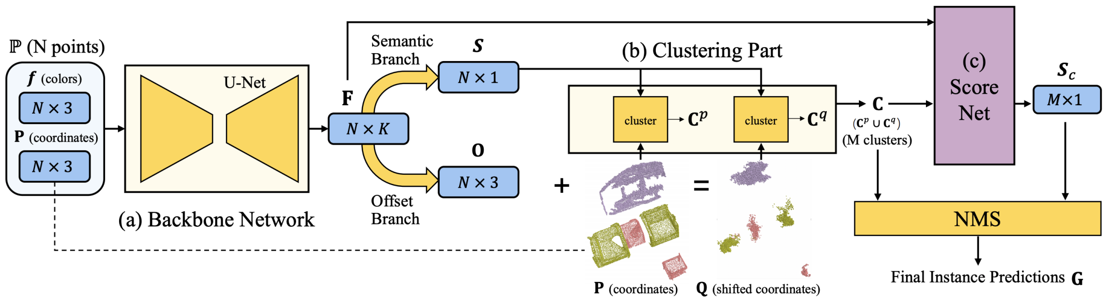


Figure 2.20: PointGroup Network [30]. PointGroup takes point cloud with N points as input to a U-Net backbone, and generate instances by Clustering, ScoreNet and NMS.

2.3.2.3 Object Detection

Object detection plays a very important role in 3D scene understanding. Same as previously introduced tasks, object detection task takes a point cloud as input, and output 3D bounding boxes for each object in the scene as showed in Figure 5.9. Knowing where the objects are is a very important step to a variety of applications, such as robotics route planning and VR/AR. Similar to instance segmentation, detection also has two main streams: top-down and bottom-up methods. In this section, we introduce a bottom-up method called VoteNet [32]. We introduce our proposed algorithm that is a top-down method in Chapter 3.

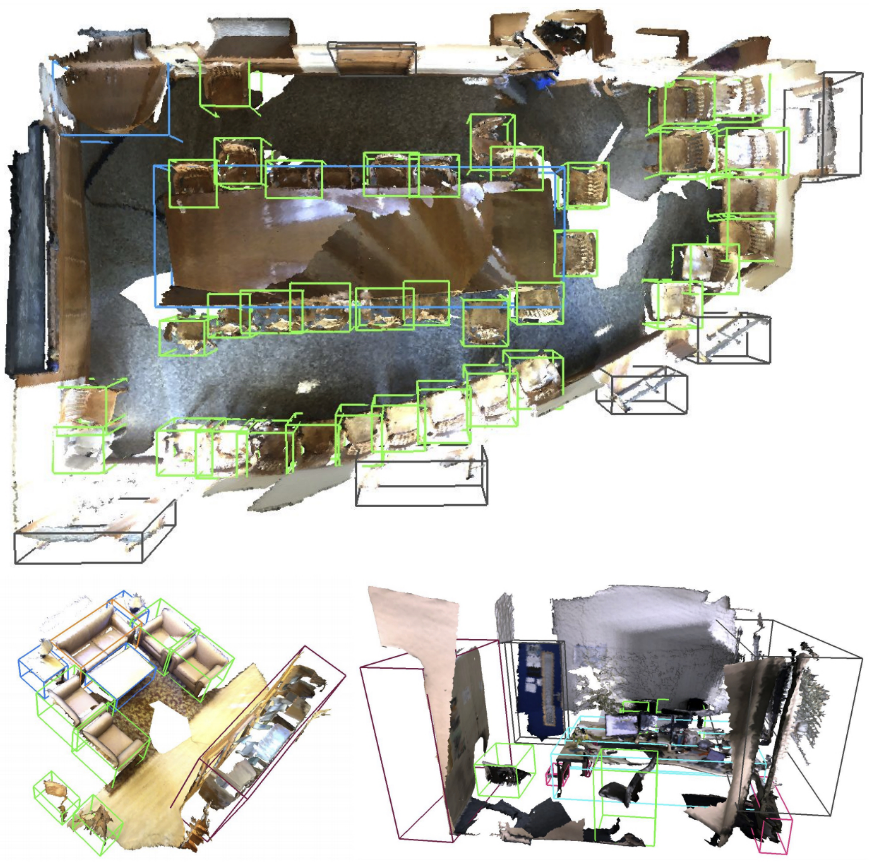


Figure 2.21: Detection Task in ScanNetV2 [32]. For each object in the scene, a 3D bounding box that includes the object' geometry is depicted.

VoteNet leverages a PointNet/PointNet++ [21], [33] backbone to extract features for each point. Furthermore, a subset of points are sampled that considered as seed points. Each seed independently generates a vote through a voting module. Then the votes are aggregated into clusters and processed by the proposal module to generate the bounding box proposals including the location, dimensions, orientation and semantic classes. A depicted pipeline can bee seen in Figure 2.23.

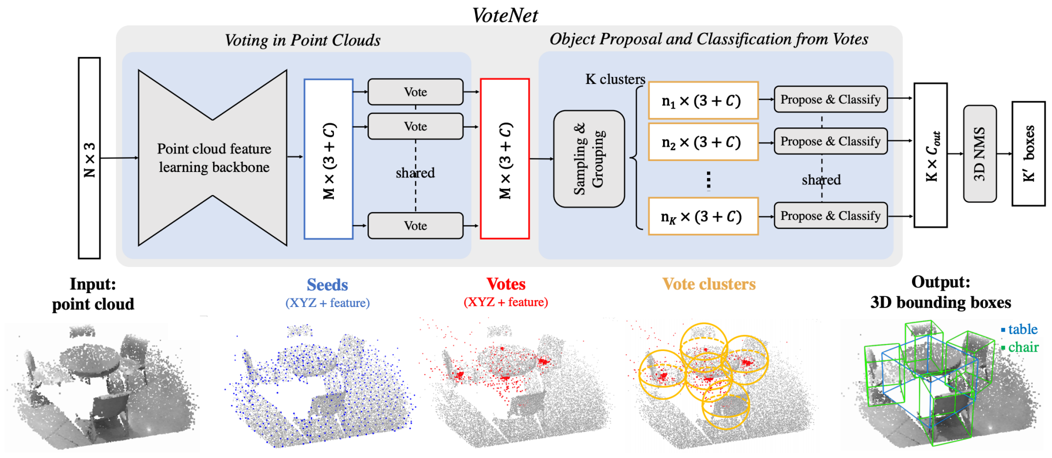


Figure 2.22: VoteNet Pipeline. Network Architecture of VoteNet [32]: from N points input, feature extraction with PointNet++ backbone, to voting module and proposal module.

2.3.2.4 Semantic Scene Completion

Tasks like semantic/instance segmentation assigns semantic labels to the input, but does not change the input geometry. However, it is quite normal that 3D reconstruction has holes and missing geometry due to the missing view angles or lighting conditions. Semantic Scene Completion (SSC) aims to complete the missing geometry and their semantic labels. In this section, we introduce two works: SSCNet [17] and RfD-Net [34]. SSCNet aims to complete whole scene, and RfD-Net completes missing geometry of each object. In our work introduced in Chapter 4, we show improvement on detection and instance segmentation tasks by completing objects' geometry.

SSCNet takes a partial depth frame as input and complete the missing geometry as well as per-voxel semantic label. It takes TSDF as input and leverages 3D CNNs for feature extraction. SSCNet renders depth map from SUNCG so that they have the ground truth of missing geometry and their semantic labels. It is the first pioneer work to target this problem.

RfD-Net aims to complete missing geometry on individual objects. Because walls and floors are normally easy to complete and less interesting for many applications. It first detect 3D objects in the scan and then leverages Shape Completion method to complete each object individually. RfD-Net uses the state-of-the-art implicit representation to reconstruct objects. To this end, they can get rid of resolution limitation and generate high-quality geometry.

2.3.3 Efficient Learning in RGB-D Data

To make better 3D scene understanding, researchers have developed sophisticated methods based on deep learning. As data-driven methods, data influences hugely on deep learning methods. On the other hand, data annotation in 3D is very expensive. To this end, efficiently using data emerges in 3D scene understanding community. The general goal is to use as few as annotated data to achieve competitive performance. In this section, we introduce the pioneer work PointContrast, the first self-supervised pre-training method on RGB-D data. Note that PointContrast is proposed to improve downstream tasks' performance, but not designed for data-efficient learning. Our work [13] is the first to propose a 3D scene understanding benchmark including detection and semantic/instance segmentation on data-efficient learning. We leverage a self-supervised pre-training to reduce the needed annotations. In Chapter 5, we explain our methods in details.

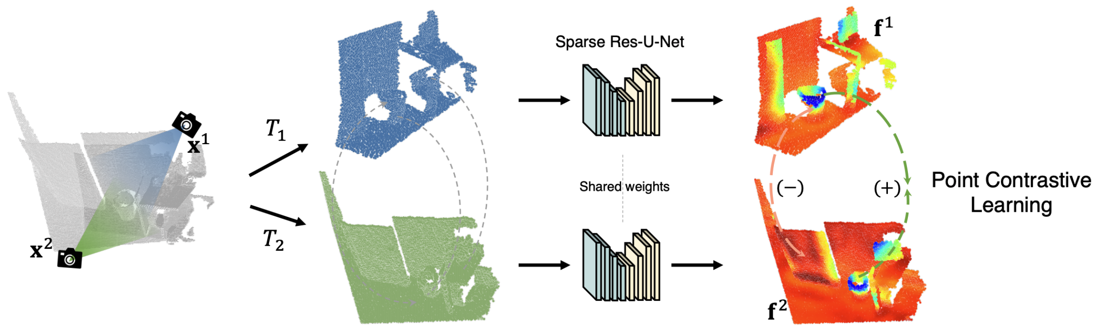


Figure 2.23: PointContrast Pipeline [35]. PointContrast takes pairs of partial depth frames, and learning correspondences matching as a pre-training task.

PointContrast takes a pair of depth frames that have at least 30% overlaps and enforce a contrastive learning loss on correspondence matching task as pre-text training. The pre-trained weights then are used as network initialization to be finetuned on downstream tasks.

Part II

3D Scene Understanding and Data-Efficient Learning

3 3D Semantic Instance Segmentation

This chapter introduces the following paper:

J. Hou, A. Dai, and M. Nießner, "3D-SIS: 3D Semantic Instance Segmentation of RGB-D Scans," in *Proceedings of Computer Vision and Pattern Recognition (CVPR), IEEE*, 2019

Abstract of paper As the most important task in 3D perception, 3D Instance Semantic Segmentation includes detecting 3D objects, classify its semantic label and segment its 3D shape out. To tackle this problem, we introduce 3D-SIS, a novel neural network architecture for 3D semantic instance segmentation in commodity RGB-D scans. The core idea of our method is to jointly learn from both geometric and color signal, thus enabling accurate instance predictions. Rather than operate solely on 2D frames, we observe that most computer vision applications have multi-view RGB-D input available, which we leverage to construct an approach for 3D instance segmentation that effectively fuses together these multi-modal inputs. Our network leverages high-resolution RGB input by associating 2D images with the volumetric grid based on the pose alignment of the 3D reconstruction. For each image, we first extract 2D features for each pixel with a series of 2D convolutions; we then backproject the resulting feature vector to the associated voxel in the 3D grid. This combination of 2D and 3D feature learning allows significantly higher accuracy object detection and instance segmentation than state-of-the-art alternatives. We show results on both synthetic and real-world public benchmarks, achieving an improvement in mAP of over 13 on real-world data.

Contribution The method development and implementation was done by the first author. Data generation and baselines were done with the help of Angela Dai. Discussions with the co-authors led to the final paper.

3.1 Introduction

Semantic scene understanding is critical to many real-world computer vision applications. It is fundamental towards enabling interactivity, which is core to robotics in both indoor and outdoor settings, such as autonomous cars, drones, and assistive robotics, as well as upcoming scenarios using mobile and AR/VR devices. In all these applications, we would not only want semantic inference of single images, but importantly, also require understanding of spatial relationships and layouts of objects in 3D environments.

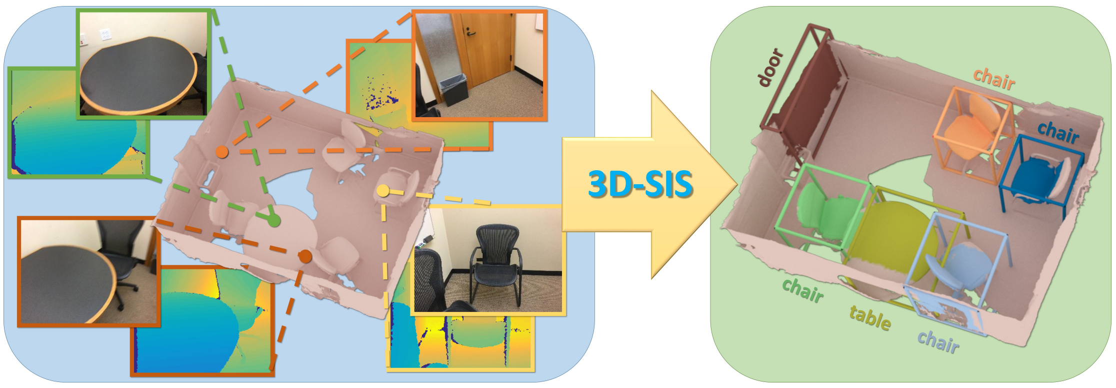


Figure 3.1: 3D Semantic Instance Segmentation. 3D-SIS performs 3D instance segmentation on RGB-D scan data, learning to jointly fuse both 2D RGB input features with 3D scan geometry features. In combination with a fully-convolutional approach enabling inference on full 3D scans at test time, we achieve accurate inference for object bounding boxes, class labels, and instance masks.

With recent breakthroughs in deep learning and the increasing prominence of convolutional neural networks, the computer vision community has made tremendous progress on analyzing images in the recent years. Specifically, we are seeing rapid progress in the tasks of semantic segmentation [36]–[38], object detection [39], [40], and semantic instance segmentation [41]. The primary focus of these impressive works lies in the analysis of visual input from a single image; however, in many real-world computer vision scenarios, we rarely find ourselves in such a single-image setting. Instead, we typically record video streams of RGB input sequences, or as in many robotics and AR/VR applications, we have 3D sensors such as LIDAR or RGB-D cameras.

In particular, in the context of semantic instance segmentation, it is quite disadvantageous to run methods independently on single images given that instance associations must be found across a sequence of RGB input frames. Instead, we aim to infer spatial relationships of objects as part of a semantic 3D map, learning prediction of spatially-consistent semantic labels and the underlying 3D layouts *jointly* from all input views and sensor data. This goal can also be seen as similar to traditional sensor fusion but for deep learning from multiple inputs.

We believe that robustly-aligned and tracked RGB frames, and even depth data, from SLAM and visual odometry provide a unique opportunity in this regard. Here, we can leverage the given mapping between input frames, and thus learn features jointly from all input modalities. In this work, we specifically focus on predicting 3D semantic instances in RGB-D scans, where we capture a series of RGB-D input frames (e.g., from a Kinect Sensor), compute 6DoF rigid poses, and reconstruct 3D models. The core of our method learns semantic features in the 3D domain from both color features, projected into 3D, and geometry features from the signed distance field of the 3D scan. This is realized by a series of 3D convolutions and ResNet blocks. From these semantic features, we obtain anchor bounding box proposals. We process these proposals with a new 3D region proposal network (3D-RPN) and 3D region of interest pooling layer (3D-RoI) to infer object bounding box locations, class labels, and per-voxel instance masks. In order to jointly learn from RGB frames, we leverage their pose alignments with respect to the volumetric grid. We first run a series of 2D convolutions, and then backproject the resulting features into the 3D grid. In 3D, we then join the 2D and 3D features in end-to-end training constrained by bounding box regression, object classification, and semantic instance mask losses.

Our architecture is fully-convolutional, enabling us to efficiently infer predictions on large 3D environments in a single shot. In comparison to state-of-the-art approaches that operate on individual RGB images, such as Mask R-CNN [41], our approach achieves significantly higher accuracy due to the joint feature learning.

To sum up, our contributions are the following:

- We present the first approach leveraging joint 2D-3D end-to-end feature learning on both geometry and RGB input for 3D object bounding box detection and semantic instance segmentation on 3D scans.
- We leverage a fully-convolutional 3D architecture for instance segmentation trained on scene parts, but with single-shot inference on large 3D environments.
- We outperform state-of-the-art by a significant margin, increasing the mAP by 13.5 on real-world data.

3.2 Related Work

3.2.1 Object Detection and Instance Segmentation

With the success of convolutional neural network architectures, we have now seen impressive progress on object detection and semantic instance segmentation in 2D images [39]–[45]. Notably, Ren et al. [42] introduced an anchor mechanism to predict objectness in a region and regress associated 2D bounding boxes while jointly classifying the object type. Mask R-CNN [41] expanded this work to semantic instance segmentation by predicting a per-pixel object instance masks. An alternative direction for detection is the popular Yolo work [40], which also defines anchors on grid cells of an image.

This progress in 2D object detection and instance segmentation has inspired work on object detection and segmentation in the 3D domain, as we see more and more video and RGB-D data become available. Song et al. proposed Sliding Shapes to predict 3D object bounding boxes from single RGB-D frame input with handcrafted feature design [46], and then expanded the approach to operate on learned features [47]. The latter direction leverages the RGB frame input to improve classification accuracy of detected objects; in contrast to our approach, there is no explicit spatial mapping between RGB and geometry for joint feature learning. An alternative approach is taken by Frustum PointNet [48], where detection is performed a 2D frame and then back-projected into 3D from which final bounding box predictions are refined. Wang et al. [31] base their SGPN approach on semantic segmentation from a PointNet++ variation. They formulate instance segmentation as a clustering problem upon a semantically segmented point cloud by introducing a similarity matrix prediction similar to the idea behind panoptic segmentation [49]. In contrast to these approaches, we explicitly map both multi-view RGB input with 3D geometry in order to jointly infer 3D instance segmentation in an end-to-end fashion.

3.2.2 3D Deep Learning

In the recent years, we have seen impressive progress in developments on 3D deep learning. Analogous to the 2D domain, one can define convolution operators on volumetric grids, which for instance embed a surface representation as an implicit signed distance field [50]. With the availability of 3D shape databases [17], [23], [51] and annotated RGB-D datasets [4], [26], [52], [53], these network architectures are now being used for 3D object classification [51], [54]–[56], semantic segmentation [4], [29], [57], and object or scene completion [17], [58], [59]. An alternative representation to volumetric grids are the popular point-based architectures, such as PointNet [21] or PointNet++ [33], which leverage a more efficient, although less structured, representation of 3D surfaces. Multi-view approaches have also been proposed to leverage RGB or RGB-D video information. Su et al. proposed one of the first multi-view architectures for object classification by view-pooling over 2D predictions [60], and Kalogerakis et al. recently proposed an approach for shape segmentation by projecting predicted 2D confidence maps onto the 3D shape, which are then aggregated through a CRF [61]. Our approach joins together many of these ideas, leveraging the power of a holistic 3D representation along with features from 2D information by combining them through their explicit spatial mapping.

3.3 Method Overview

Our approach infers 3D object bounding box locations, class labels, and semantic instance masks on a per-voxel basis in an end-to-end fashion. To this end, we propose a neural network that jointly learns features from both geometry and RGB input. In the following, we refer to bounding box regression and object classification as object detection, and semantic instance mask segmentation for each object as mask prediction.

In Sec. 3.4, we first introduce the data representation and training data that is used by our approach. Here, we consider synthetic ground truth data from SUNCG [17], as well as manually-annotated real-world data from ScanNetV2 [4]. In Sec. 4.4, we present the neural network architecture of our 3D-SIS approach. Our architecture is composed of several parts; on the one hand, we have a series of 3D convolutions that operate in

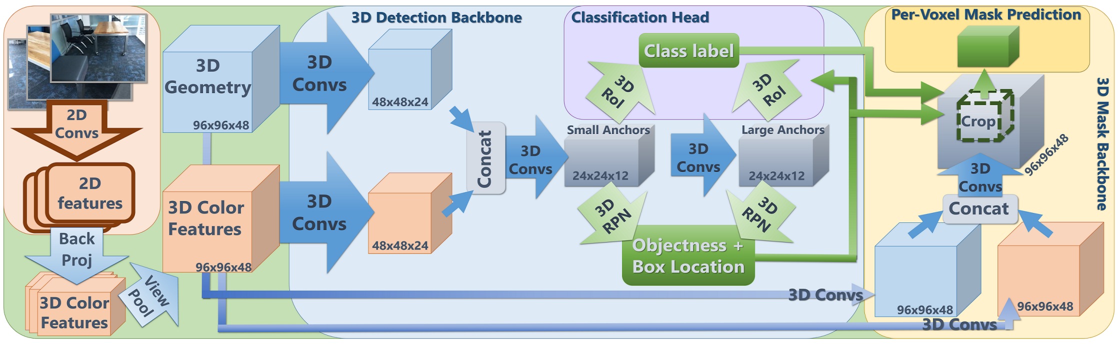


Figure 3.2: 3D-SIS Network Architecture. Our architecture is composed of a 3D detection and a 3D mask pipeline. Both 3D geometry and 2D color images are taken as input and used to jointly learn semantic features for object detection and instance segmentation. From the 3D detection backbone, color and geometry features are used to propose the object bounding boxes and their class labels through a 3D-RPN and a 3D-RoI layer. The mask backbone also uses color and geometry features, in addition to the 3D detection results, to predict per-voxel instance masks inside the 3D bounding box.

voxel grid space of the scanned 3D data. On the other hand, we learn 2D features that we backproject into the voxel grid where we join the features and thus jointly learn from both geometry and RGB data. These features are used to detect object instances; that is, associated bounding boxes are regressed through a 3D-RPN and class labels are predicted for each object following a 3D-ROI pooling layer. For each detected object, features from both the 2D color and 3D geometry are forwarded into a per-voxel instance mask network. Detection and per-voxel instance mask prediction are trained in an end-to-end fashion. In Sec. 4.5, we describe the training and implementation details of our approach, and in Sec. 4.6, we evaluate our approach.

3.4 Training Data

Data Representation We use a truncated sign distance field (TSDF) representation to encode the reconstructed geometry of the 3D scan inputs. The TSDF is stored in a regular volumetric grid with truncation of 3 voxels. In addition to this 3D geometry, we also input spatially associated RGB images. This is feasible since we know the mapping between each image pixel with voxels in the 3D scene grid based on the 6 degree-of-freedom (DoF) poses from the respective 3D reconstruction algorithm.

For the training data, we subdivide each 3D scan into chunks of 4.5m $\times$ 4.5m $\times$ 2.25m, and use a resolution of $96 \times 96 \times 48$ voxels per chunk (each voxel stores a TSDF value); i.e., our effective voxel size is $\approx 4.69\text{cm}^3$. In our experiments, for training, we associate 5 RGB images at a resolution of 328x256 pixels in every chunk, with training images selected based on the average voxel-to-pixel coverage of the instances within the region.

Our architecture is fully-convolutional (see Sec. 4.4), which allows us to run our method over entire scenes in a single shot for inference. Here, the xy-voxel resolution is derived from a given test scene's spatial extent. The z (height) of the voxel grid is fixed to 48 voxels (approximately the height of a room), with the voxel size also fixed at 4.69cm^3. Additionally, at test time, we use all RGB images available for inference. In order to evaluate our algorithm, we use training, validation, test data from synthetic and real-world RGB-D scanning datasets.

Synthetic Data For synthetic training and evaluation, we use the SUNCG [17] dataset. We follow the public train/val/test split, using 5519 train, 40 validation, and 86 test scenes (test scenes are selected to have total volume $< 600\text{m}^3$). From the train and validation scenes, we extract $97,918$ train chunks and 625 validation chunk. Each chunk contains an average of ≈ 4.3 object instances. At test time, we take the full scan data of the 86 test scenes.

In order to generate partial scan data from these synthetic scenes, we virtually render them, storing both RGB and depth frames. Trajectories are generated following the virtual scanning approach of [59], but adapted to provide denser camera trajectories to better simulate real-world scanning scenarios. Based on these trajectories, we then generate partial scans as TSDFs through volumetric fusion [50], and define the training data RGB-to-voxel grid image associations based on the camera poses. We use 23 class categories for instance segmentation, defined by their NYU40 class labels; these categories are selected for the most frequently-appearing object types, ignoring the wall and floor categories which do not have well-defined instances.

Real-world Data For training and evaluating our algorithm on real-world scenes, we use the ScanNetV2 [4] dataset. This dataset contains RGB-D scans of 1513 scenes, comprising $\approx$2.5 million RGB-D frames. The scans have been reconstructed using Bundle-Fusion [3]; both 6 DoF pose alignments and reconstructed models are available. Additionally, each scan contains manually-annotated object instance segmentation masks on the 3D mesh. From this data, we derive 3D bounding boxes which we use as constraints for our 3D region proposal.

We follow the public train/val/test split originally proposed by ScanNet of 1045 (train), 156 (val), 312 (test) scenes, respectively. From the train scenes, we extract 108241 chunks, and from the validation scenes, we extract 995 chunks. Note that due to the smaller number of train scans available in the ScanNet dataset, we augment the train scans to have 4 rotations each. We adopt the same 18-class label set for instance segmentation as proposed by the ScanNet benchmark.

Note that our method is agnostic to the respective dataset as long as semantic RGB-D instance labels are available.

3.5 Network Architecture

Our network architecture is shown in Fig. 3.2. It is composed of two main components, one for detection, and one for per-voxel instance mask prediction; each of these pipelines has its own feature extraction backbone. Both backbones are composed of a series of 3D convolutions, taking the 3D scan geometry along with the back-projected RGB color features as input. We detail the RGB feature learning in Sec. 3.5.1 and the feature backbones in Sec. 3.5.2. The learned 3D features of the detection and mask backbones are then fed into the classification and the voxel-instance mask prediction heads, respectively.

The object detection component of the network comprises the detection backbone, a 3D region proposal network (3D-RPN) to predict bounding box locations, and a 3D-region of interest (3D-RoI) pooling layer followed by classification head. The detection backbone outputs features which are input to the 3D-RPN and 3D-RoI to predict bounding box locations and object class labels, respectively. The 3D-RPN is trained by associating predefined anchors with ground-truth object annotations; here, a per-anchor loss defines whether an object exists for a given anchor. If it does, a second loss regresses the 3D object bounding box; if not, no additional loss is considered. In addition, we classify the the object class of each 3D bounding box. For the per-voxel instance mask prediction network (see Sec. 3.5.4), we use both the input color and geometry as well as the predicted bounding box location and class label. The cropped feature channels are used to create a mask prediction which has n channels for the n semantic class labels, and the final mask prediction is selected from these channels using the previously predicted class label. We optimize for the instance mask prediction using a binary cross entropy loss. Note that we jointly train the backbones, bounding box regression, classification, and per-voxel mask predictions end-to-end; see Sec. 4.5 for more detail. In the following, we describe the main components of our architecture design, for more detail regarding exact filter sizes, etc., we refer to the supplemental material.

3.5.1 Back-projection Layer for RGB Features

In order to jointly learn from RGB and geometric features, one could simply assign a single RGB value to each voxel. However, in practice, RGB image resolutions are significantly higher than the available 3D voxel resolution due to memory constraints. This 2D-3D resolution mismatch would make learning from a per-voxel color rather inefficient. Inspired by the semantic segmentation work of Dai et al. [29], we instead leverage a series of 2D convolutions to summarize RGB signal in image space. We then define a back-projection layer and map these features on top of the associated voxel grid, which are then used for both object detection and instance segmentation.

To this end, we first pre-train a 2D semantic segmentation network based on the ENet architecture [38]. The 2D architecture takes single 256×328 RGB images as input, and is trained on a semantic classification loss using the NYUv2 40 label set. From this pre-trained network, we extract a feature encoding of dimension 32×41 with 128 channels from the encoder. Using the corresponding depth image, camera intrinsics, and 6DoF poses, we then back-project each of these features back to the voxel grid (still 128 channels); the projection is from 2D pixels to 3D voxels. In order to combine features from multiple views, we perform view pooling through an element-wise max pooling over all RGB images available.

For training, the voxel volume is fixed to $96 \times 96 \times 48$ voxels, resulting in a $128 \times 96 \times 96 \times 48$ back-projected RGB feature grid in 3D; here, we use 5 RGB images for each training chunk (with image selection based on average 3D instance coverage). At test time, the voxel grid resolution is dynamic, given by the spatial extent of the environment; here, we use all available RGB images. The grid of projected features is processed by a set of 3D convolutions and is subsequently merged with the geometric features.

In ScanNet [4], the camera poses and intrinsics are provided; we use them directly for our back-projection layer. For SUNCG [17], extrinsics and intrinsics are given by the virtual scanning path. Note that our method is agnostic to the used 2D network architecture.

3.5.2 3D Feature Backbones

For jointly learning geometric and RGB features for both instance detection and segmentation, we propose two 3D feature learning backbones. The first backbone generates features for detection, and takes as input the 3D geometry and back-projected 2D features (see Sec. 3.5.1).

Both the geometric input and RGB features are processed symmetrically with a 3D ResNet block before joining them together through concatenation. We then apply a 3D convolutional layer to reduce the spatial dimension by a factor of 4, followed by a 3D ResNet block (e.g., for an input train chunk of $96 \times 96 \times 48$, we obtain a features of size $24 \times 24 \times 12$). We then apply another 3D convolutional layer, maintaining the same spatial dimensions, to provide features maps with larger receptive fields. We define anchors on these two feature maps, splitting the anchors into 'small' and 'large' anchors (small anchors $< 1\text{m}^3$), with small anchors associated with the first feature map of smaller receptive field and large anchors associated with the second feature map of larger receptive field. For selecting anchors, we apply k-means algorithm (k=14) on the ground-truth 3D bounding boxes in first 10k chunks. These two levels of features maps are then used for the final steps of object detection: 3D bounding box regression and classification.

The instance segmentation backbone also takes the 3D geometry and the back-projected 2D CNN features as input. The geometry and color features are first processed independently with two 3D convolutions, and then concatenated channel-wise and processed with another two 3D convolutions to produce a mask feature map prediction. Note that for the mask backbone, we maintain the same spatial resolution through all convolutions, which we found to be critical for obtaining high accuracy for the voxel instance predictions. The mask feature map prediction is used as input to predict the final instance mask segmentation.

In contrast to single backbone, we found that this two-backbone structure both converged more easily and produced significantly better instance segmentation performance (see Sec. 4.5 for more details about the training scheme for the backbones).

3.5.3 3D Region Proposals and 3D-RoI Pooling for Detection

Our 3D region proposal network (3D-RPN) takes input features from the detection backbone to predict and regress 3D object bounding boxes. From the detection backbone we obtain two feature maps for small and large anchors, which are separately processed by the 3D-RPN. For each feature map, the 3D-RPN uses a $1 \times 1 \times 1$ convolutional layer to reduce the channel dimension to $2 \times N_{\text{anchors}}$, where $N_{\text{anchors}} = (3, 11)$ for small and large anchors, respectively. These represent the positive and negative scores of objectness of each anchor. We apply a non-maximum suppression on these region proposals based on their objectness scores. The 3D-RPN then uses another $1 \times 1 \times 1$ convolutional layer to predict feature maps of $6 \times N_{\text{anchors}}$, which represent the 3D bounding box locations as $(\varDelta_x, \varDelta_y, \varDelta_z, \varDelta_w, \varDelta_h, \varDelta_l)$, defined in Eq. 3.1.

In order to determine the ground truth objectiveness and associated 3D bounding box locations of each anchor during training, we perform anchor association. Anchors are associated with ground truth bounding boxes by their IoU: if the IoU > 0.35, we consider an anchor to be positive (and it will be regressed to the associated box), and if the IoU < 0.15, we consider an anchor to be negative (and it will not be regressed to any box). We use a two-class cross entropy loss to measure the objectiveness, and for the bounding box regression we use a Huber loss on the prediction $(\varDelta_x, \varDelta_y, \varDelta_z, \varDelta_w, \varDelta_h, \varDelta_l)$ against the log ratios of the ground truth box and anchors $(\varDelta_x^{gt}, \varDelta_y^{gt}, \varDelta_z^{gt}, \varDelta_w^{gt}, \varDelta_h^{gt}, \varDelta_l^{gt})$, where

$$\varDelta_x = \frac{\mu - \mu_{anchor}}{\phi_{anchor}} \qquad \varDelta_w = \ln\left(\frac{\phi}{\phi_{anchor}}\right) \tag{3.1}$$

where μ is the box center point and ϕ is the box width.

Using the predicted bounding box locations, we can then crop out the respective features from the global feature map. We then unify these cropped features to the same dimensionality using our 3D Region of Interest (3D-RoI) pooling layer. This 3D-RoI pooling layer pools the cropped feature maps into $4 \times 4 \times 4$ blocks through max pooling operations. These feature blocks are then linearized for input to object classification, which is performed with an MLP.

3.5.4 Per-Voxel 3D Instance Segmentation

We perform instance mask segmentation using a separate mask backbone, which similarly as the detection backbone, takes as input the 3D geometry and projected RGB features. However, for mask prediction, the 3D convolutions maintain the same spatial resolutions, in order to maintain spatial correspondence with the raw inputs, which we found to significantly improve performance. We then use the predicted bounding box location from the 3D-RPN to crop out the associated mask features from the mask backbone, and compute a final mask prediction with a 3D convolution to reduce the feature dimensionality to n for n semantic class labels; the final mask prediction is the c^{th} channel for predicted object class c. During training, since predictions from the detection pipeline can be wrong, we only train on predictions whose predicted bounding box overlaps with the ground truth bounding box with at least 0.5 IoU. The mask targets are defined as the ground-truth mask in the overlapping region of the ground truth box and proposed box.

	cab	bed	chair	sofa	tabl	door	wind	bkshf	cntr	desk	shlf	curt	drsr	mirr	tv	nigh	toil	sink	lamp	bath	ostr	ofurn	oprop	**avg**
Seg-Cluster	16.8	16.2	15.6	11.8	14.5	10.0	11.7	27.2	20.0	25.7	10.0	0.0	15.0	0.0	20.0	27.8	39.5	22.9	10.7	38.9	10.4	0.0	12.3	16.4
Mask R-CNN [41]	14.9	19.0	19.5	13.5	12.2	11.7	14.2	35.0	15.7	18.3	13.7	0.0	24.4	**2**3.1	**2**6.0	**2**8.8	51.2	28.1	14.7	32.2	11.4	10.7	19.5	19.9
SGPN [31]	18.6	39.2	28.5	46.5	26.7	**2**1.8	**1**5.9	0.0	24.9	23.9	16.3	**2**0.8	15.1	10.7	0.0	17.7	35.1	37.0	**2**2.9	34.2	17.7	31.5	13.9	22.5
Ours(geo only)	23.2	**7**8.6	47.7	63.3	37.0	19.6	0.0	0.0	21.3	34.4	16.8	0.0	16.7	0.0	10.0	22.8	**5**9.7	49.2	10.0	77.2	10.0	0.0	**1**9.3	26.8
Ours(geo+1view)	22.2	70.8	48.5	**6**6.6	**4**4.4	10.0	0.0	**6**3.9	25.8	32.2	17.8	0.0	25.3	0.0	0.0	14.7	37.0	55.5	20.5	58.2	**1**8.0	20.0	17.9	29.1
Ours(geo+3views)	**2**6.5	78.4	48.2	59.5	42.8	26.1	0.0	30.0	22.7	**3**9.4	17.3	0.0	**3**6.2	0.0	10.0	10.0	37.0	50.8	16.8	**5**9.3	10.0	**3**6.4	17.8	29.4
Ours(geo+5views)	20.5	69.4	**5**6.2	64.5	43.8	17.8	0.0	30.0	**3**2.3	33.5	**2**1.0	0.0	34.2	0.0	10.0	20.0	56.7	**5**6.2	17.6	56.2	10.0	35.5	17.8	**3**0.6

Table 3.1: 3D instance segmentation on synthetic scans from SUNCG [17]. We evaluate the mean average precision with IoU threshold of 0.25 over 23 classes. Our joint color-geometry feature learning enables us to achieve more accurate instance segmentation performance.

	cab	bed	chair	sofa	tabl	door	wind	bkshf	pic	cntr	desk	curt	fridg	showr	toil	sink	bath	ofurn	**avg**
Mask R-CNN [41]	5.3	0.2	0.2	10.7	2.0	4.5	0.6	0.0	**23.8**	0.2	0.0	2.1	6.5	0.0	2.0	1.4	33.3	2.4	5.8
SGPN [31]	6.5	39.0	27.5	35.1	16.8	8.7	13.8	16.9	1.4	2.9	0.0	6.9	2.7	0.0	43.8	11.2	20.8	4.3	14.3
MTML	2.7	**61.4**	39.0	50.0	10.5	10.0	0.3	**33.7**	0.0	0.0	0.1	**11.8**	16.7	14.3	57.0	4.6	66.7	2.8	21.2
3D-BEVIS [62]	3.5	56.6	39.4	60.4	18.1	9.9	17.1	7.6	2.5	2.7	9.8	3.5	9.8	37.5	85.4	12.6	66.7	3.0	24.8
R-PointNet [63]	**34.8**	40.5	**58.9**	39.6	**27.5**	28.3	**24.5**	31.1	2.8	**5.4**	**12.6**	6.8	21.9	21.4	82.1	**33.1**	50.0	**29.0**	30.6
3D-SIS (Ours)	13.4	55.4	58.7	**72.8**	22.4	**30.7**	18.1	31.9	0.6	0.0	12.1	0.0	**54.1**	**100.0**	**88.9**	4.5	**66.7**	21.0	**36.2**

Table 3.2: Instance segmentation results on the official ScanNetV2 3D semantic instance benchmark (hidden test set). Our final model (geo+5views) significantly outperforms previous (Mask R-CNN, SGPN) and concurrent (MTML, 3D-BEVIS, R-PointNet) state-of-the-art methods in mAP@0.5. ScanNetV2 benchmark data accessed on 12/17/2018.

3.6 Training

To train our model, we first train the detection backbone and 3D-RPN. After pre-training these parts, we add the 3D-RoI pooling layer and object classification head, and train these end-to-end. Then, we add the per-voxel instance mask segmentation network along with the associated backbone. In all training steps, we always keep the previous losses (using 1:1 ratio between all losses), and train everything end-to-end. We found that a sequential training process resulted in more stable convergence and higher accuracy.

We use an SGD optimizer with learning rate 0.001, momentum 0.9 and batch size 64 for 3D-RPN, 16 for classification, 16 for mask prediction. The learning rate is divided by 10 every 100k steps. We use a non-maximum suppression for proposed boxes with threshold of 0.7 for training and 0.3 for test. Our network is implemented with PyTorch and runs on a single Nvidia GTX1080Ti GPU. The object detection components of the network are trained end-to-end for 10 epochs ($\approx$ 24 hours). After adding in the mask backbone, we train for an additional 5 epochs ($\approx$ 16 hours). For mask training, we also use ground truth bounding boxes to augment the learning procedure.

3.7 Results

We evaluate our approach on both 3D detection and instance segmentation predictions, comparing to several state-of-the-art approaches, on synthetic scans of SUNCG [17] data and real-world scans from the ScanNetV2 dataset [4]. To compare to previous approaches that operate on single RGB or RGB-D frames (Mask R-CNN [41], Deep Sliding Shapes [47], Frustum PointNet [48]), we first obtain predictions on each individual frame, and then merge all predictions together in the 3D space of the scene, merging predictions if the predicted class labels match and the IoU > 0.5. We further compare to SGPN [31] which performs instance segmentation on 3D point clouds. For both detection and instance segmentation tasks, we project all results into a voxel space of 4.69cm voxels and evaluate them with a mean average precision metric. We additionally show several variants of our approach for learning from both color and geometry features, varying the number of color views used during training. We consistently find that training on more color views improves both the detection and instance segmentation performance.

	cab	bed	chair	sofa	tabl	door	wind	bkshf	pic	cntr	desk	curt	fridg	showr	toil	sink	bath	ofurn	**avg**
Seg-Cluster	11.8	13.5	18.9	14.6	13.8	11.1	11.5	11.7	0.0	13.7	12.2	12.4	11.2	18.0	19.5	18.9	16.4	12.2	13.4
Mask R-CNN [41]	15.7	15.4	16.4	16.2	14.9	12.5	11.6	11.8	**19.5**	13.7	14.4	14.7	21.6	18.5	25.0	24.5	24.5	16.9	17.1
SGPN [31]	20.7	31.5	31.6	40.6	**31.9**	16.6	**15.3**	13.6	0.0	17.4	14.1	22.2	0.0	0.0	72.9	**52.4**	0.0	18.6	22.2
Ours(geo only)	22.1	48.2	64.4	52.2	16.0	13.4	0.0	17.2	0.0	20.7	17.4	13.9	23.6	33.0	45.2	47.7	61.3	14.6	28.3
Ours(geo+1view)	25.4	60.3	**66.2**	52.1	31.7	27.6	10.1	16.9	0.0	21.4	30.9	18.4	22.6	16.0	70.5	44.5	37.5	20.0	31.8
Ours(geo+3views)	28.3	52.3	65.0	**66.5**	31.4	**27.9**	10.1	**17.9**	0.0	20.3	**36.3**	20.1	28.1	31.0	68.6	41	**66.8**	**24.0**	35.3
Ours(geo+5views)	**32.0**	**66.3**	65.3	56.4	29.4	26.7	10.1	16.9	0.0	**22.1**	35.1	**22.6**	**28.6**	**37.2**	**74.9**	39.6	57.6	21.1	**35.7**

Table 3.3: 3D instance segmentation on ScanNetV2 [4]. We demonstrate mAP@0.25 on 18 classes. Our explicit leveraging of spatial mapping between 3D geometry and color features extracted through 2D CNNs enables significantly improved performance.

3.7.1 3D Instance Analysis on Synthetic Scans

We evaluate 3D detection and instance segmentation on virtual scans taken from the synthetic SUNCG dataset [17], using 23 class categories. Table 3.4 shows 3D detection performance compared to state-of-the-art approaches which operate on single frames. Table 4.4 shows a quantitative evaluation of our approach, the SGPN for point cloud instance segmentation [31], their proposed Seg-Cluster baseline, and Mask R-CNN [41] projected into 3D. For both tasks, our joint color-geometry approach along with a global view of the 3D scenes at test time enables us to achieve significantly improved detection and segmentation results.

3.7.2 3D Instance Analysis on Real-World Scans

We further evaluate our approach on ScanNet dataset [4], which contains 1513 real-world scans. For training and evaluation, we use ScanNetV2 annotated ground truth as well as the proposed 18-class instance benchmark. We show qualitative results in Figure 3.3. In Table 3.5, we quantitatively evaluate our object detection against Deep Sliding Shapes and Frustum PointNet, which operate on RGB-D frame, as well as Mask R-CNN [41]

	mAP@0.25	mAP@0.5
Deep Sliding Shapes [46]	12.8	6.2
Mask R-CNN 2D-3D [41]	20.4	10.5
Frustum PointNet [48]	24.9	10.8
Ours – 3D-SIS (geo only)	27.8	21.9
Ours – 3D-SIS (geo+1view)	30.9	23.8
Ours – 3D-SIS (geo+3views)	31.3	24.2
Ours – 3D-SIS (geo+5views)	**32.2**	**24.7**

Table 3.4: 3D detection in SUNCG [17]. We show mAP over 23 classes. Our holistic approach and the combination of color and geometric features result in significantly improved detection results over previous approaches which operate on individual input frames.

projected to 3D. Our fully-convolutional approach enabling inference on full test scenes achieves significantly better detection performance. Table 4.3 shows our 3D instance segmentation in comparison to SGPN instance segmentation [31], their proposed Seg-Cluster baseline, and Mask R-CNN [41] projected into 3D. Our formulation for learning from both color and geometry features brings notable improvement over state of the art.

	mAP@0.25	mAP@0.5
Deep Sliding Shapes [46]	15.2	6.8
Mask R-CNN 2D-3D [41]	17.3	10.5
Frustum PointNet [48]	19.8	10.8
Ours – 3D-SIS (geo only)	27.6	16.0
Ours – 3D-SIS (geo+1view)	35.1	18.7
Ours – 3D-SIS (geo+3views)	36.6	19.0
Ours – 3D-SIS (geo+5views)	**40.2**	**22.5**

Table 3.5: 3D detection on ScanNetV2 [4]. We show mAP over 18 classes. In contrast to previous approaches operating on individual frames, our approach achieves significantly improved performance.

Finally, we evaluate our model on the ScanNetV2 3D instance segmentation benchmark on the hidden test set; see Table 3.2. Our final model (geo+5views) significantly outperforms previous (Mask R-CNN [41], SGPN [31]) and concurrent (MTML, 3D-BEVIS [62], R-PointNet [63]) state-of-the-art methods in mAP@0.5. ScanNetV2 benchmark data was accessed on 12/17/2018.

3.8 Network Architecture

Table 3.8 details the layers used in our detection backbone, 3D-RPN, classification head, mask backbone, and mask prediction. Note that both the detection backbone and mask backbone are fully-convolutional. For the classification head, we use several fully-connected layers; however, due to our 3D RoI-pooling on its input, we can run our entire instance segmentation approach on full scans of varying sizes.

small anchors	big anchors
(8, 6, 8)	(12, 12, 40)
(22, 22, 16)	(8 , 60, 40)
(12, 12, 20)	(38, 12, 16)
	(62, 8 , 40)
	(46, 8 , 20)
	(46, 44, 20)
	(14, 38, 16)

Table 3.6: Anchor sizes (in voxels) used for SUNCG [17] region proposal. Sizes are given in voxel units, with voxel resolution of ≈ 4.69cm

small anchors	big anchors
(8, 8, 9)	(21, 7, 38)
(14, 14, 11)	(7, 21, 39)
(14, 14, 20)	(32, 15, 18)
	(15, 31, 17)
	(53, 24, 22)
	(24, 53, 22)
	(28, 4, 22)
	(4, 28, 22)
	(18, 46, 8)
	(46, 18, 8)
	(9, 9, 35)

Table 3.7: Anchor sizes used for region proposal on the ScanNet dataset [4]. Sizes are given in voxel units, with voxel resolution of ≈ 4.69cm

We additionally list the anchors used for the region proposal for our model trained on the ScanNet [4] and SUNCG [17] datasets in Tables 4.5 and 4.6, respectively. Anchors for each dataset are determined through k-means clustering of ground truth bounding boxes. The anchor sizes are given in voxels, where our voxel size is ≈ 4.69cm.

3.9 Training and Inference

In order to leverage as much context as possible from a input RGB-D scan, we leverage fully-convolutional detection and mask backbones to infer instance segmentation on varying-sized scans. To accommodate memory and efficiency constraints during training, we train on chunks of scans, i.e. cropped volumes out of the scans, which we use to generalize to the full scene at test time (see Figure 3.4). This also enables us to avoid inconsistencies which can arise with individual frame input, with differing views of the same object; with the full view of a test scene, we can more easily predict consistent object boundaries.

The fully-convolutional nature of our methods allows testing on very large scans such as entire floors or buildings in a single forward pass; e.g., most SUNCG scenes are actually fairy large; see Figure 3.5.

3.10 Additional Experiment Details

We additionally evaluate mean average precision on SUNCG [17] and ScanNetV2 [4] using an IoU threshold of 0.5 in Tables 3.11 and 3.10. Consistent with evaluation at an IoU threshold of 0.25, our approach leveraging joint color-geometry feature learning and inference on full scans enables significantly better instance segmentation performance. We also submit our model the ScanNet Benchmark, and we achieve the state-of-the-art in all three metrics.

We run an additional ablation study to evaluate the impact of the RGB input and the two-level anchor design; see Table. 4.3.

3.11 Conclusion

In this work, we introduce 3D-SIS, a new approach for 3D semantic instance segmentation of RGB-D scans, which is trained in an end-to-end fashion to detect object instances and infer a per-voxel 3D semantic instance segmentation. The core of our method is to jointly learn features from RGB and geometry data using multi-view RGB-D input recorded with commodity RGB-D sensors. The network is fully-convolutional, and thus can run efficiently in a single shot on large 3D environments. In comparison to existing state-of-the-art methods that typically operate on single RGB-D frame, we achieve significantly better 3D detection and instance segmentation results, improving on mAP by over 13. We believe that this is an important insight to a wide range of computer vision applications given that many of them now capture multi-view RGB and depth streams; e.g., autonomous cars, AR/VR applications, etc..

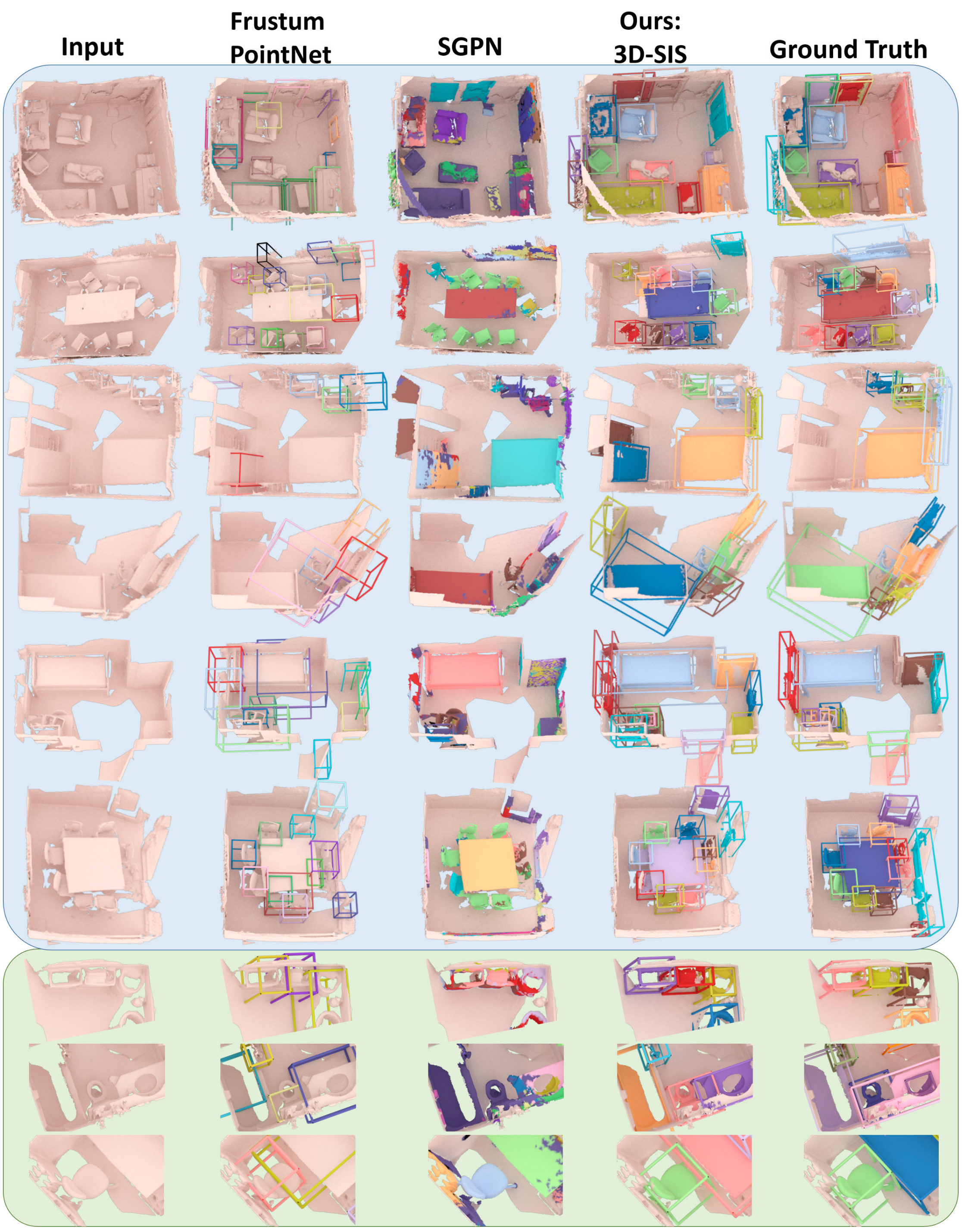


Figure 3.3: Qualitative comparison of 3D object detection and instance segmentation on ScanNetV2 [4]. Full scans above; close-ups below. Our joint color-geometry feature learning combined with our fully-convolutional approach to inference on full test scans at once enables more accurate and semantically coherent predictions. Note that different colors represent different instances, and the same instances in the ground truth and predictions are not necessarily the same color.

layer name	input layer	type	output size	kernel size	stride	padding
geo_1	TSDF	conv3d	(32, 48, 24, 48)	(2, 2, 2)	(2, 2, 2)	(0, 0, 0)
geo_2	geo_1	conv3d	(32, 48, 24, 48)	(1, 1, 1)	(1, 1, 1)	(0, 0, 0)
geo_3	geo_2	conv3d	(32, 48, 24, 48)	(3, 3, 3)	(1, 1, 1)	(1, 1, 1)
geo_4	geo_3	conv3d	(32, 48, 24, 48)	(1, 1, 1)	(1, 1, 1)	(0, 0, 0)
geo_5	geo_4	conv3d	(32, 48, 24, 48)	(1, 1, 1)	(1, 1, 1)	(0, 0, 0)
geo_6	geo_5	conv3d	(32, 48, 24, 48)	(3, 3, 3)	(1, 1, 1)	(1, 1, 1)
geo_7	geo_6	conv3d	(32, 48, 24, 48)	(1, 1, 1)	(1, 1, 1)	(0, 0, 0)
geo_8	geo_7	conv3d	(64, 24, 12, 24)	(2, 2, 2)	(2, 2, 2)	(0, 0, 0)
geo_9	geo_1	conv3d	(32, 24, 12, 24)	(1, 1, 1)	(1, 1, 1)	(0, 0, 0)
geo_10	geo_2	conv3d	(32, 24, 12, 24)	(3, 3, 3)	(1, 1, 1)	(1, 1, 1)
geo_11	geo_3	conv3d	(64, 24, 12, 24)	(1, 1, 1)	(1, 1, 1)	(0, 0, 0)
geo_12	geo_4	conv3d	(32, 24, 12, 24)	(1, 1, 1)	(1, 1, 1)	(0, 0, 0)
geo_13	geo_5	conv3d	(32, 24, 12, 24)	(3, 3, 3)	(1, 1, 1)	(1, 1, 1)
geo_14	geo_6	conv3d	(64, 24, 12, 24)	(1, 1, 1)	(1, 1, 1)	(0, 0, 0)
color_1	projected 2D features	conv3d	(64, 48, 24, 48)	(2, 2, 2)	(2, 2, 2)	(0, 0, 0)
color_2	color_1	conv3d	(32, 48, 24, 48)	(1, 1, 1)	(1, 1, 1)	(0, 0, 0)
color_3	color_2	conv3d	(32, 48, 24, 48)	(3, 3, 3)	(1, 1, 1)	(1, 1, 1)
color_4	color_3	conv3d	(64, 48, 24, 48)	(1, 1, 1)	(1, 1, 1)	(0, 0, 0)
color_5	color_4	maxpool3d	(64, 48, 24, 48)	(3, 3, 3)	(1, 1, 1)	(1, 1, 1)
color_6	color_5	conv3d	(64, 24, 12, 24)	(2, 2, 2)	(2, 2, 2)	(0, 0, 0)
color_7	color_6	conv3d	(32, 24, 12, 24)	(1, 1, 1)	(1, 1, 1)	(0, 0, 0)
color_8	color_7	conv3d	(32, 24, 12, 24)	(3, 3, 3)	(1, 1, 1)	(1, 1, 1)
color_9	color_8	conv3d	(64, 24, 12, 24)	(1, 1, 1)	(1, 1, 1)	(0, 0, 0)
color_10	color_9	maxpool3d	(64, 24, 12, 24)	(3, 3, 3)	(1, 1, 1)	(1, 1, 1)
concat_1	(geo_14, color_10)	concat	(128, 24, 12, 24)	None	None	None
combine_1	concat_1	conv3d	(128, 24, 12, 24)	(3, 3, 3)	(1, 1, 1)	(1, 1, 1)
combine_2	combine_1	conv3d	(64, 24, 12, 24)	(1, 1, 1)	(1, 1, 1)	(0, 0, 0)
combine_3	combine_2	conv3d	(64, 24, 12, 24)	(3, 3, 3)	(1, 1, 1)	(1, 1, 1)
combine_4	combine_3	conv3d	(128, 24, 12, 24)	(1, 1, 1)	(1, 1, 1)	(0, 0, 0)
combine_5	combine_4	conv3d	(64, 24, 12, 24)	(1, 1, 1)	(1, 1, 1)	(0, 0, 0)
combine_6	combine_5	conv3d	(64, 24, 12, 24)	(3, 3, 3)	(1, 1, 1)	(1, 1, 1)
combine_7	combine_6	conv3d	(128, 24, 12, 24)	(1, 1, 1)	(1, 1, 1)	(0, 0, 0)
combine_8	combine_7	maxpool3d	(128, 24, 12, 24)	(3, 3, 3)	(1, 1, 1)	(1, 1, 1)
rpn_1	combine_7	conv3d	(256, 24, 12, 24)	(3, 3, 3)	(1, 1, 1)	(1, 1, 1)
rpn_cls_1	rpn_1	conv3d	(6, 24, 12, 24)	(1, 1, 1)	(1, 1, 1)	(0, 0, 0)
rpn_bbox_1	rpn_1	conv3d	(18, 24, 12, 24)	(1, 1, 1)	(1, 1, 1)	(0, 0, 0)
rpn_2	combine_5	conv3d	(256, 24, 12, 24)	(3, 3, 3)	(1, 1, 1)	(1, 1, 1)
rpn_cls_2	rpn_2	conv3d	(22, 24, 12, 24)	(1, 1, 1)	(1, 1, 1)	(0, 0, 0)
rpn_bbox_2	rpn_2	conv3d	(66, 24, 12, 24)	(1, 1, 1)	(1, 1, 1)	(0, 0, 0)
cls_1	combine_7	FC	128x4x4x4 $\rightarrow$ 256	None	None	None
cls_2	cls_1	FC	256 $\rightarrow$ 256	None	None	None
cls_3	cls_2	FC	256 $\rightarrow$ 128	None	None	None
cls_cls	cls_3	FC	128 $\rightarrow N_{cls}$	None	None	None
cls_bbox	cls_3	FC	128 $\rightarrow N_{cls} \times 6$	None	None	None
mask_1	TSDF	conv3d	(64, 96, 48, 96)	(3, 3, 3)	(1,1,1)	(1,1,1)
mask_2	mask_1	conv3d	(64, 96, 48, 96)	(3, 3, 3)	(1,1,1)	(1,1,1)
mask_3	mask_2	conv3d	(64, 96, 48, 96)	(3, 3, 3)	(1,1,1)	(1,1,1)
mask_4	mask_3	conv3d	(64, 96, 48, 96)	(3, 3, 3)	(1,1,1)	(1,1,1)
mask_5	mask_4	conv3d	(64, 96, 48, 96)	(3, 3, 3)	(1,1,1)	(1,1,1)
mask_6	mask_5	conv3d	(N_{cls}, 96, 48, 96)	(1, 1, 1)	(1,1,1)	(0,0,0)

Table 3.8: 3D-SIS network architecture. We show detailed layer specifications of our network including kernel size, channel numbers etc.

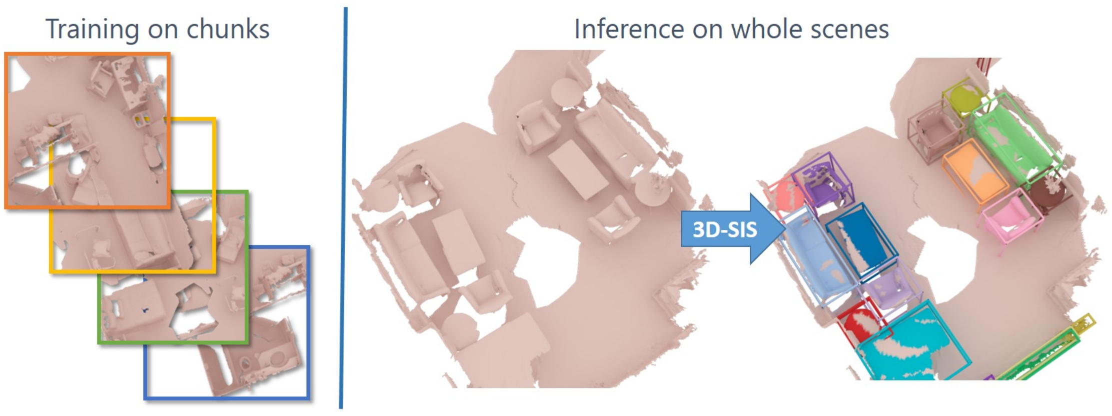


Figure 3.4: 3D-SIS Training and Inference 3D-SIS trains on chunks of a scene, and leverages fully-convolutional backbone architectures to enable inference on a full scene in a single forward pass, producing more consistent instance segmentation results.

	mAP@0.5	mAP@0.25
3D-SIS (only color-1view)	9.4	30.5
3D-SIS (only color-3view)	16.5	35.0
3D-SIS (only color-5view)	17.4	35.7
3D-SIS (only geometry)	16.0	27.6
3D-SIS (one anchor layer)	12.2	33.4
3D-SIS (final)	**22.5**	**40.2**

Table 3.9: Additional ablation study on ScanNetV2. We show combination of geometry and color signal complement each other, thus achieving the best performance.

	cab	bed	chair	sofa	tabl	door	wind	bkshf	pic	cntr	desk	curt	fridg	showr	toil	sink	bath	ofurn	**avg**
Seg-Cluster	10.4	11.9	15.5	12.8	12.4	10.1	10.1	10.3	0.0	11.7	10.4	11.4	0.0	13.9	17.2	11.5	14.2	10.5	10.8
Mask R-CNN [41]	11.2	10.6	10.6	11.4	10.8	10.3	0.0	0.0	**11.1**	**10.1**	0.0	10.0	12.8	0.0	18.9	13.1	11.8	11.6	9.1
SGPN [31]	10.1	16.4	20.2	20.7	14.7	11.1	**11.1**	0.0	0.0	10.0	10.3	12.8	0.0	0.0	48.7	16.5	0.0	0.0	11.3
Ours(geo only)	11.5	17.5	18.0	26.3	0.0	10.1	0.0	10.3	0.0	0.0	0.0	0.0	**24.4**	21.5	25.0	**17.2**	34.9	10.1	12.6
Ours(geo+1view)	12.5	15.0	17.8	23.7	0.0	**19.0**	0.0	11.0	0.0	0.0	10.5	11.1	13.0	19.4	22.5	14.0	**40.5**	10.1	13.3
Ours(geo+3views)	14.4	19.9	**48.4**	**37.3**	**16.9**	18.3	0.0	11.0	0.0	0.0	10.5	**13.1**	16.3	15.3	**51.3**	13.0	12.9	**13.4**	17.3
Ours(geo+5views)	**19.7**	**37.7**	40.5	31.9	15.9	18.1	0.0	**11.0**	0.0	0.0	**10.5**	11.1	18.5	**24.0**	45.8	15.8	23.5	12.9	**18.7**

Table 3.10: 3D instance segmentation on real-world scans from ScanNetV2 [4]. We evaluate the mean average precision with IoU threshold of 0.5 over 18 classes. Our explicit leveraging of the spatial mapping between the 3D geometry and color features extracted through 2D convolutions enables significantly improved instance segmentation performance.

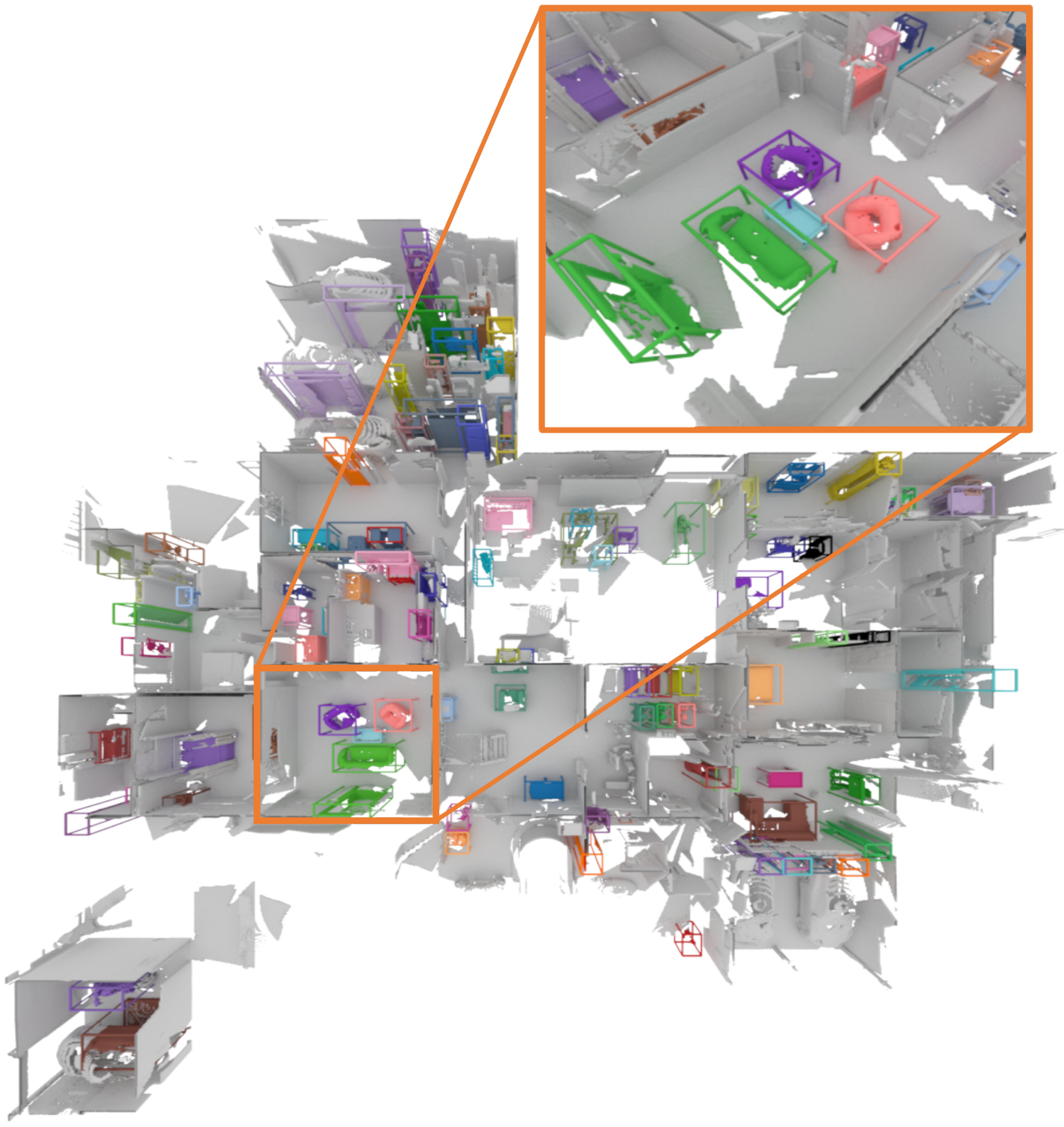


Figure 3.5: Inference on large scene. Our fully-convolutional architectures allows testing on a large SUNCG scene (45m x 45m) in about 1 second runtime.

	cab	bed	chair	sofa	tabl	door	wind	bkshf	cntr	desk	shlf	curt	drsr	mirr	tv	nigh	toil	sink	lamp	bath	ostr	ofurn	oprop	**avg**
Seg-Cluster	10.1	10.9	10.4	10.1	10.3	0.0	0.0	12.9	10.7	15.2	0.0	0.0	10.0	0	0.0	11.2	26.1	12.1	0	16.5	0	0	10	7.7
Mask R-CNN [41]	0.0	10.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**10.8**	**11.4**	10.8	18.8	13.5	0.0	11.5	0.0	0.0	10.7	4.3
SGPN [31]	15.3	28.7	23.7	29.7	17.6	15.1	**15.4**	0.0	10.8	16.0	0.0	**10.9**	0.0	0.0	0.0	12.3	33.7	25.9	**19.2**	31.7	0.0	10.4	10.5	14.2
Ours(geo only)	12.6	**60.5**	38.6	45.8	21.8	**16.8**	0.0	0.0	10.0	18.5	10.0	0.0	14.0	0.0	0.0	**14.9**	**64.2**	30.8	17.6	35.2	10.0	0.0	**16.9**	19.1
Ours(geo+1view)	13.9	42.4	35.3	**52.9**	22	10	0.0	**35.0**	**13.4**	**21.4**	10.0	0.0	13.5	0.0	0.0	10.0	33.8	29.2	**17.7**	**48.3**	10.0	16.9	10.0	19.4
Ours(geo+3views)	15.4	58.5	35.5	34.5	**24.4**	16.6	0.0	20.0	10.0	17.6	10.0	0.0	**24.3**	0.0	10.0	10.0	34.6	28.5	15.6	40.7	10.0	**24.9**	15.5	19.8
Ours(geo+5views)	**15.5**	43.6	**43.9**	48.1	20.4	10.0	0.0	30.0	10.0	17.4	**10.0**	0.0	14.5	0.0	10.0	10.0	53.5	**35.1**	17.2	39.7	**10.0**	18.9	16.2	**20.6**

Table 3.11: 3D instance segmentation on synthetic scans from SUNCG [17]. We evaluate the mean average precision with IoU threshold of 0.5 over 23 classes. Our joint color-geometry feature learning enables us to achieve more accurate instance segmentation performance.

4 Seeing Behind Objects in RGB-D Scans

This chapter introduces the following paper:

J. Hou, A. Dai, and M. Nießner, “RevealNet: Seeing Behind Objects in RGB-D Scans,” in *Proceedings of Computer Vision and Pattern Recognition (CVPR), IEEE*, 2020

Abstract of paper 3D perception requires high quality 3D reconstruction. During 3D reconstruction, it is often the case that people cannot scan each individual object from all views, resulting in missing geometry in the captured scan. This missing geometry can be fundamentally limiting for many applications, e.g., a robot needs to know the unseen geometry to perform a precise grasp on an object. Thus, we introduce the task of semantic instance completion: from an incomplete RGB-D scan of a scene, we aim to detect the individual object instances and infer their complete object geometry. This will open up new possibilities for interactions with objects in a scene, for instance for virtual or robotic agents. We tackle this problem by introducing RevealNet, a new data-driven approach that jointly detects object instances and predicts their complete geometry. This enables a semantically meaningful decomposition of a scanned scene into individual, complete 3D objects, including hidden and unobserved object parts. RevealNet is an end-to-end 3D neural network architecture that leverages joint color and geometry feature learning. The fully-convolutional nature of our 3D network enables efficient inference of semantic instance completion for 3D scans at scale of large indoor environments in a single forward pass. We show that predicting complete object geometry improves both 3D detection and instance segmentation performance. We evaluate on both real and synthetic scan benchmark data for the new task, where we outperform state-of-the-art approaches by over 15 in mAP@0.5 on ScanNet, and over 18 in mAP@0.5 on SUNCG.

Contribution The method development and implementation was done by the first author. Alternative baselines were provided by Angela Dai. Discussions with the co-authors led to the final paper.

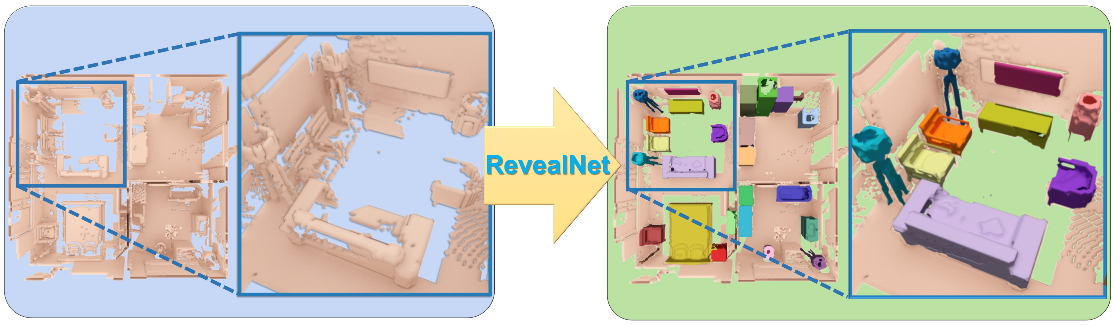


Figure 4.1: RevealNet takes an RGB-D scan as input and learns to "see behind objects": from the scan's color images and geometry (encoded as a TSDF), objects in the observed scene are detected (as 3D bounding boxes and class labels) and for each object, the complete geometry of that object is predicted as per-instance masks (in both seen and unseen regions).

4.1 Introduction

Understanding 3D environments is fundamental to many tasks spanning computer vision, graphics, and robotics. In particular, in order to effectively navigate, and moreover interact with an environment, an understanding of the geometry of a scene and the objects it comprises of is essential. This is in contrast to the partial nature of reconstructed RGB-D scans; e.g., due to sensor occlusions. For instance, for a robot exploring an environment, it needs to infer where objects are as well as what lies behind the objects it sees in order to efficiently navigate or perform tasks like grasping. That is, it needs not only instance-level knowledge of objects in the scene, but to also estimate the missing geometry of these objects. Additionally, for content creation or mixed reality applications, captured scenes must be decomposable into their complete object components, in order to enable applications such as scene editing or virtual-real object interactions; i.e., it is often insufficient to segment object instances only for observed regions.

Thus, we aim to address this task of "seeing behind objects," which we refer to as *semantic instance completion*: predicting object detection as well as instance-level completion for an input partial 3D scan of a scene. Previous approaches have addressed these tasks independently: 3D instance segmentation segments object instances from the visible surface of a partial scan [9], [31], [43], [63]–[67], and 3D scan completion approaches predict the full scene geometry [17], [59], but lack the notion of individual objects. In contrast, our approach focuses on the instance level, as knowledge of instances is essential towards enabling interaction with the objects in an environment.

In addition, the task of semantic instance completion is not only important towards enabling object-level understanding and interaction with 3D environments, but we also show that the prediction of complete object geometry informs the task of semantic instance segmentation. Thus, in order to address the task of semantic instance completion, we propose to consider instance detection and object completion in an end-to-end, fully differentiable fashion.

From an input RGB-D scan of a scene, our RevealNet model sees behind objects to predict each object's complete geometry. First, object bounding boxes are detected and regressed, followed by object classification and then a prediction of complete object geometry. Our approach leverages a unified backbone from which instance detection and object completion are predicted, enabling information to flow from completion to detection. We incorporate features from both color image and 3D geometry of a scanned scene, as well as a fully-convolutional design in order to effectively predict the complete object decomposition of varying-sized scenes. To address the task of semantic instance completion for real-world scans, where ground truth complete geometry is not readily available, we further introduce a new semantic instance completion benchmark for ScanNet [4], leveraging the Scan2CAD [25] annotations to evaluate semantic instance completion (and semantic instance segmentation).

In summary, we present a fully-convolutional, end-to-end 3D CNN formulation to predict 3D instance completion that outperforms state-of-the-art, decoupled approaches to semantic instance completion by 15.8 in mAP@0.5 on real-world scan data, and 18.5 in mAP@0.5 on synthetic data:

- We introduce the task of *semantic instance completion* for 3D scans;
- we propose a novel, end-to-end 3D convolutional network which predicts 3D semantic instance completion as object bounding boxes, class labels, and complete object geometry,
- and we show that semantic instance completion task can benefit semantic instance segmentation and detection performance.

4.2 Related Work

Object Detection and Instance Segmentation Recent advances in convolutional neural networks have now begun to drive impressive progress in object detection and instance segmentation for 2D images [39]–[45]. Combined with the increasing availability of synthetic and real-world 3D data [4], [17], [53], we are now seeing more advances in object detection [46]–[48], [68] for 3D. Sliding Shapes [46] predicted 3D object bounding boxes from a depth image, designing handcrafted features to detect objects in a sliding window fashion. Deep Sliding Shapes [47] then extended this approach to leverage learned features for object detection in a single RGB-D frame. Frustum PointNet [48] tackles the problem of object detection for an RGB-D frame by first detecting object in the 2D image before projecting the detected boxes into 3D to produce final refined box predictions. VoteNet [68] propose a reformulation of Hough voting in the context of deep learning through an end-to-end differentiable architecture for 3D detection purpose.

Recently, several approaches have been introduced to perform 3D instance segmentation, applicable to single or multi-frame RGB-D input. Wang et al. [31] introduced SGPN to operate on point clouds by clustering semantic segmentation predictions. Li et al. [63] leverages an object proposal-based approach to predict instance segmentation for a point cloud. Simultaneously, Hou et al. [9] presented an approach leveraging joint color-geometry feature learning for detection and instance segmentation on volumetric data. Lahoud et al. [65] proposes to use multi-task losses to predict instance segmentation. Yang et al. [64] and Liu et al. [69] both use bottom-up methods to predict instance segmentation for a point cloud. Our approach also leverages an anchor-based object proposal mechanism for detection, but we leverage object completion to predict instance completion, as well as show that completing object-level geometry can improve detection and instance segmentation performance on volumetric data.

3D Scan Completion Scan completion of 3D shapes has been a long-studied problem in geometry processing, particularly for cleaning up broken mesh models. In this context, traditional methods have largely focused on filling small holes by locally fitting geometric primitives, or through continuous energy minimization [70]–[72]. Surface reconstruction approaches on point cloud inputs [73], [74] can also be applied in this fashion to locally optimize for missing surfaces. Other shape completion approaches leverage priors such as symmetry and structural priors [75]–[79], or CAD model retrieval [80]–[84] to predict the scan completion.

Recently, methods leveraging generative deep learning have been developed to predict the complete geometry of 3D shapes [51], [85]–[87]. Song et al. [17] extended beyond shapes to predicting the voxel occupancy for a single depth frame leveraging the geometric occupancy prediction to achieve improved 3D semantic segmentation. Recently, Dai et al. [59] presented a first approach for data-driven scan completion of full 3D scenes, leveraging a fully-convolutional, autoregressive approach to predict complete geometry along with 3D semantic segmentation. Both Song et al. [17] and Dai et al. [59] show that inferring the complete scan geometry can improve 3D semantic segmentation. With our approach for 3D semantic instance completion, this task not only enables new applications requiring instance-based knowledge of a scene (e.g., virtual or robotic interactions with objects in a scene), but we also show that instance segmentation can benefit from instance completion.

4.3 Method Overview

Our network takes as input an RGB-D scan, and learns to join together features from both the color images as well as the 3D geometry to inform the semantic instance completion. The architecture is shown in Fig. 4.2.

The input 3D scan is encoded as a truncated signed distance field (TSDF) in a volumetric grid. To combine this with color information from the RGB images, we first extract 2D features using 2D convolutional layers on the RGB images, which are then back-projected into a 3D volumetric grid, and subsequently merged with geometric features extracted from the geometry. The joint features are then fed into an encoder-decoder backbone, which leverages a series of 3D residual blocks to learn the representation for the task of semantic instance completion. Objects are detected through anchor proposal and bounding box regression; these predicted object boxes are then used to crop and extract features from the backbone encoder to predict the object class label as well as the complete object geometry for each detected object as per-voxel occupancies.

We adopt in total five losses to supervise the learning process illustrated in Fig. 4.2. Detection contains three losses: (1) objectness using binary cross entropy to indicate that there is an object, (2) box location using a Huber loss to regress the 3D bounding box locations, and (3) classification of the class label loss using cross entropy. Following detection, the completion head contains two losses: per-instance completion loss using binary cross entropy to predict per-voxel occupancies, and a proxy completion loss using binary cross entropy to classify the surface voxels belonging to all objects in the scene.

Our method operates on a unified backbone for detection followed by instance completion, enabling object completion to inform the object detection process; this results in effective 3D detection as well as instance completion. Its fully-convolutional nature enables us to train on cropped chunks of 3D scans but test on a whole scene in a single forward pass, resulting in an efficient decomposition of a scan into a set of complete objects.

4.4 Network Architecture

From an RGB-D scan input, our network operates on the scan's reconstructed geometry, encoded as a TSDF in a volumetric grid, as well as the color images. To jointly learn from both color and geometry, color features are first extracted in 2D with a 2D semantic segmentation network [38], and then back-projected into 3D to be combined with the TSDF features, similar to [9], [29]. This enables complementary semantic features to be learned from both data modalities. These features are then input to the backbone of our network, which is structured in an encoder-decoder style.

The encoder-decoder backbone is composed of a series of five 3D residual blocks, which generates five volumetric feature maps $\mathbb{F} = \{f_i | i = 1 \dots 5\}$. The encoder results in a reduction of spatial dimension by a factor of 4, and symmetric decoder results in an expansion of spatial dimension by a factor of 4. Skip connections link spatially-corresponding encoder and decoder features. For a more detailed description of the network architecture, we refer to the appendix.

4.4.1 Color Back-Projection

As raw color data is often of much higher resolution than 3D geometry, to effectively learn from both color and geometry features, we leverage color information by back-projecting 2D CNN features learned from RGB images to 3D, similar to [9], [29]. For each voxel location $v_i = (x, y, z)$ in the 3D volumetric grid, we find its pixel location $p_i = (x, y)$ in 2D views by camera intrinsic and extrinsic matrices. We assign the voxel feature at location v_i with the learned 2D CNN feature vector at p_i. To handle multiple image observations of the same voxel v_i, we apply element-wise view pooling; this also allows our approach to handle a varying number of input images. Note that this back-

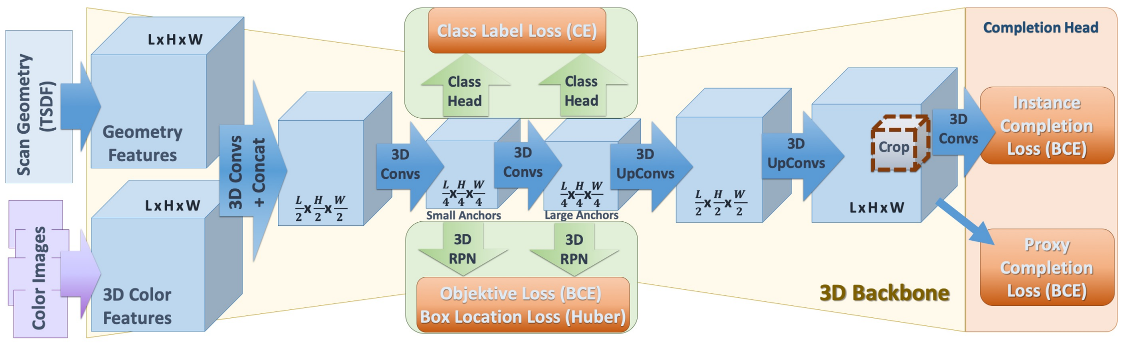


Figure 4.2: Network Architecture. Our RevealNet network architecture takes an RGB-D scan as input. Color images are processed with 2D convolutions to spatially compress the information before back-projecting into 3D, to be merged with the 3D geometry features of the scan (following [9], [29]). These joint features are used for object detection (as 3D bounding boxes and class labels) followed by per-instance geometric completion, for the task of semantic instance completion. In contrast to [9], which leverages separate backbones for detection and instance segmentation, our network maintains one unified backbone for both detection and completion head, allowing the completion task to directly inform the detection parameters.

projection is differentiable, allowing our model to be trained end-to-end and benefit from both RGB and geometric signal.

4.4.2 Object Detection

For object detection, we predict the bounding box of each detected object as well as the class label. To inform the detection, features are extracted from feature maps F_2 and F_3 of the backbone encoder. We define two set of anchors on these two features maps, $A_s = \{a_i | i = 1 \dots N_s\}$ and $A_b = \{a_i | i = 1 \dots N_b\}$ representing 'small' and 'large' anchors for the earlier F_2 and later F_3, respectively, so that the larger anchors are associated with the feature map of larger receptive field. These anchors $A_s \cup A_b$ are selected through a k-means clustering of the ground truth 3D bounding boxes. For our experiments, we use $N_s + N_b = 9$. From these $N_s + N_b$ clusters, A_b are those with any axis > 1.125m, and the rest are in A_s.

The two features maps F_2 and F_3 are then processed by a 3D region proposal to regress the 3D object bounding boxes. The 3D region proposal first employs a $1 \times 1 \times 1$ convolution layer to output objectness scores for each potential anchor, producing an objectness feature map with $2(N_s + N_b)$ channels for the positive and negative objectness probabilities. Another $1 \times 1 \times 1$ convolution layer is used to predict the 3D bounding box locations as 6-dimensional offsets from the anchors; we then apply a non-maximum suppression based on the objectness scores. We use a Huber loss on the log ratios of the offsets to the anchor sizes to regress the final bounding box predictions:

$$\Delta_x = \frac{\mu - \mu_{anchor}}{\phi_{anchor}} \qquad \Delta_w = \ln\left(\frac{\phi}{\phi_{anchor}}\right) \tag{4.1}$$

where μ is the box center point and ϕ is the box width. The final bounding box loss is then:

$$L_{\Delta} = \begin{cases} \frac{1}{2}\Delta^2, & \text{if } |\Delta| \ \leq 2 \\ |\Delta|, & \text{otherwise.} \end{cases}$$

Using these predicted object bounding boxes, we then predict the object class labels using features cropped from the bounding box locations from F_2 and F_3. We use a 3D region of interest pooling layer to unify the sizes of the cropped feature maps to a spatial dimension of $4 \times 4 \times 4$ to be input to an object classification MLP.

4.4.3 Instance Completion

For each object, we infer its complete geometry by predicting per-voxel occupancies. Here, we crop features from feature map F_5 of the backbone, which has a feature map resolution matching the input spatial resolution, using the predicted object bounding box. These features are processed through a series of five 3D convolutions which maintain the spatial resolution of their input. The complete geometry is then predicted as voxel occupancy using a binary cross entropy loss.

We predict N_{classes} potential object completions for each class category, and select the final prediction based on the predicted object class. We define ground truth bounding boxes b_i and masks m_i as $\gamma = \{(b_i, m_i)|i = 1 \ldots N_b\}$. Further, we define predicted bounding boxes $\hat{b_i}$ along with predicted masks $\hat{m_i}$ as $\hat{\gamma} = \{(\hat{b_i}, \hat{m_i})|i = 1 \ldots \hat{N_b}\}$. During training, we only train on predicted bounding boxes that overlap with the ground truth bounding boxes:

$$\begin{aligned} \Omega = \{(\hat{b_i}, \hat{m_i}, b_i, m_i) \mid \text{IoU}(\hat{b_i}, b_i) \geq 0.5, \\ \forall(\hat{b_i}, \hat{m_i}) \in \hat{\gamma}, \forall(b_i, m_i) \in \gamma\} \end{aligned}$$

We can then define the instance completion loss for each associated pair in Ω:

$$L_{\text{compl}} = \frac{1}{|\Omega|} \sum_{\Omega} \text{BCE}(\text{sigmoid}(\hat{m_i}), m_i'),$$

$$m_i'(v) = \begin{cases} m_i(v) & \text{if } v \in \hat{b_i} \cap b_i \\ 0 & \text{otherwise.} \end{cases}$$

We further introduce a global geometric completion loss on entire scene level that serves as an intermediate proxy. To this end, we use feature map F_5 as input to a binary cross entropy loss whose target is the composition of all complete object instances of the scene:

$$L_{\text{geometry}} = \text{BCE}(\text{sigmoid}(F_5), \cup_{(b_i, m_i) \in \gamma}).$$

Our intuition is to obtain a strong gradient during training by adding this additional constraint to each voxel in the last feature map F_5. We find that this global geometric completion loss further helps the final instance completion performance; see Sec 4.6.

	display	table	bathtub	trashbin	sofa	chair	cabinet	bookshelf	**avg**
Scene Completion + Instance Segmentation	1.65	0.64	4.55	11.25	9.09	9.09	0.18	5.45	5.24
Instance Segmentation + Shape Completion	2.27	3.90	1.14	1.68	14.86	9.93	7.11	3.03	5.49
Ours – RevealNet (no color)	13.16	11.28	13.64	18.19	24.79	15.87	8.60	10.60	14.52
Ours – RevealNet (no proxy)	21.94	7.63	12.55	28.24	20.38	22.58	13.42	9.51	17.03
Ours – RevealNet	**26.86**	**13.21**	**22.31**	**28.93**	**29.41**	**23.64**	**15.35**	**14.48**	**21.77**

Table 4.1: 3D Semantic Instance Completion on ScanNet [4] scans with Scan2CAD [25] targets. We use mAP@0.5 as metric. Our end-to-end formulation achieves significantly better performance than alternative, decoupled approaches that first use state-of-the-art scan completion [59] and then instance segmentation [9] method or first instance segmentation [9] and then shape completion [85].

4.5 Network Training

4.5.1 Data

The input 3D scans are represented as truncated signed distance fields (TSDFs) encoded in volumetric grids. The TSDFs are generated through volumetric fusion [50] during the 3D reconstruction process. For all our experiments, we used a voxel size of $\approx$ 4.7cm and truncation of 3 voxels. We also input the color images of the RGB-D scan, which we project to the 3D grid using their camera poses. We train our model on both synthetic and real scans, computing 9 anchors through k-means clustering; for real-world ScanNet [4] data, this results in 4 small anchors and 5 large anchors, and for synthetic SUNCG [17] data, this results in 3 small anchors and 6 large anchors.

At test time, we leverage the fully-convolutional design to input the full scan of a scene along with its color images. During training, we use random $96 \times 48 \times 96$ crops ($4.5 \times 2.25 \times 4.5$ meters) of the scanned scenes, along with a greedy selection of ≤ 5 images covering the most object geometry in the crop. Only objects with 50% of their complete geometry inside the crop are considered.

4.5.2 Optimization

We train our model jointly, end-to-end from scratch. We use an SGD optimizer with batch size 64 for object proposals and 16 for object classification, and all positive bounding box predictions (> 0.5 IoU with ground truth box) for object completion. We use a learning rate of 0.005, which is decayed by a factor of 0.1 every 100k steps. We train our model for 200k steps ($\approx$ 60 hours) to convergence, on a single Nvidia GTX 1080Ti. Additionally, we augment the data for training the object completion using ground truth bounding boxes and classification in addition to predicted object detection.

4.6 Results

We evaluate our approach on semantic instance completion performance on synthetic scans of SUNCG [17] scenes as well as on real-world ScanNet [4] scans, where we obtain ground truth object locations and geometry from CAD models aligned to ScanNet

	cab	bed	chair	sofa	tabl	door	wind	bkshf	cntr	desk	shlf	curt	drsr	mirr	tv	nigh	toil	sink	lamp	bath	ostr	ofurn	oprop	**avg**
SC + IS	3.0	0.6	19.5	0.8	18.1	**15.9**	0.00	0.0	1.0	2.3	3.0	0.0	0.5	0.0	9.2	10.4	23.9	3.4	9.1	0.0	0.0	0.0	9.1	5.5
IS + SC	0.3	0.0	7.4	0.4	3.0	9.1	0.0	0.0	0.2	0.0	0.0	0.0	2.3	0.0	3.0	0.0	2.6	0.0	1.8	0.0	0.0	0.0	4.6	1.5
no color	**19.05**	41.8	38.2	11.9	23.9	9.1	0.0	0.0	2.5	**21.6**	9.1	0.0	**12.6**	**4.6**	49.4	33.8	63.4	**36.9**	**38.8**	14.7	15.9	0.0	**23.8**	20.5
no proxy	12.9	46.1	**39.4**	26.8	**30.3**	1.0	15.9	0.0	9.1	18.2	3.4	0.0	1.1	0.0	43.6	**34.0**	**69.1**	32.4	29.6	**31.1**	14.6	0.0	23.3	20.9
Ours	14.7	**58.3**	38.2	**28.8**	29.5	0.0	**15.9**	**54.6**	**9.1**	12.1	**9.1**	**0.0**	6.2	0.0	**49.4**	33.5	61.2	34.5	29.5	27.1	**16.4**	**0.0**	23.5	**24.0**

Table 4.2: 3D Semantic Instance Completion on synthetic SUNCG [17] scans at mAP@0.5. Our semantic instance completion approach achieves significantly better performance than alternative approaches with decoupled state-of-the-art scan completion (SC) [59] followed by instance segmentation (IS) [9], as well as instance segmentation followed by shape completion [85]. We additionally evaluate our approach without color input (no color) and without a completion proxy loss on the network backbone (no proxy).

provided by Scan2CAD [25]. To evaluate semantic instance completion, we use a mean average precision metric on the complete masks (at IoU 0.5). Qualitative results are shown in Figs. 4.3 and 4.4.

	3D Detection	Instance Segmentation
3D-SIS [9]	25.70	20.78
Ours (no compl)	31.93	24.49
Ours (no color)	29.29	23.55
Ours (no proxy)	31.52	25.92
Ours	**36.39**	**30.52**

Table 4.3: 3D Detection and Instance Segmentation on ScanNet [4] scans with Scan2CAD [25] annotations at mAP@0.5. We evaluate our instance completion approach on the task of instance segmentation and detection to justify our contribution that instance completion task helps instance segmentation and detection. We evaluate our approach without completion (no compl), without color input (no color), and without a completion proxy loss on the network backbone (no proxy). Predicting instance completion notably increases performance of predicting both instance segmentation and detection (Ours vs. no compl). We additionally compare against 3D-SIS [9], a state-of-the-art approach for both 3D detection and instance segmentation on 3D dense volumetric data (the representation we use).

Comparison to state-of-the-art approaches for semantic instance completion. Tables 4.1 and 4.2 evaluate our method against state of the art for the task of semantic instance completion on our real and synthetic scans, respectively. Qualitative comparisons on ScanNet scans [4] with Scan2CAD [25] targets (which provide ground truth for complete object geometry) are shown in Fig. 4.3. We compare to state-of-the-art 3D instance segmentation and scan completion approaches used sequentially; that is, first applying a 3D instance segmentation approach followed by a shape completion method on the predicted instance segmentation, as well as first applying a scene completion approach to the input partial scan, followed by a 3D instance segmentation method. For 3D instance segmentation, we evaluate 3D-SIS [9], which achieves state-of-the-art performance on a dense volumetric grid representation (the representation we use), and for

	3D Detection	Instance Segmentation
3D-SIS [9]	24.70	20.61
Ours (no compl)	29.80	23.86
Ours (no color)	31.75	31.59
Ours (no proxy)	34.05	32.59
Ours	**37.81**	**36.28**

Table 4.4: 3D Detection and Instance Segmentation on synthetic SUNCG [17] scans at mAP@0.5. To demonstrate the benefits of instance completion task for instance segmentation and 3D detection, we evaluate our semantic instance completion approach on the task of instance segmentation and 3D detection. Predicting instance completion notably benefits 3D detection and instance segmentation (Ours vs. no compl).

scan completion we evaluate the 3D-EPN [85] shape completion approach and ScanComplete [59] scene completion approach. Our end-to-end approach for semantic instance completion results in significantly improved performance due to information flow from instance completion to object detection. For instance, this allows our instance completion to more easily adapt to some inaccuracies in detection, which strongly hinders a decoupled approach. Note that the ScanComplete model applied on ScanNet data is trained on synthetic data, due to the lack of complete ground truth scene data (Scan2CAD provides only object ground truth) for real-world scans.

Does instance completion help instance detection and segmentation? We can also evaluate our semantic instance completion predictions on the task of semantic instance segmentation by taking the intersection between the predicted complete mask and the input partial scan geometry to be the predicted instance segmentation mask. We show that predicting instance completion helps instance segmentation, evaluating our method on 3D semantic instance segmentation with and without completion, on ScanNet [4] and SUNCG [17] scans in Tables 4.3 and 4.4, as well as 3D-SIS [9], an approach jointly predicts 3D detection and instance segmentation, which also operates on dense volumetric data, achieving state-of-the-art performance on this representation. We find that predicting instance completion significantly benefits instance segmentation, due to a more unified understanding of object geometric structures.

Additionally, we evaluate the effect on 3D detection in Tables 4.3 and 4.4; predicting instance completion also significantly improves 3D detection performance. Note that in contrast to 3D-SIS [9] which uses separate backbones for detection and instance segmentation, our unified backbone helps 3D mask information (complete or non-complete) propagate through detection parameters to improve 3D detection performance.

What is the effect of a global completion proxy? In Tables 4.1 and 4.2, we demonstrate the impact of the geometric completion proxy loss; here, we see that this loss improves the semantic instance completion performance on both real and synthetic data. In

Tables 4.3 and 4.4, we can see that it also improves 3D detection and semantic instance segmentation performance.

Can color input help? Our approach takes as input the 3D scan geometry as a TSDF as well as the corresponding color images. We evaluate our approach with and without the color input stream; on both real and synthetic scans, the color input notably improves semantic instance completion performance, as shown in Tables 4.1 and 4.2.

4.7 Network Architecture

In this section, we detail our RevealNet network architecture in Section 4.7.

small anchors	big anchors
(9, 10, 9)	(47, 20, 23)
(17, 21, 17)	(23, 20, 47)
(12, 19, 13)	(16, 18, 30)
(16, 12, 15)	(17, 38, 17)
	(30, 18, 16)

Table 4.5: Anchor sizes used for region proposal on the ScanNet dataset [4]. Sizes are given in voxel units, with voxel resolution of ≈ 4.69cm

small anchors	big anchors
(8, 6, 8)	(12, 12, 40)
(22, 22, 16)	(8 , 60, 40)
(12, 12, 20)	(38, 12, 16)
	(62, 8 , 40)
	(46, 8 , 20)
	(46, 44, 20)
	(14, 38, 16)

Table 4.6: Anchor sizes (in voxels) used for SUNCG [17] region proposal. Sizes are given in voxel units, with voxel resolution of ≈ 4.69cm

Table 5.16 details the layers used in our backbone. 3D-RPN, classification head, and mask completion head are described in Table 4.11. Additionally, we leverage the residual blocks in our backbone, which is listed in Table 4.9. Note that both the backbone and mask completion head are fully-convolutional. For the classification head, we use several fully-connected layers; however, we leverage 3D RoI-pooling on its input, so that we can run our method on large 3D scans of varying sizes in a single forward pass [9].

We additionally list the anchors used for the region proposal for our model trained on our ScanNet-based semantic instance completion benchmark [4], [25] and SUNCG [17] datasets in Tables 4.5 and 4.6, respectively. Anchors for each dataset are determined through k-means clustering of ground truth bounding boxes. The anchor sizes are given in voxels, where our voxel size is ≈ 4.69cm.

4.8 Inference Timing

physical size (m)	5.8 x 6.4	8.3 x 13.9	10.9 x 20.1
voxel resolution	124 x 136	176 x 296	232 x 428
forward pass (s)	0.15	0.37	0.72

Table 4.7: Inference time on entire scenes without color signal. Timings are given in seconds, physical sizes are given in meters and spatial sizes are given in voxel units, with voxel resolution of ≈ 4.69cm

.

In this section, we present the inference timing with and without color projection in Table 4.7 and 4.8. Note that our color projection layer currently projects the color signal into 3D space sequentially, and can be further optimized using CUDA, so that it can project the color features back to 3D space in parallel. A scan typically contains several hundreds of images; hence, this optimization could significantly further improve inference time.

physical size (m)	4.7 x 7.7	7.9 x 9.6	10.7 x 16.5
voxel resolution	100 x 164	168 x 204	228 x 352
image count	49	107	121
color projection (s)	1.43	5.16	11.78
forward pass (s)	0.19	0.34	0.64
total (s)	1.62	5.50	12.42

Table 4.8: Inference timing on entire large scans with RGB input. Timings are given in seconds, physical sizes are given in meters and spatial sizes are given in voxel units, with voxel resolution of ≈ 4.69cm

.

4.9 Model-fitting vs. Prediction-based Methods

In terms of object level completion in the RGB-D scan, We discuss about differences between model-fitting and prediction-based methods. Regarding model-fitting methods, *“Scan2CAD”* [25] and *“End-to-End CAD”* [88] define a CAD alignment task for which they require a pre-defined set of CAD models for each test scene; i.e., GT objects are given at test time and only alignment needs to be inferred (cf. Scan2CAD benchmark). Prediction-based method method, e.g. RevealNet, does not have the GT objects as input in the test time.

Semantic instance completion is fundamentally more flexible as it operates on a per-voxel basis (vs. fixed CAD models); i.e., allowing construction of much more true-to-observation geometry (e.g., necessary in the context of robotics). Since prediction-based methods complete the missing surface based on observed geometry that performs as an anchor, RevealNet easily overlaps with the ground truth surface, whereas model-fitting methods, which predicts rotation/scale/translation, could have little overlap with the ground truth surface due to slight misalignment (e.g. 10 degrees rotation error). In addition, we want to highlight that CAD model alignment does not help their detection performance, but instance completion in RevealNet does.

ResBlock	Input Layer	Type	Input Size	Output Size	Kernel Size	Stride	Padding
convres0	CNN feature	Conv3d	(N,X,Y,Z)	(N/2,X,Y,Z)	(1,1,1)	(1,1,1)	(0,0,0)
normres0	convres0	InstanceNorm3d	(N/2,X,Y,Z)	(N/2,X,Y,Z)	None	None	None
relures0	normres0	ReLU	(N/2,X,Y,Z)	(N/2,X,Y,Z)	None	None	None
convres1	relures0	Conv3d	(N/2,X,Y,Z)	(N/2,X,Y,Z)	(3,3,3)	(1,1,1)	(1,1,1)
normres1	convres1	InstanceNorm3d	(N/2,X,Y,Z)	(N/2,X,Y,Z)	None	None	None
relures1	normres1	ReLU	(N/2,X,Y,Z)	(N/2,X,Y,Z)	None	None	None
convres2	relures1	Conv3d	(N/2,X,Y,Z)	(N,X,Y,Z)	(1,1,1)	(1,1,1)	(0,0,0)
normres2	convres2	InstanceNorm3d	(N,X,Y,Z)	(N,X,Y,Z)	None	None	None
relures2	normres2	ReLU	(N,X,Y,Z)	(N,X,Y,Z)	None	None	None

Table 4.9: Network Details. Residual block specification in RevealNet.

4.10 Performance Study over Degrees of Completeness

Both our SUNCG and ScanNet settings have a large variety of incompleteness. We show a histogram in Fig. 4.5 of our current results on ScanNet (numbers split from Tab. 1 main paper). From the histogram, We show that with more complete geometry input, semantic instance completion task becomes easier.

4.11 Limitations

Our approach shows significant potential in the task of semantic instance completion, but several important limitations still remain. First, we output a binary mask for the complete object geometry, which can limit the amount of detail represented by the completion; other 3D representations such as distance fields or sparse 3D representations [20] could potentially resolve greater geometric detail. Our approach also uses axis-aligned bounding boxes for object detection; it would be helpful to additionally predict the object orientation. We also do not consider object movement over time, which contains significant opportunities for semantic instance completion in the context of dynamic environments.

4.12 Conclusion

In this paper, we tackle the problem of "seeing behind objects" by predicting the missing geometry of individual objects in RGB-D scans. This opens up many possibilities for complex interactions with objects in 3D, for instance for efficient navigation or robotic grasping. To this end, we introduced the new task of semantic instance completion

BackBone	Input Layer	Type	Input Size	Output Size	Kernel Size	Stride	Padding
geometry0	TSDF	Conv3d	(2,96,48,96)	(32,48,24,48)	(2,2,2)	(2,2,2)	(0,0,0)
norm0	geometry0	InstanceNorm3d	(32,48,24,48)	(32,48,24,48)	None	None	None
relu0	norm0	ReLU	(32,48,24,48)	(32,48,24,48)	None	None	None
block0	relu0	ResBlock	(32,48,24,48)	(32,48,24,48)	None	None	None
color1	CNN feature	Conv3d	(128,96,48,96)	(32,48,24,48)	(2,2,2)	(2,2,2)	(0,0,0)
norm1	color1	InstanceNorm3d	(32,48,24,48)	(32,48,24,48)	None	None	None
relu1	norm1	ReLU	(32,48,24,48)	(32,48,24,48)	None	None	None
block1	relu1	ResBlock	(32,48,24,48)	(32,48,24,48)	None	None	None
concat2	(block0,block1)	Concatenate	(32,48,24,48)	(64,48,24,48)	None	None	None
combine2	concat2	Conv3d	(64,48,24,48)	(128,24,12,24)	(2,2,2)	(2,2,2)	(0,0,0)
norm2	combine2	InstanceNorm3d	(128,24,12,24)	(128,24,12,24)	None	None	None
relu2	norm2	ReLU	(128,24,12,24)	(128,24,12,24)	None	None	None
block2	relu2	ResBlock	(128,24,12,24)	(128,24,12,24)	None	None	None
encoder3	block2	Conv3d	(128,24,12,24)	(128,24,12,24)	(3,3,3)	(1,1,1)	(1,1,1)
norm3	combine3	InstanceNorm3d	(128,24,12,24)	(128,24,12,24)	None	None	None
relu3	norm3	ReLU	(128,24,12,24)	(128,24,12,24)	None	None	None
block3	relu3	ResBlock	(128,24,12,24)	(128,24,12,24)	None	None	None
skip4	(block, block3)	Conv3d	(128,24,12,24)	(64,48,24,48)	(2,2,2)	(2,2,2)	(0,0,0)
norm4	combine4	InstanceNorm3d	(64,48,24,48)	(64,48,24,48)	None	None	None
relu4	norm4	ReLU	(64,48,24,48)	(64,48,24,48)	None	None	None
block4	relu4	ResBlock	(64,48,24,48)	(64,48,24,48)	None	None	None
concat5	(block2,block4)	Concatenate	(64,48,24,48)	(128,48,24,48)	None	None	None
decoder5	block5	ConvTranspose3d	(128,48,24,48)	(32,96,48,96)	(2,2,2)	(2,2,2)	(0,0,0)
norm5	combine5	InstanceNorm3d	(32,96,48,96)	(32,96,48,96)	None	None	None
relu5	norm5	ReLU	(32,96,48,96)	(32,96,48,96)	None	None	None
block5	relu5	ResBlock	(32,96,48,96)	(32,96,48,96)	None	None	None
proxy5	block5	ConvTranspose3d	(32,96,48,96)	(1,96,48,96)	(1, 1, 1)	(1,1,1)	(0,0,0)

Table 4.10: Network Architecture. Backbone layer specifications in RevealNet.

along with RevealNet, a new 3D CNN-based approach to jointly detect objects and predict their complete geometry. Our proposed 3D CNN learns from both color and geometry features to detect and classify objects, then predicts the voxel occupancy for the complete geometry of the object in an end-to-end fashion, which can be run on a full 3D scan in a single forward pass. On both real and synthetic scan data, we significantly outperform state-of-the-art approaches for semantic instance completion. We believe that our approach makes an important step towards higher-level scene understanding and helps to enable object-based interactions and understanding of scenes, which we hope will open up new research avenues.

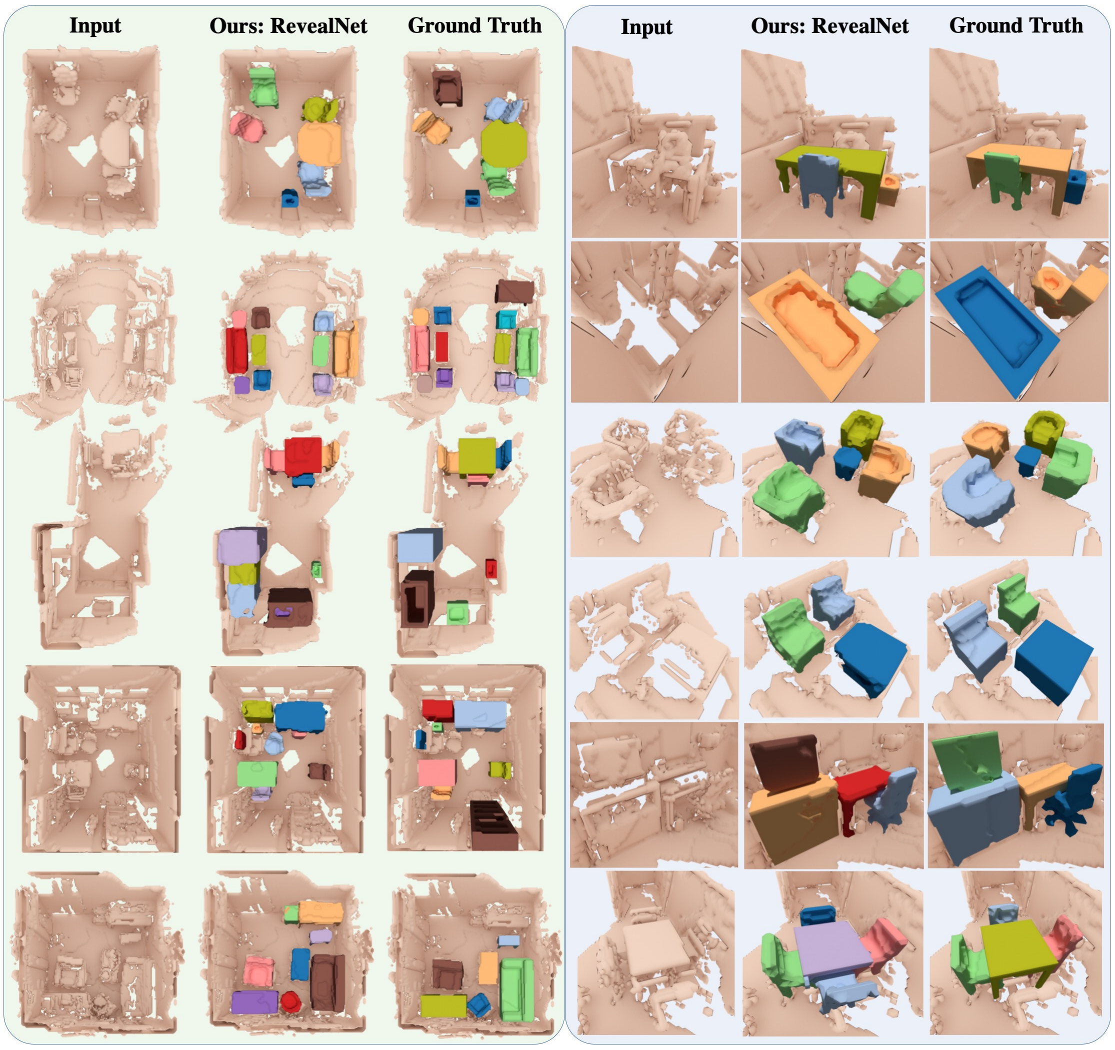


Figure 4.3: Qualitative results on real-world ScanNet [4] scenes with Scan2CAD [25] targets. Close-ups are shown on the right. Note that different colors denote distinct object instances in the visualization. Our approach effectively predicts complete individual object geometry, including missing structural components (e.g., missing chair legs), across varying degrees of partialness in input scan observations.

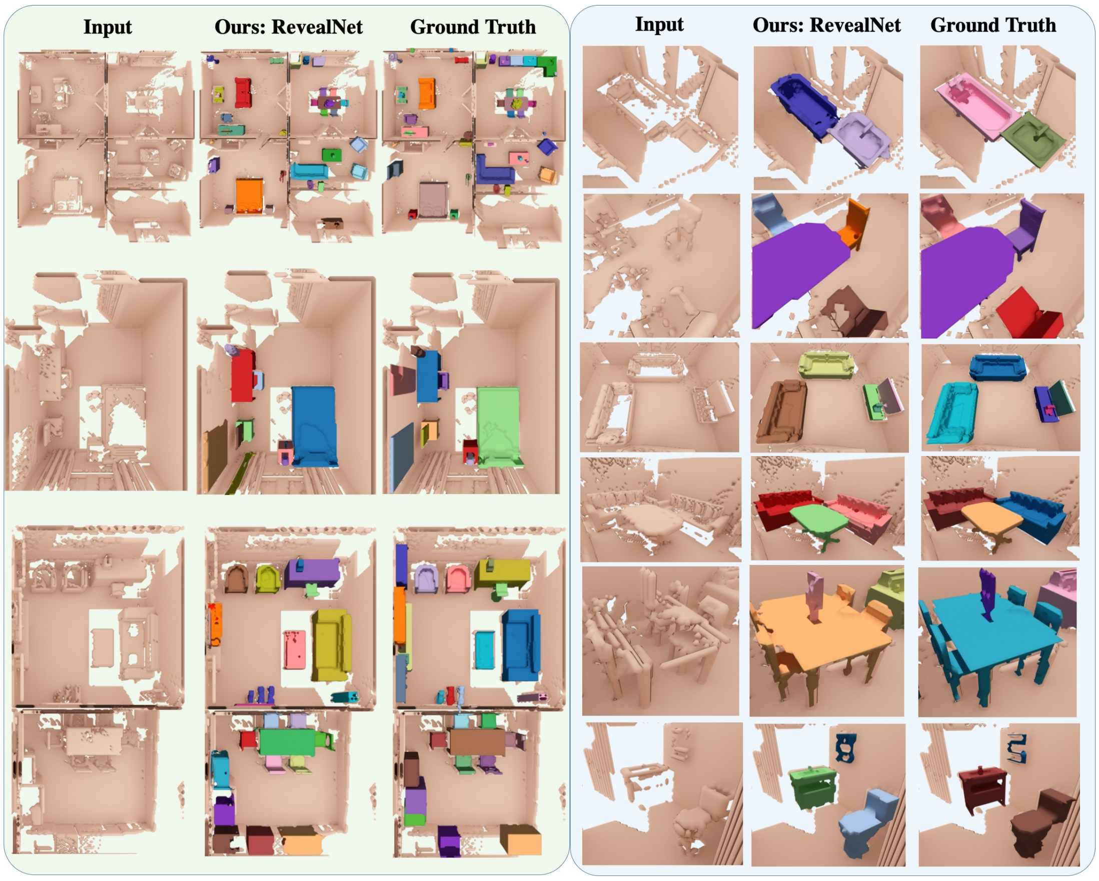


Figure 4.4: Qualitative results on SUNCG dataset [17] (left: full scans, right: close-ups). We sample RGB-D images to reconstruct incomplete 3D scans from random camera trajectories inside SUNCG scenes. Note that different colors denote distinct object instances in the visualization.

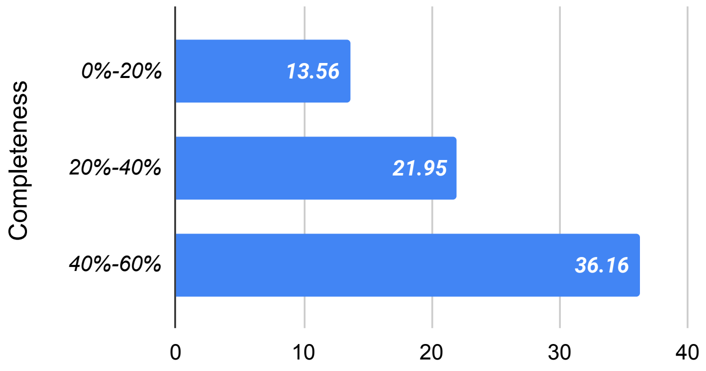


Figure 4.5: Results of Tab. 1 split by levels of object completeness in mAP@0.5. More complete input is easier.

RPN	Input Layer	Type	Input Size	Output Size	Kernel Size	Stride	Padding
rpn6	block2	Conv3d	(128,24,12,24)	(256,24,12,24)	(3,3,3)	(1,1,1)	(1,1,1)
norm6	rpn6	InstanceNorm3d	(256,24,12,24)	(256,24,12,24)	None	None	None
relu6	norm6	ReLU	(256,24,12,24)	(256,24,12,24)	None	None	None
rpncls7a	relu6	Conv3d	(256,24,12,24)	(8,24,12,24)	(1,1,1)	(1,1,1)	(0,0,0)
norm7a	rpncls7a	InstanceNorm3d	(8,24,12,24)	(8,24,12,24)	None	None	None
rpnbbox7b	relu6	Conv3d	(24,24,12,24)	(24,24,12,24)	(1,1,1)	(1,1,1)	(0,0,0)
norm7b	rpnbbox7b	InstanceNorm3d	(24,24,12,24)	(24,24,12,24)	None	None	None
rpn8	block3	Conv3d	(128,24,12,24)	(256,24,12,24)	(3,3,3)	(1,1,1)	(1,1,1)
norm8	rpn8	InstanceNorm3d	(256,24,12,24)	(256,24,12,24)	None	None	None
relu8	norm8	ReLU	(256,24,12,24)	(256,24,12,24)	None	None	None
rpncls9a	relu8	Conv3d	(256,24,12,24)	(8,24,12,24)	(1,1,1)	(1,1,1)	(0,0,0)
norm9a	rpncls9a	InstanceNorm3d	(10,24,12,24)	(10,24,12,24)	None	None	None
rpnbbox9b	relu8	Conv3d	(30,24,12,24)	(30,24,12,24)	(1,1,1)	(1,1,1)	(0,0,0)
norm9b	rpnbbox9b	InstanceNorm3d	(30,24,12,24)	(30,24,12,24)	None	None	None
Class Head	Input Layer	Type	Input Size	Output Size	Kernel Size	Stride	Padding
roipool10	block2/block3	RoI Pooling	(64,arbitrary)	(64, 4, 4, 4)	None	None	None
flat10	roipool10	Flat	(64,4,4,4)	(4096)	None	None	None
cls10a	flat10	Linear	(4096)	(256)	None	None	None
relu10a	cls10a	ReLU	(256)	(256)	None	None	None
cls10b	relu10a	Linear	(256)	(128)	None	None	None
relu10b	cls10b	ReLU	(128)	(128)	None	None	None
cls10c	relu10b	Linear	(128)	(128)	None	None	None
relu10c	cls10c	ReLU	(128)	(128)	None	None	None
clscls10	relu10c	Linear	(128)	(8)	None	None	None
clsbbox10	relu10c	Linear	(128)	(48)	None	None	None
Mask Head	Input Layer	Type	Input Size	Output Size	Kernel Size	Stride	Padding
mask11	block2/block3	Conv3d	(N,arbitrary)	(N,arbitrary)	(9,9,9)	(1,1,1)	(4,4,4)
norm11	mask11	InstanceNorm3d	(N,arbitrary)	(N,arbitrary)	None	None	None
relu11	norm11	ReLU	(N,arbitrary)	(64,arbitrary)	None	None	None
mask12	relu11	Conv3d	(N,arbitrary)	(N,arbitrary)	(7,7,7)	(1,1,1)	(3,3,3)
norm12	mask12	InstanceNorm3d	(N,arbitrary)	(N,arbitrary)	None	None	None
relu12	norm12	ReLU	(N,arbitrary)	(64,arbitrary)	None	None	None
mask13	relu12	Conv3d	(N,arbitrary)	(N,arbitrary)	(5,5,5)	(1,1,1)	(2,2,2)
norm13	mask13	InstanceNorm3d	(N,arbitrary)	(N,arbitrary)	None	None	None
relu13	norm13	ReLU	(N,arbitrary)	(64,arbitrary)	None	None	None
mask14	relu13	Conv3d	(N,arbitrary)	(N,arbitrary)	(3,3,3)	(1,1,1)	(1,1,1)
norm14	mask14	InstanceNorm3d	(N,arbitrary)	(N,arbitrary)	None	None	None
relu14	norm14	ReLU	(N,arbitrary)	(64,arbitrary)	None	None	None
mask15	relu14	Conv3d	(N,arbitrary)	(N,arbitrary)	(1,1,1)	(1,1,1)	(0,0,0)

Table 4.11: Network Structures. Head layer specifications of RPN, Classification and Mask Completion in RevealNet.

5 Data-Efficient 3D Scene Understanding

This chapter introduces the following paper:

J. Hou, B. Graham, M. Nießner, and S. Xie, “Exploring Data-Efficient 3D Scene Understanding with Contrastive Scene Contexts,” in *Proceedings of Computer Vision and Pattern Recognition (CVPR), IEEE*, 2021

Abstract of paper The rapid progress in 3D scene understanding has come with growing demand for data; however, collecting and annotating 3D scenes (e.g. point clouds) are notoriously hard. For example, the number of scenes (e.g. indoor rooms) that can be accessed and scanned might be limited; even given sufficient data, acquiring 3D labels (e.g. instance masks) requires intensive human labor. In this paper, we explore data-efficient learning for 3D point cloud. As a first step towards this direction, we propose Contrastive Scene Contexts, a 3D pre-training method that makes use of both point-level correspondences and spatial contexts in a scene. Our method achieves state-of-the-art results on a suite of benchmarks where training data or labels are scarce. Our study reveals that exhaustive labelling of 3D point clouds might be unnecessary; and remarkably, on ScanNet, even using 0.1% of point labels, we still achieve 89% (instance segmentation) and 96% (semantic segmentation) of the baseline performance that uses full annotations.

Contribution The method development and implementation was done by the first author. Discussions with the co-authors led to the final paper.

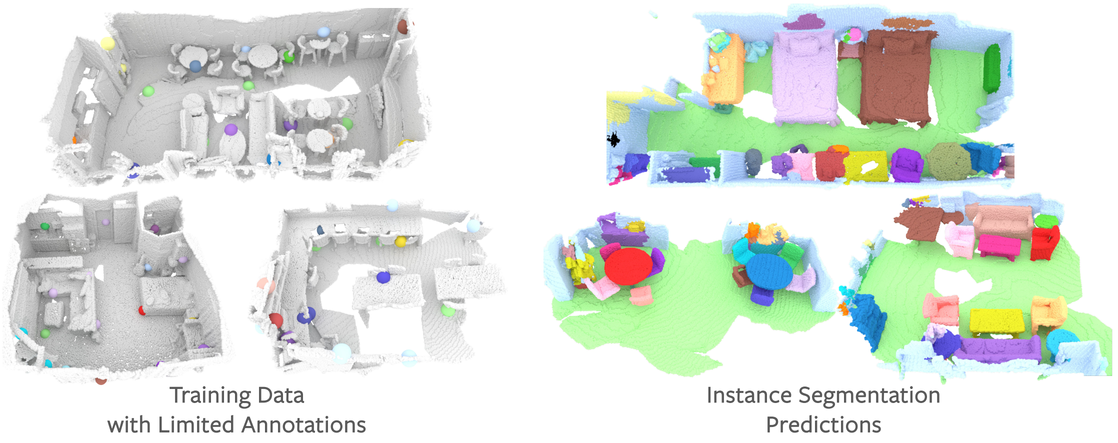


Figure 5.1: Limited Annotations Overview. How many point labels are necessary to train a 3D instance segmentation model on point clouds? It turns out not too many! With the help of unsupervised pre-training, only 20 labelled points per scene (*less than 0.1% of the total points*) are used to fine-tune an instance segmentation model on ScanNet. **Left**: Train samples; only colored points (enlarged for better visibility) are labeled. **Right**: Predictions in validation set and different colors represent different instances.

5.1 Introduction

Recent advances in deep learning on point clouds, such as those obtained from LiDAR or depth sensors, together with a proliferation of public, annotated datasets [4], [24], [27], [89]–[93], have led to swift progress in 3D scene understanding. However, compared to large-scale 2D scene understanding on images [7], [8], [94], the scale of 3D scene understanding—in terms of the amount and diversity of data and annotations, the model size, the number of semantic categories, and so on—still falls behind. We argue that one major bottleneck is the fact that collecting and annotating diverse 3D scenes are significantly more expensive. Unlike 2D images that comfortably exists on the Internet, collecting real world 3D scene datasets usually involves traversing the environment in real life and scanning with 3D sensors. Therefore, the number of indoor scenes that can be scanned might be limited. What is more concerning is that, even given sufficient data acquisition, 3D semantic labelling (e.g. bounding boxes and instance masks) requires complex pipelines [4] and labor-intensive human effort.

In this work, we explore a new learning task in 3D, i.e. data-efficient 3D scene understanding, which focuses on the problem of learning with limited data or supervision[1]. We note that the importance of data-efficient learning in 3D is two-fold. One concerns the status quo: given limited data we have right now, *can we design better methods that perform better?* The other one is more forward-looking: *is it possible to reduce the*

[1]Sometimes a distinction is drawn between *data-efficiency* and *label-efficiency*, to separate the scenarios of limited amount of data samples and limited supervision; here, we use *data-efficiency* to encompass both cases.

human labor for annotation, with a goal of creating 3D scene datasets on a much larger scale?

To formally study the problem, we first introduce a suite of scene understanding benchmarks that encompasses two complementary settings for data-efficient learning: (1) *limited scene reconstructions (LR)* and (2) *limited annotations (LA).* The first setting concerns the scenario where the bottleneck is the number of scenes that can be scanned and reconstructed. The second one focuses on the case where in each scene, the budget for labeling is constrained (e.g. one can only label a small set of points). For each setting, the evaluation is done on a diverse set of scene understanding tasks including object detection, semantic segmentation and instance segmentation.

For data-efficient learning in 2D [95], representation learning, e.g. pre-training on a rich source set and fine-tuning on a much smaller target set, often comes to the rescue; in 3D, representation learning for data-efficient learning is even more wanted but long overdue. With this perspective, we focus on studying data-efficient 3D scene understanding through the lens of representation learning.

Only recently, PointContrast [35] demonstrates that network weights pre-trained on 3D partial frames can lead to a performance boost when fine-tuned on 3D semantic segmentation and object detection tasks. Our work is inspired by PointContrast. However, we observe that the simple contrastive-learning based pretext task used in [35] only concerns point-level correspondence matching, which completely disregards the spatial configurations and contexts in a scene. In Section 5.3, we show that this design limits the scalibility and transferability; we further propose an approach that integrates the spatial information into the contrastive learning framework. The simple modification can significantly improve the performance over PointContrast, especially on complex tasks such as instance segmentation.

Our exploration in data-efficient 3D scene understanding provides some surprising observations. For example, on ScanNet, even using 0.1% of point labels, we are still able to recover 89% (instance segmentation) and 96% (semantic segmentation) of the baseline performance that uses full annotations. The results imply that exhaustive labelling of 3D point clouds might not be necessary. In both scenarios of *limited scene reconstructions (LR)* and *limited annotations (LA)*, our pre-trained network, when used as the initialization for supervised fine-tuning, offers consistent improvement across multiple tasks and datasets. In the scenario of *LA*, we also show that an active labeling strategy can be enabled by clustering the pre-trained point features.

In summary, the contributions of our work include:

- A systematic study on data-efficient 3D scene understanding with a comprehensive suite of benchmarks.
- A new 3D pre-training method that can gracefully transfer to complex tasks such as instance segmentation and outperform the state-of-the-art results.
- Given the pre-trained network, we study practical solutions for data-efficient learning in 3D through fine-tuning as well as an active labeling strategy.

5.2 Related Work

3D Scene Understanding. Research in deep learning on 3D point clouds have been recently shifted from synthetic, single object classification [21], [55], [96] to the challenge of large-scale, real-world scene understanding. A variety of datasets [4], [26], [27], [93], [97] and algorithms have been proposed for 3D object detection [32], [98]–[100], semantic segmentation [21], [28], [101]–[104] and instance segmentation **hou20193d**, [10], [30], [31], [63]–[65], [105]–[107]. In the past year, sparse convolutional networks [28], [104] stand out as a promising approach to standardize deep learning for point clouds, due to its computational efficiency and state-of-the-art performance for 3D scene understanding tasks [28], [30], [108]. In this work, we also adopt a sparse U-Net [109] backbone for our exploration.

3D Representation Learning. Compared to 2D vision, the limits of big data are far from being fully explored in 3D. In 2D representation learning, for example, transfer learning from a rich source data (e.g. ImageNet [7]) to a (typically smaller) target data, has become a dominant framework for many applications [110]. In contrast, 3D representation learning has not been widely adopted and most 3D networks are trained from scratch on the target data directly. Recently, unsupervised pre-training has made great progress and drawn significant attention in 2D [95], [111]–[120]. Following suit, recent works attempt to adapt the 2D pretext tasks to 3D, but mostly focus on single object classification tasks on ShapeNet [121]–[129]. Our work is mostly inspired by a recent contrastive-learning based method PointContrast [35], which first demonstrates the effectiveness of unsupervised pre-training on a diverse set of scene-level understanding tasks. As we will show in the later sections, the simple point-level pre-training objective in PointContrast ignores the spatial contexts of the scene (such as relative poses of objects, and distances between them) which limits its transferability for complex tasks such as instance segmentation. PointContrast also focuses on downstream tasks with 100% data and labels, while we systematically explore a new data-efficient paradigm that has practical importance.

Data-Efficient Learning. Data-efficient learning concerns the problem of learning with limited training examples or labels. This capability is known in cognitive science to be a distinctive characteristic of humans [130]. In contrast, training deep neural networks is not naturally data-efficient, as it typically relies on large amount of annotated data. Among many potential solutions towards this goal, representation learning (commonly through transfer learning) is arguably the most promising one. A good representation "*entangles the different explanatory factors of variation behind the data*" [131] and thus makes the downstream prediction easier (and less data-hungry). This concept has been validated successfully in natural language processing [132] and to some extent in 2D image classification [95]. Pursuing this direction in 3D is even more desirable, considering the potential benefit in reducing the labor of data collection and annotation. Existing work focuses on mostly single CAD model classification or part segmentation [35], [127], [133]–[137]. To the best of our knowledge, our work is the first to explore data-efficient learning in a real-world, large-scale 3D scene understanding setup.

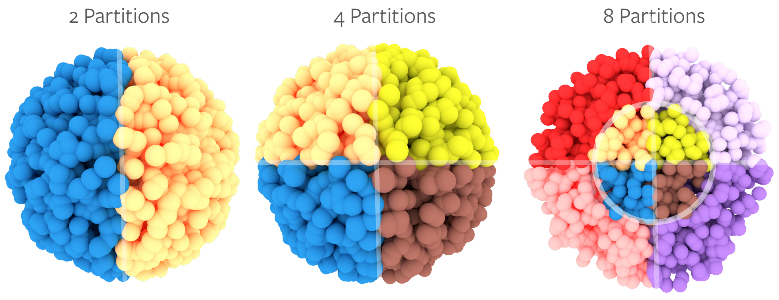


Figure 5.2: Illustration of Scene Contexts. We visualize the 2,4 and 8 spatial partitions for Scene Contexts. The anchor point is in the center. For 2 and 4 partitions, only relative angles are sufficient. For 8 partitions (a cross-section is shown), both relative angles and distances are needed.

5.3 Contrastive Scene Contexts for Pre-training

In this section, we first briefly revisit the PointContrast framework [35], and discuss the shortcomings and remedies. We then introduce our pre-training algorithm.

Revisiting PointContrast. The pre-training objective for PointContrast is to achieve point *equivariance* with respect to a set of random geometric transformations. Given a pair of overlapping partial scans, a contrastive loss for pre-training is defined over the point features. The objective is to minimizes the distance for matched points (positive pairs) and maximize the distance between unmatched ones (negative pairs). Despite the fact that strong spatial contexts exist among objects in a scene, this objective does not capture any of the spatial information: the negative pairs could be sampled from arbitrary locations across many scenes in a mini-batch. We hypothesize that this leads to some limitations: 1) the spatial contexts (e.g. relative pose, direction and distance), which could be pivotal for complex tasks such as instance segmentation, are entirely discarded from pre-training; 2) the scalibility of contrastive learning might be hampered; PointContrast cannot utilize a large number of negative points, potentially because that contrasting a pair of spatially distant and unrelated points would contribute little to learning. In fact, PointContrast uses only a random sampling of 1024 points per scene for pre-training, and it has been shown that results do not improve with more sampled points [35]. We also confirm this behavior with experiments later this section.

Contrastive Scene Contexts. We hope to integrate spatial contexts into the pre-training objective. There are many ways to achieve the goal, and here we take inspiration from the classic *ShapeContext* local descriptor [138]–[140] for shape matching. The *ShapeContext* descriptor partitions the space into spatially inhomogeneous cells, and encodes the spatial contexts about the shape at each point by computing a histogram over the number of neighboring points in each cell. We call our method *Contrastive Scene Contexts* because at a high level, our method also aims to capture *the distribution over relative locations in a scene.* We partition the scene point cloud into multiple regions, and instead of having a single contrastive loss for the entire point set sampled in a mini-batch, we perform contrastive learning in each region separately, and aggregate the losses in the end.

Concretely, given a pair of partial frame point clouds $\mathbf{x}$ and $\mathbf{y}$ from the same scene, we have correspondence mapping $(i, j) \in M_{\mathbf{xy}}$ available, where i is the index of a point $\mathbf{x}_i \in \mathcal{R}^3$ in frame $\mathbf{x}$ and j is the index of a matched point $\mathbf{y}_j \in \mathcal{R}^3$ in frame $\mathbf{y}$. Similar to PointContrast, we sample N pairs of matched points as positives. However, in our method, for each anchor point $\mathbf{x}_i$, the space is divided into multiple partitions and other points are assigned to different partitions based on their relative angles and distances to i.

The distance and angle information needed for scene context partition at anchor point $\mathbf{x}_i$ is as follows,

$$\mathcal{D}_{ik} = \sqrt{\sum_{d=1}^{3} (\mathbf{x}_i^d - \mathbf{x}_k^d)^2} \tag{5.1}$$

$$\mathcal{A}_{ik} = arctan2(\mathcal{D}_{ik}) + 2\pi \tag{5.2}$$

where $\mathcal{D}$ is the relative distance matrix. $\mathcal{D}_{ik}$ stores the distance between point i and point k and $\mathcal{A}$ is the relative angle matrix, where A_{ik} stores the relative angle between point i and point k. In Equation (1) d represents the 3D dimension. With $\mathcal{D}$ and $\mathcal{A}$, a *ShapeContext*-like spatial partitioning function can be easily constructed on-the-fly. In Figure 5.2, we show a visual illustration of how the space partitioning works. Computing 2 or 4 partitions only requires cutting the space according to relative angles based on $\mathcal{A}$; while the 8 or more partitions also require the extent of the inner regions using $\mathcal{D}$. We always partition the space uniformly along the relative angles and distances. Note that the partitioning is *relative to the anchor point i*.

Suppose there are P partitions, we denote the spatial partition functions as $par_p(\cdot)$, where $p \in \{1, \ldots, P\}$. Function $par_p(\cdot)$ takes the anchor point i as input, and return a set of points as negatives. A PointInfoNCE loss $\mathcal{L}_p$ is independently computed for each partition:

$$\mathcal{L}_p = -\sum_{(i,j)\in M} \log \frac{\exp(\mathbf{f}_i^1 \cdot \mathbf{f}_j^2/\tau)}{\sum_{(\cdot,k)\in M, k\in par_p(i)} \exp(\mathbf{f}_i^1 \cdot \mathbf{f}_k^2/\tau)} \tag{5.3}$$

Details of Equation (3) and other implementation details can be found in Appendix. The final loss is computed by aggregating all partitions $\mathcal{L} = \frac{1}{|P|}\sum_p \mathcal{L}_p$.

Analysis. We first show that by integrating the scene contexts into the objective, our pre-training method can benefit more from a larger point set. We conduct an analysis experiment by varying the number of scene context partitions and the number of points sampled for computing the contrastive loss. We pre-train our model for a short schedule (20K iters). We then fine-tune the pre-trained weights on S3DIS instance segmentation benchmark [27]. Results are shown in Figure 5.3, the green line represents a variant with *no spatial partitioning*; the left-most point represents PointContrast[2]. Similar to the observation in [35], without scene contexts, increasing the number of sampled points

[2]Not exactly identical since the matched points are sampled per scene in this experiment, rather than from the whole mini-batch as in PointContrast; we have verified that this nuance does not influence the conclusion.

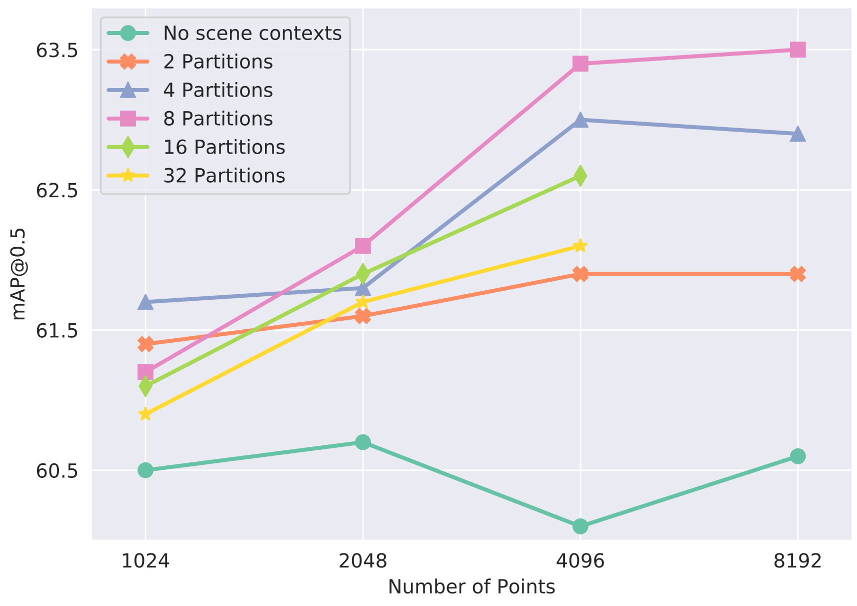


Figure 5.3: Analysis Experiment. Varying the number of partitions and sampled points for pre-training; Results are reported on the S3DIS instance segmentation task [27]. Using scene context partitions has enabled constrastive learning to utilize more points for better performance.

does not improve the performance; with more partitions, increasing # sampled points leads to a consistent boost in performance (up to 4096 points). We use 8 partitions as empirically it works best. This shows that our method leads to better *scalability* as more points can be utilized for pre-training.

We achieve state-of-the-art instance segmentation results in terms of mAP@0.5 (Table. 5.1) using a simple bottom-up clustering mechanism with voting loss (details in Appendix). We do not use any special modules such as Proposal Aggregation [106] or Scoring Network [30]. We observe a 2.9% absolute improvement over PointContrast pre-training, which brings the improvement over train-from-scratch baseline to 4.1%. This substantial margin demonstrates the effectiveness of Contrastive Scene Contexts on instance segmentation tasks. We provide more results comparing against PointContrast in Section 5.5.3.

5.4 Data-Efficient 3D Scene Understanding

To formally explore data-efficient 3D scene understanding, in this section, we propose two different learning paradigms and relevant benchmarks that are associated with two complementary settings that can occur in real world application scenarios: (1) *limited scene reconstructions (LR)* and (2) *limited annotations (LA)*. The first setting mainly

Methods	mAP@0.5
ASIS [105]	55.3
3D-BoNet [64]	57.5
PointGroup [30]	57.8
3D-MPA [106]	63.1
Train from scratch	59.3
PointContrast (PointInfoNCE) [35]	60.5 (+1.2)
Contrastive Scene Contexts	**63.4** (+4.1)

Table 5.1: Fine-tuning results for instance segmentation on S3DIS [27]. A simple clustering-based model with *Contrastive Scene Contexts* pre-trained backbone performs significantly better than the train-from-scratch baseline and PointContrast pre-training [35].

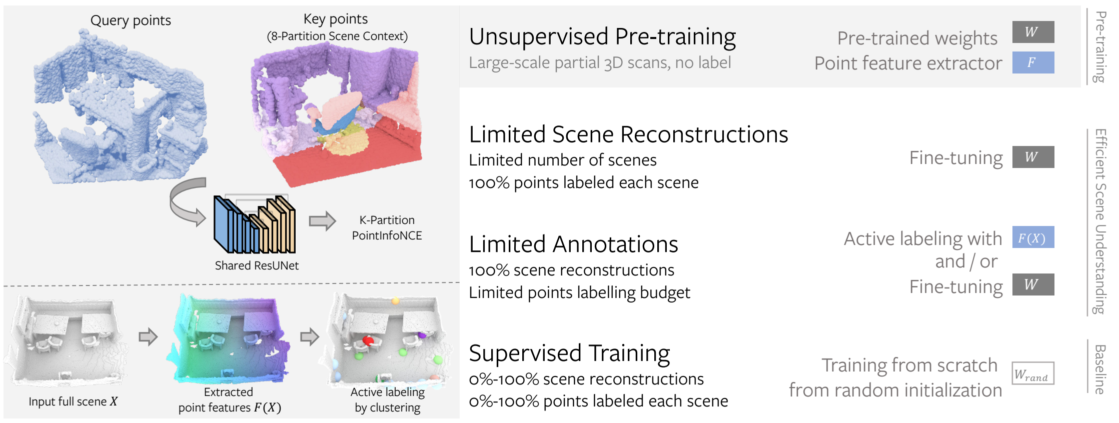

Query points
Key points
(8-Partition Scene Context)
Shared ResUNet
K-Partition
PointInfoNCE
Input full scene X
Extracted
point features $F(X)$
Active labeling
by clustering
Unsupervised Pre-training
Large-scale partial 3D scans, no label
Pre-trained weights W
Point feature extractor F
Pre-training
Limited Scene Reconstructions
Limited number of scenes
100% points labeled each scene
Fine-tuning W
Limited Annotations
100% scene reconstructions
Limited points labelling budget
Active labeling with $F(X)$
and / or
Fine-tuning W
Efficient Scene Understanding
Supervised Training
0%-100% scene reconstructions
0%-100% points labeled each scene
Training from scratch
from random initialization W_{rand}
Baseline

Figure 5.4: Overview of Data-Efficient 3D Scene Understanding. Left: Unsupervised pre-training with *Contrastive Scene Contexts*. The outputs of pre-training are 1) a pre-trained U-Net **F** (that can be used as an offline feature extractor) and 2) its associated weights **W**. **Right**: After pre-training, different learning scenarios can be applied for the downstream tasks such as learning with limited scene reconstructions (LR) or limited annotations (LA). In the case of LR, the pre-trained weights **W** are used as network initialization for fine-tuning. In the case of LA, all the scene reconstructions can be used but only a limited annotation budget is available, e.g. 20 points can be annotated (semantic labels) per scene. Again, **W** can be used as network initialization for fine-tuning; optionally the feature extractor **F** can be used in an active labeling strategy to decide which points to annotate. Baselines are standard supervised learning where models are trained from scratch.

concerns the scenario where the bottleneck of data collection is the *number of scenes* that can be scanned and reconstructed. The second one focuses on the case where in each scene, the budget for labeling is limited (e.g. one can only label a small set of points). Since 3D point labeling is human intensive, this represents a practical scenario where a data-efficient learning strategy can greatly reduce the annotation cost. An overview is presented in Figure 5.4, and details of individual benchmarks are described below.

5.4.1 Limited Annotations (LA)

In this benchmark, we explore 3D scene understanding with a limited budget for point cloud annotations. We consider a diverse set of tasks including semantic segmentation, instance segmentation and object detection. Specifically, for instance segmentation and semantic segmentation, the annotation budget is in terms of the *number of points* for labelling. This is practically useful: if an annotator only needs to label the semantic labels for 20 points, it will only require a few minutes to label a full room. Our benchmark considers four different training configurations on ScanNet including using {20, 50, 100, 200} labeled points per scene. For object detection, the annotation budget is with respect to the *number of bounding boxes* to label in each scene. Our benchmark considers four different training configurations including {1, 2, 4, 7} labeled bounding boxes. Our base dataset is ScanNetV2 [4] which has 1201 scenes for training. We evaluate the model performance on standard ScanNetV2 validation set of 312 scenes that has full labels.

5.4.2 Limited Scene Reconstructions (LR)

For current 3D scene datasets, it is common for annotators to carry commodity depth cameras and record 3D videos at private houses or furniture stores. It might be unrealistic to enter a large number of homes and obtain detailed scanning. In this case, the number of scenes might be the bottleneck and the training has to be done on limited amount of scene reconstructions. We simulate this scenario by random sampling a subset of ScanNetV2 training set. Our benchmark has four configurations {1%, 5%, 10%, 20%} (100% represents the entire ScanNet train set) for semantic segmentation and instance segmentation; and {10%, 20%, 40%, 80%} for object detection. During test time, evaluation is on all scenes in the validation set.

5.5 Experimental Results

In this section, we present our experimental results on the data-efficient 3D scene understanding benchmarks: **ScanNet-LA** with limited annotations and **ScanNet-LR** with limited scene reconstructions. In both scenarios, we compare our method against the baseline of training from scratch, and report results on semantic/instance segmentation and object detection. We also compare our models with the state-of-the-art method in the last part of the section.

Experiments Setup For pre-training, we use SGD optimizer with learning rate 0.1 and a batch-size of 32. The learning rate is decreased by a factor of 0.99 every 1000 steps. The model is trained for 60K steps. The fine-tuning experiments on instance segmentation and semantic segmentation are trained with a batch-size of 48 for a total of 10K steps. The initial learning rate is 0.1, with polynomial decay with power 0.9. For all experiments, we use data parallel on 8 NVIDIA V100 GPUs. For object detection experiments, we fine-tune the model with a batch-size of 32 for 180 epochs. The initial learning rate is set to 0.001 and decayed by a factor of 0.1 at epoch 80, 120 and 160. For all the experiments, we use the same Sparse Res-UNet [35] as the backbone. For both training and testing, the voxel size for Sparse ConvNet is set to 2.0 cm. We use Sparse ConvNet implemented by MinkowskiEngine [28].

5.5.1 Limited Annotations

As introduced in Section 5.4, the *Limited Annotation (LA)* benchmark covers two different annotation types: *Limited Point Annotations* for semantic and instance segmentation and *Limited Bounding Box Annotations* for detection. The pre-trained network (and its weights) can be used as initialization for fine-tuning, or integrate in an active labeling strategy, which we describe below.

Active labeling. Since we focus on the scenario of having limited annotation budget, it is natural to consider an *active learning* strategy during the data annotation process; i.e. one can interactively query an annotator to label some data points that can help most for subsequent training. The core idea of our approach is to perform a **balanced sampling** on the **feature space**, so that the selected points will be the most representative and exemplary ones in a scene. Our pre-trained network extracts dense features at each point of the to-be-annotated point cloud, by simply performing a forward pass. We then perform k-means clustering in this feature space to obtain K cluster centroids. We select the K centroids as the points to be provided to the annotators for labeling. We also present two baseline strategies including a simple **random sampling** strategy where K points are randomly selected to be labeled, and a similar **k-means sampling** strategy on raw (RGB+XYZ) inputs, rather than on the pre-trained features.

We note that although our experiments are simulated based on the already collected ScanNet dataset, our pre-trained feature extractor and the labeling strategy are readily useful in a real-world data annotation pipeline.

Results. In Figure 5.7 we show that compared to the naive from-scratch baselines, our proposed pre-training framework can lead to much improved performance. It is interesting to see that, for both semantic segmentation and instance segmentation, even *without* fine-tuning, the *active labeling* strategy alone provides point labels that make the trained model perform significantly better, compared to *random sampling* or *k-means sampling* baseline strategies, yielding a >10% absolute improvement in terms of mAP@0.5 and mIoU when the training data has only 20 point labels.

The fact that *active labeling* strategy performs on par with the more common pre-training and fine-tuning paradigm, suggests that finding exemplary points to label is crucial for data-efficient learning. Of course, in real applications both active labeling and fine-tuning can be used jointly, and we indeed observe a further (though admittedly smaller) boost in performance by 1) active sampling points to label and then 2) fine-tuning with the pre-trained weights.

Overall, with the help of our Contrastive Scene Contexts pre-training, even using around 0.1% of point labels (e.g. 200 labeled points out of 150K total points per scene), we are still able to achieve 50.4% mAP@0.5 for instance segmentation, and 69.0% mIoU

No. of Boxes	VoteNet (scratch)	VoteNet (ours)
all	35.4	**39.3** (+3.9)
1	9.1	**10.9** (+1.8)
2	15.9	**18.5** (+2.6)
4	22.5	**26.1** (+3.6)
7	26.5	**30.4** (+3.9)

Table 5.2: Object detection results using Limited Bounding Box Annotations on ScanNet. The metric is mAP@0.5. "Ours" denotes the fine-tuning results with our pre-trained model. We list the upper-bound performance using all annotated bounding boxes (in average about 13 bounding boxes per scene) as a reference in the first row.

for semantic segmentation. This indicates a recovery of 89% and 96% of baseline performance that uses 100% of the annotations. We show additional qualitative comparison in Figure 5.5.

Limited Bounding Box Annotations. For object detection, we use VoteNet [32] as the detector framework; follwoing [35], we replace PointNet [21] with our Sparse Res-UNet. For this part, we do not use any active labeling strategy as the labeling cost for bounding boxes are much smaller. We random sample {1, 2, 4, 7} bounding boxes per scene and train the detector. In Table 5.2, we observe that our pre-training also consistently improves over the baseline VoteNet, and the performance gap does not diminish when more box annotations are available.

5.5.2 Limited Scene Reconstructions

In this section, we report the experimental results for another scenario of data-efficient 3D scene understanding, when there is a shortage of scene reconstructions. For instance segmentation and semantic segmentation tasks, we random sample subsets of ScanNet scenes of different sizes. We sample {1%, 5%, 10%, 20%} of the entire 1201 scenes in the training set (which corresponds to 12, 60, 120, and 240 scenes, respectively). For object detection, we find it very difficult to train the detector when the scenes are too scarce. Thus we sample {10%, 20%, 40%, 80%} subsets. For each configuration, we randomly sample 3 subsets and report the averaged results to reduce variance. We also use the official ScanNetV2 validation set for evaluation.

Network fine-tuned with our pre-trained model again shows a clear gap compared to the training from scratch baseline (Table 5.3). We achieve competitive results (50.6% mAP@0.5 for instance segmentation and 64.6% mIoU for semantic segmentation) using only 20% of the total scenes.

Similar behavior can be observed on the object detection task on ScanNet, and the difference between with and without our pre-training is more pronounced in Table 5.4: the detector can barely produce any meaningful results when the data is scarce (e.g. 10% or 20%) and trained from scratch. However, fine-tuning with our pre-trained weights, VoteNet can perform significantly better (e.g. improve the mAP@0.5 by more than 16% with 20% training data).

Data Pct.	Instance Seg.		Semantic Seg.	
	Scratch	Ours	Scratch	Ours
100%	56.9	**59.4** (+2.5)	72.2	**73.8** (+1.6)
1%	9.9	**13.2** (+3.3)	26.0	**28.9** (+2.9)
5%	31.9	**36.3** (+4.4)	47.8	**49.8** (+2.0)
10%	42.7	**44.9** (+2.2)	56.7	**59.4** (+2.7)
20%	48.1	**50.6** (+2.5)	62.9	**64.6** (+1.7)

Table 5.3: 3D semantic and instance segmentation results with Limited Scene Reconstructions (ScanNet-LR). Metric is mAP@0.5 for instance segmentation and mIoU for semantic segmentation. "Scratch" denotes the training from scratch baseline, and "Ours" denotes the fine-tuning results using our pre-trained weights. Results using 100% of the data during training are listed in the first row.

Data Pct.	VoteNet (scratch)	VoteNet (ours)
100%	35.4	**39.3** (+3.9)
10%	0.3	**8.6** (+8.3)
20%	4.6	**20.9** (+16.3)
40%	22.0	**29.2** (+7.2)
80%	33.7	**36.7** (+3.0)

Table 5.4: Object detection results with Limited Scene Reconstructions on ScanNet. Metric is mAP@0.5. We show constantly improved results over training from scratch, especially so when 10% or 20% of the data are available. Results using all scenes are listed in the first row.

5.5.3 Additional Comparisons to PointContrast

As 3D-SIS is closely related to PointContrast [35], we provide additional results in this section, including comparisons on the data-efficient ScanNet benchmarks (Table 5.5) as well as on other datasets and benchmarks (Table 5.6). Our pre-training method outperforms [35] in almost every benchmark setting, sometimes by a big margin. These results further render the importance of integrating scene contexts in contrastive learning. Notably, our pre-training method on S3DIS achieves 72.2% mIoU which outperforms, for the first time, the *supervised* pre-training result reported in [35].

5.5.4 Analysis on Active Labeling: Cluttered Scenes

To better explain our active labeling strategy and show that it can work in scenes with heavy occlusion and clutter, we filter out a ScanNet subset of 200 cluttered scenes that has multiple objects per one square meter area. Compared to naive k-means sampling, active labeling performs even better on cluttered scenes. In Figure 5.8, we visualize a cluttered scene and sampled points (bottom); we also show quantitatively (top) our strategy covers more distinct objects and thus has a balancing effect.

In this supplemental document, we describe the details of our implementation in Section 5.6. We show more visualizations of our models on semantic segmentation and object detection tasks with extremely scarce data for training in Section 5.7. Detailed

Settings	Task (Metric)	SC	PC [35]	Ours
LA (200 points)	ins (mAP@0.5)	43.5	44.5 (+1.0)	**48.9** (+5.4)
LA (200 points)	sem (mIoU)	65.5	67.8 (+2.3)	**68.2** (+2.7)
LA (7 bboxes)	det (mAP@0.5)	33.4	34.9 (+1.5)	**35.9** (+2.5)
LR (240 scenes)	ins (mAP@0.5)	48.1	48.4 (+0.3)	**50.6** (+2.5)
LR (240 scenes)	sem (mIoU)	62.9	63.0 (+0.1)	**64.6** (+1.7)
LR (960 scenes)	det (mAP@0.5)	33.7	36.3 (+2.6)	**37.4** (+3.7)

Table 5.5: Comparisons to PointContrast for data-efficient 3D scene understanding on ScanNet. We compare our method with PointContrast (PC) and training from scratch (SC) in various tasks. Our method constantly achieves better results in both Limited Point Annotations (LA) and Limited Scene Reconstructions (LR) scenarios.

Datasets	Task (Metric)	SC	PC [35]	Ours
S3DIS	ins (mAP@0.5)	59.3	60.5 (+1.2)	**63.4** (+4.1)
S3DIS	sem (mIoU)	68.2	70.3 (+2.1)	**72.2** (+4.0)
SUN RGB-D	det (mAP@0.5)	31.7	34.8 (+3.1)	**36.4** (+4.7)
ScanNet	ins (mAP@0.5)	56.9	58.0 (+1.1)	**59.4** (+2.5)
ScanNet	sem (mIou)	72.2	**74.1** (+1.9)	73.8 (+1.6)
ScanNet	det (mAP@0.5)	35.4	38.0 (+2.6)	**39.3** (+3.9)

Table 5.6: Downstream fine-tuning results on other benchmarks. Contrastive Scene Contexts (Ours) achieve better or on par results compared to PointContrast (PC) [35] on instance segmentation (ins), semantic segmentation (sem) and object detection (det) across multiple datasets.

per-category results on data-efficient benchmark as well as on full data are showed in Section 5.8.

5.6 Implementation Details

Data Preprocessing. Following [35], we subsample the partial frames by every 25 frames. We find pairs of frames within each scene by computing their overlaps. In detail, every single frame is transformed to world coordinates. We iterate every pair of frames to calculate how many points are overlapped by 2.5cm threshold. For example, for each point in frame A, if we can find another point in frame B within 2.5cm in the transformed coordinate system (world), then those 2 points are stored as a correspondence pair. When 2 frames have at least 30% overlaps of points, those 2 frames are saved for training. We save and use both the xyz coordinates and rgb color for pre-training.

PointInfoNCE Loss. Here we explain the details of the PointInfoNCE loss (Equation 3 in the main paper).

$$\mathcal{L}_p = - \sum_{(i,j)\in M} \log \frac{\exp(\mathbf{f}_i^1 \cdot \mathbf{f}_j^2/\tau)}{\sum_{(\cdot,k)\in M, k\in par_p(i)} \exp(\mathbf{f}_i^1 \cdot \mathbf{f}_k^2/\tau)} \tag{5.4}$$

M denotes the set of all the corresponding matches from two frames. Denote the point features from two frames $\mathbf{f}^1$ and $\mathbf{f}^2$ respectively. In this formulation, we use the points that have at least one match as negative, and non-matched points are discarded. For a matched pair $(i, j) \in M$, point feature $\mathbf{f}_i^1$ serves as the query and $\mathbf{f}_j^2$ serves as the positive key. Point feature $\mathbf{f}_k^2$ where $\exists(\cdot, k) \in M$, $k \in par_p(i)$ and $k \neq j$ are used as the set of negative keys. In practice, we sample a subset of matched pairs from M for training.

Active Labelling. We first use our pre-trained network to make a forward pass on all the voxels of each scene in the training data, and save the 96-dim penultimate layer features at each voxel. Then we back-project the features at each voxel to the raw point cloud using nearest neighbour search. We run a k-means clustering algorithm on the features and xyz coordinates of the point cloud on each scene to get k centroids, where k is the number of points we propose to annotator to label. We run k-means for 50 iterations.

Clustering Algorithm in Instance Segmentation. We adapt the code of breadth first search from PointGroup [30]. Clustering only happens in the test time. In the test time, we cluster on points that are shifted by learned directional and distance vectors. Directional and distance vectors are learned by voting-center loss in the training time. We use 3cm-ball as threshold for every point to search its neighbouring points at each iteration. Within the ball, the points are grouped into one instance when they have the same semantic label. We don't use the ScoreNet proposed in PointGroup, so that we don't have additional network for training. We simply average the scores of semantic prediction of the points belonging to the same instance.

	cab	bed	chair	sofa	tabl	door	wind	bkshf	pic	cntr	desk	curt	fridg	showr	toil	sink	bath	ofurn	**avg**
Scratch	31.8	72.4	56.0	52.7	**55.9**	36.6	25.3	47.6	14.7	11.3	10.1	**36.4**	34.5	57.5	90.0	33.7	**80.3**	35.8	43.5
PointContrast	39.2	71.2	**63.1**	**71.4**	48.4	36.9	20.5	45.2	18.2	8.1	13.9	32.4	31.5	**64.1**	97.0	42.3	54.9	**40.1**	44.5
Ours	**43.7**	**75.2**	62.9	65.7	50.5	**43.4**	**27.4**	**52.9**	**26.9**	**19.7**	**14.4**	34.4	**39.9**	61.9	**97.4**	**49.4**	75.3	39.0	**48.9**

Table 5.7: Instance Segmentation with Limited Point Annotations (ScanNet-LA). We use mAP@0.5 as metric and demonstrate per-category performance over 18 classes on data-efficient benchmark (200 labelled points for training per scene).

	wall	floor	cab	bed	chair	sofa	tabl	door	wind	bkshf	pic	cntr	desk	curt	fridg	showr	toil	sink	bath	ofurn	**avg**
Scratch	81.6	**96.1**	57.5	79.5	88.1	82.2	67.1	55.9	54.4	76.3	24.3	59.9	52.9	67.9	39.8	55.9	86.9	58.2	82.4	42.1	65.5
PointContrast	83.0	96.0	**61.1**	**79.5**	89.5	81.9	**71.6**	57.1	**57.0**	73.0	22.6	62.0	**58.8**	**69.1**	44.4	63.6	91.5	**59.4**	**85.2**	48.5	67.8
Ours	**84.0**	95.9	60.2	79.0	**89.5**	**83.8**	69.6	**60.2**	56.7	**80.6**	**26.1**	**63.9**	55.6	63.5	**45.1**	**63.7**	**91.9**	56.9	84.7	**52.6**	**68.2**

Table 5.8: Semantic Segmentation with Limited Point Annotations (ScanNet-LA). We evaluate mean IoU over 20 classes on data-efficient benchmark (200 labelled points per scene for training).

	cab	bed	chair	sofa	tabl	door	wind	bkshf	pic	cntr	desk	curt	fridg	showr	toil	sink	bath	ofurn	**avg**
Scratch	5.4	**71.9**	64.2	59.8	37.3	17.1	6.8	32.3	0.4	**16.8**	**33.7**	**26.7**	29.2	3.3	**87.9**	20.6	70.2	17.9	33.4
PointContrast	10.3	71.8	**71.1**	61.2	43.1	**21.6**	**9.4**	34.7	**2.3**	6.8	25.7	21.2	32.6	**17.1**	84.1	20.4	74.6	20.0	34.9
Ours	**10.9**	69.5	70.2	**62.1**	44.3	18.2	9.0	**39.8**	1.0	9.2	32.9	25.3	**35.6**	10.3	78.9	**26.5**	**81.0**	**21.2**	**35.9**

Table 5.9: Object Detection with Limited Bounding Box Annotations . We evaluate mAP@0.5 over 18 classes on data-efficient benchmark (7 annotated bounding boxes for training per scene).

	ceiling	floor	wall	beam	column	window	door	chair	table	bookcase	sofa	board	**avg**
Scratch	46.8	89.5	72.5	0.0	**38.2**	72.5	89.5	88.0	39.3	34.7	72.7	85.7	59.3
PointContrast	66.0	**93.0**	73.0	0.0	18.6	72.8	88.3	**91.4**	42.3	**29.5**	63.6	88.0	60.5
Ours	**74.4**	88.0	**76.5**	**0.0**	32.4	**74.6**	**96.4**	91.0	**45.0**	28.8	**63.6**	**90.5**	**63.4**

Table 5.10: Instance Segmentation on Stanford Area 5 Test [27]. We evaluate mAP@0.5 over 12 classes.

5.7 More Visualizations

We show more visualizations of semantic segmentation and object detection predictions from our model trained with extremely scarce annotations. We show the semantic segmentation on ScanNet validation set with our model trained on 20 labelled points per scene in Figure 5.10. We also demonstrate the object detection results on ScanNet validation set predicted by our model trained on 1 bounding box annotated per scene in Figure 5.9.

5.8 Per-Category Results

In this section, we demonstrate detailed per-category performance as supplement of data-efficient benchmark. Instance segmentation on ScanNet-LA (Limited Scene Annotations, 200 labelled points for training) is showed in Table 5.7; semantic segmentation of per-category performance on ScanNet-LA is showed in Table 5.8; object detection on Limited Bounding Boxes Annotations is showed in Table 5.9.

5.9 Different Backbones.

We use Sparse Residual U-Net (SR-UNet-34, also used in [28]) as backbone architecture. 3D-MPA also uses a Sparse Residual U-Net backbone, and the performance gap is due to the additional head modules (e.g., Proposal Consolidation) which is orthogonal to our pre-training method. To show our algorithm is generic and agnostic to the specific backbone, we perform experiments with different backbones, including SR-UNet-18A and PointNet++. Models pre-trained with our method yield significant better results; see Tab. 5.16.

	ceiling	floor	wall	beam	column	window	door	chair	table	bookcase	sofa	board	clutter	**avg**
Scratch	91.5	98.6	84.1	0.0	33.0	56.9	63.9	90.1	81.7	72.5	76.5	77.9	59.6	68.2
PointContrast	93.3	**98.7**	85.6	**0.1**	**45.9**	54.4	67.9	91.6	80.1	**74.7**	78.2	81.5	62.3	70.3
Ours	**95.1**	98.4	**86.3**	0.0	40.7	**60.8**	**85.2**	**91.8**	**81.9**	73.9	**78.9**	**82.8**	**62.4**	**72.2**

Table 5.11: Semantic Segmentation on Stanford Area 5 Test [27]. We evaluate mIoU over 13 classes.

	bed	table	sofa	chair	toilet	desk	dresser	night stand	book	bathtub	**avg**
Scratch	47.8	19.6	48.1	54.6	60.0	6.3	15.8	27.3	5.4	32.1	31.7
PointContrast [35]	50.5	19.4	51.8	**54.9**	57.4	**7.5**	**16.2**	37.0	5.9	**47.6**	34.8
Ours	**55.3**	**20.3**	**53.8**	53.6	**65.9**	6.1	15.5	**38.0**	**9.1**	46.5	**36.4**

Table 5.12: Object Detection on SUN RGB-D [46]. We use mAP@0.5 as metric and show per-category AP@0.5 over 10 classes.

5.10 ScanNet Benchmark

We report validation results to directly compare with PointContrast which also evaluates on the val set. Additionally, we submitted our model to the ScanNet Benchmark (test set); see Tab. 5.15. Our method significantly outperforms 3D-MPA, despite not leveraging the special 3D-MPA proposal module.

5.11 Conclusion

In this work, we focus on data-efficient 3D scene understanding through a novel unsupervised pre-training algorithm that integrates the scene contexts in the contrastive learning framework. We show the possibility of using extremely few data or annotations to achieve competitive performance leveraging representation learning. Our results and findings are very encouraging and can potentially open up new opportunities in 3D (interactive) data collection, unsupervised 3D representation learning, and large-scale 3D scene understanding.

	cab	bed	chair	sofa	tabl	door	wind	bkshf	pic	cntr	desk	curt	fridg	showr	toil	sink	bath	ofurn	**avg**
Scratch	49.0	70.0	87.4	66.5	71.1	47.4	39.6	53.0	30.8	**32.8**	30.8	41.7	48.6	60.1	**99.9**	68.4	75.3	52.4	56.9
PointContrast	49.4	72.1	87.2	**71.7**	67.0	**49.0**	40.7	**57.8**	**35.6**	24.0	30.2	**49.9**	**53.0**	65.2	98.3	61.7	80.5	50.8	58.0
Ours	**50.8**	**74.1**	**88.7**	61.4	**67.2**	48.0	**42.0**	57.0	33.8	32.5	**42.9**	47.4	49.5	**68.9**	98.2	**71.3**	**80.5**	**54.7**	**59.4**

Table 5.13: Instance Segmentation on ScanNetV2 [4] Validation Set. We evaluate the mean average precision with IoU threshold of 0.5 over 18 classes.

	cab	bed	chair	sofa	tabl	door	wind	bkshf	pic	cntr	desk	curt	fridg	showr	toil	sink	bath	ofurn	**avg**
Scratch	9.9	70.5	70.0	60.5	43.4	**21.8**	10.5	**33.3**	0.8	15.4	33.3	26.6	**39.3**	9.7	74.7	23.7	75.8	18.1	35.4
PointContrast	13.1	**74.7**	**75.4**	**61.3**	44.8	19.8	12.9	32.0	0.9	**21.9**	31.9	**27.0**	32.6	17.5	**87.4**	23.2	80.8	**26.7**	38.0
Ours	**15.1**	74.3	71.9	60.2	**46.4**	21.2	**15.0**	32.5	**1.1**	9.4	**36.6**	21.3	37.3	**47.5**	84.3	**26.2**	**86.8**	21.2	**39.3**

Table 5.14: Object Detection on ScanNetV2 Validation Set. We use mAP@0.5 as metric and show per-category performance over 18 classes.

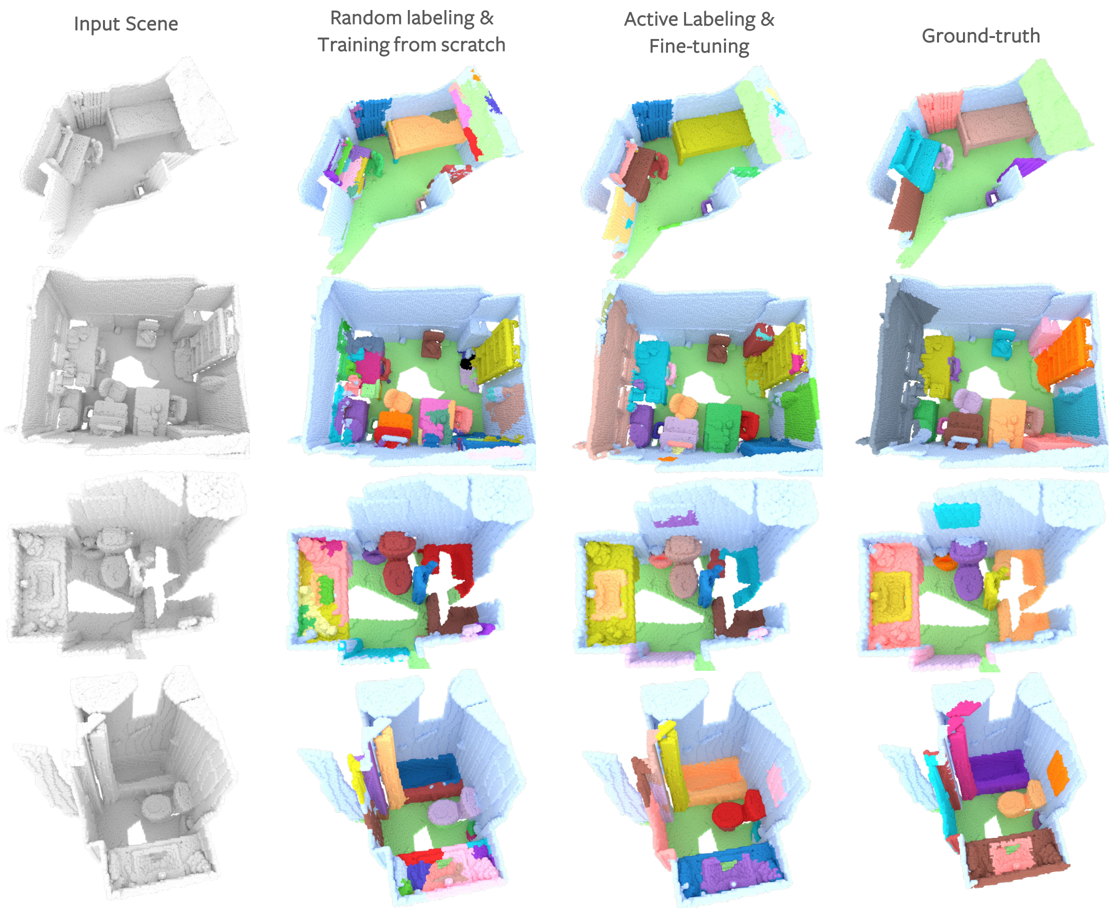


Figure 5.5: Qualitative Instance Segmentation Results (ScanNet-LA). With our pre-trained model as initialization for fine-tuning, together with an active labeling process, our approach (trained with 20 labeled points per scene) generates high-quality instance masks. Different color represents instance index only (same instances might not share the same color).

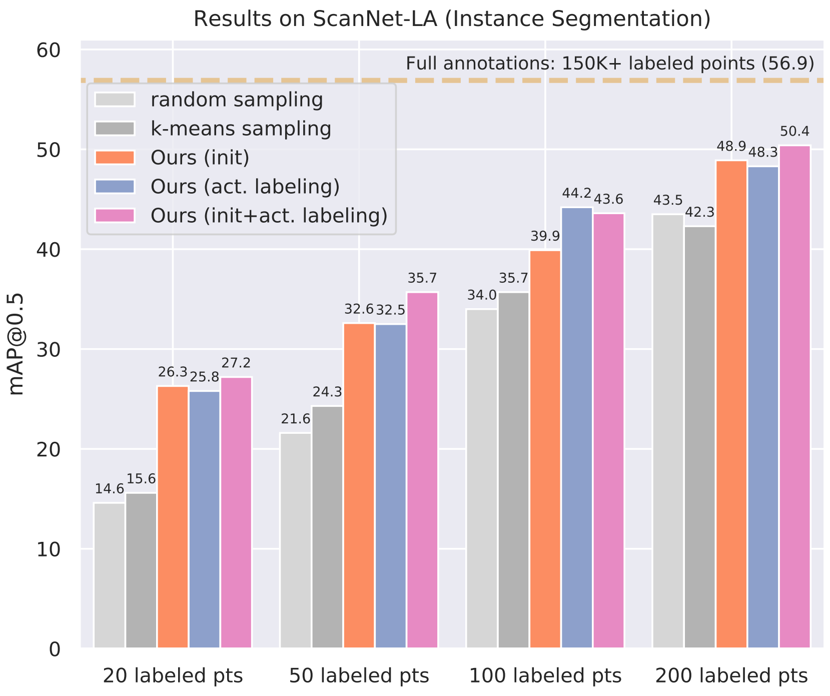


Figure 5.6: 3D Instance Segmentation with Limited Point Annotations (ScanNet-LA). Ours (init) denotes the network initialization by our pre-trained model. Ours (act. labeling) denotes the active selection of annotated points by our pre-trained model. Ours (init+act. labeling) denotes using our model as both network initialization and active labeling. We additionally mark the upper bound of using all 150K annotated points (in average) per scene as the dash line.

	AP	AP@50	AP@25
3D-MPA [106]	35.5	61.1	73.7
ours (pre-trained)	**40.5**	**64.8**	**79.1**

Table 5.15: ScanNet Instance Segmentation (test set) results.. Similar to S3DIS, we outperform 3D-MPA.

	Task	Dataset	Backone	mAP@0.5
scratch	ins	S3DIS	SR-UNet-18A	58.6
ours (pre-trained)	ins	S3DIS	SR-UNet-18A	**62.8**
scratch	det	ScanNet	PointNet++	33.5
ours (pre-trained)	det	ScanNet	PointNet++	**39.2**

Table 5.16: Pre-training with different backbones. 100% of available train data is used; we would expect larger deltas with smaller train set.

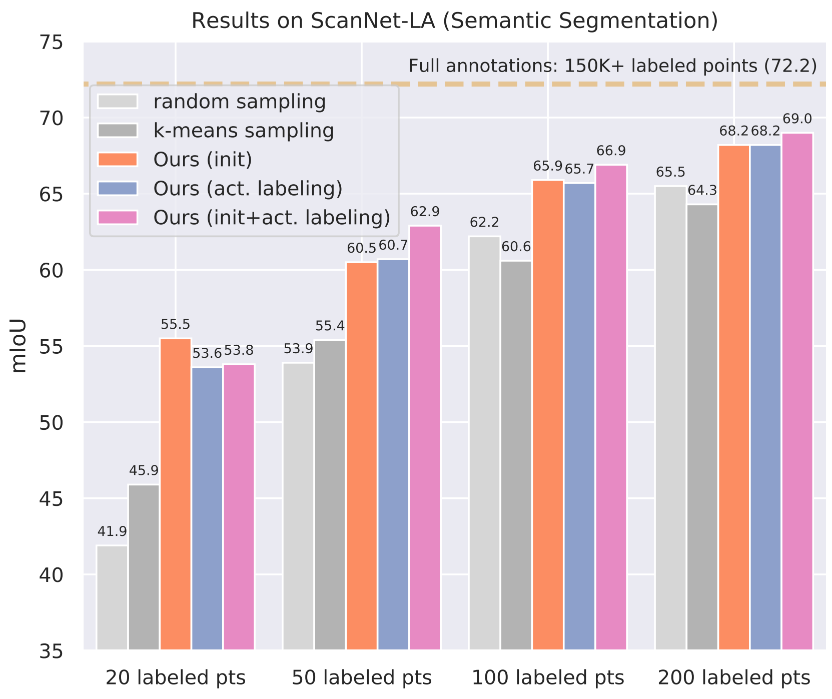


Figure 5.7: 3D Semantic Segmentation with Limited Point Annotations (ScanNet-LA). Ours (init) denotes the network initialization by our pre-trained model. Ours (act. labeling) denotes the active selection of annotated points by our pre-trained model. Ours (init+act. labeling) denotes using our model as both network initialization and active labeling. We additionally mark the upper bound of using all 150K annotated points (in average) per scene as the dash line.

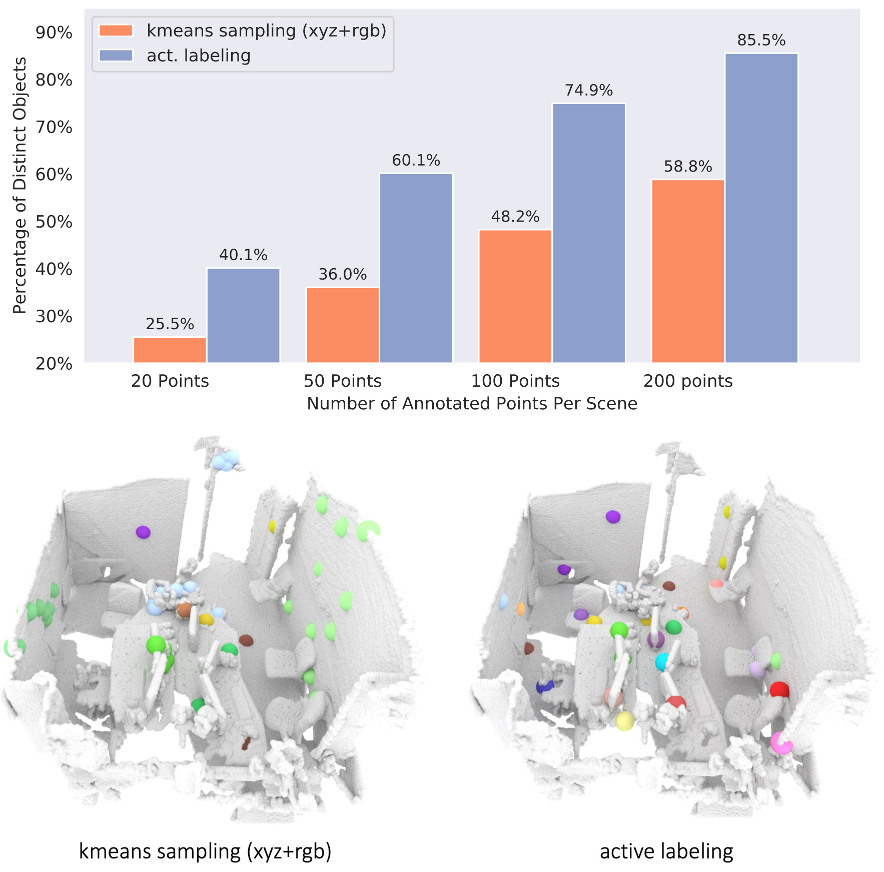


Figure 5.8: Sampled points in cluttered scenes. Top: object coverage percentage—more distinct objects are covered with active labeling; **Bottom**: Visualization of sampled points in a cluttered scene.

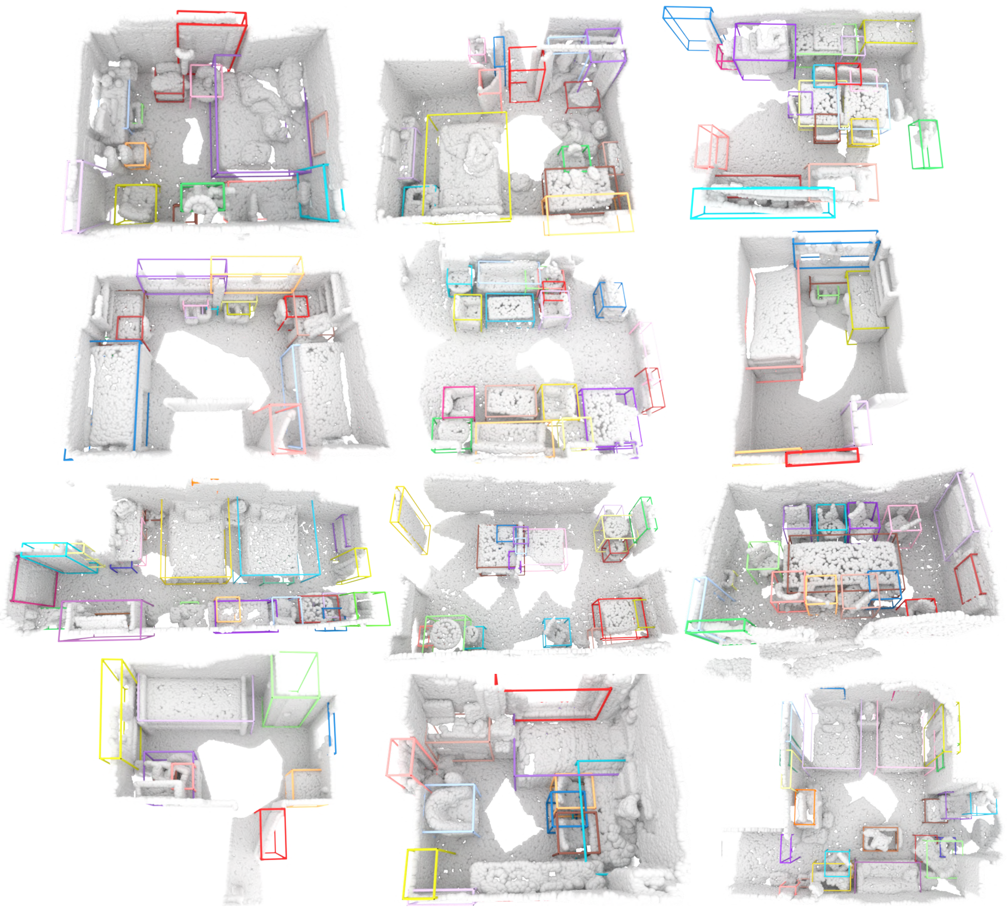


Figure 5.9: Object Detection Results (Limited Bounding Box Annotations). With our pre-trained model as initialization for fine-tuning, our approach generates high-quality detection predictions. Here our model is trained with 1 bounding box annotated per scene.

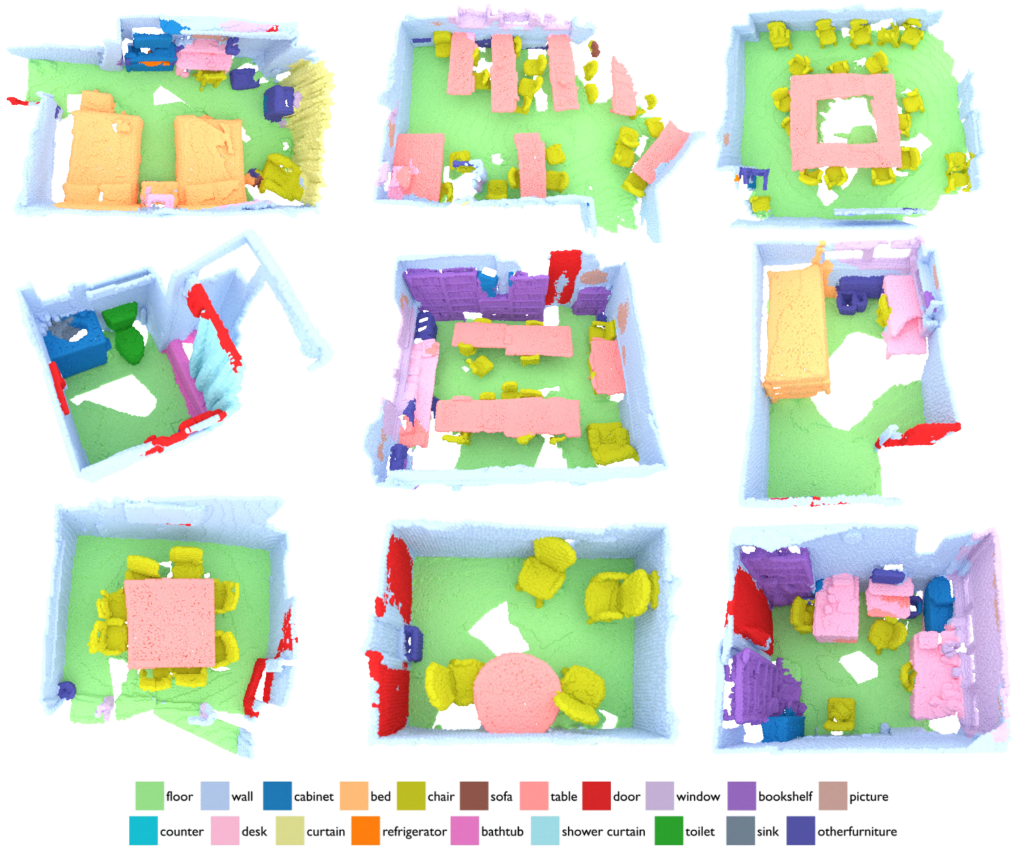


Figure 5.10: Semantic Segmentation Results (ScanNet-LA). With our pre-trained model as initialization for fine-tuning, together with an active labeling process, our approach generates high-quality semantic segmentation predictions. Here our model is fine-tuned with 20 labeled points per scene.

Part III

Conclusion & Outlook

6 Conclusion

This dissertation investigates a very important research topic: 3D Scene Understanding. We mainly focus on three problems: 3D Semantic Instance Segmentation, 3D Semantic Instance Completion and Data-Efficient 3D Scene Understanding. Each of these problems were introduced in Part II and we present concluding remarks in the following.

3D-SIS: Semantic Instance Segmentation in RGB-D Scans In Chapter 3, we introduce 3D-SIS, our 3D semantic instance segmentation algorithm. We present an 3D anchor regression mechanism for bounding boxes detection. Following Mask-RCNN way, we define a mask branch for 3D shape estimation and a classification branch for semantic label prediction. Our ablation study reveals the fusion of 2D CNNs color features and 3D geometry features futher boosts the performance. We show our algorithms generalize on both synthetic and real data and outperform our baselines by a large margin.

RevealNet: Seeing Behind Objects in RGB-D Scans In Chapter 4, we discuss instance-level completion. We present RevealNet, the first semantic instance completion algorithm. Following 3D-SIS framework, we introduced a completion branch to complete missing geometry of 3d objects in RGB-D scans. Although there are no perfect 3D reconstructions as ground truth, we leverage objects' CAD models aligned to 3D scans as target to learn instance-level completion. Our ablation study shows incorporating geometric completion on instances loss improves detection and segmentation performance. Our end-to-end joint training algorithm significantly outperforms the baselines that do completion and instance segmentation separately.

Exploring Data-Efficient 3D Scene Understanding with Contrastive Scene Contexts In Chapter 5, we explore the data-efficient scenarios for 3D scene understanding tasks. We propose the benchmark of using limited annotations and reconstructions for training. To this end, we also propose an unsupervised pre-training algorithm leveraging point-wise and spatial contexts in a contrastive learning framework. Our experiments reveal that even with less than 1% annotated points on ScanNet, we can achieve competitive results as baseline using 100% annotated points for semantic and instance segmentation tasks. For detection task, we show with only 10% available annotations and reconstructions on ScanNet, state-of-the-art methods can barely learn anything. However, it can predict meaningful results with our pre-training algorithm. Our research opens up the possibility of large-scale 3D scene understanding by saving much effort on human annotations.

7 Limitations and Future Work

"We can only see a short distance ahead, but we can see plenty there that needs to be done."

–– Alan Turing, Computing Machinery and Intelligence (1950)

7.1 3D-SIS: Semantic Instance Segmentation in RGB-D Scans

In this work, we explore the fusion of color features and geometric features represented by tsdf. We learn the color feature from a simle ENet network on a semantic segmentation task. However, there are a variety of more powerful 2D CNNs and 2D tasks that can be used for learning better 2D features, e.g. Mask-RCNN with FPN [41] on instance segmentation task. We can also hugely improve the inference time by leveraging more advanced 3D backbones rather than 3D CNNs, e.g. SparseConvs [104]. For simplicity, we only regress the locations and sizes of 3D bounding boxes. Another improvement could be also learn the pose of objects, such as rotation around world-up axis. To this end, it may help to predict better 3D object shapes.

7.2 RevealNet: Seeing Behind Objects in RGB-D Scans

In RevealNet, we learn to complete missing geometry of each 3D object in RGB-D scans. We use aligned CAD models of objects as ground truth. We compute a simple binary cross entropy loss on unseen parts to predict if it is occupied or not. Due to the completion loss, it is a deterministic way to predict missing geometry. To this end, we actually predict the average geometry of unseen parts. However, the missing geometry has multiple possible shapes. In the future work, we try to explore to use a generative model, e.g. GANs, to complete the missing geometry by multiple possibilities.

Another potential improvement of a binary mask for the complete object geometry is to use implicit neural representations. Binary mask can hugely limit the details of geometry. To this end, implicit neural representations, such as DeepSDF [18] or Occupancy Network [19], could potentially resolve greater geometric details.

We also do not consider object movement over time, which contains significant opportunities for semantic instance completion in the context of dynamic environments.

7.3 Exploring Data-Efficient 3D Scene Understanding with Contrastive Scene Contexts

In this work, we investigate the possibility of using extremely few annotated data to train 3D scene understanding models. We notice that exhaustively annotating every point might not be necessary. We propose to find the representative points in feature space by clustering, and propose this point the annotator to label. However, there is still a big unknown space to explore. For example, annotating a point on each instance might be even more representative, but could consume more time. Because annotator needs to rotate the camera to find each distinct instance. As a trade-off of time effort and performance, it is still in discussion how to find the most representative points to annotate.

We also propose a pre-training algorithm leveraging spatial contexts which is given by free in 3D. There are other more information that given by free in RGB-D data and we can also take advantage of. For instance, key point matching between depth frames and color images. It is still in debate what is a better way to jointly pre-train on cross-modality features between geometry and color, so that it can better transfer to downstream scene understanding tasks.

Bibliography

[1] J. L. Schönberger and J.-M. Frahm, "Structure-from-motion revisited," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[2] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm, "Pixelwise view selection for unstructured multi-view stereo," in *European Conference on Computer Vision (ECCV)*, 2016.

[3] A. Dai, M. Nießner, M. Zollhöfer, S. Izadi, and C. Theobalt, "Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 3, p. 24, 2017.

[4] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "Scannet: Richly-annotated 3D reconstructions of indoor scenes," in *CVPR*, 2017.

[5] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the fifth annual workshop on Computational learning theory*, 1992, pp. 144–152.

[6] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *CVPR*, 2009.

[8] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *ECCV*, 2014.

[9] J. Hou, A. Dai, and M. Nießner, "3D-SIS: 3D Semantic Instance Segmentation of RGB-D Scans," in *CVPR*, 2019.

[10] J. Hou, A. Dai, and M. Nießner, "RevealNet: Seeing Behind Objects in RGB-D Scans," in *CVPR*, 2020.

[11] J. Hou, B. Graham, M. Nießner, and S. Xie, "Exploring Data-Efficient 3D Scene Understanding with Contrastive Scene Contexts," in *CVPR*, 2021.

[12] **J. Hou**, A. Dai, and M. Nießner, "3D-SIS: 3D Semantic Instance Segmentation of RGB-D Scans," in *Proceedings of Computer Vision and Pattern Recognition (CVPR), IEEE*, 2019.

[13] **J. Hou**, A. Dai, and M. Nießner, "RevealNet: Seeing Behind Objects in RGB-D Scans," in *Proceedings of Computer Vision and Pattern Recognition (CVPR), IEEE*, 2020.

[14] **J. Hou**, B. Graham, M. Nießner, and S. Xie, "Exploring Data-Efficient 3D Scene Understanding with Contrastive Scene Contexts," in *Proceedings of Computer Vision and Pattern Recognition (CVPR), IEEE*, 2021.

[15] M. Gadelha, S. Maji, and R. Wang, "Shape generation using spatially partitioned point clouds," *arXiv preprint arXiv:1707.06267*, 2017.

[16] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall, "Semantickitti: A dataset for semantic scene understanding of lidar sequences," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9297–9307.

[17] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser, "Semantic scene completion from a single depth image," *Proceedings of 30th IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[18] 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/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 165–174.

[19] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, "Occupancy networks: Learning 3d reconstruction in function space," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4460–4470.

[20] B. Graham and L. van der Maaten, "Submanifold sparse convolutional networks," *arXiv preprint arXiv:1706.01307*, 2017.

[21] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3D classification and segmentation," *CVPR*, 2017.

[22] M. Jaderberg, K. Simonyan, A. Zisserman, *et al.*, "Spatial transformer networks," in *NeurIPS*, 2015.

[23] 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," *arXiv preprint arXiv:1512.03012*, 2015.

[24] 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," *arXiv preprint arXiv:1512.03012*, 2015.

[25] A. Avetisyan, M. Dahnert, A. Dai, M. Savva, A. X. Chang, and M. Nießner, "Scan2cad: Learning cad model alignment in rgb-d scans," in *CVPR*, 2019.

[26] S. Song, S. P. Lichtenberg, and J. Xiao, "SUN RGB-D: A RGB-D Scene Understanding Benchmark Suite," in *CVPR*, 2015.

[27] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer, and S. Savarese, "3D semantic parsing of large-scale indoor spaces," in *ICCV*, 2016.

[28] C. Choy, J. Gwak, and S. Savarese, "4D spatio-temporal convnets: Minkowski convolutional neural networks," in *CVPR*, 2019.

[29] A. Dai and M. Nießner, "3dmv: Joint 3d-multi-view prediction for 3d semantic scene segmentation," *arXiv preprint arXiv:1803.10409*, 2018.

[30] L. Jiang, H. Zhao, S. Shi, S. Liu, C.-W. Fu, and J. Jia, "PointGroup: Dual-Set Point Grouping for 3D Instance Segmentation," in *CVPR*, 2020.

[31] W. Wang, R. Yu, Q. Huang, and U. Neumann, "Sgpn: Similarity group proposal network for 3d point cloud instance segmentation," in *CVPR*, 2018.

[32] C. R. Qi, O. Litany, K. He, and L. J. Guibas, "Deep hough voting for 3D object detection in point clouds," *ICCV*, 2019.

[33] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Advances in Neural Information Processing Systems*, 2017, pp. 5105–5114.

[34] Y. Nie, **J. Hou**, X. Han, and M. Nießner, "RfD-Net: Point Scene Understanding by Semantic Instance Reconstruction," in *Proceedings of Computer Vision and Pattern Recognition (CVPR), IEEE*, 2021.

[35] S. Xie, J. Gu, D. Guo, C. R. Qi, L. J. Guibas, and O. Litany, "Pointcontrast: Unsupervised pre-training for 3D point cloud understanding," *ECCV*, 2020.

[36] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *CVPR*, 2015.

[37] F. Iandola, M. Moskewicz, S. Karayev, R. Girshick, T. Darrell, and K. Keutzer, "Densenet: Implementing efficient convnet descriptor pyramids," *arXiv preprint arXiv:1404.1869*, 2014.

[38] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, "Enet: A deep neural network architecture for real-time semantic segmentation," *arXiv:1606.02147*, 2016.

[39] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[40] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *CVPR*, 2016.

[41] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *ICCV*, 2017.

[42] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *NeurIPS*, 2015.

[43] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *ECCV*, Springer, 2016, pp. 21–37.

[44] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *CVPR*, 2017.

[45] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[46] S. Song and J. Xiao, “Sliding shapes for 3D object detection in depth images,” in *ECCV*, 2014.

[47] S. Song and J. Xiao, “Deep sliding shapes for amodal 3d object detection in rgb-d images,” *arXiv preprint arXiv:1511.02300*, 2015.

[48] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, “Frustum pointnets for 3d object detection from rgb-d data,” *arXiv preprint arXiv:1711.08488*, 2017.

[49] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár, “Panoptic segmentation,” *arXiv preprint arXiv:1801.00868*, 2018.

[50] B. Curless and M. Levoy, “A volumetric method for building complex models from range images,” in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, ACM, 1996, pp. 303–312.

[51] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, “3d shapenets: A deep representation for volumetric shapes,” in *CVPR*, 2015.

[52] I. Armeni, S. Sax, A. R. Zamir, and S. Savarese, “Joint 2d-3d-semantic data for indoor scene understanding,” *arXiv preprint arXiv:1702.01105*, 2017.

[53] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang, “Matterport3D: Learning from RGB-D data in indoor environments,” *International Conference on 3D Vision (3DV)*, 2017.

[54] D. Maturana and S. Scherer, “Voxnet: A 3d convolutional neural network for real-time object recognition,” in *IROS*, 2015.

[55] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. Guibas, “Volumetric and multi-view cnns for object classification on 3D data,” in *CVPR*, 2016.

[56] G. Riegler, A. O. Ulusoy, and A. Geiger, “Octnet: Learning deep 3d representations at high resolutions,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[57] M. Tatarchenko, J. Park, V. Koltun, and Q.-Y. Zhou, “Tangent convolutions for dense prediction in 3d,” in *CVPR*, 2018.

[58] 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.

[59] A. Dai, D. Ritchie, M. Bokeloh, S. Reed, J. Sturm, and M. Nießner, “Scancomplete: Large-scale scene completion and semantic segmentation for 3d scans,” *arXiv preprint arXiv:1712.10215*, 2018.

[60] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, “Multi-view convolutional neural networks for 3d shape recognition,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 945–953.

[61] E. Kalogerakis, M. Averkiou, S. Maji, and S. Chaudhuri, “3d shape segmentation with projective convolutional networks,” *Proc. CVPR, IEEE*, vol. 2, 2017.

[62] C. Elich, F. Engelmann, J. Schult, T. Kontogianni, and B. Leibe, “3D-BEVIS: Birds-Eye-View Instance Segmentation,” *CoRR*, vol. abs/1904.02199, 2019.

[63] L. Yi, W. Zhao, H. Wang, M. Sung, and L. Guibas, “GSPN: Generative shape proposal network for 3D instance segmentation in point cloud,” in *CVPR*, 2019.

[64] B. Yang, J. Wang, R. Clark, Q. Hu, S. Wang, A. Markham, and N. Trigoni, “Learning object bounding boxes for 3D instance segmentation on point clouds,” in *NeurIPS*, 2019.

[65] J. Lahoud, B. Ghanem, M. Pollefeys, and M. R. Oswald, “3d instance segmentation via multi-task metric learning,” in *ICCV*, 2019.

[66] G. Narita, T. Seno, T. Ishikawa, and Y. Kaji, “Panopticfusion: Online volumetric semantic mapping at the level of stuff and things,” *arXiv preprint arXiv:1903.01177*, 2019.

[67] C. Elich, F. Engelmann, J. Schult, T. Kontogianni, and B. Leibe, “3d-bevis: Birds-eye-view instance segmentation,” *arXiv preprint arXiv:1904.02199*, 2019.

[68] C. R. Qi, O. Litany, K. He, and L. J. Guibas, “Deep hough voting for 3d object detection in point clouds,” *arXiv preprint arXiv:1904.09664*, 2019.

[69] C. Liu and Y. Furukawa, “Masc: Multi-scale affinity with sparse convolution for 3d instance segmentation,” *arXiv preprint arXiv:1902.04478*, 2019.

[70] O. Sorkine and D. Cohen-Or, “Least-squares meshes,” in *Shape Modeling Applications, 2004. Proceedings*, IEEE, 2004, pp. 191–199.

[71] A. Nealen, T. Igarashi, O. Sorkine, and M. Alexa, “Laplacian mesh optimization,” in *Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia*, ACM, 2006, pp. 381–389.

[72] W. Zhao, S. Gao, and H. Lin, “A robust hole-filling algorithm for triangular mesh,” *The Visual Computer*, vol. 23, no. 12, pp. 987–997, 2007.

[73] M. Kazhdan, M. Bolitho, and H. Hoppe, “Poisson surface reconstruction,” in *Proceedings of the fourth Eurographics symposium on Geometry processing*, vol. 7, 2006.

[74] M. Kazhdan and H. Hoppe, “Screened poisson surface reconstruction,” *ACM Transactions on Graphics (TOG)*, vol. 32, no. 3, p. 29, 2013.

[75] S. Thrun and B. Wegbreit, “Shape from symmetry,” in *Tenth IEEE International Conference on Computer Vision (ICCV’05) Volume 1*, IEEE, vol. 2, 2005, pp. 1824–1831.

[76] N. J. Mitra, L. J. Guibas, and M. Pauly, “Partial and approximate symmetry detection for 3d geometry,” in *ACM Transactions on Graphics (TOG)*, ACM, vol. 25, 2006, pp. 560–568.

[77] M. Pauly, N. J. Mitra, J. Wallner, H. Pottmann, and L. J. Guibas, “Discovering structural regularity in 3d geometry,” in *ACM transactions on graphics (TOG)*, ACM, vol. 27, 2008, p. 43.

[78] I. Sipiran, R. Gregor, and T. Schreck, “Approximate symmetry detection in partial 3d meshes,” in *Computer Graphics Forum*, Wiley Online Library, vol. 33, 2014, pp. 131–140.

[79] P. Speciale, M. R. Oswald, A. Cohen, and M. Pollefeys, “A symmetry prior for convex variational 3d reconstruction,” in *European Conference on Computer Vision*, Springer, 2016, pp. 313–328.

[80] L. Nan, K. Xie, and A. Sharf, “A search-classify approach for cluttered indoor scene understanding,” *ACM Transactions on Graphics (TOG)*, vol. 31, no. 6, p. 137, 2012.

[81] T. Shao, W. Xu, K. Zhou, J. Wang, D. Li, and B. Guo, “An interactive approach to semantic modeling of indoor scenes with an rgbd camera,” *ACM Transactions on Graphics (TOG)*, vol. 31, no. 6, p. 136, 2012.

[82] Y. M. Kim, N. J. Mitra, D.-M. Yan, and L. Guibas, “Acquiring 3d indoor environments with variability and repetition,” *ACM Transactions on Graphics (TOG)*, vol. 31, no. 6, p. 138, 2012.

[83] Y. Li, A. Dai, L. Guibas, and M. Nießner, “Database-assisted object retrieval for real-time 3d reconstruction,” in *Computer Graphics Forum*, Wiley Online Library, vol. 34, 2015.

[84] Y. Shi, P. Long, K. Xu, H. Huang, and Y. Xiong, “Data-driven contextual modeling for 3d scene understanding,” *Computers & Graphics*, vol. 55, pp. 55–67, 2016.

[85] A. Dai, C. R. Qi, and M. Nießner, “Shape completion using 3d-encoder-predictor cnns and shape synthesis,” in *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017.

[86] 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 *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[87] C. Häne, S. Tulsiani, and J. Malik, “Hierarchical surface prediction for 3d object reconstruction,” *arXiv preprint arXiv:1704.00710*, 2017.

[88] A. Avetisyan, A. Dai, and M. Nießner, “End-to-end cad model retrieval and 9dof alignment in 3d scans,” in *ICCV*, 2019.

[89] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, “Indoor segmentation and support inference from RGB-D images,” *ECCV*, 2012.

[90] A. Janoch, S. Karayev, Y. Jia, J. T. Barron, M. Fritz, K. Saenko, and T. Darrell, “A category-level 3d object dataset: Putting the kinect to work,” in *Consumer depth cameras for computer vision*, 2013.

[91] J. Xiao, A. Owens, and A. Torralba, “SUN3D: A database of big spaces reconstructed using sfm and object labels,” in *ICCV*, 2013.

[92] K. Mo, S. Zhu, A. X. Chang, L. Yi, S. Tripathi, L. J. Guibas, and H. Su, "PartNet: A Large-Scale Benchmark for Fine-Grained and Hierarchical Part-Level 3D Object Understanding," in *CVPR*, 2019.

[93] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, Y. Zhang, J. Shlens, Z. Chen, and D. Anguelov, "Scalability in perception for autonomous driving: Waymo open dataset," in *CVPR*, 2020.

[94] A. Gupta, P. Dollar, and R. Girshick, "LVIS: A dataset for large vocabulary instance segmentation," in *CVPR*, 2019.

[95] O. J. Hénaff, A. Srinivas, J. De Fauw, A. Razavi, C. Doersch, S. Eslami, and A. van den Oord, "Data-efficient image recognition with contrastive predictive coding," *ICML*, 2020.

[96] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *NeurIPS*, 2017.

[97] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *CVPR*, 2012.

[98] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum pointnets for 3D object detection from RGB-D data," in *CVPR*, 2018.

[99] C. R. Qi, X. Chen, O. Litany, and L. J. Guibas, "Imvotenet: Boosting 3D object detection in point clouds with image votes," in *CVPR*, 2020.

[100] J. Gwak, C. Choy, and S. Savarese, "Generative sparse detection networks for 3D single-shot object detection," *ECCV*, 2020.

[101] L. Tchapmi, C. Choy, I. Armeni, J. Gwak, and S. Savarese, "Segcloud: Semantic segmentation of 3D point clouds," in *3DV*, 2017.

[102] W. Wu, Z. Qi, and L. Fuxin, "PointConv: Deep convolutional networks on 3D point clouds," in *CVPR*, 2019.

[103] 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 *CVPR*, 2019.

[104] B. Graham, M. Engelcke, and L. van der Maaten, "3D semantic segmentation with submanifold sparse convolutional networks," in *CVPR*, 2018.

[105] X. Wang, S. Liu, X. Shen, C. Shen, and J. Jia, "Associatively segmenting instances and semantics in point clouds," in *CVPR*, 2019.

[106] F. Engelmann, M. Bokeloh, A. Fathi, B. Leibe, and M. Nießner, "3D-MPA: Multi-Proposal Aggregation for 3D Semantic Instance Segmentation," in *CVPR*, 2020.

[107] H. Jiang, F. Yan, J. Cai, J. Zheng, and J. Xiao, "End-to-End 3D Point Cloud Instance Segmentation Without Detection," in *CVPR*, 2020.

[108] L. Han, T. Zheng, L. Xu, and L. Fang, “OccuSeg: Occupancy-aware 3D instance segmentation,” in *CVPR*, 2020.

[109] O. Ronneberger, P. Fischer, and T. Brox, “U-Net: Convolutional networks for biomedical image segmentation,” in *MICCAI*, 2015.

[110] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in *CVPR*, 2014.

[111] A. v. d. Oord, Y. Li, and O. Vinyals, “Representation learning with contrastive predictive coding,” *arXiv preprint arXiv:1807.03748*, 2018.

[112] P. Bachman, R. D. Hjelm, and W. Buchwalter, “Learning representations by maximizing mutual information across views,” in *NeurIPS*, 2019.

[113] I. Misra and L. van der Maaten, “Self-supervised learning of pretext-invariant representations,” in *CVPR*, 2020.

[114] Z. Wu, Y. Xiong, S. X. Yu, and D. Lin, “Unsupervised feature learning via non-parametric instance discrimination,” in *CVPR*, 2018.

[115] Y. Tian, D. Krishnan, and P. Isola, “Contrastive multiview coding,” *ECCV*, 2020.

[116] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, and Y. Bengio, “Learning deep representations by mutual information estimation and maximization,” *ICLR*, 2019.

[117] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, “Momentum contrast for unsupervised visual representation learning,” in *CVPR*, 2020.

[118] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, “A simple framework for contrastive learning of visual representations,” *ICML*, 2020.

[119] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin, “Unsupervised learning of visual features by contrasting cluster assignments,” in *NeurIPS*, 2020.

[120] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar, *et al.*, “Bootstrap your own latent-a new approach to self-supervised learning,” *NeurIPS*, 2020.

[121] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas, “Learning representations and generative models for 3D point clouds,” *ICML*, 2018.

[122] M. Gadelha, R. Wang, and S. Maji, “Multiresolution tree networks for 3D point cloud processing,” in *ECCV*, 2018.

[123] Y. Yang, C. Feng, Y. Shen, and D. Tian, “Foldingnet: Point cloud auto-encoder via deep grid deformation,” in *CVPR*, 2018.

[124] T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. Aubry, “A papier-mâché approach to learning 3D surface generation,” in *CVPR*, 2018.

[125] J. Li, B. M. Chen, and G. Hee Lee, “SO-Net: Self-organizing network for point cloud analysis,” in *CVPR*, 2018.

[126] Y. Wang and J. M. Solomon, "Deep closest point: Learning representations for point cloud registration," in *ICCV*, 2019.

[127] K. Hassani and M. Haley, "Unsupervised multi-task feature learning on point clouds," in *ICCV*, 2019.

[128] J. Sauder and B. Sievers, "Self-supervised deep learning on point clouds by reconstructing space," in *NeurIPS*, 2019.

[129] A. Sanghi, "Info3D: Representation learning on 3D objects using mutual information maximization and contrastive learning," *ECCV*, 2020.

[130] I. Biederman, "Recognition-by-components: A theory of human image understanding.," *Psychological review*, vol. 94, no. 2, p. 115, 1987.

[131] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *TPAMI*, vol. 35, no. 8, pp. 1798–1828, 2013.

[132] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, "Language models are few-shot learners," *NeurIPS*, 2020.

[133] C. Zhu, K. Xu, S. Chaudhuri, L. Yi, L. J. Guibas, and H. Zhang, "AdaCoSeg: Adaptive shape co-segmentation with group consistency loss," in *CVPR*, 2020.

[134] G. Sharma, E. Kalogerakis, and S. Maji, "Learning point embeddings from shape repositories for few-shot segmentation," in *2019 International Conference on 3D Vision (3DV)*, IEEE, 2019, pp. 67–75.

[135] S. Muralikrishnan, V. G. Kim, and S. Chaudhuri, "Tags2Parts: Discovering semantic regions from shape tags," in *CVPR*, 2018.

[136] Z. Chen, K. Yin, M. Fisher, S. Chaudhuri, and H. Zhang, "BAE-Net: Branched autoencoder for shape co-segmentation," in *CVPR*, 2019.

[137] M. Gadelha, A. RoyChowdhury, G. Sharma, E. Kalogerakis, L. Cao, E. Learned-Miller, R. Wang, and S. Maji, "Label-efficient learning on point clouds using approximate convex decompositions," *ECCV*, 2020.

[138] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *TPAMI*, vol. 24, no. 4, pp. 509–522, 2002.

[139] M. Körtgen, M. Novotni, and R. Klein, "3D shape matching with 3D shape contexts," in *In The 7th Central European Seminar on Computer Graphics*, Citeseer, 2003.

[140] S. Xie, S. Liu, Z. Chen, and Z. Tu, "Attentional shapecontextnet for point cloud recognition," in *CVPR*, 2018.

[141] M. A. Nielsen, *Neural networks and deep learning.* Determination press San Francisco, CA, 2015, vol. 25.

[142] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," *arXiv preprint arXiv:1603.07285*, 2016.

[143] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[144] [Online]. Available: `https://computersciencewiki.org/index.php/File:MaxpoolSample2.png`.

[145] Y. Wu and K. He, "Group normalization," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 3–19.

[146] [Online]. Available: `https://deepai.org/machine-learning-glossary-and-terms/activation-function`.

[147] [Online]. Available: `https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53`.

[148] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[149] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.

[150] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[151] S. K. Kumar, "On weight initialization in deep neural networks," *arXiv preprint arXiv:1704.08863*, 2017.

[152] S. Badillo, B. Banfai, F. Birzele, I. Davydov, L. Hutchinson, T. Kam-Thong, J. Siebourg-Polster, B. Steiert, and J. D. Zhang, "An introduction to machine learning," *Clinical Pharmacology Therapeutics*, vol. 107, Mar. 2020. DOI: `10.1002/cpt.1796`.

[153] F.-F. Li, A. Karpathy, and J. Johnson, "Cs231n: Convolutional neural networks for visual recognition," *University lecture*, 2015.

Appendix

A Open-source Code & Videos

A.1 3D-SIS: Semantic Instance Segmentation in RGB-D Scans

- Source Code: `https://github.com/Sekunde/3D-SIS`
- Video: `https://www.youtube.com/watch?v=IH9rNLD1-JE`

A.2 RevealNet: Seeing Behind Objects in RGB-D Scans

- Video: `https://www.youtube.com/watch?v=iyT_fkOA2yg`

A.3 Exploring Data-Efficient 3D Scene Understanding with Contrastive Scene Contexts

- Video: `https://www.youtube.com/watch?v=E7OxToZLgs4`
- Project: `https://sekunde.github.io/project_efficient/`

B Authored and Co-authored Publications

1. **J. Hou**, A. Dai, and M. Nießner, "3D-SIS: 3D Semantic Instance Segmentation of RGB-D Scans," in *Proceedings of Computer Vision and Pattern Recognition (CVPR), IEEE*, 2019
2. **J. Hou**, A. Dai, and M. Nießner, "RevealNet: Seeing Behind Objects in RGB-D Scans," in *Proceedings of Computer Vision and Pattern Recognition (CVPR), IEEE*, 2020
3. **J. Hou**, B. Graham, M. Nießner, and S. Xie, "Exploring Data-Efficient 3D Scene Understanding with Contrastive Scene Contexts," in *Proceedings of Computer Vision and Pattern Recognition (CVPR), IEEE*, 2021
4. **J. Hou**, S. Xie, B. Graham, A. Dai, and M. Nießner, "Pri3D: Can 3D Priors Help 2D Representation Learning?" In *Proceedings of International Conference on Computer Vision (ICCV), IEEE*, 2021
5. M. Dahnert, **J. Hou**, M. Nießner, and A. Dai, "Panoptic 3d scene reconstruction from a single rgb image," in *Thirty-Fifth Conference on Neural Information Processing Systems (NeurIPS)*, 2021
6. Y. Nie, **J. Hou**, X. Han, and M. Nießner, "RfD-Net: Point Scene Understanding by Semantic Instance Reconstruction," in *Proceedings of Computer Vision and Pattern Recognition (CVPR), IEEE*, 2021

C Original Publications

3D-SIS: 3D Semantic Instance Segmentation of RGB-D Scans

Ji Hou Angela Dai Matthias Nießner
Technical University of Munich

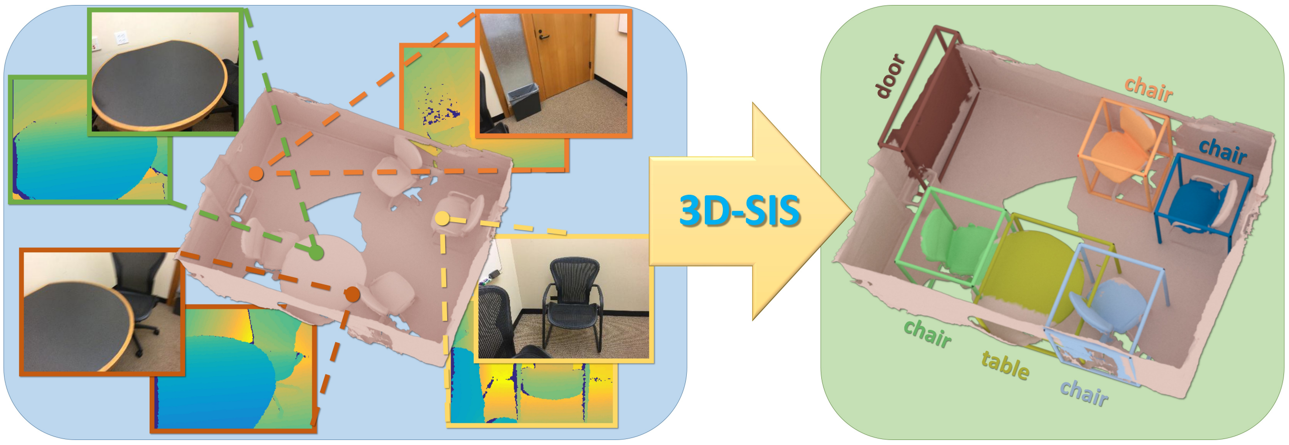


Figure 1: 3D-SIS performs 3D instance segmentation on RGB-D scan data, learning to jointly fuse both 2D RGB input features with 3D scan geometry features. In combination with a fully-convolutional approach enabling inference on full 3D scans at test time, we achieve accurate inference for object bounding boxes, class labels, and instance masks.

Abstract

We introduce 3D-SIS, a novel neural network architecture for 3D semantic instance segmentation in commodity RGB-D scans. The core idea of our method is to jointly learn from both geometric and color signal, thus enabling accurate instance predictions. Rather than operate solely on 2D frames, we observe that most computer vision applications have multi-view RGB-D input available, which we leverage to construct an approach for 3D instance segmentation that effectively fuses together these multi-modal inputs. Our network leverages high-resolution RGB input by associating 2D images with the volumetric grid based on the pose alignment of the 3D reconstruction. For each image, we first extract 2D features for each pixel with a series of 2D convolutions; we then backproject the resulting feature vector to the associated voxel in the 3D grid. This combination of 2D and 3D feature learning allows significantly higher accuracy object detection and instance segmentation than state-of-the-art alternatives. We show results on both synthetic and real-world public benchmarks, achieving an improvement in mAP of over 13 on real-world data.

1. Introduction

Semantic scene understanding is critical to many real-world computer vision applications. It is fundamental towards enabling interactivity, which is core to robotics in both indoor and outdoor settings, such as autonomous cars, drones, and assistive robotics, as well as upcoming scenarios using mobile and AR/VR devices. In all these applications, we would not only want semantic inference of single images, but importantly, also require understanding of spatial relationships and layouts of objects in 3D environments.

With recent breakthroughs in deep learning and the increasing prominence of convolutional neural networks, the computer vision community has made tremendous progress on analyzing images in the recent years. Specifically, we are seeing rapid progress in the tasks of semantic segmentation [19, 13, 21], object detection [11, 26], and semantic instance segmentation [12]. The primary focus of these impressive works lies in the analysis of visual input from a single image; however, in many real-world computer vision scenarios, we rarely find ourselves in such a single-image setting. Instead, we typically record video streams of RGB input sequences, or as in many robotics and AR/VR applications, we have 3D sensors such as LIDAR or RGB-D cameras.

In particular, in the context of semantic instance segmentation, it is quite disadvantageous to run methods independently on single images given that instance associations must be found across a sequence of RGB input frames. Instead, we aim to infer spatial relationships of objects as part of a semantic 3D map, learning prediction of spatially-consistent semantic labels and the underlying 3D layouts *jointly* from all input views and sensor data. This goal can also be seen as similar to traditional sensor fusion but for deep learning from multiple inputs.

We believe that robustly-aligned and tracked RGB frames, and even depth data, from SLAM and visual odometry provide a unique opportunity in this regard. Here, we can leverage the given mapping between input frames, and thus learn features jointly from all input modalities. In this work, we specifically focus on predicting 3D semantic instances in RGB-D scans, where we capture a series of RGB-D input frames (e.g., from a Kinect Sensor), compute 6DoF rigid poses, and reconstruct 3D models. The core of our method learns semantic features in the 3D domain from both color features, projected into 3D, and geometry features from the signed distance field of the 3D scan. This is realized by a series of 3D convolutions and ResNet blocks. From these semantic features, we obtain anchor bounding box proposals. We process these proposals with a new 3D region proposal network (3D-RPN) and 3D region of interest pooling layer (3D-RoI) to infer object bounding box locations, class labels, and per-voxel instance masks. In order to jointly learn from RGB frames, we leverage their pose alignments with respect to the volumetric grid. We first run a series of 2D convolutions, and then backproject the resulting features into the 3D grid. In 3D, we then join the 2D and 3D features in end-to-end training constrained by bounding box regression, object classification, and semantic instance mask losses.

Our architecture is fully-convolutional, enabling us to efficiently infer predictions on large 3D environments in a single shot. In comparison to state-of-the-art approaches that operate on individual RGB images, such as Mask R-CNN [12], our approach achieves significantly higher accuracy due to the joint feature learning.

To sum up, our contributions are the following:

- We present the first approach leveraging joint 2D-3D end-to-end feature learning on both geometry and RGB input for 3D object bounding box detection and semantic instance segmentation on 3D scans.
- We leverage a fully-convolutional 3D architecture for instance segmentation trained on scene parts, but with single-shot inference on large 3D environments.
- We outperform state-of-the-art by a significant margin, increasing the mAP by 13.5 on real-world data.

2. Related Work

2.1. Object Detection and Instance Segmentation

With the success of convolutional neural network architectures, we have now seen impressive progress on object detection and semantic instance segmentation in 2D images [11, 27, 18, 26, 16, 12, 17]. Notably, Ren et al. [27] introduced an anchor mechanism to predict objectness in a region and regress associated 2D bounding boxes while jointly classifying the object type. Mask R-CNN [12] expanded this work to semantic instance segmentation by predicting a per-pixel object instance masks. An alternative direction for detection is the popular Yolo work [26], which also defines anchors on grid cells of an image.

This progress in 2D object detection and instance segmentation has inspired work on object detection and segmentation in the 3D domain, as we see more and more video and RGB-D data become available. Song et al. proposed Sliding Shapes to predict 3D object bounding boxes from single RGB-D frame input with handcrafted feature design [30], and then expanded the approach to operate on learned features [31]. The latter direction leverages the RGB frame input to improve classification accuracy of detected objects; in contrast to our approach, there is no explicit spatial mapping between RGB and geometry for joint feature learning. An alternative approach is taken by Frustum PointNet [22], where detection is performed a 2D frame and then back-projected into 3D from which final bounding box predictions are refined. Wang et al. [35] base their SGPN approach on semantic segmentation from a PointNet++ variation. They formulate instance segmentation as a clustering problem upon a semantically segmented point cloud by introducing a similarity matrix prediction similar to the idea behind panoptic segmentation [15]. In contrast to these approaches, we explicitly map both multi-view RGB input with 3D geometry in order to jointly infer 3D instance segmentation in an end-to-end fashion.

2.2. 3D Deep Learning

In the recent years, we have seen impressive progress in developments on 3D deep learning. Analogous to the 2D domain, one can define convolution operators on volumetric grids, which for instance embed a surface representation as an implicit signed distance field [4]. With the availability of 3D shape databases [36, 3, 32] and annotated RGB-D datasets [29, 1, 5, 2], these network architectures are now being used for 3D object classification [36, 20, 24, 28], semantic segmentation [5, 34, 6], and object or scene completion [8, 32, 9]. An alternative representation to volumetric grids are the popular point-based architectures, such as PointNet [23] or PointNet++ [25], which leverage a more efficient, although less structured, representation of 3D surfaces. Multi-view approaches have also been proposed to leverage RGB or RGB-D video information. Su et al. proposed one of the first multi-view architectures for object classification by view-pooling over 2D predictions [33], and Kalogerakis et al. recently proposed an approach for shape segmentation by projecting predicted 2D confidence maps onto the 3D shape, which are then aggregated through a CRF [14]. Our approach joins together many of these ideas, leveraging the power of a holistic 3D representation along with features from 2D information by combining them through their explicit spatial mapping.

3. Method Overview

Our approach infers 3D object bounding box locations, class labels, and semantic instance masks on a per-voxel basis in an end-to-end fashion. To this end, we propose a neural network that jointly learns features from both geometry and RGB input. In the following, we refer to bounding box regression and object classification as object detection, and semantic instance mask segmentation for each object as mask prediction.

In Sec. 4, we first introduce the data representation and training data that is used by our approach. Here, we consider synthetic ground truth data from SUNCG [32], as well as manually-annotated real-world data from ScanNetV2 [5]. In Sec. 5, we present the neural network architecture of our 3D-SIS approach. Our architecture is composed of several parts; on the one hand, we have a series of 3D convolutions that operate in voxel grid space of the scanned 3D data. On the other hand, we learn 2D features that we backproject into the voxel grid where we join the features and thus jointly learn from both geometry and RGB data. These features are used to detect object instances; that is, associated bounding boxes are regressed through a 3D-RPN and class labels are predicted for each object following a 3D-ROI pooling layer. For each detected object, features from both the 2D color and 3D geometry are forwarded into a per-voxel instance mask network. Detection and per-voxel instance mask prediction are trained in an end-to-end fashion. In Sec. 6, we describe the training and implementation details of our approach, and in Sec. 7, we evaluate our approach.

4. Training Data

Data Representation We use a truncated sign distance field (TSDF) representation to encode the reconstructed geometry of the 3D scan inputs. The TSDF is stored in a regular volumetric grid with truncation of 3 voxels. In addition to this 3D geometry, we also input spatially associated RGB images. This is feasible since we know the mapping between each image pixel with voxels in the 3D scene grid based on the 6 degree-of-freedom (DoF) poses from the respective 3D reconstruction algorithm.

For the training data, we subdivide each 3D scan into chunks of 4.5m $\times$ 4.5m $\times$ 2.25m, and use a resolution of 96 $\times$ 96 $\times$ 48 voxels per chunk (each voxel stores a TSDF value); i.e., our effective voxel size is $\approx$ 4.69cm^3. In our experiments, for training, we associate 5 RGB images at a resolution of 328x256 pixels in every chunk, with training images selected based on the average voxel-to-pixel coverage of the instances within the region.

Our architecture is fully-convolutional (see Sec. 5), which allows us to run our method over entire scenes in a single shot for inference. Here, the xy-voxel resolution is derived from a given test scene's spatial extent. The z (height) of the voxel grid is fixed to 48 voxels (approximately the height of a room), with the voxel size also fixed at 4.69cm^3. Additionally, at test time, we use all RGB images available for inference. In order to evaluate our algorithm, we use training, validation, test data from synthetic and real-world RGB-D scanning datasets.

Synthetic Data For synthetic training and evaluation, we use the SUNCG [32] dataset. We follow the public train/val/test split, using 5519 train, 40 validation, and 86 test scenes (test scenes are selected to have total volume < 600m^3). From the train and validation scenes, we extract $97,918$ train chunks and 625 validation chunk. Each chunk contains an average of $\approx$ 4.3 object instances. At test time, we take the full scan data of the 86 test scenes.

In order to generate partial scan data from these synthetic scenes, we virtually render them, storing both RGB and depth frames. Trajectories are generated following the virtual scanning approach of [9], but adapted to provide denser camera trajectories to better simulate real-world scanning scenarios. Based on these trajectories, we then generate partial scans as TSDFs through volumetric fusion [4], and define the training data RGB-to-voxel grid image associations based on the camera poses. We use 23 class categories for instance segmentation, defined by their NYU40 class labels; these categories are selected for the most frequently-appearing object types, ignoring the wall and floor categories which do not have well-defined instances.

Real-world Data For training and evaluating our algorithm on real-world scenes, we use the ScanNetV2 [5] dataset. This dataset contains RGB-D scans of 1513 scenes, comprising $\approx$2.5 million RGB-D frames. The scans have been reconstructed using BundleFusion [7]; both 6 DoF pose alignments and reconstructed models are available. Additionally, each scan contains manually-annotated object instance segmentation masks on the 3D mesh. From this data, we derive 3D bounding boxes which we use as constraints for our 3D region proposal.

We follow the public train/val/test split originally proposed by ScanNet of 1045 (train), 156 (val), 312 (test)

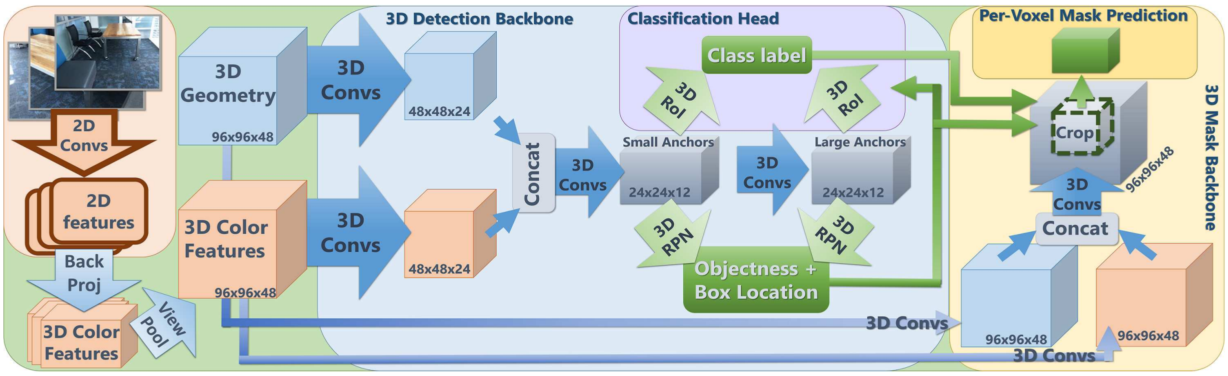


Figure 2: 3D-SIS network architecture. Our architecture is composed of a 3D detection and a 3D mask pipeline. Both 3D geometry and 2D color images are taken as input and used to jointly learn semantic features for object detection and instance segmentation. From the 3D detection backbone, color and geometry features are used to propose the object bounding boxes and their class labels through a 3D-RPN and a 3D-RoI layer. The mask backbone also uses color and geometry features, in addition to the 3D detection results, to predict per-voxel instance masks inside the 3D bounding box.

scenes, respectively. From the train scenes, we extract 108241 chunks, and from the validation scenes, we extract 995 chunks. Note that due to the smaller number of train scans available in the ScanNet dataset, we augment the train scans to have 4 rotations each. We adopt the same 18-class label set for instance segmentation as proposed by the ScanNet benchmark.

Note that our method is agnostic to the respective dataset as long as semantic RGB-D instance labels are available.

5. Network Architecture

Our network architecture is shown in Fig. 2. It is composed of two main components, one for detection, and one for per-voxel instance mask prediction; each of these pipelines has its own feature extraction backbone. Both backbones are composed of a series of 3D convolutions, taking the 3D scan geometry along with the back-projected RGB color features as input. We detail the RGB feature learning in Sec. 5.1 and the feature backbones in Sec. 5.2. The learned 3D features of the detection and mask backbones are then fed into the classification and the voxel-instance mask prediction heads, respectively.

The object detection component of the network comprises the detection backbone, a 3D region proposal network (3D-RPN) to predict bounding box locations, and a 3D-region of interest (3D-RoI) pooling layer followed by classification head. The detection backbone outputs features which are input to the 3D-RPN and 3D-RoI to predict bounding box locations and object class labels, respectively. The 3D-RPN is trained by associating predefined anchors with ground-truth object annotations; here, a per-anchor loss defines whether an object exists for a given anchor. If it does, a second loss regresses the 3D object bounding box; if not, no additional loss is considered. In addition, we classify the the object class of each 3D bounding box. For the per-voxel instance mask prediction network (see Sec. 5.4), we use both the input color and geometry as well as the predicted bounding box location and class label. The cropped feature channels are used to create a mask prediction which has n channels for the n semantic class labels, and the final mask prediction is selected from these channels using the previously predicted class label. We optimize for the instance mask prediction using a binary cross entropy loss. Note that we jointly train the backbones, bounding box regression, classification, and per-voxel mask predictions end-to-end; see Sec. 6 for more detail. In the following, we describe the main components of our architecture design, for more detail regarding exact filter sizes, etc., we refer to the supplemental material.

5.1. Back-projection Layer for RGB Features

In order to jointly learn from RGB and geometric features, one could simply assign a single RGB value to each voxel. However, in practice, RGB image resolutions are significantly higher than the available 3D voxel resolution due to memory constraints. This 2D-3D resolution mismatch would make learning from a per-voxel color rather inefficient. Inspired by the semantic segmentation work of Dai et al. [6], we instead leverage a series of 2D convolutions to summarize RGB signal in image space. We then define a back-projection layer and map these features on top of the associated voxel grid, which are then used for both object detection and instance segmentation.

To this end, we first pre-train a 2D semantic segmentation network based on the ENet architecture [21]. The 2D architecture takes single 256×328 RGB images as input, and is trained on a semantic classification loss using the NYUv2 40 label set. From this pre-trained network, we extract a feature encoding of dimension 32×41 with 128 channels from the encoder. Using the corresponding depth image, camera intrinsics, and 6DoF poses, we then back-project each of these features back to the voxel grid (still 128 channels); the projection is from 2D pixels to 3D voxels. In order to combine features from multiple views, we perform view pooling through an element-wise max pooling over all RGB images available.

For training, the voxel volume is fixed to $96 \times 96 \times 48$ voxels, resulting in a $128 \times 96 \times 96 \times 48$ back-projected RGB feature grid in 3D; here, we use 5 RGB images for each training chunk (with image selection based on average 3D instance coverage). At test time, the voxel grid resolution is dynamic, given by the spatial extent of the environment; here, we use all available RGB images. The grid of projected features is processed by a set of 3D convolutions and is subsequently merged with the geometric features.

In ScanNet [5], the camera poses and intrinsics are provided; we use them directly for our back-projection layer. For SUNCG [32], extrinsics and intrinsics are given by the virtual scanning path. Note that our method is agnostic to the used 2D network architecture.

5.2. 3D Feature Backbones

For jointly learning geometric and RGB features for both instance detection and segmentation, we propose two 3D feature learning backbones. The first backbone generates features for detection, and takes as input the 3D geometry and back-projected 2D features (see Sec. 5.1).

Both the geometric input and RGB features are processed symmetrically with a 3D ResNet block before joining them together through concatenation. We then apply a 3D convolutional layer to reduce the spatial dimension by a factor of 4, followed by a 3D ResNet block (e.g., for an input train chunk of $96 \times 96 \times 48$, we obtain a features of size $24 \times 24 \times 12$). We then apply another 3D convolutional layer, maintaining the same spatial dimensions, to provide features maps with larger receptive fields. We define anchors on these two feature maps, splitting the anchors into 'small' and 'large' anchors (small anchors $< 1\text{m}^3$), with small anchors associated with the first feature map of smaller receptive field and large anchors associated with the second feature map of larger receptive field. For selecting anchors, we apply k-means algorithm (k=14) on the ground-truth 3D bounding boxes in first 10k chunks. These two levels of features maps are then used for the final steps of object detection: 3D bounding box regression and classification.

The instance segmentation backbone also takes the 3D geometry and the back-projected 2D CNN features as input. The geometry and color features are first processed independently with two 3D convolutions, and then concatenated channel-wise and processed with another two 3D convolutions to produce a mask feature map prediction. Note that for the mask backbone, we maintain the same spatial resolution through all convolutions, which we found to be critical for obtaining high accuracy for the voxel instance predictions. The mask feature map prediction is used as input to predict the final instance mask segmentation.

In contrast to single backbone, we found that this two-backbone structure both converged more easily and produced significantly better instance segmentation performance (see Sec. 6 for more details about the training scheme for the backbones).

5.3. 3D Region Proposals and 3D-RoI Pooling for Detection

Our 3D region proposal network (3D-RPN) takes input features from the detection backbone to predict and regress 3D object bounding boxes. From the detection backbone we obtain two feature maps for small and large anchors, which are separately processed by the 3D-RPN. For each feature map, the 3D-RPN uses a $1 \times 1 \times 1$ convolutional layer to reduce the channel dimension to $2 \times N_{\text{anchors}}$, where $N_{\text{anchors}} = (3, 11)$ for small and large anchors, respectively. These represent the positive and negative scores of objectness of each anchor. We apply a non-maximum suppression on these region proposals based on their objectness scores. The 3D-RPN then uses another $1 \times 1 \times 1$ convolutional layer to predict feature maps of $6 \times N_{\text{anchors}}$, which represent the 3D bounding box locations as $(\Delta_x, \Delta_y, \Delta_z, \Delta_w, \Delta_h, \Delta_l)$, defined in Eq. 1.

In order to determine the ground truth objectiveness and associated 3D bounding box locations of each anchor during training, we perform anchor association. Anchors are associated with ground truth bounding boxes by their IoU: if the IoU > 0.35, we consider an anchor to be positive (and it will be regressed to the associated box), and if the IoU < 0.15, we consider an anchor to be negative (and it will not be regressed to any box). We use a two-class cross entropy loss to measure the objectiveness, and for the bounding box regression we use a Huber loss on the prediction $(\Delta_x, \Delta_y, \Delta_z, \Delta_w, \Delta_h, \Delta_l)$ against the log ratios of the ground truth box and anchors $(\Delta_x^{gt}, \Delta_y^{gt}, \Delta_z^{gt}, \Delta_w^{gt}, \Delta_h^{gt}, \Delta_l^{gt})$, where

$$\Delta_x = \frac{\mu - \mu_{anchor}}{\phi_{anchor}} \qquad \Delta_w = \ln\left(\frac{\phi}{\phi_{anchor}}\right) \tag{1}$$

where μ is the box center point and ϕ is the box width.

Using the predicted bounding box locations, we can then crop out the respective features from the global feature map. We then unify these cropped features to the same dimensionality using our 3D Region of Interest (3D-RoI) pooling layer. This 3D-RoI pooling layer pools the cropped feature maps into $4 \times 4 \times 4$ blocks through max pooling operations. These feature blocks are then linearized for input to object classification, which is performed with an MLP.

	cab	bed	chair	sofa	tabl	door	wind	bkshf	cntr	desk	shlf	curt	drsr	mirr	tv	nigh	toil	sink	lamp	bath	ostr	ofurn	oprop	**avg**
Seg-Cluster	16.8	16.2	15.6	11.8	14.5	10.0	11.7	27.2	20.0	25.7	10.0	0.0	15.0	0.0	20.0	27.8	39.5	22.9	10.7	38.9	10.4	0.0	12.3	16.4
Mask R-CNN [12]	14.9	19.0	19.5	13.5	12.2	11.7	14.2	35.0	15.7	18.3	13.7	0.0	24.4	**23.1**	**26.0**	**28.8**	51.2	28.1	14.7	32.2	11.4	10.7	19.5	19.9
SGPN [35]	18.6	39.2	28.5	46.5	26.7	**21.8**	**15.9**	0.0	24.9	23.9	16.3	**20.8**	15.1	10.7	0.0	17.7	35.1	37.0	**22.9**	34.2	17.7	31.5	13.9	22.5
Ours(geo only)	23.2	**78.6**	47.7	63.3	37.0	19.6	0.0	0.0	21.3	34.4	16.8	0.0	16.7	0.0	10.0	22.8	**59.7**	49.2	10.0	77.2	10.0	0.0	**19.3**	26.8
Ours(geo+1view)	22.2	70.8	48.5	**66.6**	**44.4**	10.0	0.0	**63.9**	25.8	32.2	17.8	0.0	25.3	0.0	0.0	14.7	37.0	55.5	20.5	58.2	**18.0**	20.0	17.9	29.1
Ours(geo+3views)	**26.5**	78.4	48.2	59.5	42.8	26.1	0.0	30.0	22.7	**39.4**	17.3	0.0	**36.2**	0.0	10.0	10.0	37.0	50.8	16.8	**59.3**	10.0	**36.4**	17.8	29.4
Ours(geo+5views)	20.5	69.4	**56.2**	64.5	43.8	17.8	0.0	30.0	**32.3**	33.5	**21.0**	0.0	34.2	0.0	10.0	20.0	56.7	**56.2**	17.6	56.2	10.0	35.5	17.8	**30.6**

Table 1: 3D instance segmentation on synthetic scans from SUNCG [32]. We evaluate the mean average precision with IoU threshold of 0.25 over 23 classes. Our joint color-geometry feature learning enables us to achieve more accurate instance segmentation performance.

5.4. Per-Voxel 3D Instance Segmentation

We perform instance mask segmentation using a separate mask backbone, which similarly as the detection backbone, takes as input the 3D geometry and projected RGB features. However, for mask prediction, the 3D convolutions maintain the same spatial resolutions, in order to maintain spatial correspondence with the raw inputs, which we found to significantly improve performance. We then use the predicted bounding box location from the 3D-RPN to crop out the associated mask features from the mask backbone, and compute a final mask prediction with a 3D convolution to reduce the feature dimensionality to n for n semantic class labels; the final mask prediction is the c^{th} channel for predicted object class c. During training, since predictions from the detection pipeline can be wrong, we only train on predictions whose predicted bounding box overlaps with the ground truth bounding box with at least 0.5 IoU. The mask targets are defined as the ground-truth mask in the overlapping region of the ground truth box and proposed box.

6. Training

To train our model, we first train the detection backbone and 3D-RPN. After pre-training these parts, we add the 3D-RoI pooling layer and object classification head, and train these end-to-end. Then, we add the per-voxel instance mask segmentation network along with the associated backbone. In all training steps, we always keep the previous losses (using 1:1 ratio between all losses), and train everything end-to-end. We found that a sequential training process resulted in more stable convergence and higher accuracy.

We use an SGD optimizer with learning rate 0.001, momentum 0.9 and batch size 64 for 3D-RPN, 16 for classification, 16 for mask prediction. The learning rate is divided by 10 every 100k steps. We use a non-maximum suppression for proposed boxes with threshold of 0.7 for training and 0.3 for test. Our network is implemented with PyTorch and runs on a single Nvidia GTX1080Ti GPU. The object detection components of the network are trained end-to-end for 10 epochs ($\approx$ 24 hours). After adding in the mask backbone, we train for an additional 5 epochs ($\approx$ 16 hours). For mask training, we also use ground truth bounding boxes to augment the learning procedure.

7. Results

We evaluate our approach on both 3D detection and instance segmentation predictions, comparing to several state-of-the-art approaches, on synthetic scans of SUNCG [32] data and real-world scans from the ScanNetV2 dataset [5]. To compare to previous approaches that operate on single RGB or RGB-D frames (Mask R-CNN [12], Deep Sliding Shapes [31], Frustum PointNet [22]), we first obtain predictions on each individual frame, and then merge all predictions together in the 3D space of the scene, merging predictions if the predicted class labels match and the IoU > 0.5. We further compare to SGPN [35] which performs instance segmentation on 3D point clouds. For both detection and instance segmentation tasks, we project all results into a voxel space of 4.69cm voxels and evaluate them with a mean average precision metric. We additionally show several variants of our approach for learning from both color and geometry features, varying the number of color views used during training. We consistently find that training on more color views improves both the detection and instance segmentation performance.

	cab	bed	chair	sofa	tabl	door	wind	bkshf	pic	cntr	desk	curt	fridg	showr	toil	sink	bath	ofurn	**avg**
Mask R-CNN [12]	5.3	0.2	0.2	10.7	2.0	4.5	0.6	0.0	**23.8**	0.2	0.0	2.1	6.5	0.0	2.0	1.4	33.3	2.4	5.8
SGPN [35]	6.5	39.0	27.5	35.1	16.8	8.7	13.8	16.9	1.4	2.9	0.0	6.9	2.7	0.0	43.8	11.2	20.8	4.3	14.3
MTML	2.7	**61.4**	39.0	50.0	10.5	10.0	0.3	**33.7**	0.0	0.0	0.1	**11.8**	16.7	14.3	57.0	4.6	66.7	2.8	21.2
3D-BEVIS [10]	3.5	56.6	39.4	60.4	18.1	9.9	17.1	7.6	2.5	2.7	9.8	3.5	9.8	37.5	85.4	12.6	66.7	3.0	24.8
R-PointNet [37]	**34.8**	40.5	**58.9**	39.6	**27.5**	28.3	**24.5**	31.1	2.8	**5.4**	**12.6**	6.8	21.9	21.4	82.1	**33.1**	50.0	**29.0**	30.6
3D-SIS (Ours)	13.4	55.4	58.7	**72.8**	22.4	**30.7**	18.1	31.9	0.6	0.0	12.1	0.0	**54.1**	**100.0**	**88.9**	4.5	**66.7**	21.0	**36.2**

Table 2: Instance segmentation results on the official ScanNetV2 3D semantic instance benchmark (hidden test set). Our final model (geo+5views) significantly outperforms previous (Mask R-CNN, SGPN) and concurrent (MTML, 3D-BEVIS, R-PointNet) state-of-the-art methods in mAP@0.5. ScanNetV2 benchmark data accessed on 12/17/2018.

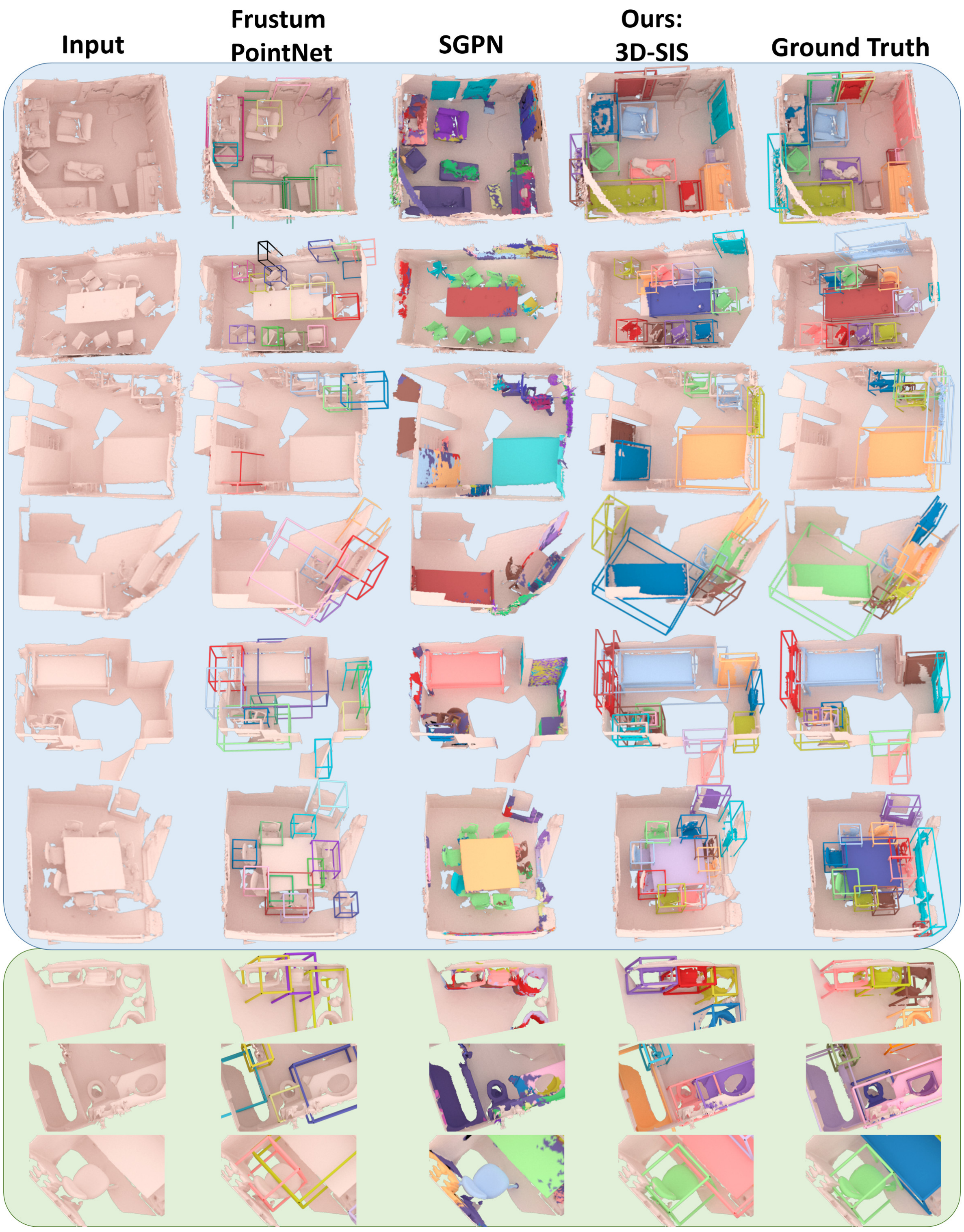


Figure 3: Qualitative comparison of 3D object detection and instance segmentation on ScanNetV2 [5] (full scans above; close-ups below). Our joint color-geometry feature learning combined with our fully-convolutional approach to inference on full test scans at once enables more accurate and semantically coherent predictions. Note that different colors represent different instances, and the same instances in the ground truth and predictions are not necessarily the same color.

	cab	bed	chair	sofa	tabl	door	wind	bkshf	pic	cntr	desk	curt	fridg	showr	toil	sink	bath	ofurn	**avg**
Seg-Cluster	11.8	13.5	18.9	14.6	13.8	11.1	11.5	11.7	0.0	13.7	12.2	12.4	11.2	18.0	19.5	18.9	16.4	12.2	13.4
Mask R-CNN [12]	15.7	15.4	16.4	16.2	14.9	12.5	11.6	11.8	**19.5**	13.7	14.4	14.7	21.6	18.5	25.0	24.5	24.5	16.9	17.1
SGPN [35]	20.7	31.5	31.6	40.6	**31.9**	16.6	**15.3**	13.6	0.0	17.4	14.1	22.2	0.0	0.0	72.9	**52.4**	0.0	18.6	22.2
Ours(geo only)	22.1	48.2	64.4	52.2	16.0	13.4	0.0	17.2	0.0	20.7	17.4	13.9	23.6	33.0	45.2	47.7	61.3	14.6	28.3
Ours(geo+1view)	25.4	60.3	**66.2**	52.1	31.7	27.6	10.1	16.9	0.0	21.4	30.9	18.4	22.6	16.0	70.5	44.5	37.5	20.0	31.8
Ours(geo+3views)	28.3	52.3	65.0	**66.5**	31.4	**27.9**	10.1	**17.9**	0.0	20.3	**36.3**	20.1	28.1	31.0	68.6	41	**66.8**	**24.0**	35.3
Ours(geo+5views)	**32.0**	**66.3**	65.3	56.4	29.4	26.7	10.1	16.9	0.0	**22.1**	35.1	**22.6**	**28.6**	**37.2**	**74.9**	39.6	57.6	21.1	**35.7**

Table 3: 3D instance segmentation on ScanNetV2 [5] with mAP@0.25 on 18 classes. Our explicit leveraging of spatial mapping between 3D geometry and color features extracted through 2D CNNs enables significantly improved performance.

7.1. 3D Instance Analysis on Synthetic Scans

We evaluate 3D detection and instance segmentation on virtual scans taken from the synthetic SUNCG dataset [32], using 23 class categories. Table 4 shows 3D detection performance compared to state-of-the-art approaches which operate on single frames. Table 1 shows a quantitative evaluation of our approach, the SGPN for point cloud instance segmentation [35], their proposed Seg-Cluster baseline, and Mask R-CNN [12] projected into 3D. For both tasks, our joint color-geometry approach along with a global view of the 3D scenes at test time enables us to achieve significantly improved detection and segmentation results.

	mAP@0.25	mAP@0.5
Deep Sliding Shapes [30]	12.8	6.2
Mask R-CNN 2D-3D [12]	20.4	10.5
Frustum PointNet [22]	24.9	10.8
Ours – 3D-SIS (geo only)	27.8	21.9
Ours – 3D-SIS (geo+1view)	30.9	23.8
Ours – 3D-SIS (geo+3views)	31.3	24.2
Ours – 3D-SIS (geo+5views)	**32.2**	**24.7**

Table 4: 3D detection in SUNCG [32], using mAP over 23 classes. Our holistic approach and the combination of color and geometric features result in significantly improved detection results over previous approaches which operate on individual input frames.

7.2. 3D Instance Analysis on Real-World Scans

We further evaluate our approach on ScanNet dataset [5], which contains 1513 real-world scans. For training and evaluation, we use ScanNetV2 annotated ground truth as well as the proposed 18-class instance benchmark. We show qualitative results in Figure 3. In Table 5, we quantitatively evaluate our object detection against Deep Sliding Shapes and Frustum PointNet, which operate on RGB-D frame, as well as Mask R-CNN [12] projected to 3D. Our fully-convolutional approach enabling inference on full test scenes achieves significantly better detection performance. Table 3 shows our 3D instance segmentation in comparison to SGPN instance segmentation [35], their proposed Seg-Cluster baseline, and Mask R-CNN [12] projected into 3D. Our formulation for learning from both color and geometry features brings notable improvement over state of the art.

	mAP@0.25	mAP@0.5
Deep Sliding Shapes [30]	15.2	6.8
Mask R-CNN 2D-3D [12]	17.3	10.5
Frustum PointNet [22]	19.8	10.8
Ours – 3D-SIS (geo only)	27.6	16.0
Ours – 3D-SIS (geo+1view)	35.1	18.7
Ours – 3D-SIS (geo+3views)	36.6	19.0
Ours – 3D-SIS (geo+5views)	**40.2**	**22.5**

Table 5: 3D detection on ScanNetV2 [5], using mAP over 18 classes. In contrast to previous approaches operating on individual frames, our approach achieves significantly improved performance.

Finally, we evaluate our model on the ScanNetV2 3D instance segmentation benchmark on the hidden test set; see Table 2. Our final model (geo+5views) significantly outperforms previous (Mask R-CNN [12], SGPN [35]) and concurrent (MTML, 3D-BEVIS [10], R-PointNet [37]) state-of-the-art methods in mAP@0.5. ScanNetV2 benchmark data was accessed on 12/17/2018.

8. Conclusion

In this work, we introduce 3D-SIS, a new approach for 3D semantic instance segmentation of RGB-D scans, which is trained in an end-to-end fashion to detect object instances and infer a per-voxel 3D semantic instance segmentation. The core of our method is to jointly learn features from RGB and geometry data using multi-view RGB-D input recorded with commodity RGB-D sensors. The network is fully-convolutional, and thus can run efficiently in a single shot on large 3D environments. In comparison to existing state-of-the-art methods that typically operate on single RGB frame, we achieve significantly better 3D detection and instance segmentation results, improving on mAP by over 13. We believe that this is an important insight to a wide range of computer vision applications given that many of them now capture multi-view RGB and depth streams; e.g., autonomous cars, AR/VR applications, etc..

Acknowledgements

This work was supported by a Google Research Grant, an Nvidia Professor Partnership, a TUM Foundation Fellowship, a TUM-IAS Rudolf Mößbauer Fellowship, and the ERC Starting Grant *Scan2CAD (804724)*.

References

[1] Iro Armeni, Sasha Sax, Amir R Zamir, and Silvio Savarese. Joint 2d-3d-semantic data for indoor scene understanding. *arXiv preprint arXiv:1702.01105*, 2017. 2

[2] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3D: Learning from RGB-D data in indoor environments. *International Conference on 3D Vision (3DV)*, 2017. 2

[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. 2

[4] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312. ACM, 1996. 2, 3

[5] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017. 2, 3, 5, 6, 7, 8

[6] Angela Dai and Matthias Nießner. 3dmv: Joint 3d-multi-view prediction for 3d semantic scene segmentation. *arXiv preprint arXiv:1803.10409*, 2018. 2, 4

[7] Angela Dai, Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Christian Theobalt. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ACM Transactions on Graphics (TOG)*, 36(3):24, 2017. 3

[8] 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

[9] Angela Dai, Daniel Ritchie, Martin Bokeloh, Scott Reed, Jürgen Sturm, and Matthias Nießner. Scancomplete: Large-scale scene completion and semantic segmentation for 3d scans. *arXiv preprint arXiv:1712.10215*, 2018. 2, 3

[10] Cathrin Elich, Francis Engelmann, Jonas Schult, Theodora Kontogianni, and Bastian Leibe. 3D-BEVIS: Birds-Eye-View Instance Segmentation. *CoRR*, abs/1904.02199, 2019. 6, 8

[11] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. 1, 2

[12] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2980–2988. IEEE, 2017. 1, 2, 6, 8

[13] Forrest Iandola, Matt Moskewicz, Sergey Karayev, Ross Girshick, Trevor Darrell, and Kurt Keutzer. Densenet: Implementing efficient convnet descriptor pyramids. *arXiv preprint arXiv:1404.1869*, 2014. 1

[14] Evangelos Kalogerakis, Melinos Averkiou, Subhransu Maji, and Siddhartha Chaudhuri. 3d shape segmentation with projective convolutional networks. *Proc. CVPR, IEEE*, 2, 2017. 3

[15] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollár. Panoptic segmentation. *arXiv preprint arXiv:1801.00868*, 2018. 2

[16] Tsung-Yi Lin, Piotr Dollár, Ross B Girshick, Kaiming He, Bharath Hariharan, and Serge J Belongie. Feature pyramid networks for object detection. In *CVPR*, volume 1, page 4, 2017. 2

[17] Tsung-Yi Lin, Priyal Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *IEEE transactions on pattern analysis and machine intelligence*, 2018. 2

[18] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. 2

[19] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015. 1

[20] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 922–928. IEEE, 2015. 2

[21] Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation. *arXiv preprint arXiv:1606.02147*, 2016. 1, 4

[22] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. *arXiv preprint arXiv:1711.08488*, 2017. 2, 6, 8

[23] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 1(2):4, 2017. 2

[24] Charles Ruizhongtai Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2016. 2

[25] 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*, pages 5105–5114, 2017. 2

[26] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. 1, 2

[27] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 2

[28] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 2

[29] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 567–576, 2015. 2

[30] Shuran Song and Jianxiong Xiao. Sliding shapes for 3d object detection in depth images. In *European conference on computer vision*, pages 634–651. Springer, 2014. 2, 8

[31] Shuran Song and Jianxiong Xiao. Deep sliding shapes for amodal 3d object detection in rgb-d images. *arXiv preprint arXiv:1511.02300*, 2015. 2, 6

[32] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. *Proceedings of 30th IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 2, 3, 5, 6, 8

[33] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 945–953, 2015. 3

[34] Maxim Tatarchenko, Jaesik Park, Vladlen Koltun, and Qian-Yi Zhou. Tangent convolutions for dense prediction in 3d. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3887–3896, 2018. 2

[35] Weiyue Wang, Ronald Yu, Qiangui Huang, and Ulrich Neumann. Sgpn: Similarity group proposal network for 3d point cloud instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2569–2578, 2018. 2, 6, 8

[36] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1912–1920, 2015. 2

[37] Li Yi, Wang Zhao, He Wang, Minhyuk Sung, and Leonidas Guibas. Gspn: Generative shape proposal network for 3d instance segmentation in point cloud. *arXiv preprint arXiv:1812.03320*, 2018. 6, 8

RevealNet: Seeing Behind Objects in RGB-D Scans

Ji Hou Angela Dai Matthias Nießner
Technical University of Munich

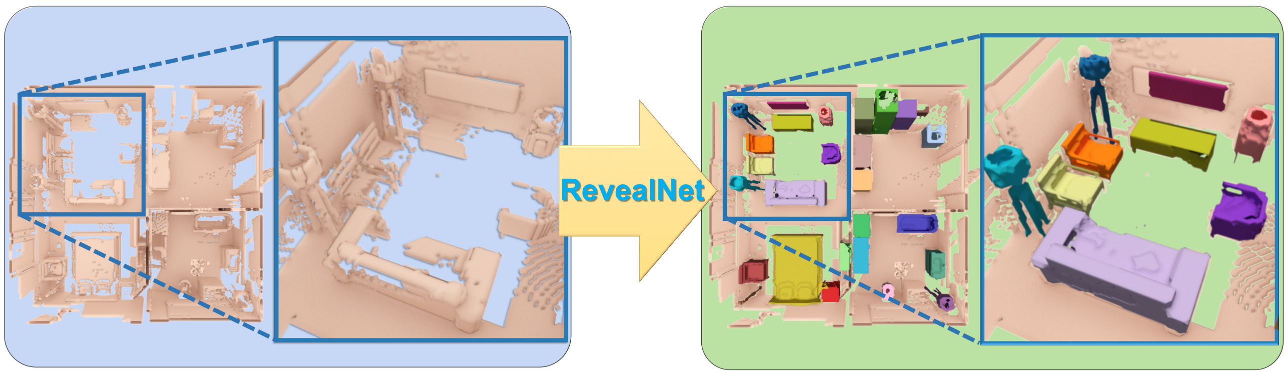


Figure 1: RevealNet takes an RGB-D scan as input and learns to "see behind objects": from the scan's color images and geometry (encoded as a TSDF), objects in the observed scene are detected (as 3D bounding boxes and class labels) and for each object, the complete geometry of that object is predicted as per-instance masks (in both seen and unseen regions).

Abstract

During 3D reconstruction, it is often the case that people cannot scan each individual object from all views, resulting in missing geometry in the captured scan. This missing geometry can be fundamentally limiting for many applications, e.g., a robot needs to know the unseen geometry to perform a precise grasp on an object. Thus, we introduce the task of semantic instance completion: from an incomplete RGB-D scan of a scene, we aim to detect the individual object instances and infer their complete object geometry. This will open up new possibilities for interactions with objects in a scene, for instance for virtual or robotic agents. We tackle this problem by introducing RevealNet, a new data-driven approach that jointly detects object instances and predicts their complete geometry. This enables a semantically meaningful decomposition of a scanned scene into individual, complete 3D objects, including hidden and unobserved object parts. RevealNet is an end-to-end 3D neural network architecture that leverages joint color and geometry feature learning. The fully-convolutional nature of our 3D network enables efficient inference of semantic instance completion for 3D scans at scale of large indoor environments in a single forward pass. We show that predicting complete object geometry improves both 3D detection and instance segmentation performance. We evaluate on both real and synthetic scan benchmark data for the new task, where we outperform state-of-the-art approaches by over 15 *in mAP@0.5 on ScanNet, and over 18 in mAP@0.5 on SUNCG.*

1. Introduction

Understanding 3D environments is fundamental to many tasks spanning computer vision, graphics, and robotics. In particular, in order to effectively navigate, and moreover interact with an environment, an understanding of the geometry of a scene and the objects it comprises of is essential. This is in contrast to the partial nature of reconstructed RGB-D scans; e.g., due to sensor occlusions. For instance, for a robot exploring an environment, it needs to infer where objects are as well as what lies behind the objects it sees in order to efficiently navigate or perform tasks like grasping. That is, it needs not only instance-level knowledge of objects in the scene, but to also estimate the missing geometry of these objects. Additionally, for content creation or mixed reality applications, captured scenes must be decomposable into their complete object components, in order to enable applications such as scene editing or virtual-real object interactions; i.e., it is often insufficient to segment object instances only for observed regions.

Thus, we aim to address this task of "seeing behind objects," which we refer to as *semantic instance completion*: predicting object detection as well as instance-level completion for an input partial 3D scan of a scene. Previous approaches have addressed these tasks independently: 3D instance segmentation segments object instances from the visible surface of a partial scan [43, 14, 46, 45, 18, 26, 23, 8], and 3D scan completion approaches predict the full scene geometry [39, 7], but lack the notion of individual objects. In contrast, our approach focuses on the instance level, as knowledge of instances is essential towards enabling interaction with the objects in an environment.

In addition, the task of semantic instance completion is not only important towards enabling object-level understanding and interaction with 3D environments, but we also show that the prediction of complete object geometry informs the task of semantic instance segmentation. Thus, in order to address the task of semantic instance completion, we propose to consider instance detection and object completion in an end-to-end, fully differentiable fashion.

From an input RGB-D scan of a scene, our RevealNet model sees behind objects to predict each object's complete geometry. First, object bounding boxes are detected and regressed, followed by object classification and then a prediction of complete object geometry. Our approach leverages a unified backbone from which instance detection and object completion are predicted, enabling information to flow from completion to detection. We incorporate features from both color image and 3D geometry of a scanned scene, as well as a fully-convolutional design in order to effectively predict the complete object decomposition of varying-sized scenes. To address the task of semantic instance completion for real-world scans, where ground truth complete geometry is not readily available, we further introduce a new semantic instance completion benchmark for ScanNet [4], leveraging the Scan2CAD [1] annotations to evaluate semantic instance completion (and semantic instance segmentation).

In summary, we present a fully-convolutional, end-to-end 3D CNN formulation to predict 3D instance completion that outperforms state-of-the-art, decoupled approaches to semantic instance completion by 15.8 in mAP@0.5 on real-world scan data, and 18.5 in mAP@0.5 on synthetic data:

- We introduce the task of *semantic instance completion* for 3D scans;
- we propose a novel, end-to-end 3D convolutional network which predicts 3D semantic instance completion as object bounding boxes, class labels, and complete object geometry,
- and we show that semantic instance completion task can benefit semantic instance segmentation and detection performance.

2. Related Work

Object Detection and Instance Segmentation Recent advances in convolutional neural networks have now begun to drive impressive progress in object detection and instance segmentation for 2D images [9, 33, 23, 32, 20, 13, 21]. Combined with the increasing availability of synthetic and real-world 3D data [4, 39, 2], we are now seeing more advances in object detection [37, 38, 31, 30] for 3D. Sliding Shapes [37] predicted 3D object bounding boxes from a depth image, designing handcrafted features to detect objects in a sliding window fashion. Deep Sliding Shapes [38] then extended this approach to leverage learned features for object detection in a single RGB-D frame. Frustum PointNet [31] tackles the problem of object detection for an RGB-D frame by first detecting object in the 2D image before projecting the detected boxes into 3D to produce final refined box predictions. VoteNet [30] propose a reformulation of Hough voting in the context of deep learning through an end-to-end differentiable architecture for 3D detection purpose.

Recently, several approaches have been introduced to perform 3D instance segmentation, applicable to single or multi-frame RGB-D input. Wang et al. [43] introduced SGPN to operate on point clouds by clustering semantic segmentation predictions. Li et al. [46] leverages an object proposal-based approach to predict instance segmentation for a point cloud. Simultaneously, Hou et al. [14] presented an approach leveraging joint color-geometry feature learning for detection and instance segmentation on volumetric data. Lahoud et al. [18] proposes to use multi-task losses to predict instance segmentation. Yang et al. [45] and Liu et al. [22] both use bottom-up methods to predict instance segmentation for a point cloud. Our approach also leverages an anchor-based object proposal mechanism for detection, but we leverage object completion to predict instance completion, as well as show that completing object-level geometry can improve detection and instance segmentation performance on volumetric data.

3D Scan Completion Scan completion of 3D shapes has been a long-studied problem in geometry processing, particularly for cleaning up broken mesh models. In this context, traditional methods have largely focused on filling small holes by locally fitting geometric primitives, or through continuous energy minimization [40, 27, 47]. Surface reconstruction approaches on point cloud inputs [15, 16] can also be applied in this fashion to locally optimize for missing surfaces. Other shape completion approaches leverage priors such as symmetry and structural priors [42, 24, 29, 36, 41], or CAD model retrieval [25, 34, 17, 19, 35] to predict the scan completion.

Recently, methods leveraging generative deep learning have been developed to predict the complete geometry of 3D shapes [44, 6, 11, 12]. Song et al. [39] extended beyond shapes to predicting the voxel occupancy for a single depth frame leveraging the geometric occupancy prediction to achieve improved 3D semantic segmentation. Recently, Dai et al. [7] presented a first approach for data-driven scan completion of full 3D scenes, leveraging a fully-convolutional, autoregressive approach to predict complete geometry along with 3D semantic segmentation. Both Song et al. [39] and Dai et al. [7] show that inferring the complete scan geometry can improve 3D semantic segmentation. With our approach for 3D semantic instance completion, this task not only enables new applications requiring instance-based knowledge of a scene (e.g., virtual or robotic interactions with objects in a scene), but we also show that instance segmentation can benefit from instance completion.

3. Method Overview

Our network takes as input an RGB-D scan, and learns to join together features from both the color images as well as the 3D geometry to inform the semantic instance completion. The architecture is shown in Fig. 2.

The input 3D scan is encoded as a truncated signed distance field (TSDF) in a volumetric grid. To combine this with color information from the RGB images, we first extract 2D features using 2D convolutional layers on the RGB images, which are then back-projected into a 3D volumetric grid, and subsequently merged with geometric features extracted from the geometry. The joint features are then fed into an encoder-decoder backbone, which leverages a series of 3D residual blocks to learn the representation for the task of semantic instance completion. Objects are detected through anchor proposal and bounding box regression; these predicted object boxes are then used to crop and extract features from the backbone encoder to predict the object class label as well as the complete object geometry for each detected object as per-voxel occupancies.

We adopt in total five losses to supervise the learning process illustrated in Fig. 2. Detection contains three losses: (1) objectness using binary cross entropy to indicate that there is an object, (2) box location using a Huber loss to regress the 3D bounding box locations, and (3) classification of the class label loss using cross entropy. Following detection, the completion head contains two losses: per-instance completion loss using binary cross entropy to predict per-voxel occupancies, and a proxy completion loss using binary cross entropy to classify the surface voxels belonging to all objects in the scene.

Our method operates on a unified backbone for detection followed by instance completion, enabling object completion to inform the object detection process; this results in effective 3D detection as well as instance completion. Its fully-convolutional nature enables us to train on cropped chunks of 3D scans but test on a whole scene in a single forward pass, resulting in an efficient decomposition of a scan into a set of complete objects.

4. Network Architecture

From an RGB-D scan input, our network operates on the scan's reconstructed geometry, encoded as a TSDF in a volumetric grid, as well as the color images. To jointly learn from both color and geometry, color features are first extracted in 2D with a 2D semantic segmentation network [28], and then back-projected into 3D to be combined with the TSDF features, similar to [5, 14]. This enables complementary semantic features to be learned from both data modalities. These features are then input to the backbone of our network, which is structured in an encoder-decoder style.

The encoder-decoder backbone is composed of a series of five 3D residual blocks, which generates five volumetric feature maps $\mathbb{F} = \{f_i | i = 1 \dots 5\}$. The encoder results in a reduction of spatial dimension by a factor of 4, and symmetric decoder results in an expansion of spatial dimension by a factor of 4. Skip connections link spatially-corresponding encoder and decoder features. For a more detailed description of the network architecture, we refer to the appendix.

4.1. Color Back-Projection

As raw color data is often of much higher resolution than 3D geometry, to effectively learn from both color and geometry features, we leverage color information by back-projecting 2D CNN features learned from RGB images to 3D, similar to [5, 14]. For each voxel location $v_i = (x, y, z)$ in the 3D volumetric grid, we find its pixel location $p_i = (x, y)$ in 2D views by camera intrinsic and extrinsic matrices. We assign the voxel feature at location v_i with the learned 2D CNN feature vector at p_i. To handle multiple image observations of the same voxel v_i, we apply element-wise view pooling; this also allows our approach to handle a varying number of input images. Note that this back-projection is differentiable, allowing our model to be trained end-to-end and benefit from both RGB and geometric signal.

4.2. Object Detection

For object detection, we predict the bounding box of each detected object as well as the class label. To inform the detection, features are extracted from feature maps F_2 and F_3 of the backbone encoder. We define two set of anchors on these two features maps, $A_s = \{a_i | i = 1 \dots N_s\}$ and

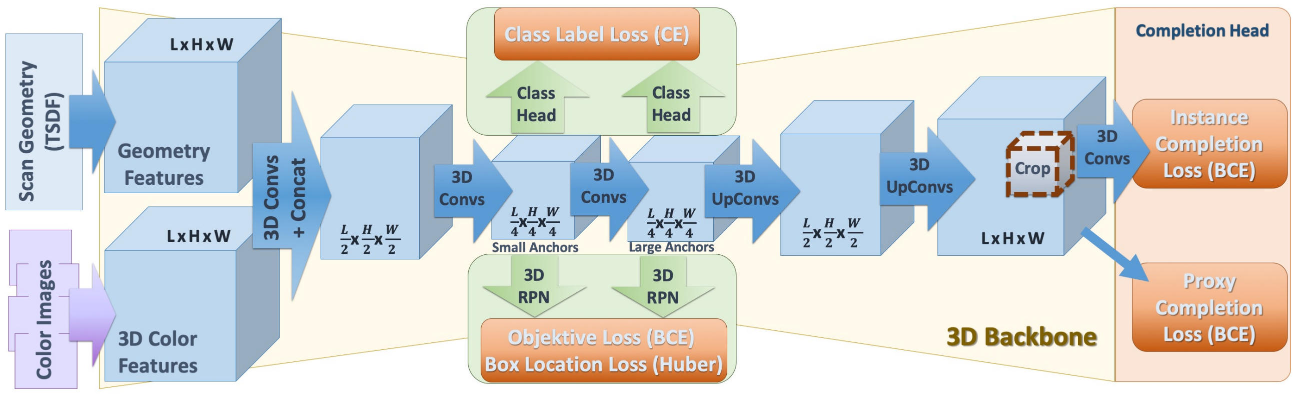


Figure 2: Our RevealNet network architecture takes an RGB-D scan as input. Color images are processed with 2D convolutions to spatially compress the information before back-projecting into 3D, to be merged with the 3D geometry features of the scan (following [5, 14]). These joint features are used for object detection (as 3D bounding boxes and class labels) followed by per-instance geometric completion, for the task of semantic instance completion. In contrast to [14], which leverages separate backbones for detection and instance segmentation, our network maintains one unified backbone for both detection and completion head, allowing the completion task to directly inform the detection parameters.

$A_b = \{a_i | i = 1 \dots N_b\}$ representing 'small' and 'large' anchors for the earlier F_2 and later F_3, respectively, so that the larger anchors are associated with the feature map of larger receptive field. These anchors $A_s \cup A_b$ are selected through a k-means clustering of the ground truth 3D bounding boxes. For our experiments, we use $N_s + N_b = 9$. From these $N_s + N_b$ clusters, A_b are those with any axis > 1.125m, and the rest are in A_s.

The two features maps F_2 and F_3 are then processed by a 3D region proposal to regress the 3D object bounding boxes. The 3D region proposal first employs a $1 \times 1 \times 1$ convolution layer to output objectness scores for each potential anchor, producing an objectness feature map with $2(N_s + N_b)$ channels for the positive and negative objectness probabilities. Another $1 \times 1 \times 1$ convolution layer is used to predict the 3D bounding box locations as 6-dimensional offsets from the anchors; we then apply a non-maximum suppression based on the objectness scores. We use a Huber loss on the log ratios of the offsets to the anchor sizes to regress the final bounding box predictions:

$$\Delta_x = \frac{\mu - \mu_{anchor}}{\phi_{anchor}} \qquad \Delta_w = \ln(\frac{\phi}{\phi_{anchor}})$$

where μ is the box center point and ϕ is the box width. The final bounding box loss is then:

$$L_\Delta = \begin{cases} \frac{1}{2}\Delta^2, & \text{if } |\Delta| \leq 2 \\ |\Delta|, & \text{otherwise.} \end{cases}$$

Using these predicted object bounding boxes, we then predict the object class labels using features cropped from the bounding box locations from F_2 and F_3. We use a 3D region of interest pooling layer to unify the sizes of the cropped feature maps to a spatial dimension of $4 \times 4 \times 4$ to be input to an object classification MLP.

4.3. Instance Completion

For each object, we infer its complete geometry by predicting per-voxel occupancies. Here, we crop features from feature map F_5 of the backbone, which has a feature map resolution matching the input spatial resolution, using the predicted object bounding box. These features are processed through a series of five 3D convolutions which maintain the spatial resolution of their input. The complete geometry is then predicted as voxel occupancy using a binary cross entropy loss.

We predict N_{classes} potential object completions for each class category, and select the final prediction based on the predicted object class. We define ground truth bounding boxes b_i and masks m_i as $\gamma = \{(b_i, m_i) | i = 1 \dots N_b\}$. Further, we define predicted bounding boxes $\hat{b_i}$ along with predicted masks $\hat{m_i}$ as $\hat{\gamma} = \{(\hat{b_i}, \hat{m_i}) | i = 1 \dots \hat{N_b}\}$. During training, we only train on predicted bounding boxes that overlap with the ground truth bounding boxes:

$$\Omega = \{(\hat{b_i}, \hat{m_i}, b_i, m_i) \mid \text{IoU}(\hat{b_i}, b_i) \geq 0.5, \\ \forall(\hat{b_i}, \hat{m_i}) \in \hat{\gamma}, \forall(b_i, m_i) \in \gamma\}$$

We can then define the instance completion loss for each

	display	table	bathtub	trashbin	sofa	chair	cabinet	bookshelf	**avg**
Scene Completion + Instance Segmentation	1.65	0.64	4.55	11.25	9.09	9.09	0.18	5.45	5.24
Instance Segmentation + Shape Completion	2.27	3.90	1.14	1.68	14.86	9.93	7.11	3.03	5.49
Ours – RevealNet (no color)	13.16	11.28	13.64	18.19	24.79	15.87	8.60	10.60	14.52
Ours – RevealNet (no proxy)	21.94	7.63	12.55	28.24	20.38	22.58	13.42	9.51	17.03
Ours – RevealNet	**26.86**	**13.21**	**22.31**	**28.93**	**29.41**	**23.64**	**15.35**	**14.48**	**21.77**

Table 1: 3D Semantic Instance Completion on ScanNet [4] scans with Scan2CAD [1] targets at mAP@0.5. Our end-to-end formulation achieves significantly better performance than alternative, decoupled approaches that first use state-of-the-art scan completion [7] and then instance segmentation [14] method or first instance segmentation [14] and then shape completion [6].

associated pair in Ω:

$$L_{\text{compl}} = \frac{1}{|\Omega|}\sum_{\Omega} \text{BCE}(\text{sigmoid}(\hat{m}_i), m_i'),$$

$$m_i'(v) = \begin{cases} m_i(v) & \text{if } v \in \hat{b_i} \cap b_i \\ 0 & \text{otherwise.} \end{cases}$$

We further introduce a global geometric completion loss on entire scene level that serves as an intermediate proxy. To this end, we use feature map F_5 as input to a binary cross entropy loss whose target is the composition of all complete object instances of the scene:

$$L_{\text{geometry}} = \text{BCE}(\text{sigmoid}(F_5), \cup_{(b_i, m_i)\in\gamma}).$$

Our intuition is to obtain a strong gradient during training by adding this additional constraint to each voxel in the last feature map F_5. We find that this global geometric completion loss further helps the final instance completion performance; see Sec 6.

5. Network Training

5.1. Data

The input 3D scans are represented as truncated signed distance fields (TSDFs) encoded in volumetric grids. The TSDFs are generated through volumetric fusion [3] during the 3D reconstruction process. For all our experiments, we used a voxel size of $\approx$ 4.7cm and truncation of 3 voxels. We also input the color images of the RGB-D scan, which we project to the 3D grid using their camera poses. We train our model on both synthetic and real scans, computing 9 anchors through k-means clustering; for real-world ScanNet [4] data, this results in 4 small anchors and 5 large anchors, and for synthetic SUNCG [39] data, this results in 3 small anchors and 6 large anchors.

At test time, we leverage the fully-convolutional design to input the full scan of a scene along with its color images. During training, we use random $96 \times 48 \times 96$ crops ($4.5 \times 2.25 \times 4.5$ meters) of the scanned scenes, along with a greedy selection of $\leq$ 5 images covering the most object geometry in the crop. Only objects with 50% of their complete geometry inside the crop are considered.

5.2. Optimization

We train our model jointly, end-to-end from scratch. We use an SGD optimizer with batch size 64 for object proposals and 16 for object classification, and all positive bounding box predictions ($>$ 0.5 IoU with ground truth box) for object completion. We use a learning rate of 0.005, which is decayed by a factor of 0.1 every 100k steps. We train our model for 200k steps ($\approx$ 60 hours) to convergence, on a single Nvidia GTX 1080Ti. Additionally, we augment the data for training the object completion using ground truth bounding boxes and classification in addition to predicted object detection.

6. Results

We evaluate our approach on semantic instance completion performance on synthetic scans of SUNCG [39] scenes as well as on real-world ScanNet [4] scans, where we obtain ground truth object locations and geometry from CAD models aligned to ScanNet provided by Scan2CAD [1]. To evaluate semantic instance completion, we use a mean average precision metric on the complete masks (at IoU 0.5). Qualitative results are shown in Figs. 3 and 4.

Comparison to state-of-the-art approaches for semantic instance completion. Tables 1 and 2 evaluate our method against state of the art for the task of semantic instance completion on our real and synthetic scans, respectively. Qualitative comparisons on ScanNet scans [4] with Scan2CAD [1] targets (which provide ground truth for complete object geometry) are shown in Fig. 3. We compare to state-of-the-art 3D instance segmentation and scan completion approaches used sequentially; that is, first applying a 3D instance segmentation approach followed by a shape completion method on the predicted instance segmentation, as well as first applying a scene completion approach to the input partial scan, followed by a 3D instance segmentation method. For 3D instance segmentation, we evaluate 3D-SIS [14], which achieves state-of-the-art performance on a dense volumetric grid representation (the representation we use), and for scan completion we evaluate the 3D-

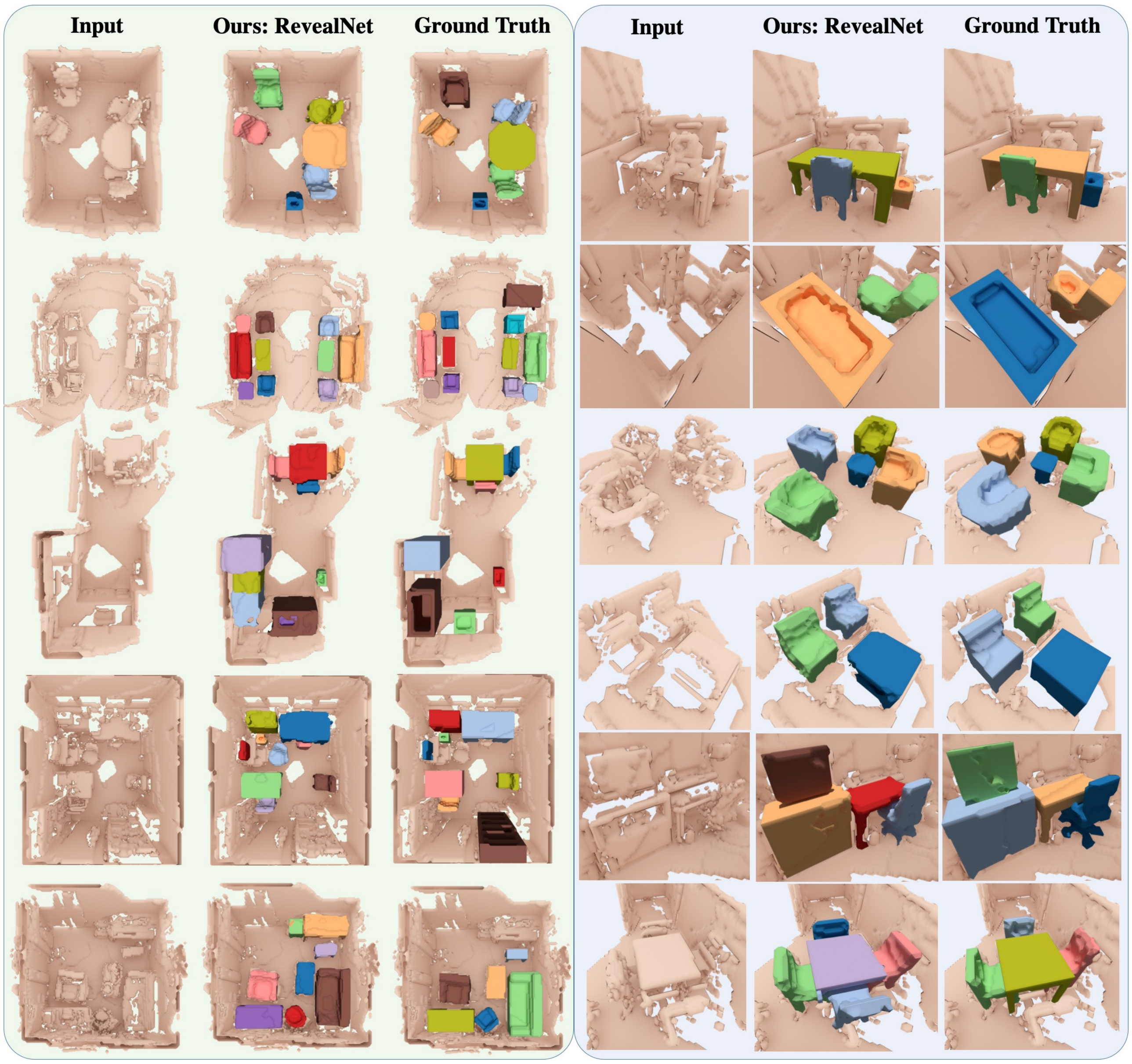


Figure 3: Qualitative results on real-world ScanNet [4] scenes with Scan2CAD [1] targets. Close-ups are shown on the right. Note that different colors denote distinct object instances in the visualization. Our approach effectively predicts complete individual object geometry, including missing structural components (e.g., missing chair legs), across varying degrees of partialness in input scan observations.

EPN [6] shape completion approach and ScanComplete [7] scene completion approach. Our end-to-end approach for semantic instance completion results in significantly improved performance due to information flow from instance completion to object detection. For instance, this allows our instance completion to more easily adapt to some inaccuracies in detection, which strongly hinders a decoupled approach. Note that the ScanComplete model applied on ScanNet data is trained on synthetic data, due to the lack of complete ground truth scene data (Scan2CAD provides only object ground truth) for real-world scans.

Does instance completion help instance detection and segmentation? We can also evaluate our semantic instance completion predictions on the task of semantic instance segmentation by taking the intersection between the predicted complete mask and the input partial scan geom-

	cab	bed	chair	sofa	tabl	door	wind	bkshf	cntr	desk	shlf	curt	drsr	mirr	tv	nigh	toil	sink	lamp	bath	ostr	ofurn	oprop	**avg**
SC + IS	3.0	0.6	19.5	0.8	18.1	**15.9**	0.00	0.0	1.0	2.3	3.0	0.0	0.5	0.0	9.2	10.4	23.9	3.4	9.1	0.0	0.0	0.0	9.1	5.5
IS + SC	0.3	0.0	7.4	0.4	3.0	9.1	0.0	0.0	0.2	0.0	0.0	0.0	2.3	0.0	3.0	0.0	2.6	0.0	1.8	0.0	0.0	0.0	4.6	1.5
no color	**19.05**	41.8	38.2	11.9	23.9	9.1	0.0	0.0	2.5	**21.6**	9.1	0.0	**12.6**	**4.6**	49.4	33.8	63.4	**36.9**	**38.8**	14.7	15.9	0.0	**23.8**	20.5
no proxy	12.9	46.1	**39.4**	26.8	**30.3**	1.0	15.9	0.0	9.1	18.2	3.4	0.0	1.1	0.0	43.6	**34.0**	**69.1**	32.4	29.6	**31.1**	14.6	0.0	23.3	20.9
Ours	14.7	**58.3**	38.2	**28.8**	29.5	0.0	**15.9**	**54.6**	**9.1**	12.1	**9.1**	**0.0**	6.2	0.0	**49.4**	33.5	61.2	34.5	29.5	27.1	**16.4**	**0.0**	23.5	**24.0**

Table 2: 3D Semantic Instance Completion on synthetic SUNCG [39] scans at mAP@0.5. Our semantic instance completion approach achieves significantly better performance than alternative approaches with decoupled state-of-the-art scan completion (SC) [7] followed by instance segmentation (IS) [14], as well as instance segmentation followed by shape completion [6]. We additionally evaluate our approach without color input (no color) and without a completion proxy loss on the network backbone (no proxy).

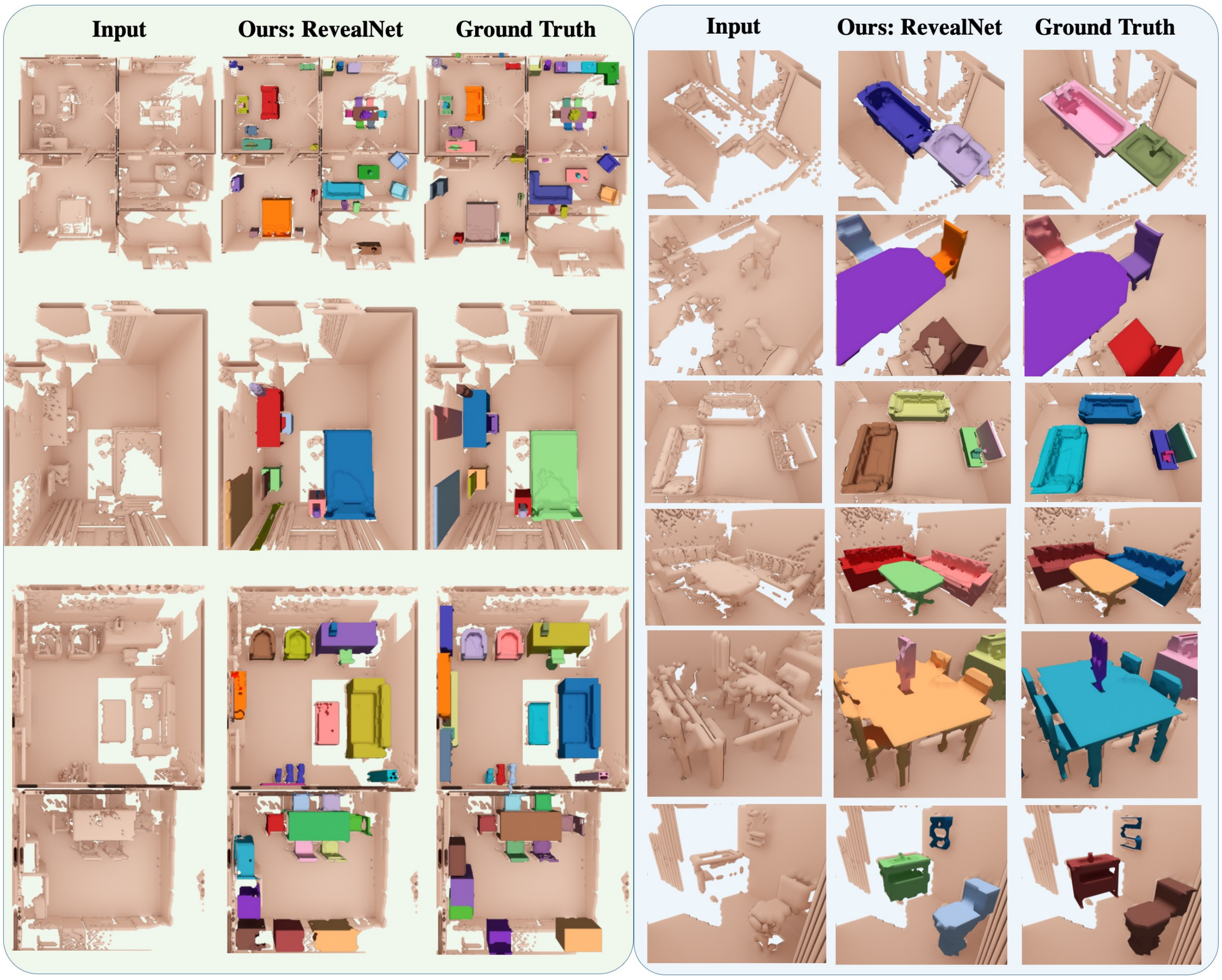


Figure 4: Qualitative results on SUNCG dataset [39] (left: full scans, right: close-ups). We sample RGB-D images to reconstruct incomplete 3D scans from random camera trajectories inside SUNCG scenes. Note that different colors denote distinct object instances in the visualization.

etry to be the predicted instance segmentation mask. We show that predicting instance completion helps instance segmentation, evaluating our method on 3D semantic instance segmentation with and without completion, on ScanNet [4] and SUNCG [39] scans in Tables 3 and 4, as well as 3D-SIS [14], an approach jointly predicts 3D detection and instance segmentation, which also operates on dense volumetric data, achieving state-of-the-art performance on this

	3D Detection	Instance Segmentation
3D-SIS [14]	25.70	20.78
Ours (no compl)	31.93	24.49
Ours (no color)	29.29	23.55
Ours (no proxy)	31.52	25.92
Ours	**36.39**	**30.52**

Table 3: 3D Detection and Instance Segmentation on ScanNet [4] scans with Scan2CAD [1] annotations at mAP@0.5. We evaluate our instance completion approach on the task of instance segmentation and detection to justify our contribution that instance completion task helps instance segmentation and detection. We evaluate our approach without completion (no compl), without color input (no color), and without a completion proxy loss on the network backbone (no proxy). Predicting instance completion notably increases performance of predicting both instance segmentation and detection (Ours vs. no compl). We additionally compare against 3D-SIS [14], a state-of-the-art approach for both 3D detection and instance segmentation on 3D dense volumetric data (the representation we use).

	3D Detection	Instance Segmentation
3D-SIS [14]	24.70	20.61
Ours (no compl)	29.80	23.86
Ours (no color)	31.75	31.59
Ours (no proxy)	34.05	32.59
Ours	**37.81**	**36.28**

Table 4: 3D Detection and Instance Segmentation on synthetic SUNCG [39] scans at mAP@0.5. To demonstrate the benefits of instance completion task for instance segmentation and 3D detection, we evaluate our semantic instance completion approach on the task of instance segmentation and 3D detection. Predicting instance completion notably benefits 3D detection and instance segmentation (Ours vs. no compl).

representation. We find that predicting instance completion significantly benefits instance segmentation, due to a more unified understanding of object geometric structures.

Additionally, we evaluate the effect on 3D detection in Tables 3 and 4; predicting instance completion also significantly improves 3D detection performance. Note that in contrast to 3D-SIS [14] which uses separate backbones for detection and instance segmentation, our unified backbone helps 3D mask information (complete or non-complete) propagate through detection parameters to improve 3D detection performance.

What is the effect of a global completion proxy? In Tables 1 and 2, we demonstrate the impact of the geometric completion proxy loss; here, we see that this loss improves the semantic instance completion performance on both real and synthetic data. In Tables 3 and 4, we can see that it also improves 3D detection and semantic instance segmentation performance.

Can color input help? Our approach takes as input the 3D scan geometry as a TSDF as well as the corresponding color images. We evaluate our approach with and without the color input stream; on both real and synthetic scans, the color input notably improves semantic instance completion performance, as shown in Tables 1 and 2.

7. Limitations

Our approach shows significant potential in the task of semantic instance completion, but several important limitations still remain. First, we output a binary mask for the complete object geometry, which can limit the amount of detail represented by the completion; other 3D representations such as distance fields or sparse 3D representations [10] could potentially resolve greater geometric detail. Our approach also uses axis-aligned bounding boxes for object detection; it would be helpful to additionally predict the object orientation. We also do not consider object movement over time, which contains significant opportunities for semantic instance completion in the context of dynamic environments.

8. Conclusion

In this paper, we tackle the problem of "seeing behind objects" by predicting the missing geometry of individual objects in RGB-D scans. This opens up many possibilities for complex interactions with objects in 3D, for instance for efficient navigation or robotic grasping. To this end, we introduced the new task of semantic instance completion along with RevealNet, a new 3D CNN-based approach to jointly detect objects and predict their complete geometry. Our proposed 3D CNN learns from both color and geometry features to detect and classify objects, then predicts the voxel occupancy for the complete geometry of the object in an end-to-end fashion, which can be run on a full 3D scan in a single forward pass. On both real and synthetic scan data, we significantly outperform state-of-the-art approaches for semantic instance completion. We believe that our approach makes an important step towards higher-level scene understanding and helps to enable object-based interactions and understanding of scenes, which we hope will open up new research avenues.

Acknowledgments

This work was supported by the ZD.B, a Google Research Grant, an Nvidia Professor Partnership, a TUM-IAS Rudolf Mößbauer Fellowship, and the ERC Starting Grant *Scan2CAD (804724)*.

References

[1] Armen Avetisyan, Manuel Dahnert, Angela Dai, Manolis Savva, Angel X. Chang, and Matthias Nießner. Scan2cad: Learning cad model alignment in rgb-d scans. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2019. 2, 5, 6, 8

[2] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3D: Learning from RGB-D data in indoor environments. *International Conference on 3D Vision (3DV)*, 2017. 2

[3] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312. ACM, 1996. 5

[4] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017. 2, 5, 6, 7, 8

[5] Angela Dai and Matthias Nießner. 3dmv: Joint 3d-multi-view prediction for 3d semantic scene segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. 3, 4

[6] Angela Dai, Charles Ruizhongtai Qi, and Matthias Nießner. Shape completion using 3d-encoder-predictor cnns and shape synthesis. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017. 3, 5, 6, 7

[7] 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. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2018. 2, 3, 5, 6, 7

[8] Cathrin Elich, Francis Engelmann, Jonas Schult, Theodora Kontogianni, and Bastian Leibe. 3d-bevis: Birds-eye-view instance segmentation. *arXiv preprint arXiv:1904.02199*, 2019. 2

[9] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. 2

[10] Benjamin Graham and Laurens van der Maaten. Submanifold sparse convolutional networks. *arXiv preprint arXiv:1706.01307*, 2017. 8

[11] 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 *IEEE International Conference on Computer Vision (ICCV)*, 2017. 3

[12] Christian Häne, Shubham Tulsiani, and Jitendra Malik. Hierarchical surface prediction for 3d object reconstruction. *arXiv preprint arXiv:1704.00710*, 2017. 3

[13] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2980–2988. IEEE, 2017. 2

[14] Ji Hou, Angela Dai, and Matthias Nießner. 3d-sis: 3d semantic instance segmentation of rgb-d scans. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2019. 2, 3, 4, 5, 7, 8

[15] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, volume 7, 2006. 2

[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. 3

[18] Jean Lahoud, Bernard Ghanem, Marc Pollefeys, and Martin R Oswald. 3d instance segmentation via multi-task metric learning. *arXiv preprint arXiv:1906.08650*, 2019. 2

[19] Yangyan Li, Angela Dai, Leonidas Guibas, and Matthias Nießner. Database-assisted object retrieval for real-time 3d reconstruction. In *Computer Graphics Forum*, volume 34, pages 435–446. Wiley Online Library, 2015. 3

[20] Tsung-Yi Lin, Piotr Dollár, Ross B Girshick, Kaiming He, Bharath Hariharan, and Serge J Belongie. Feature pyramid networks for object detection. In *CVPR*, volume 1, page 4, 2017. 2

[21] Tsung-Yi Lin, Priyal Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *IEEE transactions on pattern analysis and machine intelligence*, 2018. 2

[22] Chen Liu and Yasutaka Furukawa. Masc: Multi-scale affinity with sparse convolution for 3d instance segmentation. *arXiv preprint arXiv:1902.04478*, 2019. 2

[23] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. 2

[24] Niloy J Mitra, Leonidas J Guibas, and Mark Pauly. Partial and approximate symmetry detection for 3d geometry. In *ACM Transactions on Graphics (TOG)*, volume 25, pages 560–568. ACM, 2006. 3

[25] Liangliang Nan, Ke Xie, and Andrei Sharf. A search-classify approach for cluttered indoor scene understanding. *ACM Transactions on Graphics (TOG)*, 31(6):137, 2012. 3

[26] Gaku Narita, Takashi Seno, Tomoya Ishikawa, and Yohsuke Kaji. Panopticfusion: Online volumetric semantic mapping at the level of stuff and things. *arXiv preprint arXiv:1903.01177*, 2019. 2

[27] Andrew Nealen, Takeo Igarashi, Olga Sorkine, and Marc Alexa. Laplacian mesh optimization. In *Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia*, pages 381–389. ACM, 2006. 2

[28] Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation. *arXiv preprint arXiv:1606.02147*, 2016. 3

[29] Mark Pauly, Niloy J Mitra, Johannes Wallner, Helmut Pottmann, and Leonidas J Guibas. Discovering structural

regularity in 3d geometry. In *ACM transactions on graphics (TOG)*, volume 27, page 43. ACM, 2008. 3

[30] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep hough voting for 3d object detection in point clouds. *arXiv preprint arXiv:1904.09664*, 2019. 2

[31] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. *arXiv preprint arXiv:1711.08488*, 2017. 2

[32] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. 2

[33] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 2

[34] Tianjia Shao, Weiwei Xu, Kun Zhou, Jingdong Wang, Dongping Li, and Baining Guo. An interactive approach to semantic modeling of indoor scenes with an rgbd camera. *ACM Transactions on Graphics (TOG)*, 31(6):136, 2012. 3

[35] Yifei Shi, Pinxin Long, Kai Xu, Hui Huang, and Yueshan Xiong. Data-driven contextual modeling for 3d scene understanding. *Computers & Graphics*, 55:55–67, 2016. 3

[36] Ivan Sipiran, Robert Gregor, and Tobias Schreck. Approximate symmetry detection in partial 3d meshes. In *Computer Graphics Forum*, volume 33, pages 131–140. Wiley Online Library, 2014. 3

[37] Shuran Song and Jianxiong Xiao. Sliding shapes for 3d object detection in depth images. In *European conference on computer vision*, pages 634–651. Springer, 2014. 2

[38] Shuran Song and Jianxiong Xiao. Deep sliding shapes for amodal 3d object detection in rgb-d images. *arXiv preprint arXiv:1511.02300*, 2015. 2

[39] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. *Proceedings of 30th IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 2, 3, 5, 7, 8

[40] Olga Sorkine and Daniel Cohen-Or. Least-squares meshes. In *Shape Modeling Applications, 2004. Proceedings*, pages 191–199. IEEE, 2004. 2

[41] Pablo Speciale, Martin R Oswald, Andrea Cohen, and Marc Pollefeys. A symmetry prior for convex variational 3d reconstruction. In *European Conference on Computer Vision*, pages 313–328. Springer, 2016. 3

[42] Sebastian Thrun and Ben Wegbreit. Shape from symmetry. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 2, pages 1824–1831. IEEE, 2005. 3

[43] Weiyue Wang, Ronald Yu, Qiangui Huang, and Ulrich Neumann. Sgpn: Similarity group proposal network for 3d point cloud instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2569–2578, 2018. 2

[44] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1912–1920, 2015. 3

[45] Bo Yang, Jianan Wang, Ronald Clark, Qingyong Hu, Sen Wang, Andrew Markham, and Niki Trigoni. Learning object bounding boxes for 3d instance segmentation on point clouds. *arXiv preprint arXiv:1906.01140*, 2019. 2

[46] Li Yi, Wang Zhao, He Wang, Minhyuk Sung, and Leonidas Guibas. Gspn: Generative shape proposal network for 3d instance segmentation in point cloud. *arXiv preprint arXiv:1812.03320*, 2018. 2

[47] Wei Zhao, Shuming Gao, and Hongwei Lin. A robust hole-filling algorithm for triangular mesh. *The Visual Computer*, 23(12):987–997, 2007. 2

Exploring Data-Efficient 3D Scene Understanding with Contrastive Scene Contexts

Ji Hou[1] Benjamin Graham[2] Matthias Nießner[1] Saining Xie[2]
[1]Technical University of Munich [2]Facebook AI Research

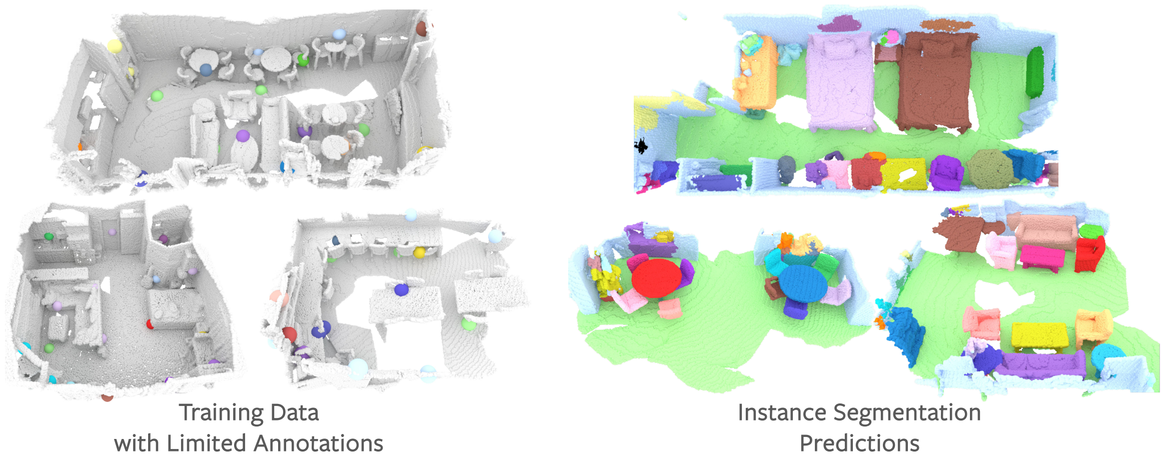


Figure 1: How many point labels are necessary to train a 3D instance segmentation model on point clouds? It turns out not too many! With the help of unsupervised pre-training, only 20 labelled points per scene (*less than 0.1% of the total points*) are used to fine-tune an instance segmentation model on ScanNet. **Left**: Train samples; only colored points (enlarged for better visibility) are labeled. **Right**: Predictions in validation set and different colors represent different instances.

Abstract

*The rapid progress in 3D scene understanding has come with growing demand for data; however, collecting and annotating 3D scenes (*e.g.* point clouds) are notoriously hard. For example, the number of scenes (*e.g.* indoor rooms) that can be accessed and scanned might be limited; even given sufficient data, acquiring 3D labels (*e.g.* instance masks) requires intensive human labor. In this paper, we explore data-efficient learning for 3D point cloud. As a first step towards this direction, we propose Contrastive Scene Contexts, a 3D pre-training method that makes use of both point-level correspondences and spatial contexts in a scene. Our method achieves state-of-the-art results on a suite of benchmarks where training data or labels are scarce. Our study reveals that exhaustive labelling of 3D point clouds might be unnecessary; and remarkably, on ScanNet, even using 0.1% of point labels, we still achieve 89% (instance segmentation) and 96% (semantic segmentation) of the baseline performance that uses full annotations.*

1. Introduction

Recent advances in deep learning on point clouds, such as those obtained from LiDAR or depth sensors, together with a proliferation of public, annotated datasets [9, 13, 53, 32, 64, 2, 40, 55], have led to swift progress in 3D scene understanding. However, compared to large-scale 2D scene understanding on images [14, 38, 23], the scale of 3D scene understanding—in terms of the amount and diversity of data and annotations, the model size, the number of semantic categories, and so on—still falls behind. We argue that one major bottleneck is the fact that collecting and annotating diverse 3D scenes are significantly more expensive. Unlike 2D images that comfortably exists on the Internet, collecting real world 3D scene datasets usually involves traversing the environment in real life and scanning with 3D sensors. Therefore, the number of indoor scenes that can be scanned might be limited. What is more concerning is that, even given sufficient data acquisition, 3D semantic labelling (*e.g.* bounding boxes and instance masks) requires complex pipelines [13] and labor-intensive human effort.

In this work, we explore a new learning task in 3D, *i.e.*

data-efficient 3D scene understanding, which focuses on the problem of learning with limited data or supervision[1]. We note that the importance of data-efficient learning in 3D is two-fold. One concerns the status quo: given limited data we have right now, *can we design better methods that perform better?* The other one is more forward-looking: *is it possible to reduce the human labor for annotation*, with a goal of creating 3D scene datasets on a much larger scale?

To formally study the problem, we first introduce a suite of scene understanding benchmarks that encompasses two complementary settings for data-efficient learning: (1) *limited scene reconstructions (LR)* and (2) *limited annotations (LA)*. The first setting concerns the scenario where the bottleneck is the number of scenes that can be scanned and reconstructed. The second one focuses on the case where in each scene, the budget for labeling is constrained (*e.g.* one can only label a small set of points). For each setting, the evaluation is done on a diverse set of scene understanding tasks including object detection, semantic segmentation and instance segmentation.

For data-efficient learning in 2D [28], representation learning, *e.g.* pre-training on a rich source set and fine-tuning on a much smaller target set, often comes to the rescue; in 3D, representation learning for data-efficient learning is even more wanted but long overdue. With this perspective, we focus on studying data-efficient 3D scene understanding through the lens of representation learning.

Only recently, PointContrast [65] demonstrates that network weights pre-trained on 3D partial frames can lead to a performance boost when fine-tuned on 3D semantic segmentation and object detection tasks. Our work is inspired by PointContrast. However, we observe that the simple contrastive-learning based pretext task used in [65] only concerns point-level correspondence matching, which completely disregards the spatial configurations and contexts in a scene. In Section 3, we show that this design limits the scalibility and transferability; we further propose an approach that integrates the spatial information into the contrastive learning framework. The simple modification can significantly improve the performance over PointContrast, especially on complex tasks such as instance segmentation.

Our exploration in data-efficient 3D scene understanding provides some surprising observations. For example, on ScanNet, even using 0.1% of point labels, we are still able to recover 89% (instance segmentation) and 96% (semantic segmentation) of the baseline performance that uses full annotations. The results imply that exhaustive labelling of 3D point clouds might not be necessary. In both scenarios of *limited scene reconstructions (LR)* and *limited annotations (LA)*, our pre-trained network, when used as the initialization for supervised fine-tuning, offers consistent improvement across multiple tasks and datasets. In the scenario of *LA*, we also show that an active labeling strategy can be enabled by clustering the pre-trained point features.

[1]Sometimes a distinction is drawn between *data-efficiency* and *label-efficiency*, to separate the scenarios of limited amount of data samples and limited supervision; here, we use *data-efficiency* to encompass both cases.

In summary, the contributions of our work include:

- A systematic study on data-efficient 3D scene understanding with a comprehensive suite of benchmarks.
- A new 3D pre-training method that can gracefully transfer to complex tasks such as instance segmentation and outperform the state-of-the-art results.
- Given the pre-trained network, we study practical solutions for data-efficient learning in 3D through fine-tuning as well as an active labeling strategy.

2. Related Work

3D Scene Understanding. Research in deep learning on 3D point clouds have been recently shifted from synthetic, single object classification [47, 46, 48] to the challenge of large-scale, real-world scene understanding. A variety of datasets [2, 13, 54, 18, 55] and algorithms have been proposed for 3D object detection [45, 44, 43, 24], semantic segmentation [46, 56, 62, 57, 20, 12] and instance segmentation [59, 30, 69, 36, 60, 67, 31, 15, 34, 33]. In the past year, sparse convolutional networks [20, 12] stand out as a promising approach to standardize deep learning for point clouds, due to its computational efficiency and state-of-the-art performance for 3D scene understanding tasks [12, 25, 34]. In this work, we also adopt a sparse U-Net [49] backbone for our exploration.

3D Representation Learning. Compared to 2D vision, the limits of big data are far from being fully explored in 3D. In 2D representation learning, for example, transfer learning from a rich source data (*e.g.* ImageNet [14]) to a (typically smaller) target data, has become a dominant framework for many applications [19]. In contrast, 3D representation learning has not been widely adopted and most 3D networks are trained from scratch on the target data directly. Recently, unsupervised pre-training has made great progress and drawn significant attention in 2D [42, 3, 39, 28, 63, 58, 29, 27, 10, 8, 21]. Following suit, recent works attempt to adapt the 2D pretext tasks to 3D, but mostly focus on single object classification tasks on ShapeNet [1, 17, 68, 22, 37, 61, 26, 51, 50]. Our work is mostly inspired by a recent contrastive-learning based method PointContrast [65], which first demonstrates the effectiveness of unsupervised pre-training on a diverse set of scene-level understanding tasks. As we will show in the later sections, the simple point-level pre-training objective in PointContrast ignores the spatial contexts of the scene (such as relative poses of objects, and distances between them) which limits its transferability for complex tasks such as instance segmentation. PointContrast also focuses on downstream tasks with 100% data and labels, while we systematically explore a new data-efficient paradigm that has practical importance.

Data-Efficient Learning. Data-efficient learning concerns the problem of learning with limited training examples or labels. This capability is known in cognitive science to be a distinctive characteristic of humans [6]. In contrast, training deep neural networks is not naturally data-efficient, as it typically relies on large amount of annotated data. Among many potential solutions towards this goal, representation learning (commonly through transfer learning) is arguably the most promising one. A good representation "*entangles the different explanatory factors of variation behind the data*" [5] and thus makes the downstream prediction easier (and less data-hungry). This concept has been validated successfully in natural language processing [7] and to some extent in 2D image classification [28]. Pursuing this direction in 3D is even more desirable, considering the potential benefit in reducing the labor of data collection and annotation. Existing work focuses on mostly single CAD model classification or part segmentation [70, 52, 41, 11, 26, 16, 65]. To the best of our knowledge, our work is the first to explore data-efficient learning in a real-world, large-scale 3D scene understanding setup.

3. Contrastive Scene Contexts for Pre-training

In this section, we first briefly revisit the PointContrast framework [65], and discuss the shortcomings and remedies. We then introduce our pre-training algorithm.

Revisiting PointContrast. The pre-training objective for PointContrast is to achieve point *equivariance* with respect to a set of random geometric transformations. Given a pair of overlapping partial scans, a contrastive loss for pre-training is defined over the point features. The objective is to minimizes the distance for matched points (positive pairs) and maximize the distance between unmatched ones (negative pairs). Despite the fact that strong spatial contexts exist among objects in a scene, this objective does not capture any of the spatial information: the negative pairs could be sampled from arbitrary locations across many scenes in a mini-batch. We hypothesize that this leads to some limitations: 1) the spatial contexts (*e.g.* relative pose, direction and distance), which could be pivotal for complex tasks such as instance segmentation, are entirely discarded from pre-training; 2) the scalibility of contrastive learning might be hampered; PointContrast cannot utilize a large number of negative points, potentially because that contrasting a pair of spatially distant and unrelated points would contribute little to learning. In fact, PointContrast uses only a random sampling of 1024 points per scene for pre-training, and it has been shown that results do not improve with more sampled points [65]. We also confirm this behavior with experiments later this section.

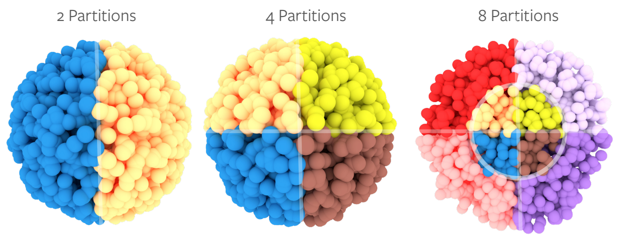


Figure 2: **Illustration of Scene Contexts.** We visualize the 2,4 and 8 spatial partitions for Scene Contexts. The anchor point is in the center. For 2 and 4 partitions, only relative angles are sufficient. For 8 partitions (a cross-section is shown), both relative angles and distances are needed.

Contrastive Scene Contexts. We hope to integrate spatial contexts into the pre-training objective. There are many ways to achieve the goal, and here we take inspiration from the classic *ShapeContext* local descriptor [4, 35, 66] for shape matching. The *ShapeContext* descriptor partitions the space into spatially inhomogeneous cells, and encodes the spatial contexts about the shape at each point by computing a histogram over the number of neighboring points in each cell. We call our method *Contrastive Scene Contexts* because at a high level, our method also aims to capture *the distribution over relative locations in a scene*. We partition the scene point cloud into multiple regions, and instead of having a single contrastive loss for the entire point set sampled in a mini-batch, we perform contrastive learning in each region separately, and aggregate the losses in the end.

Concretely, given a pair of partial frame point clouds $\mathbf{x}$ and $\mathbf{y}$ from the same scene, we have correspondence mapping $(i,j) \in M_{\mathbf{xy}}$ available, where i is the index of a point $\mathbf{x}_i \in \mathcal{R}^3$ in frame $\mathbf{x}$ and j is the index of a matched point $\mathbf{y}_j \in \mathcal{R}^3$ in frame $\mathbf{y}$. Similar to PointContrast, we sample N pairs of matched points as positives. However, in our method, for each anchor point $\mathbf{x}_i$, the space is divided into multiple partitions and other points are assigned to different partitions based on their relative angles and distances to i.

The distance and angle information needed for scene context partition at anchor point $\mathbf{x}_i$ is as follows,

$$\mathcal{D}_{ik} = \sqrt{\sum_{d=1}^{3}(\mathbf{x}_i^d - \mathbf{x}_k^d)^2} \tag{1}$$

$$\mathcal{A}_{ik} = arctan2(\mathcal{D}_{ik}) + 2\pi \tag{2}$$

where $\mathcal{D}$ is the relative distance matrix. $\mathcal{D}_{ik}$ stores the distance between point i and point k and $\mathcal{A}$ is the relative angle matrix, where A_{ik} stores the relative angle between point i and point k. In Equation (1) d represents the 3D dimension. With $\mathcal{D}$ and $\mathcal{A}$, a *ShapeContext*-like spatial partitioning function can be easily constructed on-the-fly. In Figure 2, we show a visual illustration of how the space partitioning works. Computing 2 or 4 partitions only requires cutting the space according to relative angles based on $\mathcal{A}$; while the 8 or more partitions also require the extent of the inner regions using $\mathcal{D}$. We always partition the space uniformly along the relative angles and distances. Note that the partitioning is *relative to the anchor point* i.

Suppose there are P partitions, we denote the spatial partition functions as $par_p(\cdot)$, where $p \in \{1, \ldots, P\}$. Function $par_p(\cdot)$ takes the anchor point i as input, and return a set of points as negatives. A PointInfoNCE loss $\mathcal{L}_p$ is independently computed for each partition:

$$\mathcal{L}_p = -\sum_{(i,j)\in M} \log \frac{\exp(\mathbf{f}_i^1 \cdot \mathbf{f}_j^2/\tau)}{\sum_{(\cdot,k)\in M, k\in par_p(i)} \exp(\mathbf{f}_i^1 \cdot \mathbf{f}_k^2/\tau)} \quad (3)$$

Details of Equation (3) and other implementation details can be found in Appendix. The final loss is computed by aggregating all partitions $\mathcal{L} = \frac{1}{|P|}\sum_p \mathcal{L}_p$.

Analysis. We first show that by integrating the scene contexts into the objective, our pre-training method can benefit more from a larger point set. We conduct an analysis experiment by varying the number of scene context partitions and the number of points sampled for computing the contrastive loss. We pre-train our model for a short schedule (20K iters). We then fine-tune the pre-trained weights on S3DIS instance segmentation benchmark [2]. Results are shown in Figure 3, the green line represents a variant with *no spatial partitioning*; the left-most point represents PointContrast[2]. Similar to the observation in [65], without scene contexts, increasing the number of sampled points does not improve the performance; with more partitions, increasing # sampled points leads to a consistent boost in performance (up to 4096 points). We use 8 partitions as empirically it works best. This shows that our method leads to better *scalability* as more points can be utilized for pre-training.

We achieve state-of-the-art instance segmentation results in terms of mAP@0.5 (Table. 1) using a simple bottom-up clustering mechanism with voting loss (details in Appendix). We do not use any special modules such as Proposal Aggregation [15] or Scoring Network [34]. We observe a 2.9% absolute improvement over PointContrast pre-training, which brings the improvement over train-from-scratch baseline to 4.1%. This substantial margin demonstrates the effectiveness of Contrastive Scene Contexts on instance segmentation tasks. We provide more results comparing against PointContrast in Section 5.3.

4. Data-Efficient 3D Scene Understanding

To formally explore data-efficient 3D scene understanding, in this section, we propose two different learning paradigms and relevant benchmarks that are associated with two complementary settings that can occur in real world application scenarios: (1) *limited scene reconstructions (LR)* and (2) *limited annotations (LA)*. The first setting mainly concerns the scenario where the bottleneck of data collection is the *number of scenes* that can be scanned and reconstructed. The second one focuses on the case where in each scene, the budget for labeling is limited (*e.g.* one can only label a small set of points). Since 3D point labeling is human intensive, this represents a practical scenario where a data-efficient learning strategy can greatly reduce the annotation cost. An overview is presented in Figure 4, and details of individual benchmarks are described below.

[2]Not exactly identical since the matched points are sampled per scene in this experiment, rather than from the whole mini-batch as in PointContrast; we have verified that this nuance does not influence the conclusion.

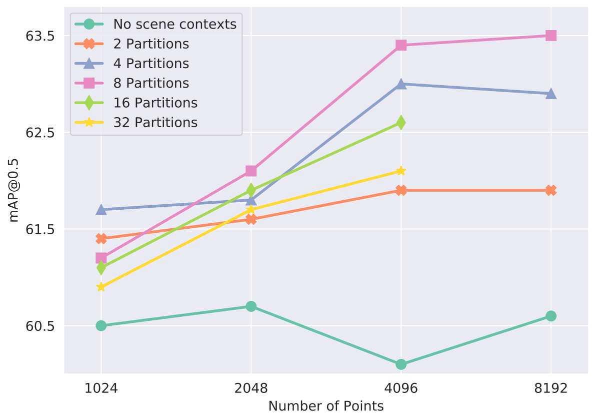


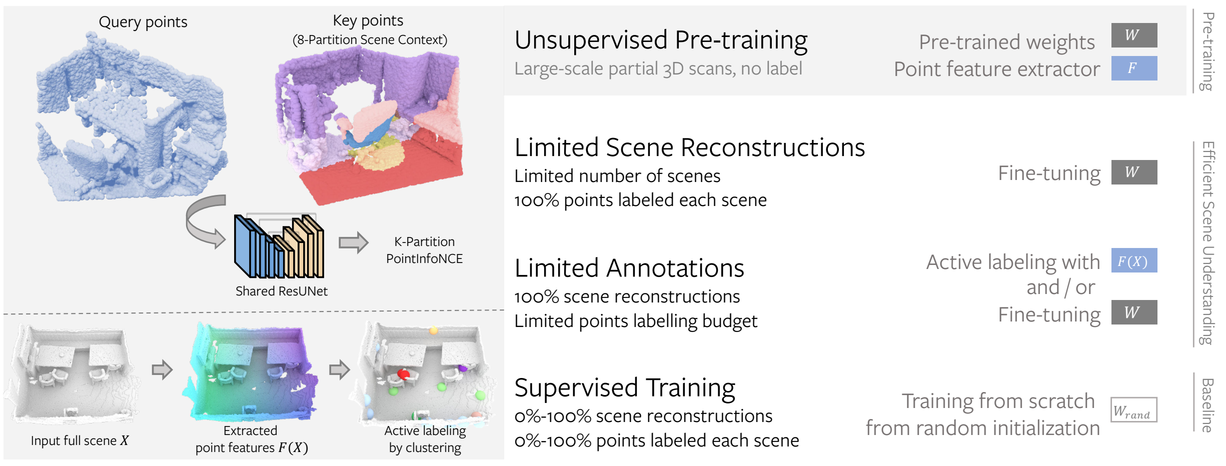
Figure 3: **Analysis Experiment.** Varying the number of partitions and sampled points for pre-training; Results are reported on the S3DIS instance segmentation task [2]. Using scene context partitions has enabled constrastive learning to utilize more points for better performance.

Methods	mAP@0.5
ASIS [60]	55.3
3D-BoNet [67]	57.5
PointGroup [34]	57.8
3D-MPA [15]	63.1
Train from scratch	59.3
PointContrast (PointInfoNCE) [65]	60.5 (+1.2)
Contrastive Scene Contexts	**63.4** (+4.1)

Table 1: **Fine-tuning results for instance segmentation on S3DIS [2].** A simple clustering-based model with *Contrastive Scene Contexts* pre-trained backbone performs significantly better than the train-from-scratch baseline and PointContrast pre-training [65].

4.1. Limited Annotations (LA)

In this benchmark, we explore 3D scene understanding with a limited budget for point cloud annotations. We consider a diverse set of tasks including semantic segmentation, instance segmentation and object detection. Specifically, for instance segmentation and semantic segmentation, the annotation budget is in terms of the *number of points* for labelling. This is practically useful: if an annotator only



Query points
Key points
(8-Partition Scene Context)
Shared ResUNet
K-Partition
PointInfoNCE
Input full scene X
Extracted
point features $F(X)$
Active labeling
by clustering
Unsupervised Pre-training
Large-scale partial 3D scans, no label
Pre-trained weights W
Point feature extractor F
Pre-training
Limited Scene Reconstructions
Limited number of scenes
100% points labeled each scene
Fine-tuning W
Limited Annotations
100% scene reconstructions
Limited points labelling budget
Active labeling with $F(X)$
and / or
Fine-tuning W
Efficient Scene Understanding
Supervised Training
0%-100% scene reconstructions
0%-100% points labeled each scene
Training from scratch
from random initialization W_{rand}
Baseline

Figure 4: **Overview of Data-Efficient 3D Scene Understanding. Left**: Unsupervised pre-training with *Contrastive Scene Contexts*. The outputs of pre-training are 1) a pre-trained U-Net **F** (that can be used as an offline feature extractor) and 2) its associated weights **W**. **Right**: After pre-training, different learning scenarios can be applied for the downstream tasks such as learning with limited scene reconstructions (*LR*) or limited annotations (*LA*). In the case of *LR*, the pre-trained weights **W** are used as network initialization for fine-tuning. In the case of *LA*, all the scene reconstructions can be used but only a limited annotation budget is available, *e.g*. 20 points can be annotated (semantic labels) per scene. Again, **W** can be used as network initialization for fine-tuning; optionally the feature extractor **F** can be used in an active labeling strategy to decide which points to annotate. Baselines are standard supervised learning where models are trained from scratch.

needs to label the semantic labels for 20 points, it will only require a few minutes to label a full room. Our benchmark considers four different training configurations on ScanNet including using $\{20, 50, 100, 200\}$ labeled points per scene. For object detection, the annotation budget is with respect to the *number of bounding boxes* to label in each scene. Our benchmark considers four different training configurations including $\{1, 2, 4, 7\}$ labeled bounding boxes. Our base dataset is ScanNetV2 [13] which has 1201 scenes for training. We evaluate the model performance on standard ScanNetV2 validation set of 312 scenes that has full labels.

4.2. Limited Scene Reconstructions (LR)

For current 3D scene datasets, it is common for annotators to carry commodity depth cameras and record 3D videos at private houses or furniture stores. It might be unrealistic to enter a large number of homes and obtain detailed scanning. In this case, the number of scenes might be the bottleneck and the training has to be done on limited amount of scene reconstructions. We simulate this scenario by random sampling a subset of ScanNetV2 training set. Our benchmark has four configurations $\{1\%, 5\%, 10\%, 20\%\}$ (100% represents the entire ScanNet train set) for semantic segmentation and instance segmentation; and $\{10\%, 20\%, 40\%, 80\%\}$ for object detection. During test time, evaluation is on all scenes in the validation set.

5. Experimental Results

In this section, we present our experimental results on the data-efficient 3D scene understanding benchmarks: **ScanNet-LA** with limited annotations and **ScanNet-LR** with limited scene reconstructions. In both scenarios, we compare our method against the baseline of training from scratch, and report results on semantic/instance segmentation and object detection. We also compare our models with the state-of-the-art method in the last part of the section.

Experiments Setup For pre-training, we use SGD optimizer with learning rate 0.1 and a batch-size of 32. The learning rate is decreased by a factor of 0.99 every 1000 steps. The model is trained for 60K steps. The fine-tuning experiments on instance segmentation and semantic segmentation are trained with a batch-size of 48 for a total of 10K steps. The initial learning rate is 0.1, with polynomial decay with power 0.9. For all experiments, we use data parallel on 8 NVIDIA V100 GPUs. For object detection experiments, we fine-tune the model with a batch-size of 32 for 180 epochs. The initial learning rate is set to 0.001 and decayed by a factor of 0.1 at epoch 80, 120 and 160. For all the experiments, we use the same Sparse Res-UNet [65] as the backbone. For both training and testing, the voxel size for Sparse ConvNet is set to 2.0 cm. We use Sparse ConvNet implemented by MinkowskiEngine [12].

5.1. Limited Annotations

As introduced in Section 4, the *Limited Annotation (LA)* benchmark covers two different annotation types: *Limited*

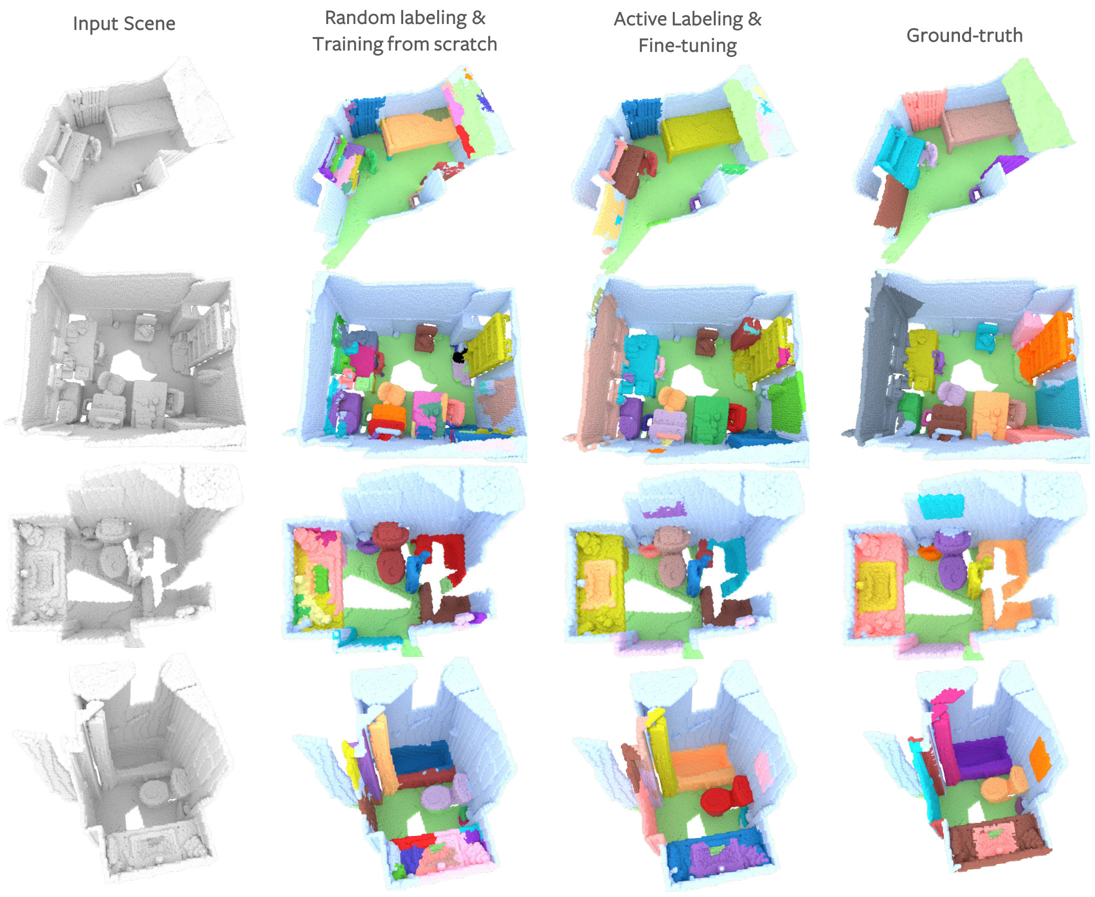

Input Scene
Random labeling & Training from scratch
Active Labeling & Fine-tuning
Ground-truth

Figure 5: **Qualitative Instance Segmentation Results (ScanNet-LA).** With our pre-trained model as initialization for fine-tuning, together with an active labeling process, our approach (trained with 20 labeled points per scene) generates high-quality instance masks. Different color represents instance index only (same instances might not share the same color).

Point Annotations for semantic and instance segmentation and *Limited Bounding Box Annotations* for detection. The pre-trained network (and its weights) can be used as initialization for fine-tuning, or integrate in an active labeling strategy, which we describe below.

Active labeling. Since we focus on the scenario of having limited annotation budget, it is natural to consider an *active learning* strategy during the data annotation process; *i.e.* one can interactively query an annotator to label some data points that can help most for subsequent training. The core idea of our approach is to perform a **balanced sampling** on the **feature space**, so that the selected points will be the most representative and exemplary ones in a scene. Our pre-trained network extracts dense features at each point of the to-be-annotated point cloud, by simply performing a forward pass. We then perform k-means clustering in this feature space to obtain K cluster centroids. We select the K centroids as the points to be provided to the annotators for labeling. We also present two baseline strategies including a simple **random sampling** strategy where K points are randomly selected to be labeled, and a similar **k-means sampling** strategy on raw (RGB+XYZ) inputs, rather than on the pre-trained features.

We note that although our experiments are simulated based on the already collected ScanNet dataset, our pre-trained feature extractor and the labeling strategy are readily useful in a real-world data annotation pipeline.

Results. In Figure 6 we show that compared to the naive from-scratch baselines, our proposed pre-training framework can lead to much improved performance. It is interesting to see that, for both semantic segmentation and instance segmentation, even *without* fine-tuning, the *active labeling* strategy alone provides point labels that make the trained model perform significantly better, compared to *random sampling* or *k-means sampling* baseline strategies, yielding a $>$10% absolute improvement in terms of mAP@0.5 and mIoU when the training data has only 20 point labels.

The fact that *active labeling* strategy performs on

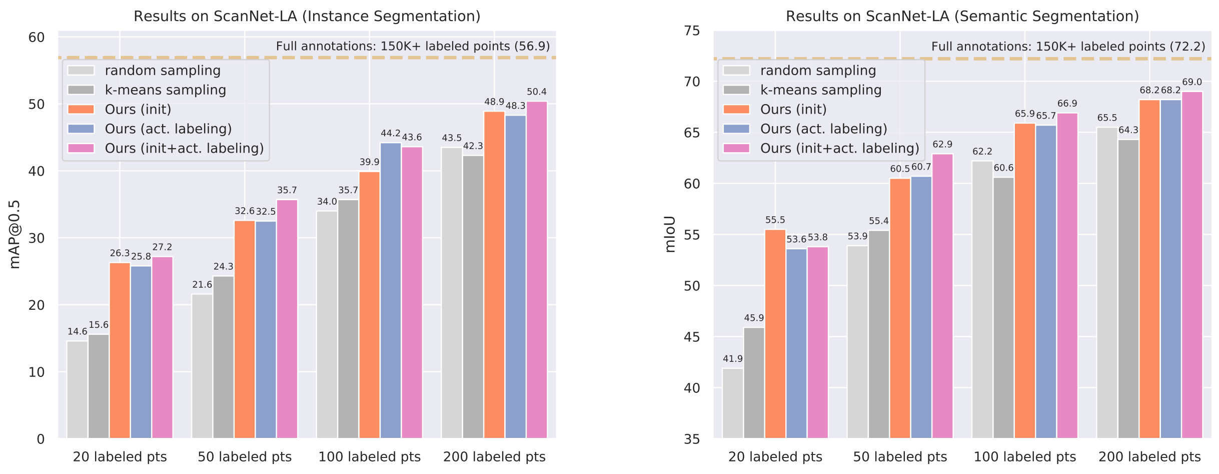


Figure 6: **3D Instance and Semantic Segmentation with Limited Point Annotations (ScanNet-LA).** Ours (init) denotes the network initialization by our pre-trained model. Ours (act. labeling) denotes the active selection of annotated points by our pre-trained model. Ours (init+act. labeling) denotes using our model as both network initialization and active labeling. We additionally mark the upper bound of using all 150K annotated points (in average) per scene as the dash line.

No. of Boxes	VoteNet (scratch)	VoteNet (ours)
all	35.4	**39.3** (+3.9)
1	27.5	**30.3** (+2.8)
2	30.9	**32.4** (+1.5)
4	32.5	**34.6** (+2.1)
7	33.4	**35.9** (+2.5)

Table 2: **Object detection results using Limited Bounding Box Annotations on ScanNet.** The metric is mAP@0.5. "Ours" denotes the fine-tuning results with our pre-trained model. We list the upper-bound performance using all annotated bounding boxes (in average about 13 bounding boxes per scene) as a reference in the first row.

par with the more common pre-training and fine-tuning paradigm, suggests that finding exemplary points to label is crucial for data-efficient learning. Of course, in real applications both active labeling and fine-tuning can be used jointly, and we indeed observe a further (though admittedly smaller) boost in performance by 1) active sampling points to label and then 2) fine-tuning with the pre-trained weights.

Overall, with the help of our Contrastive Scene Contexts pre-training, even using around 0.1% of point labels (*e.g.* 200 labeled points out of 150K total points per scene), we are still able to achieve 50.4% mAP@0.5 for instance segmentation, and 69.0% mIoU for semantic segmentation. This indicates a recovery of 89% and 96% of baseline performance that uses 100% of the annotations. We show additional qualitative comparison in Figure 5.

Limited Bounding Box Annotations. For object detection, we use VoteNet [44] as the detector framework; following [65], we replace PointNet [46] with our Sparse ResUNet. For this part, we do not use any active labeling strategy as the labeling cost for bounding boxes are much smaller. We random sample {1, 2, 4, 7} bounding boxes per scene and train the detector. In Table 2, we observe that our pre-training also consistently improves over the baseline VoteNet, and the performance gap does not diminish when more box annotations are available.

5.2. Limited Scene Reconstructions

In this section, we report the experimental results for another scenario of data-efficient 3D scene understanding, when there is a shortage of scene reconstructions. For instance segmentation and semantic segmentation tasks, we random sample subsets of ScanNet scenes of different sizes. We sample {1%, 5%, 10%, 20%} of the entire 1201 scenes in the training set (which corresponds to 12, 60, 120, and 240 scenes, respectively). For object detection, we find it very difficult to train the detector when the scenes are too scarce. Thus we sample {10%, 20%, 40%, 80%} subsets. For each configuration, we randomly sample 3 subsets and report the averaged results to reduce variance. We also use the official ScanNetV2 validation set for evaluation.

Network fine-tuned with our pre-trained model again shows a clear gap compared to the training from scratch baseline (Table 3). We achieve competitive results (50.6% mAP@0.5 for instance segmentation and 64.6% mIoU for semantic segmentation) using only 20% of the total scenes.

Similar behavior can be observed on the object detection task on ScanNet, and the difference between with and without our pre-training is more pronounced in Table 4: the detector can barely produce any meaningful results when the data is scarce (*e.g.* 10% or 20%) and trained from scratch. However, fine-tuning with our pre-trained weights, VoteNet

Data Pct.	Instance Seg.		Semantic Seg.	
	Scratch	Ours	Scratch	Ours
100%	56.9	**59.4** (+2.5)	72.2	**73.8** (+1.6)
1%	9.9	**13.2** (+3.3)	26.0	**28.9** (+2.9)
5%	31.9	**36.3** (+4.4)	47.8	**49.8** (+2.0)
10%	42.7	**44.9** (+2.2)	56.7	**59.4** (+2.7)
20%	48.1	**50.6** (+2.5)	62.9	**64.6** (+1.7)

Table 3: **3D semantic and instance segmentation results with Limited Scene Reconstructions (ScanNet-LR).** Metric is mAP@0.5 for instance segmentation and mIoU for semantic segmentation. "Scratch" denotes the training from scratch baseline, and "Ours" denotes the fine-tuning results using our pre-trained weights. Results using 100% of the data during training are listed in the first row.

can perform significantly better (*e.g.* improve the mAP@0.5 by more than 16% with 20% training data).

Data Pct.	VoteNet (scratch)	VoteNet (ours)
100%	35.4	**39.3** (+3.9)
10%	0.3	**8.6** (+8.3)
20%	4.6	**20.9** (+16.3)
40%	22.0	**29.2** (+7.2)
80%	33.7	**36.7** (+3.0)

Table 4: **Object detection results with Limited Scene Reconstructions on ScanNet.** Metric is mAP@0.5. We show constantly improved results over training from scratch, especially so when 10% or 20% of the data are available. Results using all scenes are listed in the first row.

5.3. Additional Comparisons to PointContrast

As Contrastive Scene Contexts is closely related to PointContrast [65], we provide additional results in this section, including comparisons on the data-efficient ScanNet benchmarks (Table 5) as well as on other datasets and benchmarks (Table 6). Our pre-training method outperforms [65] in almost every benchmark setting, sometimes by a big margin. These results further render the importance of integrating scene contexts in contrastive learning. Notably, our pre-training method on S3DIS achieves 72.2% mIoU which outperforms, for the first time, the *supervised* pre-training result reported in [65].

5.4. Analysis on Active Labeling: Cluttered Scenes

To better explain our active labeling strategy and show that it can work in scenes with heavy occlusion and clutter, we filter out a ScanNet subset of 200 cluttered scenes that has multiple objects per one square meter area. Compared to naive k-means sampling, active labeling performs even better on cluttered scenes. In Figure 7, we visualize a cluttered scene and sampled points (bottom); we also show quantitatively (top) our strategy covers more distinct objects and thus has a balancing effect.

Settings	Task (Metric)	SC	PC [65]	Ours
LA (200 points)	ins (mAP@0.5)	43.5	44.5 (+1.0)	**48.9** (+5.4)
LA (200 points)	sem (mIoU)	65.5	67.8 (+2.3)	**68.2** (+2.7)
LA (7 bboxes)	det (mAP@0.5)	33.4	34.9 (+1.5)	**35.9** (+2.5)
LR (240 scenes)	ins (mAP@0.5)	48.1	48.4 (+0.3)	**50.6** (+2.5)
LR (240 scenes)	sem (mIoU)	62.9	63.0 (+0.1)	**64.6** (+1.7)
LR (960 scenes)	det (mAP@0.5)	33.7	36.3 (+2.6)	**37.4** (+3.7)

Table 5: **Comparisons to PointContrast for data-efficient 3D scene understanding on ScanNet.** We compare our method with PointContrast (PC) and training from scratch (SC) in various tasks. Our method constantly achieves better results in both Limited Point Annotations (LA) and Limited Scene Reconstructions (LR) scenarios.

Datasets	Task (Metric)	SC	PC [65]	Ours
S3DIS	ins (mAP@0.5)	59.3	60.5 (+1.2)	**63.4** (+4.1)
S3DIS	sem (mIoU)	68.2	70.3 (+2.1)	**72.2** (+4.0)
SUN RGB-D	det (mAP@0.5)	31.7	34.8 (+3.1)	**36.4** (+4.7)
ScanNet	ins (mAP@0.5)	56.9	58.0 (+1.1)	**59.4** (+2.5)
ScanNet	sem (mIou)	72.2	**74.1** (+1.9)	73.8 (+1.6)
ScanNet	det (mAP@0.5)	35.4	38.0 (+2.6)	**39.3** (+3.9)

Table 6: **Downstream fine-tuning results on other benchmarks.** Contrastive Scene Contexts (Ours) achieve better or on par results compared to PointContrast (PC) [65] on instance segmentation (ins), semantic segmentation (sem) and object detection (det) across multiple datasets.

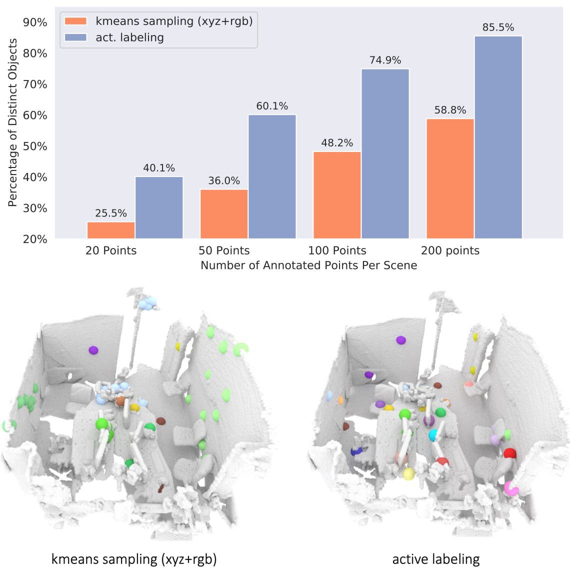


Figure 7: **Top**: object coverage percentage—more distinct objects are covered with active labeling; **Bottom**: Visualization of sampled points in a cluttered scene.

Acknowledgments Work done during Ji's internship at FAIR. Matthias Nießner was supported by ERC Starting Grant *Scan2CAD (804724)*. The authors would like to thank Norman Müller, Manuel Dahnert, Yawar Siddiqui and Angela Dai and anonymous reviewers for their constructive feedback.

References

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3D point clouds. *ICML*, 2018. 2

[2] Iro Armeni, Ozan Sener, Amir R. Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3D semantic parsing of large-scale indoor spaces. In *ICCV*, 2016. 1, 2, 4

[3] Philip Bachman, R Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. In *NeurIPS*, 2019. 2

[4] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *TPAMI*, 24(4):509–522, 2002. 3

[5] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *TPAMI*, 35(8):1798–1828, 2013. 3

[6] Irving Biederman. Recognition-by-components: a theory of human image understanding. *Psychological review*, 94(2):115, 1987. 3

[7] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *NeurIPS*, 2020. 3

[8] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *NeurIPS*, 2020. 2

[9] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. *arXiv preprint arXiv:1512.03012*, 2015. 1

[10] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *ICML*, 2020. 2

[11] Zhiqin Chen, Kangxue Yin, Matthew Fisher, Siddhartha Chaudhuri, and Hao Zhang. BAE-Net: Branched autoencoder for shape co-segmentation. In *CVPR*, 2019. 3

[12] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4D spatio-temporal convnets: Minkowski convolutional neural networks. In *CVPR*, 2019. 2, 5

[13] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3D reconstructions of indoor scenes. In *CVPR*, 2017. 1, 2, 5

[14] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 1, 2

[15] Francis Engelmann, Martin Bokeloh, Alireza Fathi, Bastian Leibe, and Matthias Nießner. 3D-MPA: Multi-Proposal Aggregation for 3D Semantic Instance Segmentation. In *CVPR*, 2020. 2, 4

[16] Matheus Gadelha, Aruni RoyChowdhury, Gopal Sharma, Evangelos Kalogerakis, Liangliang Cao, Erik Learned-Miller, Rui Wang, and Subhransu Maji. Label-efficient learning on point clouds using approximate convex decompositions. *ECCV*, 2020. 3

[17] Matheus Gadelha, Rui Wang, and Subhransu Maji. Multiresolution tree networks for 3D point cloud processing. In *ECCV*, 2018. 2

[18] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012. 2

[19] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jagannath Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 2

[20] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3D semantic segmentation with submanifold sparse convolutional networks. In *CVPR*, 2018. 2

[21] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *NeurIPS*, 2020. 2

[22] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. A papier-mâché approach to learning 3D surface generation. In *CVPR*, 2018. 2

[23] Agrim Gupta, Piotr Dollar, and Ross Girshick. LVIS: A dataset for large vocabulary instance segmentation. In *CVPR*, 2019. 1

[24] JunYoung Gwak, Christopher Choy, and Silvio Savarese. Generative sparse detection networks for 3D single-shot object detection. *ECCV*, 2020. 2

[25] Lei Han, Tian Zheng, Lan Xu, and Lu Fang. OccuSeg: Occupancy-aware 3D instance segmentation. In *CVPR*, 2020. 2

[26] Kaveh Hassani and Mike Haley. Unsupervised multi-task feature learning on point clouds. In *ICCV*, 2019. 2, 3

[27] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020. 2

[28] Olivier J Hénaff, Aravind Srinivas, Jeffrey De Fauw, Ali Razavi, Carl Doersch, SM Eslami, and Aaron van den Oord. Data-efficient image recognition with contrastive predictive coding. *ICML*, 2020. 2, 3

[29] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. *ICLR*, 2019. 2

[30] Ji Hou, Angela Dai, and Matthias Nießner. 3D-SIS: 3D Semantic Instance Segmentation of RGB-D Scans. In *CVPR*, 2019. 2

[31] Ji Hou, Angela Dai, and Matthias Nießner. RevealNet: Seeing Behind Objects in RGB-D Scans. In *CVPR*, 2020. 2

[32] Allison Janoch, Sergey Karayev, Yangqing Jia, Jonathan T Barron, Mario Fritz, Kate Saenko, and Trevor Darrell. A category-level 3d object dataset: Putting the kinect to work. In *Consumer depth cameras for computer vision*, 2013. 1

[33] Haiyong Jiang, Feilong Yan, Jianfei Cai, Jianmin Zheng, and Jun Xiao. End-to-End 3D Point Cloud Instance Segmentation Without Detection. In *CVPR*, 2020. 2

[34] Li Jiang, Hengshuang Zhao, Shaoshuai Shi, Shu Liu, Chi-Wing Fu, and Jiaya Jia. PointGroup: Dual-Set Point Grouping for 3D Instance Segmentation. In *CVPR*, 2020. 2, 4

[35] Marcel Körtgen, Marcin Novotni, and Reinhard Klein. 3D shape matching with 3D shape contexts. In *In The 7th Central European Seminar on Computer Graphics*. Citeseer, 2003. 3

[36] Jean Lahoud, Bernard Ghanem, Marc Pollefeys, and Martin R Oswald. 3d instance segmentation via multi-task metric learning. In *ICCV*, 2019. 2

[37] Jiaxin Li, Ben M Chen, and Gim Hee Lee. SO-Net: Self-organizing network for point cloud analysis. In *CVPR*, 2018. 2

[38] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014. 1

[39] Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. In *CVPR*, 2020. 2

[40] Kaichun Mo, Shilin Zhu, Angel X. Chang, Li Yi, Subarna Tripathi, Leonidas J. Guibas, and Hao Su. PartNet: A Large-Scale Benchmark for Fine-Grained and Hierarchical Part-Level 3D Object Understanding. In *CVPR*, 2019. 1

[41] Sanjeev Muralikrishnan, Vladimir G Kim, and Siddhartha Chaudhuri. Tags2Parts: Discovering semantic regions from shape tags. In *CVPR*, 2018. 3

[42] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. 2

[43] Charles R Qi, Xinlei Chen, Or Litany, and Leonidas J Guibas. Imvotenet: Boosting 3D object detection in point clouds with image votes. In *CVPR*, 2020. 2

[44] Charles R. Qi, Or Litany, Kaiming He, and Leonidas J. Guibas. Deep hough voting for 3D object detection in point clouds. *ICCV*, 2019. 2, 7

[45] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3D object detection from RGB-D data. In *CVPR*, 2018. 2

[46] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3D classification and segmentation. *CVPR*, 2017. 2, 7

[47] Charles Ruizhongtai Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas Guibas. Volumetric and multi-view cnns for object classification on 3D data. In *CVPR*, 2016. 2

[48] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *NeurIPS*, 2017. 2

[49] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 2

[50] Aditya Sanghi. Info3D: Representation learning on 3D objects using mutual information maximization and contrastive learning. *ECCV*, 2020. 2

[51] Jonathan Sauder and Bjarne Sievers. Self-supervised deep learning on point clouds by reconstructing space. In *NeurIPS*, 2019. 2

[52] Gopal Sharma, Evangelos Kalogerakis, and Subhransu Maji. Learning point embeddings from shape repositories for few-shot segmentation. In *2019 International Conference on 3D Vision (3DV)*, pages 67–75. IEEE, 2019. 3

[53] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from RGB-D images. *ECCV*, 2012. 1

[54] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. SUN RGB-D: A RGB-D Scene Understanding Benchmark Suite. In *CVPR*, 2015. 2

[55] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *CVPR*, 2020. 1, 2

[56] Lyne Tchapmi, Christopher Choy, Iro Armeni, JunYoung Gwak, and Silvio Savarese. Segcloud: Semantic segmentation of 3D point clouds. In *3DV*, 2017. 2

[57] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. KPConv: Flexible and deformable convolution for point clouds. In *CVPR*, 2019. 2

[58] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. *ECCV*, 2020. 2

[59] Weiyue Wang, Ronald Yu, Qiangui Huang, and Ulrich Neumann. Sgpn: Similarity group proposal network for 3d point cloud instance segmentation. In *CVPR*, 2018. 2

[60] Xinlong Wang, Shu Liu, Xiaoyong Shen, Chunhua Shen, and Jiaya Jia. Associatively segmenting instances and semantics in point clouds. In *CVPR*, 2019. 2, 4

[61] Yue Wang and Justin M Solomon. Deep closest point: Learning representations for point cloud registration. In *ICCV*, 2019. 2

[62] Wenxuan Wu, Zhongang Qi, and Li Fuxin. PointConv: Deep convolutional networks on 3D point clouds. In *CVPR*, 2019. 2

[63] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *CVPR*, 2018. 2

[64] Jianxiong Xiao, Andrew Owens, and Antonio Torralba. SUN3D: A database of big spaces reconstructed using sfm and object labels. In *ICCV*, 2013. 1

[65] Saining Xie, Jiatao Gu, Demi Guo, Charles R Qi, Leonidas J Guibas, and Or Litany. Pointcontrast: Unsupervised pre-training for 3D point cloud understanding. *ECCV*, 2020. 2, 3, 4, 5, 7, 8

[66] Saining Xie, Sainan Liu, Zeyu Chen, and Zhuowen Tu. Attentional shapecontextnet for point cloud recognition. In *CVPR*, 2018. 3

[67] Bo Yang, Jianan Wang, Ronald Clark, Qingyong Hu, Sen Wang, Andrew Markham, and Niki Trigoni. Learning object bounding boxes for 3D instance segmentation on point clouds. In *NeurIPS*, 2019. 2, 4

[68] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *CVPR*, 2018. 2

[69] Li Yi, Wang Zhao, He Wang, Minhyuk Sung, and Leonidas Guibas. GSPN: Generative shape proposal network for 3D instance segmentation in point cloud. In *CVPR*, 2019. 2

[70] Chenyang Zhu, Kai Xu, Siddhartha Chaudhuri, Li Yi, Leonidas J Guibas, and Hao Zhang. AdaCoSeg: Adaptive shape co-segmentation with group consistency loss. In *CVPR*, 2020. 3

D Deep Learning Basics

Neural network is in the core position of deep learning study. Thus, we will first introduce the fundamentals of neural network in Section D.1. Data-driven approaches, such as neural network, consume huge amount of data to train. In Section D.2, we discuss the basic algorithms used for training a neural network.

D.1 Neural Network

Due to the popularity of mobile devices, people nowadays can easily take pictures or record videos, and share them in social media. As a result, vast amount of data live in the internet. With the huge amount of available data, data-driven approaches, such as machine learning, are evolving very fast. Rather than analytically design the mathematical models, learning algorithms are invented to make computers develop solutions automatically by themselves through extracting features from training data. As a further step of machine learning, deep learning is proposed to consume even more data for better learning. Different from traditional machine learning algorithms that have a bunch of models, such as Random Forest, SVM etc, deep learning builds a single type of model, i.e. neural network.

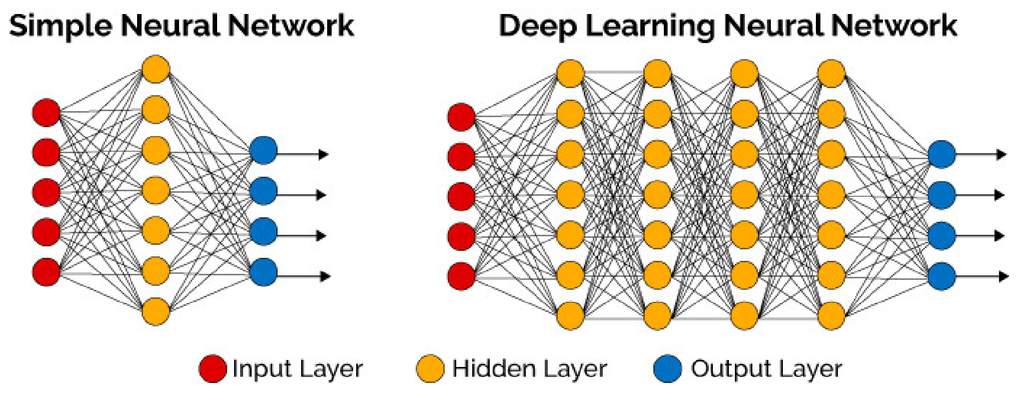


Figure D.1: Illustration of Neural Network [141]. Neural Network is a Mult-Layer Perceptron.

Neural network is structure by layers. The previous layer's output is used as input to the next layer. In the beginning, due to the limitation of hardware, very shallow neural network is generally built called perception. Perceptron is a single layer neural network and a multi-layer perceptron is called Neural Networks (see Figure D.1. Neural Network can simulate arbitrary functions. Let $x \in R^m$ be the input vector, $y \in R^k$ be the output. Network parameters are defined as $\theta \in R^n$. Neural Network models a function mapping $f : R^m \times R^k \to R^n$. As mentioned before, Network f as a multi-layer perceptron is constructed as a composition of functions. Each layer can be seen as function g, of which the weights (neurons) are defined mathematically as a matrix $W \in R^{n\times m}$, and bias is defined as $b \in R^m$. Let input of this layer be $x \in R^n$. Output is then computed as $Wx + b$. The network f of l layers is $f = g_0 \circ g_1 ... g_{l-1}$.

D.1.1 Convolutional Layer

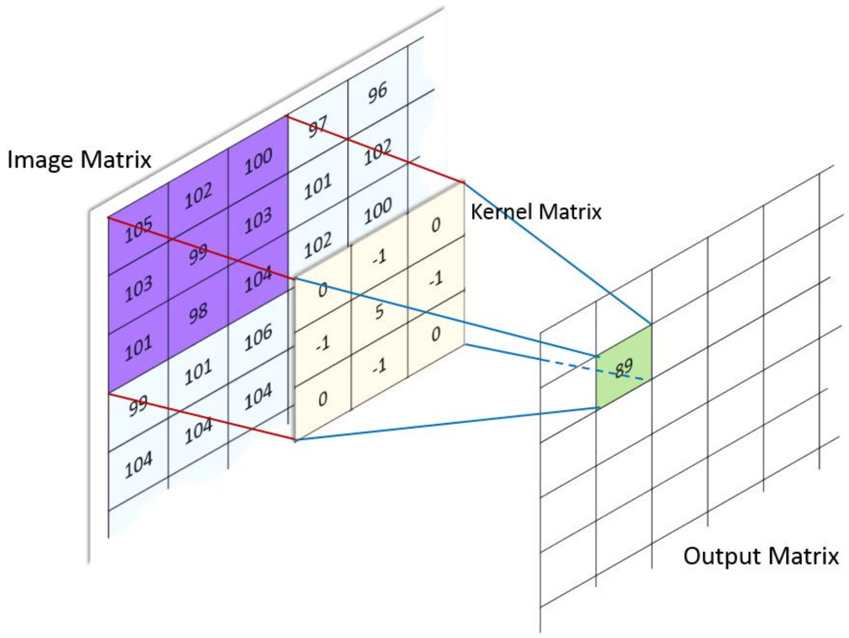


Figure D.2: Convolution Operation [142]. Kernel Matrix is convoluted with Image Matrix in a sliding window fashion.

In computer vision, Convolutional Neural Network [143] is widely used for image-based tasks. In Figure D.1, we show a multi-layer perceptron where each neuron is connected by all the neurons in previous layer (fully connected). Different from perceptron, a Convolutional Neural Network (CNNs) is composited by a series of convlutional layers. In CNNs, weights of each layer are defined as kernels (or filters). Kernel is defined as a $n \times m$ matrix. Matrix size $n \times m$ is also called kernel size. The kernel will operate on the input matrix in a sliding window fashion, as illustrated in Figure D.2. With sliding windows, kernel weights are shared between pixels so that it reduces weights number compared to perceptron. Then output of a convolutional layer is called feature map. Convolution is defined as following.

$$O[m,n] = \sum_j \sum_k k[j,k]F[m-j,n-k] \tag{D.1}$$

Where $O[m,n]$ is the value at $[m,n]$ location in the output feature map, F is the input feature map. Term j and k define the kernel size.

Convolution layers have a couple of parameters must be predefined, such as kernel size. Besides, padding is a necessary operation applied before convolution to keep the input size after convolution. Padding is defined as a size that extends the input feature map on both sides. It can be zero padding, adding zeros on the boundaries or mirror padding, replicating the border values of feature maps. Each convolutional layer can have multiple kernels which is called channels. Each kernel will generate one channel in the output feature map. For instance, 2D convolutional layer generate a three dimensional output feature map with size $m \times n \times c$, where m is the matrix height, n is the matrix width, c is the channel number. Another important parameter is stride, which defines how many pixels we slide the filter. When the stride is 1, then we move the filters one pixel at a time. When the stride is 2, then the filters jump 2 pixels at a time as we slide them around. This will produce smaller output volumes spatially. Knowing the filter size, padding and strides, we can easily compute the output feature map size in the following equation.

$$O_s = \frac{W - F + 2P}{S} + 1 \tag{D.2}$$

where O_s denotes the output size; W indicates the input size; F is the kernel size; P means padding, and S is the stride.

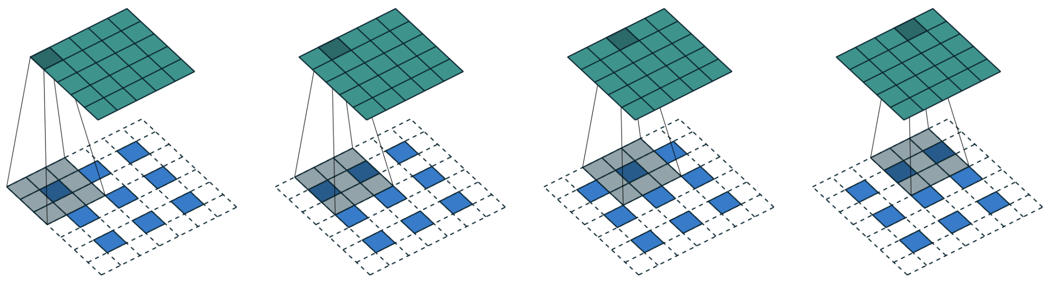


Figure D.3: Deconvolution Operation [142]. Deconvolution increases the input size from 3×3 to 5×5 with kernel size 3, padding 1, and stride 2.

Transposed Convolution (Deconvolution). Convolution layer normally reduces the input feature map size. When up-sampling is desired, people often use transposed convolution. This operation can be seen as a reverse of convolution, which increases the input feature map size. For instance, input size can be decreased by half by setting stride as 2 in convolution. But in deconvolution with stride 2, it doubles the input size as presented in Figure D.3. In deconvolution, we can also easily compute the output size by reversing the the Equation D.2.

D.1.2 Pooling Layer

A pooling layer is a new layer added after the convolutional layer. specifically after a non-linearity. For example, the layers in a model may look as following order:

- Input Image
- Convolutional Layer
- Nonlinearity
- Pooling Layer

The addition of a pooling layer after the non-linearity is a common pattern used for down sampling the feature map size. Pooling involves selecting a pooling operation with a pre-defined kernel. Similar to convolution, it also requires paremeters like stride, padding etc. Thus, we can use the same Equation D.2 to compute output size after pooling layer. Different from convolution, the pooling operation has no learnable weights, but has to be specified. Two common functions used in the pooling operation are:

Average Pooling. Calculate the average value for each patch on the feature map. Mathematically, it is defined as following,

$$O[m,n] = \frac{\sum_{j=0\ldots h}\sum_{k=0\ldots w} F[m-j,n-k]}{h \times w} \tag{D.3}$$

where $O[m,n]$ denotes the value after pooling layer at location $[m,n]$ and F is the input feature map; h,w means the filter's height and width.

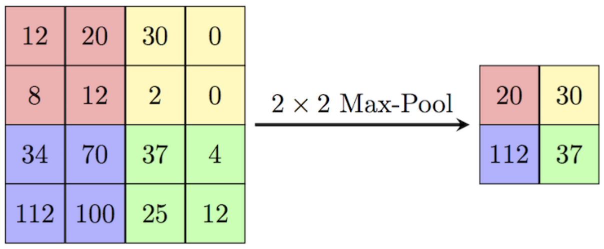


Figure D.4: Max Pooling Operation [144]. Max Pooling Operation takes the maximum value in each divided sub-region. In implantation, it also needs to track down the indices for back propagation purpose.

Max Pooling. Calculate the maximum value for each patch of the feature map. We show a graphical illustration in Figure D.4.

$$O[m,n] = \max_{i=0\ldots h, j=0\ldots w} F[m-i,n-j] \tag{D.4}$$

where $O[m, n]$ denotes the value after pooling layer at location $[m, n]$ and F is the input feature map; h, w means the filter's height and width.

D.1.3 Linear Layer

Linear Layer (Fully Connected Layer) in a neural networks is the layer where all the neurons from previous layer are connected to all the neurons in the next layer (see Figure D.1). Fully connected layers are normally used in classification tasks. In classification model, the output is a one-dimensional fixed-length vector indicating the probabilities for each class label. The feature map is usually linearized after convolutional layer and then mapped to the fixed-length vector via linear layer. Linear layer consumes huge memory compared to convolutional layer, since it connects to very previous neurons. In some other tasks, such as semantic segmentation, people usually use fully convolutional network, in which there are no linear layers. Because the input image could be arbitrary sizes and the output needs to be a one-to-one mapping to input in such task.

D.1.4 Normalization Layer

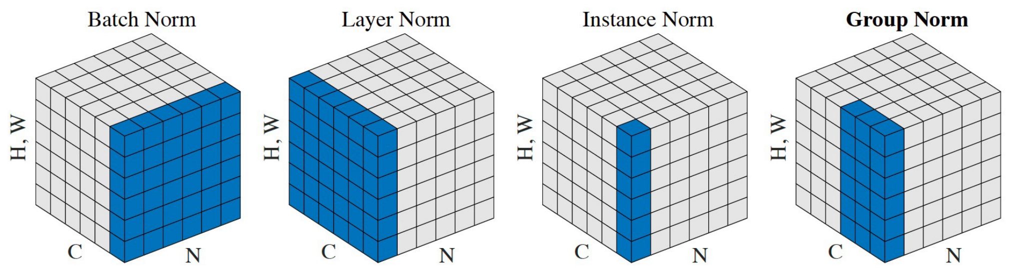


Figure D.5: Differences between Normalization Layers. Different normalization techniques normalize across different dimensions in the feature map [145].

Normalization layer plays a very important role in neural network. Normalization layer standardizes the inputs distribution to a learned mean and variance. Through normalization Layers, feature maps become more statistically pleasing, and easier to learn as well as faster to converge. In this section, we introduce the commonly used normalization layers, such as Batch Norm, Layer Norm, Instance Norm and Group Norm. Illustration of differences between those normalization layers can been seen in Figure D.5

Batch Normalization (BN) normalizes each input in the current mini-batch by subtracting the input mean and dividing it by the standard deviation in the current mini-batch. Each layer does not expect inputs with zero mean and unit variance, but instead some other mean and variance throughout the training data. Hence the BN layer also introduces two learnable parameters γ and β. Consider a mini-batch $\mathcal{B}$ of size m. To this end, we have m values output from activation function in the mini-batch,

$$\mathcal{B} = \{x_{1...m}\}.$$

Let the normalized values be $y_{1...m}$. We demonstrate the BN in Algorithm 1. In the algorithm, ϵ is a constant added to the mini-batch variance for numerical stability. In short words, Batch Normalization calculates mean and variance for each individual channel across all samples and both spatial dimensions.

```
Require: Values of $x$ over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
         Parameters to be learned: $\gamma$, $\beta$
Ensure: $\{y_i = BN(x_i, \gamma, \beta)\}$

    $\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$                                   mini-batch mean
    $\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2$       mini-batch variance
    $y_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}$       normalize
```

Algorithm 1: Batch Normalization. Normalization is applied to the output x over a mini-batch after activation functions.

Layer Normalization (LN) directly computes the normalization statistics from the summed inputs to the neurons within a hidden layer so the normalization does not introduce any new dependencies between training cases. It works well for modelling sequences, e.g. in RNNs or transformers. Layer normalization statistics is computed over all the hidden units in the same layer as follows:

$$\mu_i = \frac{1}{m}\sum_{j=1}^{m} x_{ij} \tag{D.5}$$

$$\sigma_i^2 = \frac{1}{m}\sum_{j=1}^{m}(x_{ij} - \mu_i)^2 \tag{D.6}$$

$$y_{ij} = \frac{x_{ij} - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}} \tag{D.7}$$

In short words, Layer Normalization estimates mean and variance for each individual sample across all channels and both spatial dimensions. In layer normalization, all the hidden units in a layer share the same normalization terms. Unlike batch normalization, layer normalization does not impose any constraint on the size of the mini-batch and it can be used when batch size is 1.

Instance Normalization (IN) calculates mean and variance for each individual channel for each individual sample across both spatial dimensions. Unlike batch normalization, the instance normalization layer is applied at test time as well, due to non-dependency of mini-batch. This technique is originally devised for style transfer, the problem instance normalization tries to address is that the network should be agnostic to the contrast of the original image.

Group Normalization (GN) divides channels into groups and normalizes the features within each group. GN does not exploit the batch dimension, thus its computation is independent of batch sizes. In the case where the group size is 1, it is equivalent to Instance Normalization.

D.1.5 Activation Functions

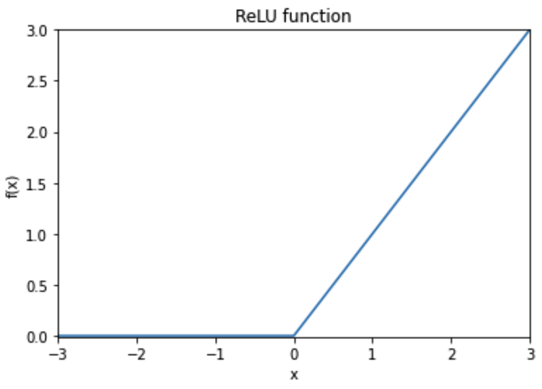


Figure D.6: ReLU Activation Function [146]. The gradient is cut in the negative axis.

An activation function is a function used in artificial neural networks which introduces non-linearity. If the inputs lies in a certain range of values, the activation function ”fires”, otherwise it does nothing or nearly nothing. In other words, an activation function is like a gate that checks if an incoming value is in a certain range. Activation functions are very useful because they add non-linearities into neural networks, which allows neural networks to learn powerful operations. Without activation functions, the entire network could be re-factored to a simple linear operation or matrix transformation on its input, and it would no longer be capable of performing complex tasks such as image recognition. Well-known activations include the rectified linear unit (ReLU) function, and the family of sigmoid functions, such as the logistic sigmoid function.

Rectified Linear Unit (ReLU) is a piece-wise linear function that outputs zero if its input is negative, and directly outputs the input otherwise. Mathematically, it is defined as following (see its graph in Figure D.6).

$$f(x) = max(0, x) \tag{D.8}$$

ReLU cuts off the gradients in back-propagation when $x < 0$ as showed in Equation D.9. When $x < 0$, it passes 0 gradients to network weights, thus does not update the weights at all.

$$\frac{df(x)}{dx} = \begin{cases} 1, & \text{if } x > 0. \\ 0, & \text{if } x < 0. \\ \text{undefined, otherwise.} \end{cases} \tag{D.9}$$

Leaky Rectified Linear Unit (LReLU) is proposed to solve dying ReLU (no gradients when $x < 0$). LReLU still has very small gradient in negative region, so that the gradient is nonzero at all points except 0 where it is undefined (see Figure D.7.)

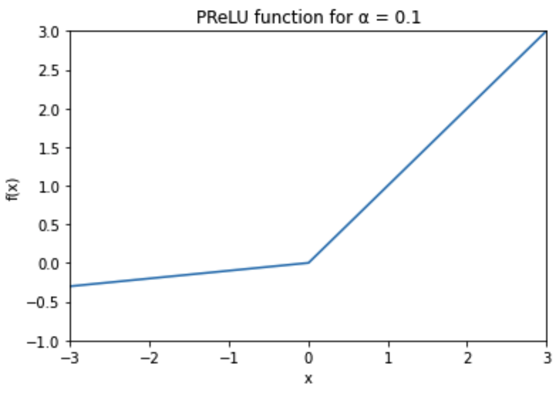


Figure D.7: Leaky ReLU Activation Function [146]. Simlar to ReLu, but the gradient changes slowly in negative part.

$$S(x) = \frac{e^x}{e^x + 1} \tag{D.10}$$

Logistic Sigmoid Function has a useful property that its gradient is defined everywhere. Besides, its output is conveniently between 0 and 1 for all inputs. Because of this nice property, it is often used at the last to convert logits to probabilities. The logistic sigmoid function is easier to work with mathematically, but the exponential functions make it computationally intensive to compute. In practice, simpler functions such as ReLU are often preferred. The mathematical definition is in Equation D.10 (see Figure D.8).

Softmax is a function that converts multiple values into probabilities and guarantees the summation of those values equals to 1. Mathematical definition is showed in Equation D.11. It is often used at the end to convert logits to probabilities for the multi-label classification task. Notably, softmax is a generalization of sigmoid. When there are 2 classes (binary classification), softmax is equivalent to sigmoid function.

$$S(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{C} e^{x_j}} \tag{D.11}$$

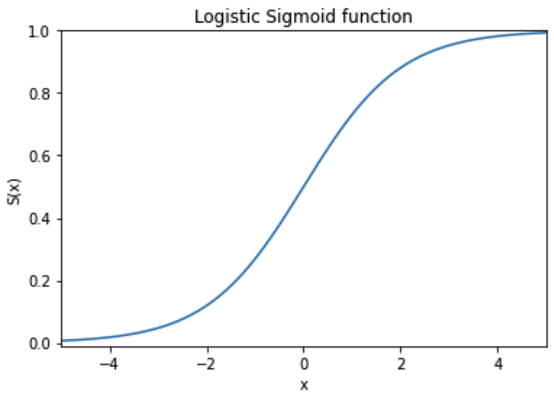


Figure D.8: Sigmoid Function [146]. Values are activated sharply when x is centered around 0 and get flat when $x \rightarrow +\infty$ or $x \rightarrow -\infty$.

D.2 Training Neural Network

In this section, we introduce the algorithms used for trianing a neural network model. We first illustrate a simple network for digit recognition in Figure D.9. When training a network in computer vision, we input an image that goes through a serious convolutional layers (Conv_1 and Conv_2 in Figure D.9) and max pooing layers(Max-Pooing). The feature map out of the last convolutional layer is linearized into 1D vector. Fully-Connected layers (fc_3 and fc_4) further operates on the vector to generate a 1D vector with length 10 (there are 10 digits from 0 to 9 in total). This procedure is called forward pass.

The output of forward pass is compared with ground-truth to compute a loss. We introduce commonly used loss in Section D.2.3. Then the network weights are updated through back-propagation (in Section D.2.1) and Stochastic Gradient Descent (in Section D.2.2). Before training, there are multiple way to initialize network weights. We introduce network initialization in Section D.2.4. In the training, learning rates are updated as well. Different learning rate schedulers are presented in Section D.2.6. To prevent over-fitting problem, we introduce several regularization techniques commonly used in Section D.2.5.

D.2.1 Back-propagation

In 1986, back-propagation was proposed to train a multi-layer neural network [148]. Since decades, it is still the fundamental algorithms to train a neural network. The core concept for back-propagation is to apply chain rule in the gradient computations. For

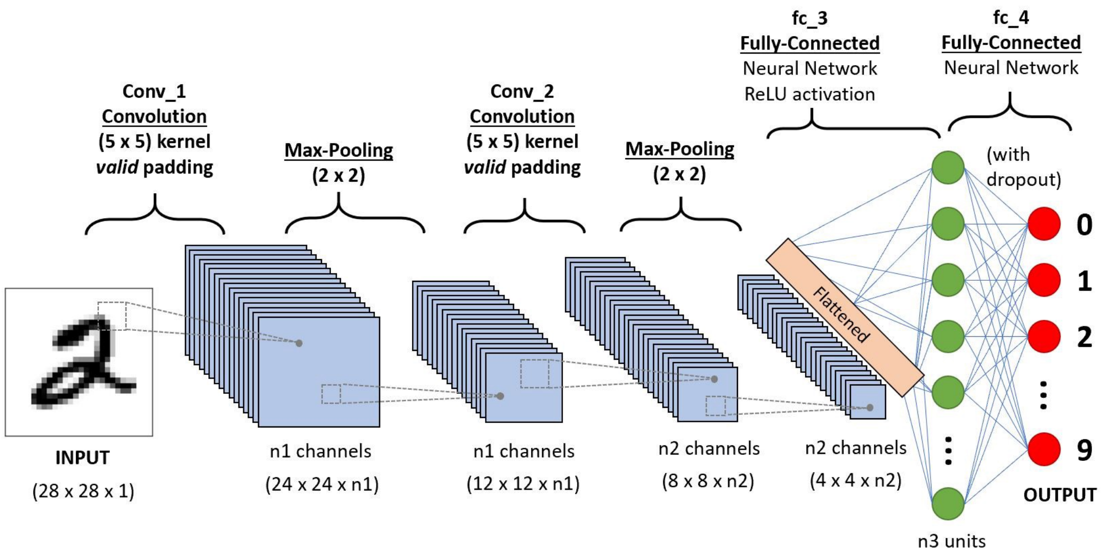


Figure D.9: Illustration of a Neural Network Model [147]. We show a simple network model to better explain each parts in training process.

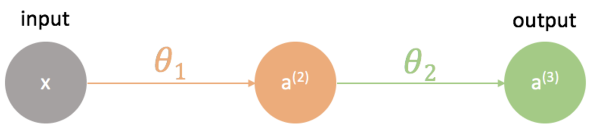


Figure D.10: Illustration of Back-Propagation. We show a naive case in back-propagation to show how the chain rule works [141].

example, we demonstrate a naive case in Figure D.10. Let $a^{(3)}$ be the output, and we want to update θ_1. We first need to compute a loss based on output $a^{(3)}$ and the ground-truth. For simplicity, let this loss be $L(a^{(3)})$. Then we compute the partial derivatives of $L(a^{(3)})$ with respect to θ_1 along the path as presented in Equation D.12.

$$\frac{\partial L(a^{(3)})}{\partial \theta_1} = \frac{\partial L(a^{(3)})}{\partial a^{(3)}} \frac{\partial a^3}{\partial a^2} \frac{\partial a^2}{\partial \theta_1} \tag{D.12}$$

In neural network, this is very efficient to update weights. Because the weights are updated layer by layer, the gradients on the path don't need to be computed twice. For instance, we can re-use $\frac{\partial L(a^{(3)})}{\partial a^{(3)}}$ for computing $\frac{\partial L(a^{(3)})}{\theta_1}$, since we would have already computed it when updating θ_2.

D.2.2 Stochastic Gradient Descent

Gradient Descent is commonly used in differential calculus. It expresses a relationship between two variables: "y over x". In this case, the y is the error produced by the neural network, and x is the parameter of the neural network. The parameter has a relationship to the error, and by changing the parameter, we can increase or decrease the error. Back-propagation tells us how to compute the gradient, and Stochastic Gradient Descent (SGD) tells how to update the weights by gradients for each training sample. Following the convention from last section D.2.1, we can update the weights θ_1 with SGD as presented in Equation D.13.

$$\theta_1 = \theta_1 - \eta \cdot \frac{\partial L(a^{(3)}; x^{(i)}, y^{(i)})}{\partial \theta_1} \tag{D.13}$$

where $x^{(i)}$ and $y^{(i)}$ are the input and target in one training sample, and η is the learning rate indicating the step size.

D.2.3 Training Loss

In previous sections, we show how to compute the gradient from loss. Neural network is capable to solve a variety of tasks. Different tasks require different losses. In this section, we introduce the commonly used losses for different tasks.

L_2 Loss is the most commonly used loss function for regression. It aims to reduce the $L2$ distance between prediction and ground-truth as presented in Equation D.14. As it computes the mean of the squared values, it is also called Mean Squared Error (MSE).

$$L(y, \hat{y}) = \frac{1}{N}\sum_{i=0}^{N}(y_i - \hat{y}_i)^2 \tag{D.14}$$

L_1 Loss is also commonly used in regression task. Different from L_2 loss, it computes the mean of L_1 norm between predicted values and ground-truth values, as showed in Equation D.15. Compared to L_2 loss, L_1 loss is less sensitive to outliers. As it uses absolute values between predicted values and ground-truth, it is also called Mean Absolute Error (MAE).

$$L(y, \hat{y}) = \frac{1}{N}\sum_{i=0}^{N}|y_i - \hat{y}_i| \tag{D.15}$$

Cross Entropy Loss is usually used in classification task. We show its mathematical definition in Equation D.16.

$$CE = -\sum_{i=1}^{C} t_i log(softmax(s_i)) \tag{D.16}$$

where t_i (either 1 or 0) denotes the ground-truth label, and s_i is the output of network.

Binary Cross Entropy Loss is a special case of CE. When there are 2 classes (binary classification), CE is transformed into BCE. Mathematical definition of BCE is showed in Equation D.17.

$$BCE = -\sum_{i=1}^{C=2} t_i log(f(s_i)) = -t_1 log(f(s_1)) - (1 - t_1) log(1 - f(s_2)) \tag{D.17}$$

where f denotes the sigmoid function.

D.2.4 Network Initialization

This optimization always requires a starting point. Weight initialization is a procedure to set the weights of a neural network to some values that define the starting point for the optimization, in another work training. Network Initialization is a very important factor that could significantly influence network training. One of the most successful story in computer vision is transfer learning. Normally, people like to initialize the network weights learned from other data or loss, so that it can transfer across knowledge domains. In fact, most downstream tasks in 2D computer vision will use ImageNet Pre-trained weights as initialization. And it turns out very useful especially for fine-tunning on smaller datasets. In this section, we will introduce several popular network initialization methods and give out their mathematical definitions for detailed analysis.

Xavier Initialization is calculated as a random number with a uniform probability distribution between the range [149]. Let n be the number of inputs to the neurons. Weights are sampled from a uniform distribution between the range $[-\frac{1}{\sqrt{n}}, +\frac{1}{\sqrt{n}}]$. As weights are multiplied with input values, small weights make the values through the neurons small, and too large weights make signal grow too fast until it's too massive to be useful. Xavier initialization makes sure the weights are just right, keeping the signal in a reasonable range even through many layers in the first forward pass.

He Initialization is calculated as a random number with a Gaussian probability distribution with a mean of 0.0 and a standard deviation of $\sqrt{\frac{2}{n}}$, where n is the number of inputs to neurons [150]. The way of He Initialization looks very similar to Xavier, however, they play different roles. It is related to the non-linearities used in the network. Kumal et al. [151] proves mathematically that He Initialization is the best initialization strategy for the ReLU activation function, which is non-differentiable at 0.

Pre-trained Initialization uses a pre-trained weights to initialize the network. It turns out to outperform previously introduced initialization by a large margin, as it has learned priors. However, this method is not flexible as others. In general, your network structure must be very similar to some pre-defined architectures. For instance, ImageNet Pre-trained weights normally require a standard backbone, e.g. ResNet50 or ResNet101. In practise, it also forces the naming rules to be the same, so that the pre-trained weights can be matched.

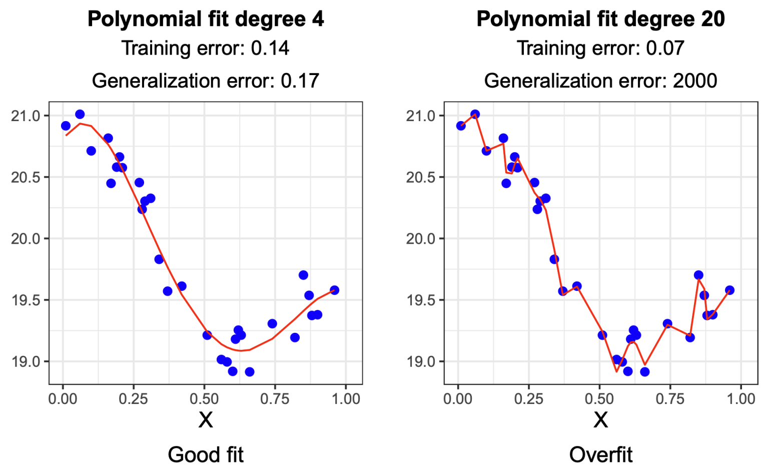


Figure D.11: Over-fitting. We show a classic over-fitting case of polynomial fit [152]

D.2.5 Regularization

Regularization techniques are widely used to prevent over-fitting problems. Over-fitting can be well explained by Figure D.11. As illustrated in the figure, we want to fit a polynomial equation for some sampled points. Over-fitted curves would go through every points, so that it has lower training error. However, it does not really represents the curves, from which the points are sampled. To this end, it has very high generalization errors. In this section, we introduce normally used regularization techniques in neural network training.

Weight Decay reduces the weights regularly during the training. As a observation of over-fitting cases, the weights usually have extremely large numbers for fitting every seen data points. To this end, decaying the weights by L_2 norm is often used to prevent large weights as well as over-fitting. In practice, we multiply the sum of squares of weights with so small number and this small number is called weight decay. This term is added in loss term as presented in Equation D.18.

$$Loss = MSE(\hat{y}, y) + w_d \sum |w|^2 \tag{D.18}$$

where MSE is mean squared error as introduced previously, and it could be arbitrary loss; w_d is so called weight decay. In practice, it should be a very small number, e.g. 0.005.

Data Augmentation aims to augment training data. Training with more data is always a good strategy for preventing over-fitting. However, training data is normally very expensive. Thus, data augmentation, such as rotation, crop and rescale etc., is invented to artificially generate more data.

Dropout introduces randomness in the training. During training, a number of nodes are randomly ignored or "dropped out". By dropping, it refers to temporarily removing it from the network, along with all its incoming and outgoing connections. To this end, this makes the layer act like another layer with a different number of nodes and connectivity. In effect, each update to a layer during training is performed with a different "view" of the configured layer. By introducing randomness in the training with this way, dropout can significantly prevent network from over-fitting.

D.2.6 Learning Rate Scheduler

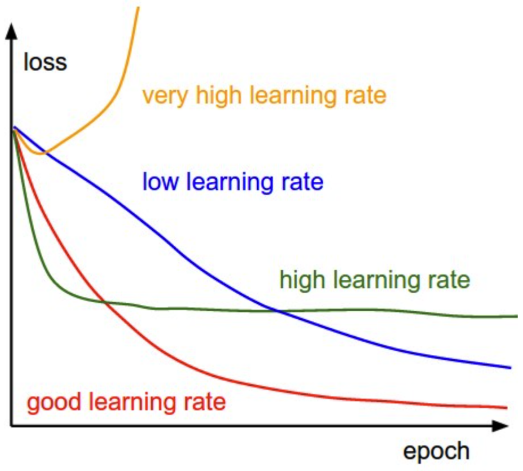


Figure D.12: Learning Rates. Choosing the correct learning rates can significantly influence training process [153].

During optimization, the learning rates are adapted as training steps, which can hugely influence the training results. As showed in Figure D.12, having a good learning rate decides the Learning rate scheduler, as its name suggests, is used to adjust the learning rate. In this section, we introduce several commonly used learning rate schedulers. We further show the curves of different learning rate schedulers in Figure D.13.

Constant Learning Rate Scheduler maintains the learning rate to be the same as its initialized learning rate throughout the whole training procedure. In practice, it is the default setup, if no learning rate scheduler is specified.

Exponential Learning Rate Scheduler decays the learning rate in a step-based fashion. It reduces the learning rates by scales computed by a fixed hyper-parameter every iteration, as presented in Equation D.19.

$$lr = lr_0 \cdot e^{-kt} \tag{D.19}$$

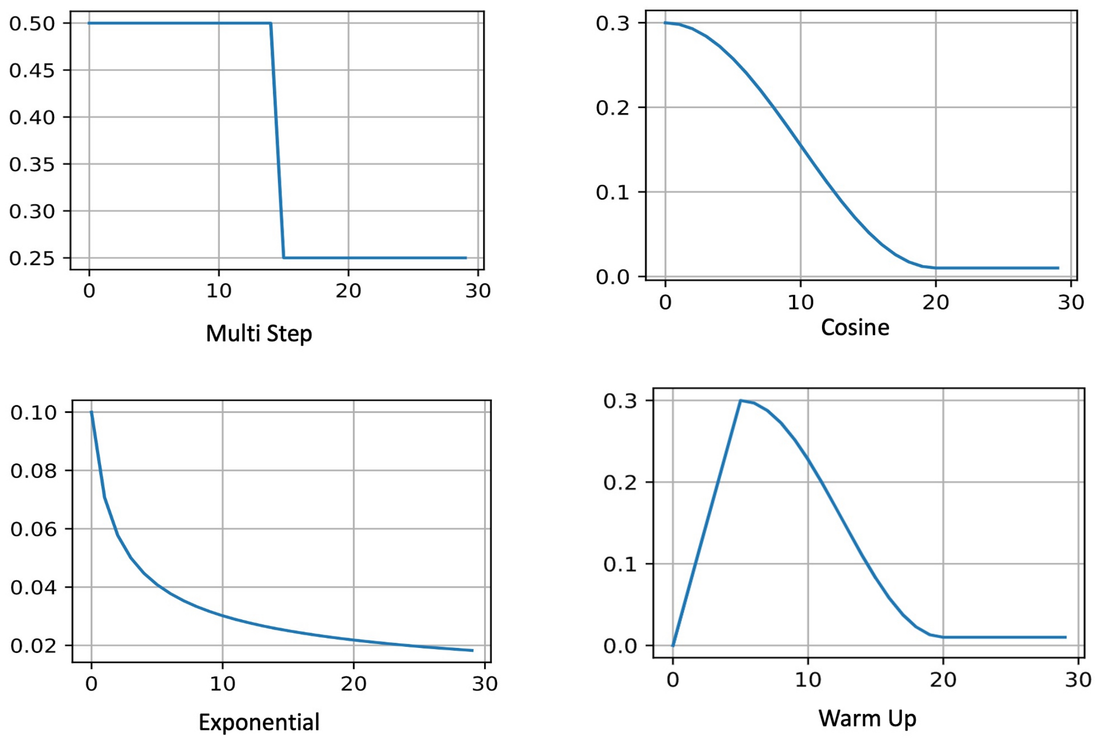


Figure D.13: Learning Rate Schedulers [141]. X-axis denotes training epochs, and y-axis demonstrates the change of learning rates.

where lr_0 is the initialized learning rate; k is the hyper-parameter and t is the iteration number.

Multi-Step Learning Rate Scheduler reduces the learning rate by scale s every n steps. Hyper-parameters s and n need to be specified before training starts. This scheduler can be seen as a piece-wise function of training step. In a specific range of training steps, learning rate remains the same, and learning rate decays by a fixed ratio.

Cosine Learning Rate Scheduler reduces the learning rate in a cosine-like curve as showed in Figure D.13. It is inspired by the observation that it is not so ideal to decrease the learning rate too drastically in the beginning and in the end we might want to "refine" the solution using a very small learning rate. This results in a cosine-like schedule with the following functional form for learning rates.

$$lr_n = lr_N + \frac{lr_0 - lr_N}{2}(1 + cos(\frac{\pi n}{N})), \quad n \in (0, N) \tag{D.20}$$

where n denotes the current training step, and N means the maximum training steps. Thus, lr_0 is the initialized learning rate, and lr_N is the learning rate when training ends. Notably, lr_0 is bigger than lr_N, as learning rate decays during the training.

Warmup Learning Rate Scheduler increases the learning rate in the beginning phase of training. As illustrated in Figure D.12, a large learning rate may lead to unstable optimization problems, whereas a low learning rate could make the training process extremely slow. To prevent divergence in the beginning as well as slow convergence, a rather simple fix for this dilemma is proposed, namely a warmup period. In the start, the learning rate increases to its initial maximum and then cool down until the end of the optimization process. For simplicity, a linear increase can be adopted, and this leads to a schedule of the form indicated in Figure D.13.

Acronyms

1D, 2D, 3D, ...	n spatial dimentions.
Adam	Adaptive Moment Estimation.
BCE	Binary Cross Entropy.
BN	Batch Normalization.
CE	Cross Entropy.
CNNs	Convolutional Neural Networks.
CPU	Central Processing Unit.
FN	False Negative.
FP	False Positive.
GN	Group Normalization.
GPU	Graphical Processing Unit.
IMU	Inertial Measurement Unit.
IN	Instance Normalization.
LA	Limited Annotations.
LiDAR	Light Detection and Ranging.
LN	Layer Normalization.
LR	Limited Reconstructions.
LReLU	Leaky Rectified Linear Unit.
MAE	Mean Absolute Error.
MLP	Multi-Layer Perception.
MSE	Mean Squared Error.
MVS	Multi-View Stereo.
NMS	Non Maximum Suppression.
PReLU	Parametric Rectified Linear Unit.
ReLU	Rectified Linear Unit.

SDF	Signed Distance Field.
SfM	Structure from Motion.
SGD	Stochastic Gradient Descent.
SIFT	Scale-Invariant Feature Transform.
SSC	Semantic Scene Completion.
ST	Spatial Transformer.
STNs	Spatial Transformer Networks.
SVM	Support Vector Machine.
TP	True Positive.
TSDF	Truncated Signed Distance Field.

List of Tables

List of Figures