DEPARTMENT OF INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Master's Thesis in Data Engineering and Analytics

Semantic Image Manipulation Using Objectwise Features

Bc. Pavel Jahoda

DEPARTMENT OF INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Master's Thesis in Data Engineering and Analytics

Semantic Image Manipulation Using Objectwise Features

Semantische Bildmanipulation mit objektbezogenen Merkmalen

Author:	Bc. Pavel Jahoda
Supervisor:	M.Sc. Azade Farshad
Advisor:	Prof. Dr. Nassir Navab
Submission Date:	April 14th, 2023

I confirm that this master's thesis in data engineering and analytics is my own work and I have documented all sources and material used.

Munich, April 14th, 2023 Bc. Pavel Jahoda

Acknowledgments

It has been a pleasurable and stress-free experience working with Azade Farshad. A great advisor who skillfully managed to find a balance between guidance and providing independence. I want to thank Professor Nassir Navab for this opportunity.

I am very grateful to my family and friends for their love and support. My mom, for always checking up on me. My dad, for always providing me with tangible and informational support, especially for sharing my programming interest during my high-school years. My brother for always sharing his interest in my hobbies. My long-term friends Patrik Simonek, Blahoslav Rataj, Thi Phuong Vo, Amanda Robillard, and Matej Skovran for always being there for me.

I am deeply indebted to Vanda Hendrychova, Ngan Vu, and Martin Svarc for their emotional support when I fought cancer during the second semester of TUM.

I want to express my gratitude to Yixuan Liu, Cosima Raedler, Ezgi Köse, Xuanpu Hong, and Borui Li who accompanied me in the library while I was working on my thesis. Thanks to them, I was looking forward to visiting the library.

I also want to thank all the people that uplifted my spirit through fun activities, including but not limited to, drone development at Horyzn, Chinese learning, bouldering, and cooking.

Last but not least, I would like to thank my girlfriend Yuchen Chou for being there for me, going to the library with me, and giving me fuel to work on my goals.

-Pavel

Abstract

The utilization of scene graphs for image manipulation represents a highly promising direction in the field of computer vision. However, generating high-quality images from scene graphs can be challenging due to the complexity of the scenes and the high diversity of the objects in datasets such as Visual Genome. To address these challenges, we present a novel Progressive Restoration framework for Scene Graph-based Image Manipulation (PRISM). As a part of PRISM, we've developed and extensively evaluated several novel approaches that individually improve the image manipulation capabilities of the system. Our end-to-end framework leverages image reconstruction through a progressive restoration process, providing additional context information that enables more precise image manipulation. We took advantage of the outer part of the masked to-be-manipulated area as they have a stronger correlation with the context of the scene, and in our end-to-end framework, we designed a progressive denoising process for image reconstruction that continuously decreases the size of the masked region in the image. Moreover, our multi-task architecture simultaneously reconstructs the entire image as well as selected image objects in detail, generating high-quality and detailed images. Our model outperforms the state-of-the-art methods in the semantic image manipulation task on the CLEVR and Visual Genome datasets. Our results demonstrate the potential of our approach for enhancing the quality and precision of scene graph-based image manipulation. Finally, we propose a new research avenue by showcasing the benefits of incorporating progressive generation into diffusion processes.

Contents

1. Introduction

1.1. Motivation and Problem Statement

As early as the 1960s, computer vision researchers experimented with the first methods for image manipulation tasks such as image restoration and image enhacement [1]. Since then, there has been rapid progress due to deep learning methods which have transformed the computer vision research landscape [2, 3]. The progress of deep learning methods has led to generative models [4, 5, 6, 7] capable of manipulating and generating first semi-realistic images. Generative Adversarial Networks [6] or GANs for short in particular have transformed the computer vision research landscape through their potential to generate natural-looking images. With the GAN improvements in terms of training stability [8, 9], aided by newly introduced losses that focus on high-level differences between images [9, 10], GANs have become models of choice for image generation and image manipulation tasks.

GANs have been used in several image alteration tasks such as semantic image manipulation [11, 12], image inpainting [13, 14], or image outpainting [15]. There are two prevalent approaches to image manipulation tasks. First, an approach that generates images according to user modification specification done directly on the image [16, 17, 18]. Second, an approach that first generates an intermediate feature representation of the image, and then generates an image based on the user's intermediate representation modification of the image [12, 15, 19].

Until 2018, state-of-the-art methods focused on generating images from text descriptions. Such representation comes with many disadvantages. Firstly, describing scenes with a lot of objects in a way that captures all the relationships between the objects requires long paragraphs of text. Furthermore, it is rather difficult to accurately describe the positional relationship between objects with a high degree of precision. On the other hand, segmentation maps, which offer a high level of precision, lack high-level abstraction. Furthermore, manipulating images through segmentation maps requires a high level of user input. Scene Graphs [20] are a prominent feature representation that alleviates these shortcomings. Scene graphs are a structural representation of a scene typically in an image via graphs where a node in the graph usually represents an object. These nodes along with their corresponding attributes are encoded using a feature vector encoding and are connected via edges which signify a relationship or connection between the node objects. They encode spatial relationship information between objects and provide attribute binding for objects [21]. For instance, <"food",

“on”, “ceramic plate”>, <“table”, “under”, “ceramic plate”> is an example of such a scene graph. Having such a powerful representation allows the user to achieve a number of abstract modifications while simultaneously having a level of control unachievable by other approaches.

In this work, we address the problem of image manipulation from scene graphs, in which a user can modify images by applying changes in the nodes or edges of the semantic graph representation of the image. This work builds upon the work of Dhamo et al. called Semantic Image Manipulation Using Scene Graphs [12], or SIMSG for short. While many image manipulation works require image pairs of the before and after image changes, SIMSG is trained by reconstructing partially masked input images so it does not require the image-pair data. The work offers a complete solution for image manipulation in low resolution. We propose a framework for Progressive Restoration for Scene Graph-based Image Manipulation (PRISM). PRISM's advancement over SIMSG is manifold. First, we present a novel multi-task approach for image manipulation. The multi-task approach is realized through a Generative Adversarial Network (GAN) featuring a decoder comprised of two heads. One head focuses on generating a detailed reconstruction of the masked image parts (which corresponds to the manipulated parts during inference) while the other head focuses on holistic image reconstruction. Second, we utilize a progressive restoration process for image reconstruction, which provides additional context information that enables more precise image manipulation. We analyze these techniques quantitatively in an image restoration task and qualitatively through performing image manipulation. We combine these techniques into a novel progressive multi-task image manipulation approach that significantly outperforms the state-of-the-art methods, resulting in high-quality and detailed images that are more faithful to the original scene graph. An example of the manipulated images generated by PRISM and SIMSG can be seen in fig. 1.1. Finally, we explore a novel research direction of incorporating progressive generation methods into diffusion processes.

1.2. Thesis Outline

The subsequent thesis text is divided into six chapters and has the following structure.

Chapter 2 First, we provide comprehensive theoretical background information. The chapter gives the necessary knowledge required for an understanding of the different parts of the proposed system. It gives an overview of self-supervised learning, followed by an explanation of different generative models in section 2.3. This is finally followed by a primer on Graph Convolutional Networks and Scene Graphs in section 2.4 and section 2.5 respectively.

Chapter 3 In this chapter, we provide a survey of different subfields that were

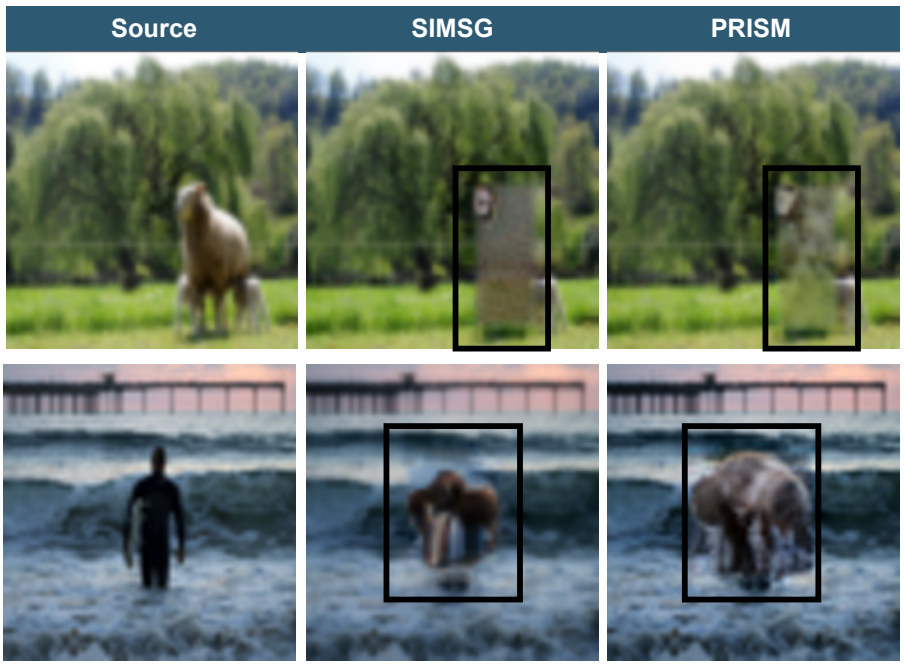


Figure 1.1.: Our method demonstrates significant improvement in capturing image details compared to SIMSG [12] as shown through two different image manipulation tasks: object removal **(top row)**, replacing man for an elephant **(bottom row).**

introduced in the theoretical background chapter. In the chapter, we mainly focus on state-of-the-art image manipulation and generation methods that make use of graph convolutional networks. Especially we focus on methods that take advantage of an intermediate scene graph representation of the manipulated images.

Chapter 4 This is followed by a description of the final proposed system as well as all the methods we've implemented.

Chapter 5 We then provide quantitative and qualitative analysis of the proposed system, including several ablation studies, and a discussion of these results.

Chapter 6 Finally, we give the conclusion and future directions derived from the thesis.

2. Theoretical Background

This chapter provides the necessary definitions, theory, and comprehensive background information for this work. In section 2.1 we will provide an explanation for important terms used throughout the thesis. Then we will touch on self-supervised learning in section 2.2. In section 2.3 called Generative Models, we will go over Normalizing Flows, Variational Autoencoders (VAEs) before diving deeper into Generative Adversarial Networks (GANs), and Diffusion Models. Afterward, we will discuss Graph Convolutional Networks in section 2.4 before finally describing Scene Graph representation in section 2.5.

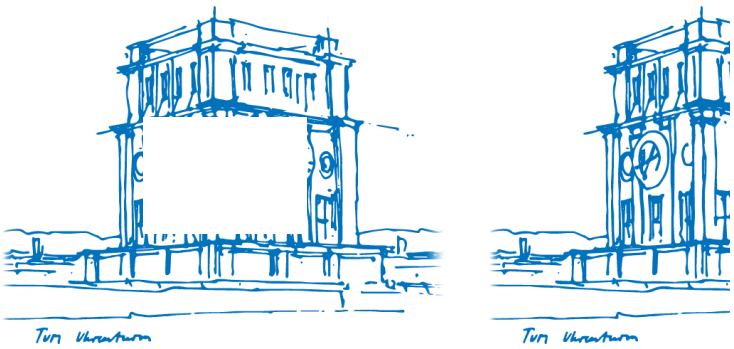
2.1. Simplified Explanations

Image Inpainting is a process where missing parts of an image are filled in to present a complete and realistic image. It is often performed as a reconstructing task where the goal is to create a realistic image that resembles the original image that does not contain any missing parts. In the image inpainting task, the missing regions of the image are completely surrounded by non-missing (and non-damaged) parts.

Image Outpainting is a process where the outside of an image is continuously filled to extend the image (into a realistic-looking image) with the image's existing visual elements taken into account. The difference between the inpainting and outpainting task is depicted in fig. 2.1.

Variational Inference is a technique to approximate complex distributions. The goal of variational inference is to approximate the complex probability distribution such as image data with a simpler, tractable distribution by optimizing parameters of the tractable distribution in a way that best approximates the complex distribution.

Expectation–Maximization (EM) Algorithm is an iterative approach that cycles between two modes to estimate the maximum likelihood of parameters in statistical models, where the model depends on unobserved latent variables. The goal is to find the model parameters such that the statistical model can capture/model some distribution. The first mode attempts to estimate the missing or latent variables and is called the estimation-step or E-step. The second mode attempts to optimize the parameters of the model to best explain the data and is called the maximization-step or M-step. It is an optimization algorithm that is often used when dealing with complex distributions



(a) Image Inpainting (b) Image Outpainting

Figure 2.1.: Illustration of Image Inpainting **(a)** and Image Outpainting **(b)** as a restoration task of the same original image

(non-differentiable functions).

Scene Graph G consist of non-empty set V called vertices (also called nodes) and set E of two-element subset of V called edges. The two-element pair that makes up an edge is used to indicate a connection (or a relationship) between two vertices. Graphs are used to represent a relationship between pairs of objects. The objects represent items of interest such as people, cities, or web pages, and we place an edge between a pair of nodes if they are somehow related (for example, placing an edge between two nodes that represent people that are connected on social media). We write $G = (V, E)$.

Semantic Layout is a 2D image representation often used to guide image generation. It consists of bounding boxes that indicate the shape and size of each object. The pixels inside these bounding boxes are feature vectors that describe the objects' visual and semantic information.

2.2. Self-Supervised Learning

Neural network models benefit from labeled data to the point of outperforming humans in numerous tasks such as emotion detection from facial images [22] or outperforming domain experts in image classification [23]. Their success relies heavily on the availability of large quantities of annotated data that is both time-consuming and expensive to acquire. However, people do not always learn by having access to labeled data. For instance, in language acquisition, a process that requires learning good representation, people benefit from context or auxiliary data to extract useful information.

Self-supervised learning is a machine learning paradigm for learning feature representation without having access to labeled data. Although there are several categories of self-supervised learning, we will focus on a generative approach in which the goal is to

have a model recover original information from a modified input.

Dating back to a quote made by John Rupert Firth in 1957 – You shall know a word by the company it keeps – self-supervised approaches have been used in multiple domains. As the use of the quote suggests, self-supervised approaches have made a significant contribution to the field of natural language processing [24, 25]. They have also been used in speech recognition [26] and computer vision tasks such image classification [27] and even in sub-tasks of autonomous driving [28].

In our context, a common supervised approach for image manipulation relies on having access to pre-modification and after-modification image pairs. However, having image pairs that would cover all the possible semantic image manipulation scenarios would require large amounts of data. Collecting these image pairs is not only extremely time-consuming but also expensive. Our work addresses these limitations and does not require image pairs for training. Instead, similarly to Baevski et al. [26], we perform input reconstruction by masking part of the input. This approach offers greater flexibility in terms of data acquisition and allows for training on larger datasets and therefore letting the model benefit from having to access more information. The specific approach is described in section 4.2.1.

2.3. Generative Models

We are surrounded by large quantities of digital data. This data includes for example text on social media, user behavioral history, and images on the internet. One of the main goals of modern artificial intelligence research is to develop models that are capable of analyzing and understanding such data. It can be argued that some of the main intuitive aspects of data understanding are the ability to differentiate data, generate them, and reason about them. They are not only capable of discriminating data across a set of categories but they are also able to generate new samples. Generative models are the best approaches for reaching the goal. This is further supported by OpenAI, who claim that generative models might be substantially better at understanding intrinsically the world [29]. In the following section, we will briefly discuss two approaches called Variational Autoencoders [5] (VAEs) and Normalizing Flows before diving deeper into Generative Adversarial Networks [6] (GANs) and Diffusion Models [7].

Normalizing Flows

Normalizing flows were popularized in 2015 in the context of variational inference by Rezende and Mohamed [30]. That said, the framework was introduced five years before by Tabak and Vanden-Eijnden [4] and in the same year already used for classification [31]. Normalizing flow is a generative process that attempts to model complex distribution (image data) by learning how to transform simple distribution (e.g. uniform, Gaussian) into a complex distribution. The transformation in Normalizing Flow is typically a

sequence of invertible functions. The process is built on the fact that if the transformation can be arbitrarily complex, then it is possible to generate complex distribution from a base distribution given reasonable assumptions about both distributions [32].

In essence, the training is done in the following way. We pick a family of transformations and then do a series of inverse transformations to map the training distribution into the base distribution such that the likelihood is maximized. In the process, the model learns the parameters of the selected transformation. During inference, we generate new data by sampling from the base distribution and performing a series of transformations with the learning parameters.

The biggest advantage of Normalizing Flows compared to VAEs and GANs is the ability to infer the value of the latent variables that correspond to a datapoint as well as the exact log-likelihood of the datapoint [33]. The tractability of the log-likelihood makes the Normalizing Flow conceptually attractive. By tractability, we mean that the distribution used in Normalizing flow to approximate the training data distribution is in a closed-form expression and the probability of this distribution can be calculated in polynomial time. Despite the attractiveness, Normalizing Flows are not as popular as their VAE, GAN, and diffusion-based model counterparts. That is because there still exists a performance gap between flow models and the state-of-the-art methods [34].

Variational Inference and Variational AutoEncoders

In generative processes, we are trying to model often complex intractable probability distribution. For example distribution of images. Consider the following non-trivial distribution depicted in fig. 2.2.

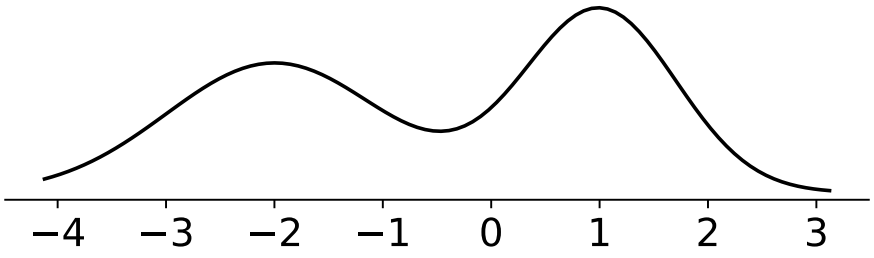


Figure 2.2.: An example of non-trivial distribution created as a mixture of Gaussian distributions

Modeling $p(x)$ of such distribution would be difficult, we can see that the distribution is a mixture of Gaussian distributions, and therefore it is much easier to model the conditional distribution $p(x|z)$ with the latent variable z. In this case, $p(x|z)$ is a simple Gaussian distribution. Therefore we can see that $p(x) = \int p(x,z)dz = \int p(x|z) \cdot p(z)dz$ and generating data from $p(x)$ is matter of two steps. First, sampling latent variable $\hat{z} \sim p(z)$ and subsequently using the sampled latent variable to sample from the conditional distribution $\hat{x} \sim p(x|\hat{z})$.

In **Variational Inference** techniques, the process is somewhat similar. We are given some complex (intractable) probability distribution p (for example distribution of images) and we will try to model the distribution by using a class of tractable distributions Q. Our goal is then to find a $q \in Q$ with their parameters that is most similar to p.

In Variational Inference, the distribution q is found as a lower bound approximation to the p distribution by observing that:

$$\log p(x) = \mathbb{E}_{z\sim q(z)}[\log p(x)] \tag{2.1}$$

$$= \mathbb{E}_{z\sim q(z)}[\log p(x,z) - \log q(z)] + \mathbb{KL}(q(z)||p(z|x)) \tag{2.2}$$

Since Kullback–Leibler divergence is non-negative we see that $L(q) = \mathbb{E}_{z\sim q(z)}[\log p(x,z) - \log q(z)]$ is a lower bound approximation to $\log p(x)$. Up until now, we've omitted that the distribution $p(x)$ depicted in the toy example in fig. 2.2 is not only represented by the conditional distribution $p(x|z)$ with the latent variable z but also by its parameters θ (for example mean and variance of a Gaussian). So in fact, we are finding the best lower bound approximation of $p_\theta(x)$ as a $\max_{q_\phi} \max_\theta L(q_\phi, \theta)$. This is done by using the iterative expectation–maximization (EM) algorithm.

In **Variational Autoencoders** the process is done by a framework consisting of Encoder and Decoder models. The Encoder learns the parameters of the latent distribution q_ϕ to sample latent variables z while the Decoder learns to approximate the p_θ distribution, i.e., to generate new sample $\hat{x}$. The flow of the process is depicted in fig. 2.3.

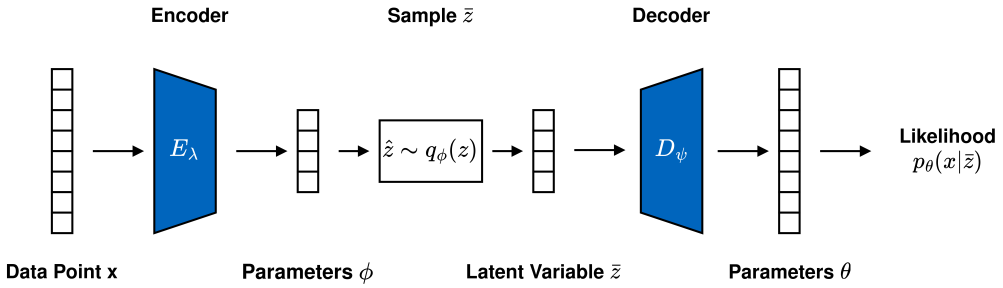


Figure 2.3.: Variational Autoencoder architecture

During inference, the process is as follows. If we want to draw samples from the whole distribution of images $p(x)$, we first sample latent variable $\hat{z} \sim q(z) \approx p(z) = \mathcal{N}(0,1)$ and then and then simply pass the latent variable to the Decoder, which generates a new sample. If we want to generate a sample from a specific class, we could for instance pass a sample x through the decoder to get the parameters ϕ and use them to generate a new sample from the specific class.

2.3.1. Generative Adversarial Networks

Introduced in 2014 by Goodfellow et al., Generative Adversarial Networks, or GAN for short is a framework for estimating generative models [6]. They are part of a

class of models called generative models that were explained earlier at the beginning of section 2.3.

GANs have transformed the computer vision research landscape through their ability to generate realistic images from scratch. GANs have also been among other things used to transfer style between images [35], generate images from text [36, 37], image in-painting [38, 39], image out-painting [15], image blending [40], and image editing [11, 17, 12]. The widespread adoption of GANs has been arguably achieved by establishing a higher quality and generality compared to VAEs and Normalizing flows [41].

At the core of the framework are two competing models called Generator (G) and Discriminator (D). The goal of model G is to capture the training data distribution, i.e., generate realistic images whereas the goal of model D is to discriminate whether an image came from the training data or whether it was generated from model G. The overview of the architecture is depicted in fig. 2.4.

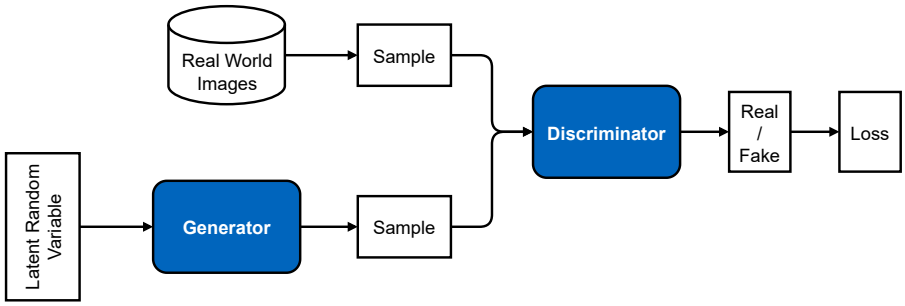


Figure 2.4.: Overview of the Generative Adversarial Network framework

In a sense, the competing models are playing a two-player minimax game with objective function V(G, D) defined as follows:

$$\min_G \max_D V(G,D) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[1 - \log D(G(z))] \tag{2.3}$$

Where x represents the training data distribution and z represents the latent random variable. Informally, the first part of the equation $\mathbb{E}_{x\sim p_{data}(x)}[\log D(x)]$ optimizes model D to recognize real images, while the second part $\mathbb{E}_{z\sim p_z(z)}[1 - \log D(G(z))]$ teaches model G to generate images that are indistinguishable from real images. Once we've defined the objective function, the models are jointly optimized in alternating gradient descent. In the current setting, the generator simply learns how to map from a latent variable space z to the training distribution, but there is no easy way to generate images from a specific class.

This is where Conditional Generative Adversarial Network (CGAN) comes into play [42]. Informally, given a target class, the goal of model G is to generate realistic images from the target class. In other words, the goal of the model G in the CGAN setup is to generate data $\hat{x} \sim G(z|y)$ such that $p(\hat{x}) = p(x|y)$. This is achieved by modifying the objective function depicted in eq. (2.3) as:

$$\min_G \max_D V(G,D) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x|y)] + \mathbb{E}_{z\sim p_z(z)}[1 - \log D(G(z|y))] \tag{2.4}$$

Deep Convolutional Generative Adversarial Networks - DCGANs

Year after the introduction of the GAN framework, Radford et al. introduced deep convolutional architecture to GAN's [8]. In the work, the authors improve previous state-of-the-art by modifying the architecture of models D and G in several ways. First, they up-sample the latent vector representation in the model G by using transposed (or fractionally-strided) convolutions [43]. In the standard convolution setting, we place a kernel (matrix of size $S \times S$) on the upper leftmost $S \times S$ region of the input and perform element-wise multiplication between the two matrices, and subsequently sum the result. Afterward, we move the kernel by one (this is called stride 1) to the right and repeat the process until we have reached the end of the row. Then we place the kernel one row below (the leftmost region) and continue the process until we reach the end of the input. Transposed convolutions can be thought of as a standard convolutions but with a modified input feature map. One example of such modification can be enlarging the input by adding zero-valued rows and columns to each of the four sides of the input which is depicted in fig. 2.5.

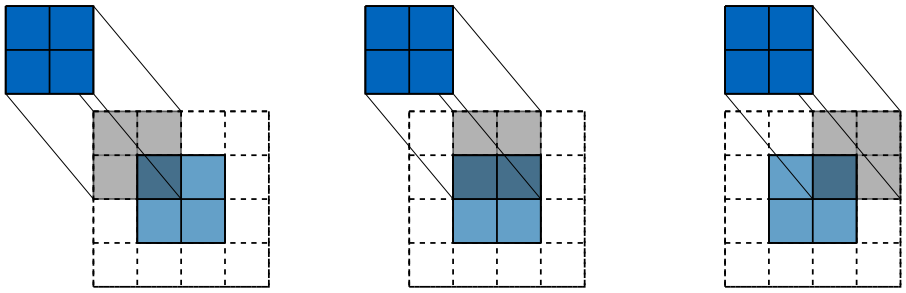


Figure 2.5.: Transposed convolution with 1 stride, kernel size 2×2 in blue, input image of size 2×2 in light blue surrounded by zeros depicted as white squares

Additionally, the authors down-sample a generator image with model D using strided convolutions. In the strided convolution, the stride i.e., the size of the step kernel takes after performing each convolution is larger than 1 which down-scales the image. Finally, the authors removed fully connected hidden layers and used ReLu activation in model G and LeakyReLu activation in model D. Nowadays, this deep and convolutional GAN setting is currently used in most state-of-the-art approaches.

2.3.2. Diffusion Models

Informally, diffusion describes dynamic movement driven by the thermal motion of molecules of a substance from an area of higher concentration to an area with lower

concentration. For example, a spray of perfume or room freshener that gets diffused into the air. Diffusion models have been inspired by thermodynamics [7]. Although introduced in 2015, diffusion models have started to rapidly gain popularity mainly from 2021 [44]. It can be said that these generative models raised the bar to a new level in the area of generative modeling. Especially in regard to the detail of generated images.

The diffusion framework consists of three different formulations of diffusion models: denoising diffusion probabilistic models (DDPMs), noise conditioned score networks (NCSNs), and stochastic differential equations (SDEs) [44]. In explaining the general framework, we will focus on DPPM formulation popularized by the work of Ho et al. [45]. The underlying principle of diffusion models is a Markov chain of diffusion steps that slowly adds random noise to data and then learns to reverse the diffusion process to construct desired data samples from the noise. In other words, they learn to reverse a process that gradually degrades the training data structure by adding noise at different scales. In the forward diffusion process we slowly in n consecutive steps add noise to the training data until it becomes a normal $\mathcal{N}(0, \mathbb{I})$ distribution. On the hand in the backward process, we reverse train a model that learns to revert the process and in n consecutive steps approximately model the training distribution given the $\mathcal{N}(0, \mathbb{I})$ distribution.

In the **forward** diffusion process, we first draw data from a training distribution $x_0 \sim q(x)$ and then we add Gaussian noise to the sample in T steps described in eq. (2.5):

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t \cdot \mathbb{I}) \quad q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1}) \tag{2.5}$$

Where $\beta \in (0, 1)$ and typically $0 < \beta_0 < \beta_1 < \beta_2 < \cdots < \beta_T < 1$. In practice, we set $T \approx 1000$ [45]. The advantage of this definition is that it allows us to get x_t at any time step directly in a closed form using the reparametrization trick as written in eq. (2.6):

$$x_t = \sqrt{\overline{\alpha_t}} \cdot x_0 + (1 - \overline{\alpha_t})\epsilon, \quad \epsilon \sim \mathcal{N}(0, 1) \tag{2.6}$$

Where $\alpha_t = 1 - \beta_t$ and $\overline{\alpha_t} = \prod_{s=0}^{t} \alpha_s$.
In the **backward** process, we want to know the true distribution $q(x_{t-1}|x_t)$, however, this is intractable. Instead, we employ neural network p_θ to learn to approximate these conditional probabilities. If β_t is small enough, the distribution $q(x_{t-1}|x_t)$ also corresponds to a Gaussian [46], so the goal of the neural network is to only learn the mean and the variance parameters of the Gaussian. Similarly to the GANs, diffusion models can also be extended to a conditional setting. For example to generate a color image conditioned on its grayscale equivalent [41].

Diffusion Models have achieved state-of-the-art results in image generation, with the most famous models being Google's Imagen [47] and OpenAI's DALLE [48] and DALLE-2 [18] which will be explained later in the section 3.5.1. They are also challenging

GAN-based approaches in image inpainting [49, 50, 18] and outpainting tasks [18]. In addition to generative tasks, the latent variable which has the same dimensionality as the original data has been found to be useful in discriminative tasks such as image generation, classification, and anomaly detection [44].

2.4. Graph Convolutional Networks

Graph Convolutional Network or GCN for short is a class of neural network that focuses on data that can be represented in a graph. GCNs were introduced in 2016 by Kipf and Welling [51] based on the spectral graph CNN work by Bruna et al. [52] (2014) which was later extended in 2016 by Defferrard et al. [53].

GCN models have been used to analyze graph structures such as citation networks [54] (where nodes typically represent publication and edges are directed from one document toward another that it cites). They have been used in natural language processing to classify document labels or to analyze the internal graph structure of a text such as a syntactic dependency tree [55]. In computer vision, GCN models have been mainly used in point cloud classification as well as to understand semantic relations between objects in a visual scene through scene graph generation [56]. Scene graphs will be further discussed in section 2.5.

Graph Neural Networks are based on an underlying concept of **differentiable message passing** [57]. In the message-passing framework, each node is represented by a hidden state - a vector of numbers (feature vector). In each step, each node aggregates hidden states from all of its neighboring nodes. The aggregation can be thought of as a form of message passing as the node receives messages (hidden states) from its neighbors. The aggregation provides a node with contextual information about its neighborhood. It must be a permutation invariant function such as a sum. Precisely the aggregation step at time k for node v can be seen in eq. (2.7):

$$m_v^{(k)} = \sum_{u \in N(v)} M(h_v^{(k-1)}, h_u^{(k-1)}, e_{vu}) \tag{2.7}$$

where M is an arbitrary learnable function. In the GCN setting, function M oftentimes takes the form of matrix multiplication between the network weights and the hidden states. For instance $m_v^{(k)} = \sum_{u \in N(v)} W_{nodes}^{(k)} \cdot h_u^{(k-1)}$ is a valid instantiation of the framework. Afterwards, the aggregated message m_v is used to update the hidden state of the node v as depicted in eq. (2.8)

$$h_v^{(k)} = U(h_v^{(k-1)}, m_v^{(k)}) \tag{2.8}$$

where function U can be for example instantiated as the $ReLu$ activation function $h_v^{(k)} = ReLu(Q^{(k)} \cdot h_v^{(k-1)} + m_v^{(k)})$. The instantiation and extension of the framework vary significantly. Sometimes the edges also have a hidden representation which is updated by aggregating the hidden representations of the nodes it connects.

The message-passing framework on a graph is analogous to the convolution operation used in convolution neural networks on images. That is if you think of each pixel as a node that's connected to the 8 adjacent neighboring pixels the difference being that the pixels in the image are ordered as opposed to the nodes in the graph. The generalization of the graph convolution from the 2D image convolution is illustrated in fig. 2.6.

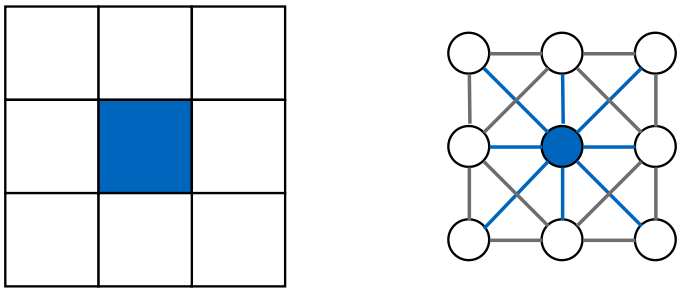


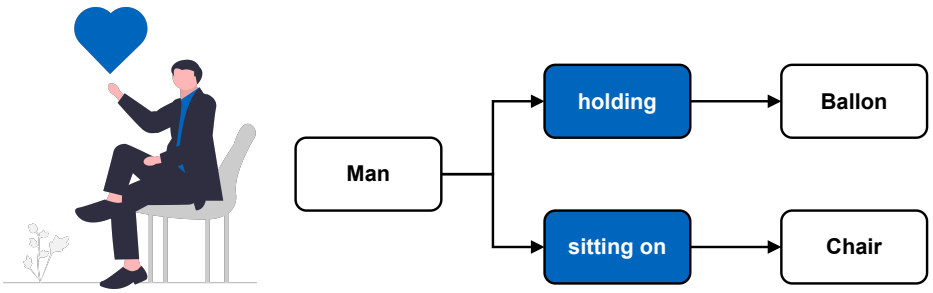
Figure 2.6.: A 3×3 image (left) can be considered as a special case of a graph (right) where pixels are connected by adjacent pixels. Similar to 2D convolution, one may perform graph convolutions by taking the weighted average of a node's neighborhood information

In the Graph Neural Network framework, each layer passes information to a node from all of its neighbors. In other words, the hidden representation of each node is recursively defined in terms of the hidden representation of its neighbors. This means that after k GCN layers, the representation $h_v^{(k)}$ of node v is based on information from all nodes in its k-hop neighborhood. As the number of layers increases, the influence of each node becomes proportional to the stationary distribution of the graph as a Markov chain which leads to a potential problem where a set of nodes influence every other node the same way regardless of their distance. This causes an over-smoothing issue [58] (indistinguishable representations of nodes in different classes). Another problem that comes from having too many layers is the bottleneck problem [59]. When we are aggregating messages across a long path, there might be single a node that tries to capture the exponentially growing information into fixed-size vectors. As a result, GNNs can fail to propagate messages originating from distant nodes and perform poorly on long-range interaction dependant tasks. Although there have been advancements [60] in attempting to tackle these issues, it is important to note these common plights of graph learning architectures.

2.5. Scene Graphs

Scene graphs are a structural representation of a scene typically in an image, video, or point cloud data via graphs. A node in the graph usually represents a person, a place, a thing, or a part of an object (for example arm of a person). Each node can also have attributes that describe the state of the objects such as their shape, pose, color, and so

on. These nodes along with their corresponding attributes are encoded using a feature vector encoding. The nodes in scene graphs are connected via edges which signify a relationship or connection between the node objects. An example of such connection can be an action (girl *swinging* a racket) or a position (person *on top of* a mountain). An example scene graph with a corresponding image is depicted in fig. 2.7.



(a) An image (b) Corresponding scene graph

Figure 2.7.: An image **(a)** with corresponding scene graph **(b)** where edges are depicted in blue and nodes are depicted in white. The images were adopted from[1]

In 2015, Johnson et al. introduces scene graphs to improve image-based retrieval from a text search query [20]. They argue that scene graphs provide semantically rich representation that would is oftentimes equivalent to paragraphs-long text which is too cumbersome to work with. In addition to being used for image retrieval [61], scene graphs have been used to reason about 3D point cloud scenes [62], video reasoning [63], video action recognition, image captioning and other relevant computer vision tasks [21]. In the context of image generation [64, 65] and manipulation [12, 15, 19], they place especially important role they provide attribute binding to object instances and a sense of direction and position. Therefore they can deal with complex scenes containing multiple objects and desired layouts [21].

To be able to perform complex operations such as image generation from a scene graph it is important to first describe the scene graph construction process. In short, they are two lines of approaches. The first approach is a sequential process that has many parts. First, the scene objects are detected (let's assume there are n objects), then these objects are described using visual features. Afterward, these visual features are used to describe the $n \times n$ possible relationships. The visual features are subsequently used for the classification of the n objects as well as the classification of the $n \times n$ edge relationships. The other approach involves jointly detecting and recognizing the objects and their relationships. Once the detection has been made, the scene graph generation model is then typically trained by using cross-entropy classification loss on the n object predictions and $n \times n$ relationship predictions [21].

[1]Image available at https://undraw.co/illustrations

3. Related Work

This chapter provides a survey of different state-of-the-art methods related to image manipulation. First, in section 3.1 we discuss methods that perform image manipulation with GANs without intermediate representation. This is followed in section 3.2 by works related to a scene graph representation, which includes an overview of methods for scene graph generation and their subsequent use for image generation and manipulation. In section 3.3 we introduce works that are related to processing the intermediate scene graph representation through graph convolutional networks. We then discuss the subfield of image manipulation called image inpainting in section 3.4, which is finally followed by a survey of the state-of-the-art diffusion methods in section 3.5.

3.1. Image Manipulation using GANs

Image manipulation involves the transformation or alteration of a photograph to get the desired output according to the user's specifications. Among state-of-the-art image manipulation works, there are two main approaches. One line of work focuses on extracting intermediate scene graph information from an image so that the user can subsequently modify the graph which is finally used to produce the modified image [12, 15, 19]. The other line of work generates novel images according to the modification specification that the user performs directly on the image [16, 17, 18]. The division of approaches according to the intermediate representation can also be extended to whether an image manipulation system also generates a semantic layout before generating the final modified image. Hong et al. [66] show the benefits of generating the intermediate layout representation in an ablation study and Johnson et al. [64] use it to generate semantically meaningful images.

The majority of state-of-the-art image manipulation approaches use either GAN or diffusion models to perform image generation. In the current section, we will first go through some of the first GAN approaches for image manipulation. Namely, we will discuss the contributions of Hong et al. [11] who manipulate images based on semantic image maps and the contributions of the SESAME method [17] that only requires only semantic information from the manipulated region. Then, we will explore methods that condition the image generation on the intermediate layout representation [12, 15]. Afterward, we will go through Scene Graph state-of-the-art image manipulation approaches [20, 64, 65, 21]. We will also explore advancements in graph convolutional neural networks for processing the scene graph representation [67, 68, 19]. Then we

will branch out and explore state-of-the-art approaches in related sub-fields that could be relevant for image manipulation such as image inpainting [38, 14, 69] before finally addressing diffusion model approaches [48, 70, 50].

Learning Hierarchical Semantic Image Manipulation through Structured Representations

Many state-of-the-art image manipulation methods are guided by learned binary mask representation of an object to be manipulated. The work of Hong et al. [11] is an example of such a method. They train a model, called an Image generator, by masking a region of interest they want to reconstruct and feed the masked image into the model that then re-creates the original image [11]. In addition to the masked input RGB image, the model is fed a corresponding learned semantic image. This approach is inspired by Wang and Gupta who note that images consist of two separate characteristics: **structure**, which encodes the underlying geometric representation of the image scene, and **style**, which encodes the texture and illumination of the objects [16].

In addition to training the Image generator, Hong et al. also train a semantic image generator called the Structure generator. The Structure generator is trained with the same approach as the Image generator. The Structure generator is a two-output stream convolutional neural network that produces the binary mask of the foreground object in one stream and a semantic label map in the other stream. The architecture can be seen in fig. 3.1. The foreground object binary mask is used to define a precise pixel-wise boundary of the object and the final semantic labels are generated in accordance with the binary mask. The approach of consecutively first generating a binary mask that is used to guide semantic/RGB image generator can be seen in many state-of-the-art methods and will be discussed further in later chapters.

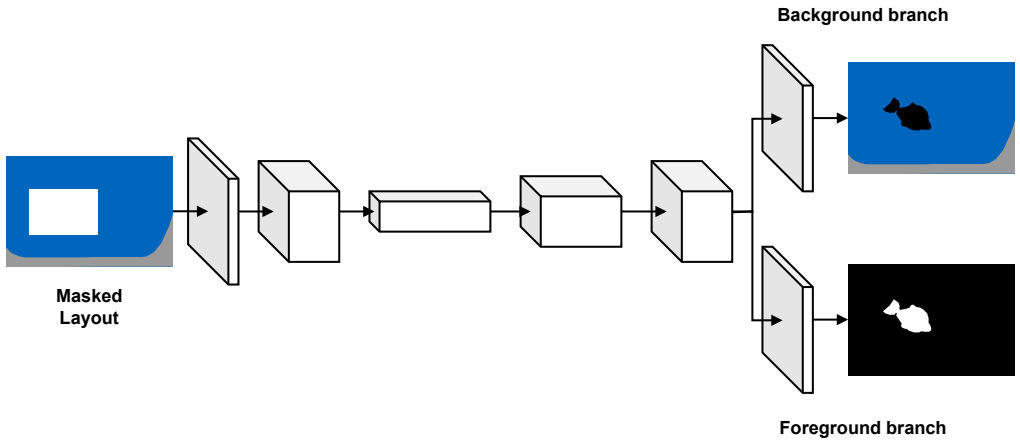


Figure 3.1.: The Structure generator architecture proposed by Hong et al. [11]

In inference time, both the semantic image as well as the RGB image of the image to be modified are required. This makes the approach impractical for use in many real-world scenarios as obtaining accurate semantic labels is a costly and time-consuming process.

SESAME: Semantic Editing of Scenes by Adding, Manipulating or Erasing Objects

SESAME is a method that enables users to perform free-form semantic editing of images [17]. Inspired by layout-driven editing used by Hong et al., SESAME performs image reconstruction (inpainting) of a masked bounding-box region. However, compared to the method by Hong et al., it does not require semantic information of the whole image, but only semantic information of the masked region. They improve image manipulation the-state-of-art by incorporating many improvements from the computer vision field. Such as using dilated convolutions [71], SPADE layers [72], multi-scale discriminator [73] or perceptual loss function [10].

Incorporating the most recent advancement from computer vision into image manipulation systems remains a rich research endeavor. In the following text, we will highlight some ideas from related literature that could be potentially used or seen as an inspiration for creating a state-of-the-art image manipulation system.

3.2. Scene Graph Models

Scene graphs provide a semantically rich representation of images, describing an image scene containing objects and relationships between them. They play an important role in image generation and particularly in image manipulation, encoding spatial relationship information between objects and providing attribute binding for objects. In this section, we will give a literature review related to scene graphs. In section 3.2.1 and section 3.2.2 we provide an overview of state-of-the-art scene graph generation and scene graph image generation methods respectively. This is followed by section 3.2.3 where we describe the most significant image manipulation works that take advantage of the benefits provided by having a scene graph as an intermediate representation of images.

3.2.1. Scene Graph Generation

Scene graph generation works either do produce a scene graph in a multi-step fashion by for example first detecting the objects in the scene and then they reason about their relationships or generate the scene graph directly [21]. Dai et al. first detect objects, then extract visual features and spatial information such as the position and size of these objects to perform classification of the relationship between the objects to generate a scene graph. This is extended by the introduction of an edge-relationship embedding module in LinkNet jointly learns the connections between all related objects [74]. LinkNet also introduces a global context encoding module and a geometrical layout encoding module, which extracts global context information and spatial information (such as relative position and scale) between object proposals from the entire image.

Similarly, Xu et al. first use a regional proposal network to detect objects and then reason about their relationship by using graph convolutional network [75]. On the other hand, Liu et al. propose a fully convolutional scene graph generation framework that detects objects and recognize the relationships between them simultaneously [76].

As opposed to the two main lines of work for scene graph generation, Garg et al. develop a deep auto-regressive model that takes a seed object as input and generates a scene graph in a sequence of steps, each step generating an object node with corresponding edges to the previously generated nodes[65]. They convert the graph representation into a sequence representation and learn the probability distribution over sequences. Learning to model the distribution not only allows for new scene graph sample generation but also enables the detection of out-of-distribution anomaly samples.

Herzig et al. expand the original scene graph representation by in addition to having edges corresponding to the relationship between two nodes, append the graph with edges that correspond to the inverse relationship [77]. This canonical representation of scene graphs has stronger invariance properties. This leads to improved robustness to graph size and noise in comparison to existing methods. .

3.2.2. Image Generation Using Scene Graphs

Until 2018, state-of-the-art methods focused on generating images from text descriptions. Describing scenes with many objects in a way that captures all the relational and positional connections between objects requires long paragraphs of text and using these texts by models is rather challenging. In 2018, Johnson et al. developed seminal work on how to generate images using scene graph representation called Sg2Im [64]. Sg2Im is an end-to-end method for generating images from scene graphs. In short, Sg2Im takes scene graph representation as an input and processes it with a graph convolutional network to generate a scene layout that is finally used to generate an image. The image is generated from the layout representation using Cascade Refinement Network [78] (CRN). The CRN is responsible for generating images that respect the object's position in the scene layout. It consists of a series of convolutional refinement modules with an increasing spatial resolution that ensures that image generation is happening in a coarse-to-fine manner. The Sg2Im [64] model is trained adversarially using a generator and discriminator framework. Specifically, it uses two discriminators – one that focuses on image patches containing individual objects and the other that focuses on the image as a whole. The authors performed an ablation study that showed that each of the intermediate steps is beneficial for image generation.

Although Sg2Im [64] is crucial work in the image generation field, it requires a complete semantic segmentation map for training the model. This was improved by the Layout2Im [79] work that uses coarse bounding-box layout representation. Layout2Im is also an end-to-end method that disentangles each object representation into a category

part and an appearance part. Another limitation of Sg2Im is the fact it cannot deal with an incrementally additive scene description provided by the user. Mittal et al. extend the Sg2Im architecture [80]. They implement an approach for dealing with incremental scene graph expansion that ensures that the image generated in the current step preserves the visual context from the previous step. Namely, they use a recurrent architecture that passes the RGB image channels generated in the previous step to the CRN [78] generator network in a current time step. This can be viewed as a form of progressive image generation which will be described in section 3.4.1. Another way Sg2Im [64] was improved was by not only using the scene layout for the generation by CRN [78] but also a context from the scene graph itself [81]. Specifically, Tripathi et al. pooled scene graph feature representation and processed it by a fully connected layer before feeding it alongside the scene layout to the generator. The authors argue, that the context not only encourages the images to respect the scene graph relationships but also prevents mode collapse during the GAN training.

A different approach to improve state-of-the-art was introduced by pasteGAN [82]. PasteGAN is a semi-parametric method that generates images by generating individual image patches containing objects before using an Image Fuser module that uses an attention mechanism to combine these patches. Specifically, the Image Fuser uses a per-pixel attention mechanism that captures the extent to which the fuser attends the individual image patch at every pixel. In contrast to all the methods mentioned before, Farshad et al. focus on meta-learning few-shot image generation [83]. The goal of the approach is to come up with a model trained on a variety of images that quickly adapts to a specific task with just a few training samples. They do this by forming subsets of the training set based on certain criteria and subsequently tackle the problem of how well will each model perform on image generation tasks for a certain sub-set.

Image generation is an active field of research and scene graph representation provides control over the process that methods that do not use such intermediate representation lack. Although many limitations have been addressed by the presented methods, generating high-resolution high-fidelity images remains an open-ended challenge.

3.2.3. Image Manipulation Using Scene Graphs

Image manipulation is a sub-field of image generation that transforms images according to user specifications. Many of the techniques introduced in image generation are then also used for image manipulation. In this subsection, we will discuss state-of-the-art works in image manipulation that leverage scene graph representation. Namely, we focus on seminal scene graph image manipulation work by Dhamo et al. [12] that offers a complete solution to the problem in low dimensional setting trained by performing image reconstruction and work by Yang et al. [15] that employs the image reconstruction principle along with a self-attention mechanism to the image outpainting task.

Semantic Image Manipulation Using Scene Graphs

This thesis builds upon the work of Dhamo et al. called Semantic Image Manipulation Using Scene Graphs [12], or SIMSG for short. The SIMSG was the first work that used an intermediate scene graph image representation before performing semantic image manipulation.

The SIMSG system architecture is partly inspired by Sg2Im [64] and consists of several modules. During inference, first, the scene graph representing the query image is generated using Factorizable Net [84]. A user can then subsequently modify the scene graph to achieve any of the available semantic operations: object addition, object removal, repositioning of an object, or a change of relationship between the objects. On top of the node embedding representation obtained by the F-Net, each node embedding is appended by a feature vector obtained by feeding the corresponding image object region into a pre-trained VGG network [85]. The scene graph representation is then fed into a graph convolutional network-based solution called Spatio-semantic Scene Graph Network (SGN). Alongside improved feature embeddings, SGN predicts bounding boxes for the desired position of the objects. This is then used to generate a scene layout. Subsequently, the scene layout is fed into a SPADE [72] decoder that generates the desired output image. The authors trained the pipeline by image reconstruction described earlier in section 4.2.1. The image reconstruction setting is convenient, as it does not require image pairs of "before" and "after" manipulation ground-truth images.

Scene Graph Expansion for Semantics-Guided Image Outpainting

Inspired by the recent employment of transformers for scene graph generation [86], Yang et al. incorporate an attention mechanism for an image outpanting task using scene graph [15]. Similarly to SIMSG [12], they train the model by doing restoration (masking the outer parts of an image and then restoring the masked part). Likewise, they extract scene graph representation from an image, process it, then generate a layout that is used to generate the final image. However, their main contribution lies in processing the scene graph representation. They employ a transformer self-attention mechanism to process the scene graph. Specifically, they apply self-attention across nodes under the guidance of corresponding edges and attention across edges conditioned on the sharing node. In addition to scene graph processing, Yang et al. were inspired by recent advancements in scene graph representation [77]. In addition to having edge features representation relationship between nodes, they append the graph with edges corresponding to the inverse relationship. Adding both of these improvements allow them to achieve state-of-the-art results in layout generation as well as in image outpainting task.

3.3. Graph Convolutional Networks models

Images often depict a scene with multiple objects and corresponding relationships between these objects. Such images can be represented as a scene graph described in sec-

tion 2.5. Graph Convolutional Networks (GCN) enable extracting feature representation of scene graphs for subsequent operations such as image manipulation. In this section, we will describe advancements in processing scene graphs that are useful for subsequent semantic image manipulation.

3.3.1. Disentangled Neural Networks

The structure of real-world graphs is typically based on the interaction of many latent factors. As a result, within a real-world graph structure, there could be several nodes with similar feature representations belonging. In 2019, Ma et al. noticed that state-of-the-art graph convolutional networks fail to take into account this group entanglement while doing message updates [67]. They proposed a novel neighborhood routing mechanism that identifies and takes into account the latent factor entanglement.

Specifically, the disentangled convolutional neural network [67] (DisenGCN) updates each node u representation x_u such that the new representation is composed of the concatenation of K latent factor components ($x_{u\_new} = [c_1, c_2, \ldots, c_K]$). Each of these K components represents a neighborhood of nodes connected by the latent factor. For each node u, DisenGCN divides all the neighboring nodes of u into K clusters and uses the center of each cluster as the cluster representation c. With the introduction of the routing mechanism, DisenGCN achieved state-of-the-art results in semi-supervised classification on numerous citation datasets. Performance of which was improved one year later with the introduction of FactorGCN [68] by Yang et al., which decomposes the graph into several interpretable factor graphs.

In 2022, Farshad et al. incorporate the disentanglement approach into a GNN that learns scene graph representation for image manipulation tasks [19]. The authors present DisPositioNet [19] that extends the SIMSG [12] work described in section 3.2.3. The DisPositioNet disentangles the image object nodes into several latent factors that are subsequently used by two separate encoders to encode each object's pose and appearance. In contrast to the previous disentangled graph neural network works, DisPositioNet also considers the edge features in the disentangled feature extraction process. As a result, DisPositioNet is able to outperform SIMSG in an image reconstruction task on the COCO [87] dataset and in an image manipulation task on Visual Genome [88] (VG) dataset.

3.4. Image Inpainting

In the image manipulation task, one of the possibilities is to train the manipulation model through the image restoration task. Image inpainting is a restoration task in which we are trying to restore a continuous corrupted region inside the image. The technique can also be used for removing objects. In computer vision, the term has been used since 2000 by the analogy of the process used in art restoration [89]. Elharrouss et al.

categorize image inpainting approaches into three different categories: sequential-based approaches, CNN-based approaches, and GAN-based approaches [90].

One of the sequential-based approaches is the patch-based approach. Patch-based methods fill the missing region patch-by-patch by searching for suitable replacement patches in the undamaged part of the image and copying them to corresponding locations. The sequential-based methods succeed in some parts of image inpainting like filling texture details with promising results, however, the problem of capturing the global structure still poses an unresolved challenge. To tackle this issue, many CNN-based approaches have been proposed. Most of these approaches use a decoder-encoder approach with an adopted U-Net [91] style architecture to obtain fine details as well as to capture global structure.

GAN-based approaches are another popular line of work that can capture global structure. Pathak et al. show a significant improvement for image inpainting when combining a standard pixel-wise loss with an adversarial loss [13]. An example of GAN-based approaches would be a work by Dhamo et al. [14]. The authors tackle the issue of inpainting color and depth of an area occluded by an object. In other words, the goal is to obtain Layered Depth Image representation [92]. The authors train a CNN-based network to jointly predict a conventional depth map and a foreground mask. Then they use the mask so that only the visible background regions of the RGB-D image are fed into a GAN with architecture based on the work of Isola et al. [93].

3.4.1. Progressive Image Generation Models

Recently, we've seen a surge of state-of-the-art image inpainting systems based on progressive image generation, where they restore the corrupted region in multiple steps. Zhang et al. progressively restore the image by first generating the outer border parts before restoring the center of the missing region [69]. They argue that generating the outer part (the border sub-region) is easier to generate because it borders image parts that provide informational context for image generation. They achieve this by having four progressively smaller predefined handcrafted masks that are used to mask the uncorrupted image. The model after being fed an image with the largest mask attempts to generate an image corresponding to the image with the second largest mask, and so on. The process is depicted in fig. 3.2.

While it is intuitive to generate the outer regions first, the handcrafted approach of Zhang et al. cannot deal with irregular masks. In addition, the distance from the middle of the missing region cannot encapsulate all the information describing the difficulty of generating a particular pixel. To solve these issues, Guo et al. propose a full-resolution residual network [94] (FRRN) that learns to progressively update each pixel of the mask individually. This approach not enables higher flexibility but also outperforms the work of Zhang et al. in image inpainting restoration tasks.

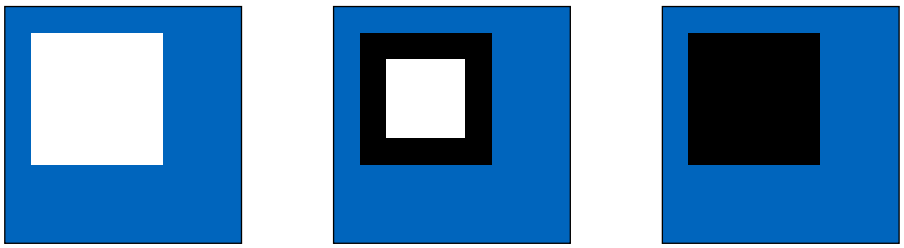


Figure 3.2.: Progressive image inpainting restoration process (from left to right). Blue regions represent ground-truth image pixels, white regions represent masked pixels, and black regions represent generated pixels. The leftmost image is fed into a model that generates the middle image, which is then fed into the model that generates the final restored image on the right.

3.5. Diffusion Models

Diffusion-based approaches raised the bar for image generation generative models by their ability to produce high-quality images with a very high level of detail. In this section, we focus on a selection of diffusion-based seminal works. Namely, we introduce the first two versions of DALLE [48, 18] with its underlying technique called CLIP [70] that has alongside GLIDE [50] that has significantly improved diffusion landscape in terms of the quality of the generated images.

3.5.1. DALLE 2

The work introduced by OpenAI called DALLE [48] with its successor DALLE-2 [18] has been getting significant attention for its ability to generate images and to perform image manipulation with an exceptional level of detail. The diffusion-based system allows users to not only generate novel images from a text but to perform image inpainting as well as image outpainting. The system is built on two technologies: diffusion and CLIP [70].

CLIP learns task-agnostic multi-modal embedding that can be used for several visual and natural language processing tasks. It builds on a critical insight to leverage natural language as a flexible prediction space to enable generalization and transfer. The main source of inspiration is the work of Li et al. [95] who used natural language supervision to allow zero-shot transfer to perform classification on datasets such as the ImageNet [96]. They achieved this by training CNN to assign a likelihood $p(w|I)$ to each possible phrase (n-gram) w given an image I. In other words, they taught the model to predict a wide set of visual concepts given text and then used the model to perform a different task (classification) in a zero-shot fashion.

Like Li et al., CLIP [70] uses text-to-image pairs on the internet to create very general embedding. Specifically, they used more than 400 million text-to-image pairs collected

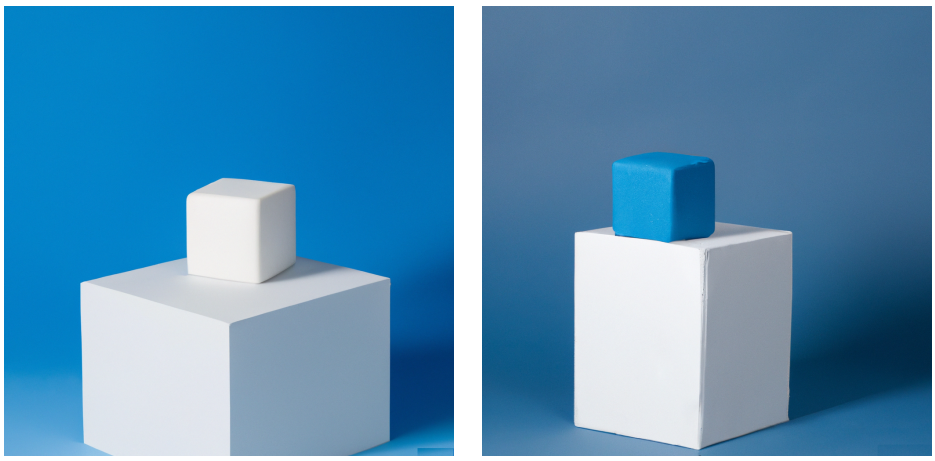
from the internet to learn state-of-the-art image representation. They trained the system to predict which out of a set of 32768 randomly sampled text snippets a query image belongs to. At its core, the system consists of encoders. One encodes the query image and the other encodes a text embedding. The system is then trained to generate similar embeddings for matching pairs and dissimilar embeddings (for example in terms of cosine similarity) for non-matching pairs. Intuitively the system learns visual concepts and associates them with their corresponding names. In addition to being robust and usable for a number of different visual tasks, CLIP matches the performance of the original ResNet [97] on ImageNet without using any of the original labeled samples.

DALLE-2 [18] uses a two-stage process to generate images. Given a text caption of an image, DALLE-2 first uses a model that generates the corresponding image CLIP [70] embedding of the text. Then, subsequently, it uses a diffusion model that restores the original image. Specifically, the diffusion model receives the CLIP embedding as well as a corrupted version (using the diffusion process of Gaussian noise addition) of the image during each training step. According to the authors, there are several benefits of the two-stage process. The CLIP encoder that generates the image encoding from a text, learns high-level features of the image. These CLIP embeddings can be used to perform a multitude of high-level operations in the high-level embedding concept vector space introduced by word2vec authors Mikolov et al. [24]. The word2vec authors show an example where the resulting vector embedding of arithmetic operation "king" − "man" + "woman" is similar to the vector embedding of "queen". Since CLIP captures embeddings for both text and images, it enables to conduct concept space operations using both image and text embeddings. To for example perform a gradual transformation of an image style.

There are however downsides to using CLIP [70] embedding. First, CLIP embedding does not preserve information about which attributes correspond to which image objects (attribute binding). Second, CLIP embedding does not preserve the relative position information of the objects. Images generated using DALLE-2 showcasing these limitations can be seen in fig. 3.3. Both of these issues are not present when using a scene graph representation of an image.

GLIDE

In 2021, Dhariwal and Nichol showed that having a classifier trained on diffusion-corrupted data and subsequently taking the gradient of the classifier with respect to its input to guide the diffusion can help produce high-fidelity images [98]. Later that year it has been shown that the diffusion model can leverage its knowledge to guide a so-called classifier-free diffusion generation [99]. As a result, Nichol et al. introduced GLIDE [50] which leverages classifier-free guidance to generate high-fidelity images. The scaled-down model of GLIDE which uses fewer parameters than the first version of DALLE [48] and does not need the CLIP [70] embeddings outperforms the first version of DALLE in image generation (judged by human evaluators as well as quantitatively



(a) (b)

Figure 3.3.: Result of DALLE-2 image generation given a query "a white cube on top of a blue cube". Showcasing object attribute-binding issues **(a)** and inability to reason about position **(b)**

by FID [100] score). Despite this, GLIDE, which could just as well be called DALLE 1.5, suffers from all the limitations of the DALLE systems.

4. Method

In this chapter, we will give a detailed description of the system. The architecture of our system, which is depicted in fig. 4.1, is based on the SIMSG [12] architecture for which we've provided an overview in section 3.2.3. Here, we will detail the system architecture, describe each component of the system, and subsequently discuss all additions and improvements of the system over SIMSG.

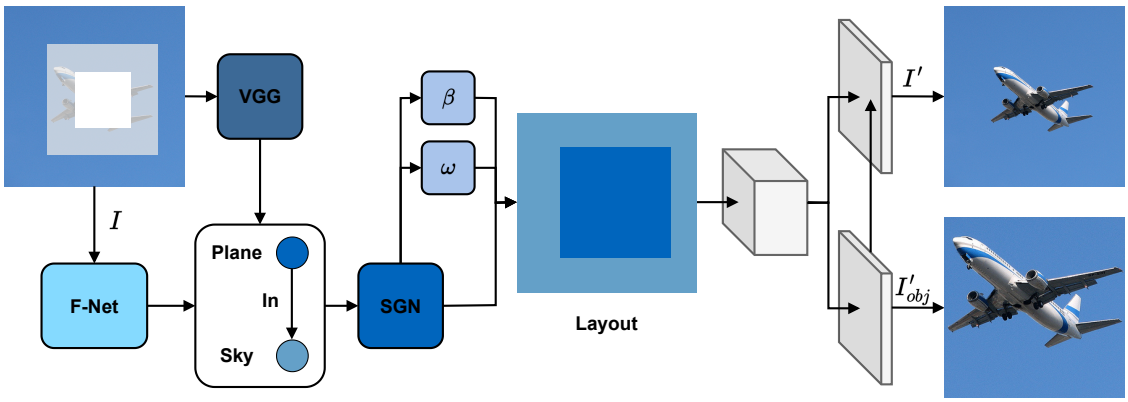


Figure 4.1.: **An overview of PRISM.** We propose a progressive multi-task model for semantic image manipulation. During training, the system takes masked input (on the left), uses a multi-step decoder to create a layout representation, and then through a series of SPADE [72] ResNet blocks progressively fills in the border regions of masked input by replacing the mask with generated image pixels. In this way, the model progressively reconstructs the entire original image (top-right) and detailed reconstruction of the masked object (bottom-right).

4.1. Architecture Flow

We are given a dataset $\mathbb{D} = (I, \mathcal{G})$ of input images $I \in \mathbb{R}^{W \times H \times c}$, and their corresponding scene graphs $\mathcal{G} = (V, E)$ where V and E are the vertices and edges in the scene graphs. The network is denoted by θ, where it has two outputs $I', I'_{obj} = \theta(I, \mathcal{G})$. The scene graph $\mathcal{G}$ is predicted using a pre-trained scene graph generation model. Each scene graph $\mathcal{G}_i$ consists of a set of triplets (o_{ij}, p_{ij}, s_{ij}) where o, p, s are the object, predicate and subject

and $j \in J$, where J is the total number of triplets in the given scene graph.

In the training process, the model is trained through a reconstruction process where the input image I and scene graph nodes are randomly masked and the masked area is reconstructed by the model from the extracted scene graph features. We denote masked image I_M. The reconstructed object and image are denoted by I', I'_{obj} respectively. At the inference time, the modification type is defined based on the masking performed on the input image and graph.

Image to Scene Graph

In the first part of the training/inference pipeline, we first preprocess the images and then feed them into a Factorizable Net [84] that extracts a scene graph representation of the images. We normalize each image per channel with the per-channel mean and variance of all the images in the ImageNet dataset [96]. Specifically, the output image is defined in eq. (4.1):

$$I_{norm} = \frac{I_c - \bar{\Psi}_c}{\sigma_{\Psi_c}} \tag{4.1}$$

, where I represents the input image, c represent an image channel, Ψ corresponds to the ImageNet [96] distribution of images, and I_{norm} represents the input image after normalization.

Afterward, we resize the images into a uniform size of $H \times W \times c$ pixels, where $H = W$. Subsequently, the processed images are fed into a state-of-the-art scene graph prediction network called Factorizable Net [84]. Factorizable Net first uses a Region Proposal Network [101] to make object region proposals. It then uses a bottom-up approach where these region proposals are clustered into subgraph scene predictions. Messages are then passed between subgraphs to maintain spatial information and to refine the feature representation of the subgraphs. Finally, objects are predicted from the object features, and predicates are inferred based on the object features and the subgraph features.

Scene Graph Representation

Scene graphs are a structural representation of a scene consisting of a set of nodes typically corresponding to objects and a set of edges that describe the relationship between the objects. In our work, a node feature vector representation consists of three parts. It consists of the following: (1) the object's features obtained from the Factorizable Net [84] described in section 4.1, (2) the object's bounding box, and (3) a visual feature encoding of the object obtained from the pre-trained convolutional neural network.

Scene Graph Processing

Once we have the scene graph representation, we further process it by a graph convolutional network called Spatio-semantic Scene Graph Network (SGN) that allows feature representation information to flow through the graph. The edges are update is given in eq. (4.2) as:

$$(\alpha_{ij}^{(t+1)}, \rho_{ij}^{(t+1)}, \beta_{ij}^{(t+1)}) = \tau_{edges}(v_i^{(t)}, \rho_{ij}^{(t)}, v_j^{(t)}) \tag{4.2}$$

where $v_i^(0)$ corresponds to the i-th node feature vector before scene graph processing, τ_{edges} transformation is a implemented as a multilayer perceptron. The node update depicted in eq. (4.2) is computed as an average of the edge update result:

$$v_j^{(t+1)} = \tau_{nodes}(\frac{1}{N_i}(\sum \alpha_{ij}^{(t+1)} + \sum \beta_{ki}^{(t+1)})) \tag{4.3}$$

where N_i is the number neighbors of the i-th node and transformation τ_{nodes} is implemented as a multilayer perceptron. As a part of each object feature representation, SGN produces a bounding box prediction for each object. This is subsequently used to produce a scene layout.

Layout Generation

The processed scene graph representation is then used to produce a 2D scene representation called scene layout. In the scene layout, each pixel is a summation of the node features belonging to objects whose bounding box contains the pixel. The pixels that do not belong to any object's bounding box have a value of zero. This 2D spatial arrangement of features is then appended by low-level visual features of the query image obtained by passing the query image through a convolutional neural network.

Layout to Modified Image

Finally, the layout is decoded into an image by a decoder based on the SPADE [72] architecture. Introduced in 2019, SPADE is a methodology for improving the training of generative adversarial networks. In our work, the SPADE decoder consists of multiple residual spade blocks which focus on the normalization in the image generation process. In contrast to Batch Normalization [102], which is unconditional and isn't spatially sensitive, the original SPADE architecture conditions the normalization on the segmentation map. In Batch Normalization, several learned affine layers are applied after the normalization step. In SPADE, these layers are learned as the output of convolution on the semantic segmentation map. Such a conditional approach makes the affine layer spatially adaptive. In our work, instead of using semantic segmentation, the generation is conditioned on the RGB input image as well as the predicted scene layout. During training, the query RGB image corresponds to a masked image. To be specific, we first pass the RGB image through a single layer of 2D convolution. Then, in each

residual SPADE block of the decoder, the shallow feature representation of the masked RGB image is fed alongside the predicted semantic scene layout and the output of the previous residual SPADE block as the input. In the first layer, Gaussian noise is used as an input instead of the output from the previous layer. The architecture of the SPADE decoder is depicted in fig. 4.2. For a detailed description of the decoder, see table A.1.

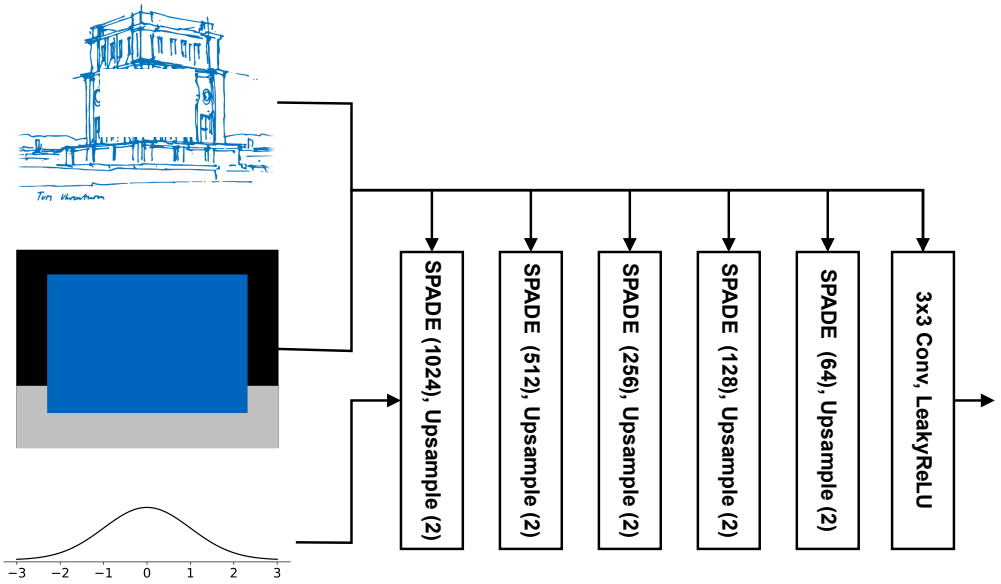


Figure 4.2.: Architecture of the SPADE decoder, which through 5 SPADE residual blocks upscales a 4×4 Gaussian noise conditioned on the scene layout and the input image into a 64×64 generated output image

4.2. Methodology

We perform training of the image manipulation system by image restoration which is described in detail in section 4.2.1. During training, we mask the visual feature encoding part of an object's node representation with a probability p_ρ. Afterward, we mask corresponding regions in the input image with Gaussian noise. Additionally, we independently mask the object's bounding boxes with probability p_b. As a result, the SGN performs a form of image reconstruction as it attempts to predict the bounding boxes of each object. Therefore the system performs two types of reconstruction – visual and spatial.

4.2.1. Image Manipulation Training Overview

There are two line approaches with respect to the training data used for training semantic image manipulation models. The first approach, which is resource costly involves having an image pair of an image before modification and an image after modification. The

model is simply fed an image before modification and attempts to generate the desired modification according to the ground-truth after-modification image. In our work, we do not require such image pairs. Instead, we train the model using a single image and performing image reconstruction. In the following text, we will describe how to train a model to perform specific desired image manipulations using image reconstruction.
Addition If the goal is to generate a new object in the image then the approach is as follows. First, we take an image that already contains the object. Then, we mask the region of the image that contains the object. The masked image is then fed into a generative model (for example GAN) that attempts to generate a realistic image that resembles the original unmasked image that includes the object. Finally, once the model is trained, the model can take an image and generate a new image that contains the additional object, such as a car, person, and so on.
Deletion To train a model capable of deleting objects. We take an image, select a region that contains a background, mask the region, and subsequently feed it into a model (for example GAN) that re-creates the original unmasked image. In inference time, when we want to delete an object from an image, we mask the region of the object and feed it into the trained model which then generates a novel image without the undesired object.
Replace The replacement functionality is trained exactly as the addition. In the inference, we mask the region of the to-be-replaced object and condition the restoration generation process with the desired object class.
Reposition If the new location of the to-be-re-positioned object is not overlapping with the current location, we could perform two operations - deletion of the object in the current location and addition of the object in the new location. In the scene graph settings, in addition to learning the desired layout, repositioning objects could require us to learn the change of relationships between the objects. For example, in the current setting, the objects are on top of each other, while in the new setting, the objects are next to each other.

Training losses

We employ several losses used for training state-of-the-art generative adversarial networks. For the generator, we use mean absolute error (L1 loss) between the generated and ground-truth images depicted in eq. (4.4).

$$\mathcal{L}_{rec} = \|I' - I\| \tag{4.4}$$

We use mean squared error (L2 loss) between predicted bounding boxes of objects and ground-truth depicted in eq. (4.5).

$$\mathcal{L}_{bbox} = \|\beta(I) - b\|_2 \tag{4.5}$$

, where b is the ground truth bounding box information. We also employ Perceptual Loss [10], Feature Matching Loss [9, 103], and Hinge Loss the objective main objective function of the generative adversarial network. Finally, we use an auxiliary classification

loss [104] that performs the classification of the individual reconstructed objects. The overall training loss function is depicted in eq. (4.6). The system is trained using a multi-scale discriminator [73].

$$\begin{aligned}\mathcal{L}_{total} = \mathcal{L}_{GAN,img} + \mathcal{L}_{GAN,obj} + \mathcal{L}_{bbox} \\ + \mathcal{L}_{rec} + \mathcal{L}_{aux,obj}\end{aligned} \tag{4.6}$$

Perceptual Loss is used to compare the high-level differences between two images. Used for learning style transfer between images [10], Perceptual Loss is also a powerful tool for image restoration and image generation. It works by feeding two images into a VGG [85] network that was pre-trained on ImageNet classification and comparing the intermediate feature representation of these images from multiple layers of the VGG network. The loss is depicted in eq. (4.7):

$$\mathcal{L}_{perceptual}(G) = \mathbb{E}_{(z,I)} \sum_{i=1}^{T} \frac{1}{N_i} \|\phi^{(i)}(I) - \phi^{(i)}(G(I_M, z))\|_2^2 \tag{4.7}$$

where G is the generator, $\phi^{(i)}$ is i-th layer representation of the pre-trained VGG [85] network, T is the number of layers of the VGG network, N_i denotes the number of elements of each layer, and $G(I_M, z)$ is the corresponding generated image from a latent variable z conditioned on the masked input image.

Similar to Perceptual Loss, **Feature matching Loss** works by feeding a real image and a corresponding generated image into a discriminator. Then extracting an intermediate feature representation of these images from multiple layers of the discriminator and making them match. The loss, which stabilizes the training, is depicted in eq. (4.8):

$$\mathcal{L}_{FM}(G, D) = \mathbb{E}_{(z,I)} \sum_{i=1}^{T} \frac{1}{N_i} \|D^{(i)}(I) - D^{(i)}(G(I_M, z))\|_1 \tag{4.8}$$

where G is the generator, D is the discriminator, T is the number of layers of the discriminator, N_i denotes the number of elements of each layer,and $G(I_M, z)$ is the corresponding generated image from a latent variable z conditioned on the masked input image.

We've also experimented with **Total Variational Loss** [105]. The total variational loss is the sum of the absolute differences for neighboring pixel-values of an image as shown in eq. (4.9). The goal of the loss is to encourage a spatial smoothness of generated images.

$$\mathcal{L}_{tv} = \frac{1}{H \cdot W} \sum \sum (I'_{i,j} - I'_{i-1,j})^2 + (I'_{i,j} - I'_{i,j-1})^2 \tag{4.9}$$

where $I_{i,j}$ represents the value of the pixel at i-th row and j-th column. However, using total variational loss proved to be to no avail. Possibly due to the relatively small size of the generated images.

4.2.2. Multi-task Learning

Multi-task learning is an inductive transfer mechanism that improves generalization by leveraging the domain-specific information contained in the training signals of related tasks [106]. It is inspired by how creatures, such as humans, learn. When we learn new tasks, we usually take advantage of the knowledge we have gained by learning related tasks. Similarly, in machine learning, the idea is that learning tasks jointly improve performance over learning tasks individually. In computer vision, multi-task learning has been used in many different areas. Not only working with images but with videos as well. It has for example been used for object categorization, image segmentation, visual tracking, and even scene classification [107]. Although the multi-task approach has been used for a related task of image reconstruction, the authors were unable to outperform the state-of-the-art single-task approaches [108]. To the best of our knowledge, the multi-task approach has not been used for the problem of image manipulation or within-the-scene graph settings.

Our goal is not only to improve the image manipulation capabilities of the original SIMSG [12] system but to also allow for the system to work in higher resolution. Therefore our multi-task learning setting reflects these goals. In short, we've modified the decoder into a two-headed architecture. With the first head, the decoder tries to reconstruct the original entire image. In the second head, the decoder focuses on reconstructing only a single image region containing a previously masked object. The architecture can be seen depicted in fig. 4.3. Generating the magnified object allows the network to gain additional fine detail information about the objects it tries to reconstruct.

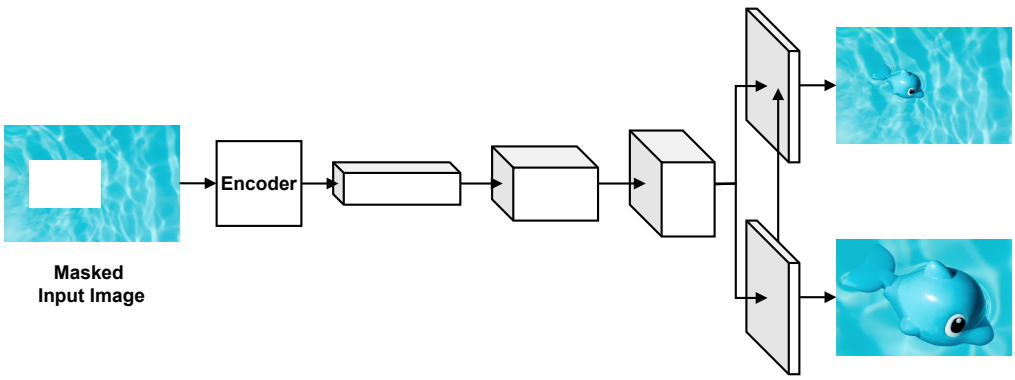


Figure 4.3.: Architecture of the multi-task two-headed approach. During training, the system takes masked input (on the left), uses a multi-step decoder to create a layout representation, and then through a series of SPADE [72] ResNet blocks reconstructs the entire original image (top-right) and detailed reconstruction of the masked object (bottom-right). The RGB image was taken by Andres Victorero[1]on Pexels[2]

4.2.3. Progressive Generation

In section 3.4.1, we've seen that it's beneficial to do image reconstruction in multiple steps. For example, by first restoring the borders of the to-be-restored regions and subsequently using information from the generated borders to generate the inner regions. The state-of-the-art approaches for progressive image reconstruction are divided into two main lines of work. In the first line of work, the progressive generation (masking) is handcrafted, whereas, in the second line of work, it is learned. We've implemented both approaches. Specifically, we've implemented a variation of the handcrafted approach introduced by Zhang et al. [69] and the automatic system called full-resolution residual network developed by Guo et al. [94].

In our system, an encoder first generates an intermediate layout representation which is then given alongside a masked image feature representation to the decoder that generates the resulting predicted image. In the variation of the handcrafted approach, we first generate an image prediction by the decoder and use this prediction to update the masked image feature representation that is again passed to the decoder alongside the layout. We obtain masked image feature representation by taking the original image that has the bounding boxes of the object bounding boxes on which we perform image reconstruction masked by Gaussian noise. This masked image is then fed into a shallow one-layer convolutional neural network that gives us a low-level feature representation. In the first pass, we pass the fully masked image into the convolutional neural network, whereas in the second pass, we replace the outer borders of the image with pixels of the predicted image of the first pass through the decoder. Specifically, we replace a border region with the thickness of 1 / 4 of the size of the image. This can be seen depicted in fig. 4.4.

4.2.4. Progressive Multi-task Generation

The confluence of methodologies described in section 4.2.2 and section 4.2.3 presented an instinctive trajectory to pursue. Specifically, in the first head that reconstructs the entire image, we progressively mask the object regions as described in section 4.2.3. On the other hand, the second head, which reconstructs only a single object region, attempts to reconstruct a border region with the thickness of 1 / 4 of the size of the image in the first pass and the entire object region in the second pass.

Progressive Diffusion Inspired by the successes of GANs in progressive image reconstruction, we've leveraged these principles in diffusion models. As described in sec-

[1]Profile available at https://www.pexels.com/@andres-victorero-2102935/
[2]Image available at https://www.pexels.com/photo/blue-and-white-toy-dolphin-floating-on-water-3730754/
[3]Profile available at https://www.pexels.com/@drone-trotter-2765040/
[4]Image available at https://www.pexels.com/photo/white-yacht-in-azure-sea-5640310/

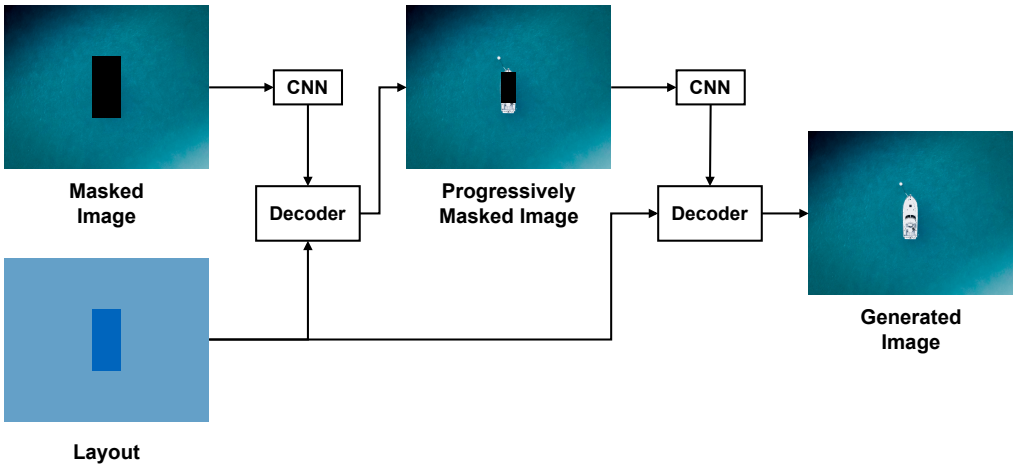
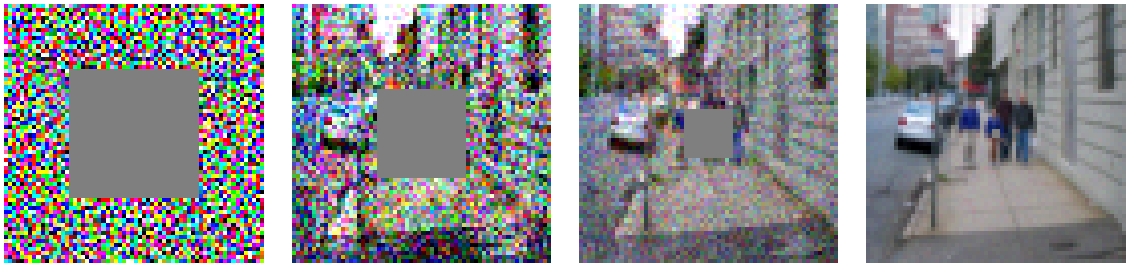


Figure 4.4.: Architecture of the handcrafted progressive generation approach. The system progressively fills in the border regions of masked input by replacing the mask with generated image pixels. The RGB image was taken by Drone Trotter[3]on Pexels[4]

tion 2.3.2, diffusion models work by learning how to reverse a diffusion process that slowly adds random noise to data. Similarly to image reconstruction, the reversal diffusion process is applied to the entire image. As in progressive image reconstruction, we believe that reversing the diffusion process in certain regions of the image could be more challenging in others. Specifically, we assume reversing the process in the center of the image, where the objects are typically located could pose a bigger challenge without additional information. Therefore, we progressively mask the image during the denoising steps to first let the system focus on reversing the diffusion process in the borders of the image. The approach is depicted in fig. 4.5.



(a) $t = 0$ (b) $t = 333$ (c) $t = 666$ (d) $t = 1000$

Figure 4.5.: Visual Genome sample generation through progression diffusion process visualized at different diffusion timesteps

5. Experiments

5.1. Data

In this section, we will describe and analyze image datasets for visual understanding and reasoning called CLEVR [109] and Visual Genome [88]. As opposed to VG, which contains real-life images, CLEVR is a synthetic dataset with a limited complexity of the depicted scenes. Compared to VG, CLEVR has a lower variety of object classes, types of relationships as well as the average number of objects present in an image. Both of these datasets provided annotated scene graph annotation for each image, making these datasets suitable for image manipulation tasks. An example of images from both of these datasets can be seen in fig. 5.1

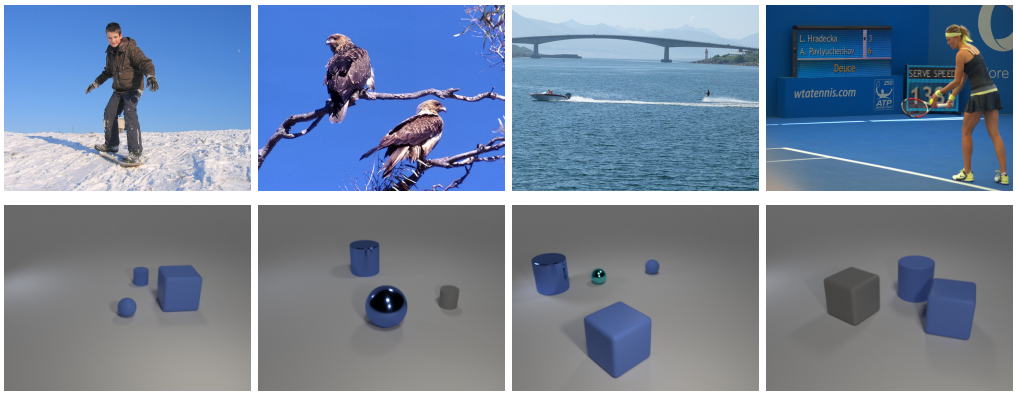


Figure 5.1.: Sample images from Visual Genome (top row) and CLEVR (bottom row) datasets

5.1.1. CLEVR

Introduced in 2017, CLEVR [109] is a diagnostic dataset for compositional language and elementary visual reasoning. CLEVR is a dataset that contains 100k images depicting scenes with objects of three different shapes (cube, sphere, and cylinder) with various attributes (size, material). Each image has a 128×128 dimension with three RGB channels. In our work, we use a sub-set of 21310 images, each containing 3 to 7 objects. The analysis of the objects in terms of their shapes and attributes can be seen in fig. 5.2. Furthermore, we calculate the size of each object in the dataset as $size = width \cdot height$ and depict the resulting distribution in fig. 5.3.

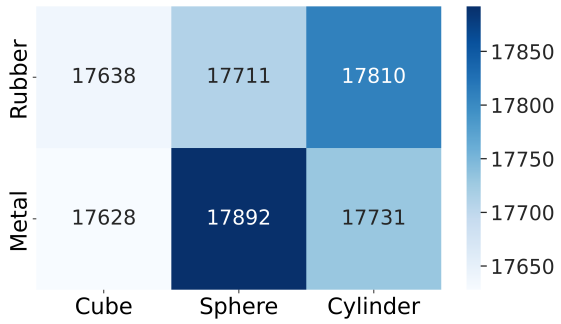


Figure 5.2.: Heatmap analyzing the number of object types in the CLEVR dataset by shape and material

Each image has also a corresponding scene graph annotation. The relationship in the scene graph describes the spatial relationship between two objects. Namely, "left", "right", "behind", and "in front". The synthetic nature of the CLEVR [109] framework allows us to generate before and after modification image pairs. However, for training, we do not use the image pairs but rather perform image reconstruction as described in section 4.2.1.

The dataset is primarily used as a benchmark for visual reasoning, therefore it contains a set of questions and answers about the scenes. An example question might be "Are there an equal number of large things and metal spheres?". However, in our work, we only use the images with their corresponding scene graph representation, bounding boxes, object classes, and object attributes. The limited number of possible shapes simplifies recognition and makes this dataset an ideal benchmark for prototyping new image manipulation approaches. For a detailed description of the dataset, please refer to dataset publication [109].

5.1.2. Visual Genome

Visual Genome [88] or VG for short is a dataset containing images and descriptions that outline the objects, attributes, and relationships within the images through a scene graph annotation. Compared to CLEVR [109], VG is a much more complex dataset consisting of 108k images where each image has an average of 35 objects, 26 attributes, and 21 pairwise relationships between objects. Compared to the very limited number of categories in CLEVR, VG contains more than 33k different object categories, more than 68k attribute categories, and more than 42k relationship categories. The diversity of VG, especially in terms of the number of object classes, surpasses all other datasets including MS-COCO [110] that uses the same images. VG is the largest dataset in terms of the number of relationships. However, inspired by Johnson et al. [64] we've avoided working with object and relationship types that are occurring less than 2000 and 500 times respectively. Furthermore, we only work with images that contain 3 to 30 objects. This leaves us with 45 relationships and 178 object types.

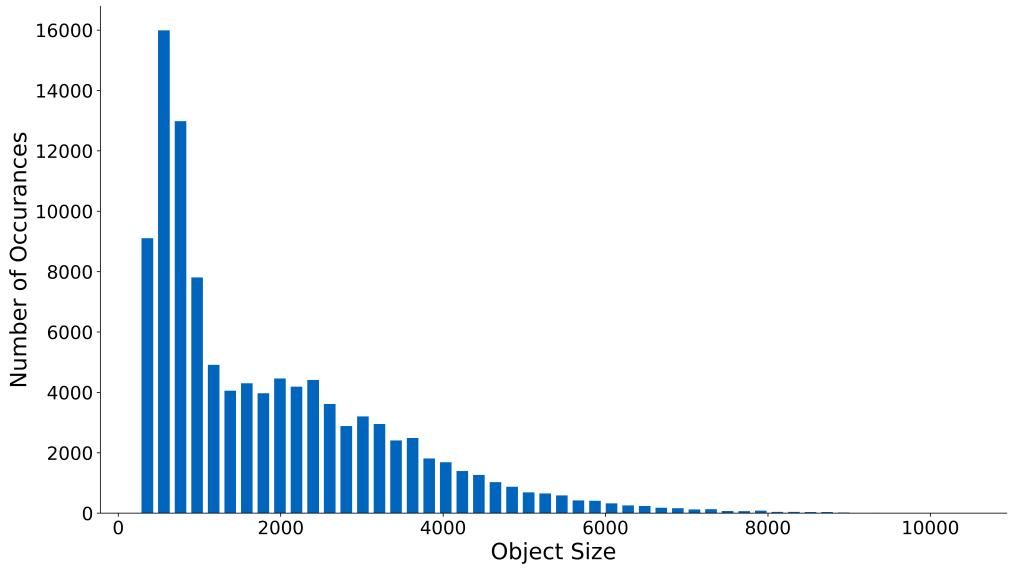


Figure 5.3.: Histogram of object sizes in the CLEVR dataset, where the size is calculated as $size = width \cdot height$

Compared to CLEVR [109], which has only positional relationships, VG [88] has a variety of different relationships. These include actions, spatial, comparative relationships, and verbs ("wears"). The distribution of the most common relationships can be seen in fig. 5.4

The authors not only provide scene graph annotation for the entire image but also provide a description and a scene graph for 42 for different regions of the images. In our work, we only utilize the scene graph representing the entire image scene. We split the data into 80% train, 10% validation, and 10% test sets. Such data split corresponds to 62565 train, 5506 validation, and 5088 test images with an average of ten objects and five relationships per image.

Overall, VG [88] provides a multi-layered understanding of pictures and it is an ideal dataset for developing models with a broader understanding of our visual world. The diversity of the dataset makes it a challenging benchmark for our image manipulation system. For a detailed description of the dataset, please refer to dataset publication [88].

5.2. Evaluation Metrics

We've evaluated our models with multiple metrics used for the assessment of image generation models. Namely, we've used Mean Absolute Error (MAE) between generated and ground-truth images, Structural Similarity Index Measure [111] (SSIM), Learned Perceptual Image Patch Similarity [112] (LPIPS), and Fréchet Inception Distance [100]

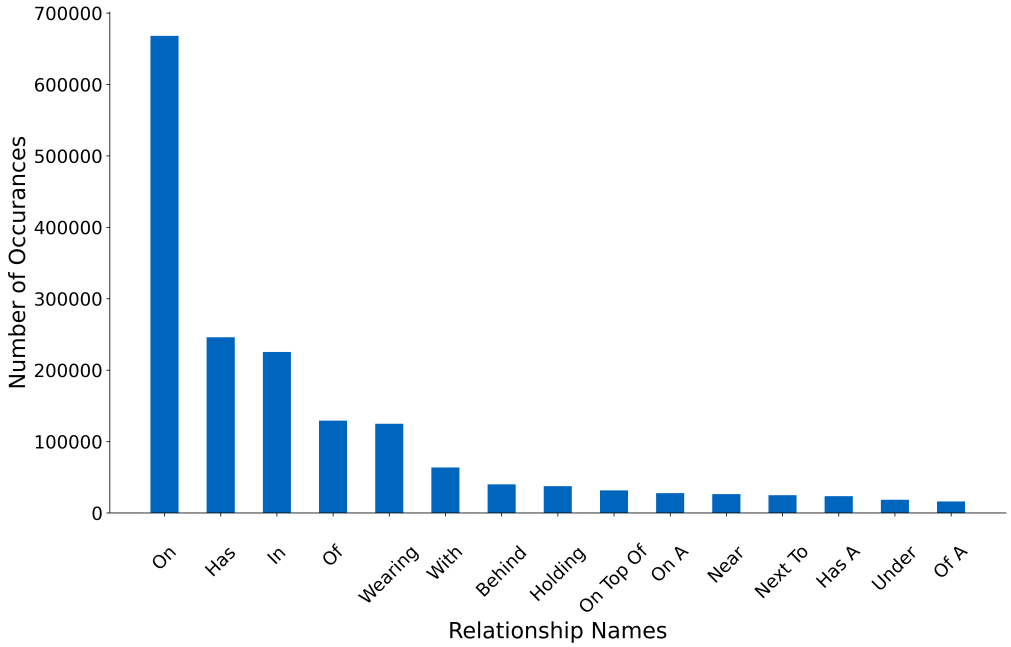


Figure 5.4.: The top 15 most common relationships in the Visual Genome dataset

(FID).

Mean Absolute Error

Mean Absolute Error or MAE for short is still one of the most commonly used evaluation metrics mainly for its ease of implementation and understanding. To calculate the MAE of a generated and ground-truth image of size $H \times W \times 3$ we define MAE in eq. (5.1):

$$\text{MAE} = \frac{1}{H \cdot W \cdot 3} \sum_{i=1}^{H} \sum_{j=1}^{W} \sum_{k=1}^{3} |I'_{ijk} - I_{ijk}| \tag{5.1}$$

where I' corresponds to the generated image and I to the ground-truth image.

5.2.1. Structural Similarity Index Measure

Introduced in 2004, SSIM is a popular metric that attempts to compare structural differences between the generated and ground-truth images, rather than looking at each pixel individually. The assessment is based on structural information that tries to mimic how people extract structural information from a scene [111].

The authors claim that for image quality assessment, it is useful to compute structural statistics locally rather than globally. Therefore, they use a Gaussian window of size $N \times N$ which moves pixel-by-pixel over the entire image and computes the SSIM [111]

for each $N \times N$ patch and subsequently takes the average of the calculated SSIM values. In our work, we set $N = 3$

The SSIM [111] is composed of three structural features. Namely, luminance, contrast, and structure. The luminance is represented as the Gaussian weighted average of the pixel over the window. It is defined as $\mu_x = \frac{1}{N \cdot N} \sum_{i=1}^{N} \sum_{j=1}^{N} w_{ij} \cdot I_{ij}$, where I_{ij} are the pixels values of i-th row and j-th column of the window and the w_{ij} are the corresponding gaussian weights. We defined contrast as weighted standard deviation σ_I and structure as $(I - \mu_I)/\sigma_I$. The higher the SSIM, the better. In essence, we derive SSIM as:

$$\begin{aligned}
\text{SSIM}(I, I') &= \text{Luminance function} + \text{Contrast function} + \text{Structure function} && (5.2)\\
&= [l(I, I')]^{\alpha} + [c(I, I')]^{\beta} + [s(I, I')]^{\gamma} && (5.3)\\
&= \frac{2 \cdot \mu_I \cdot \mu_{I'} + C_1}{\mu_I^2 + \mu_{I'}^2 + C_1} + \frac{2 \cdot \sigma_I \cdot \sigma_{I'} + C_2}{\sigma_I^2 + \sigma_{I'}^2 + C_2} + \frac{\sigma_{II'} + C_3}{\sigma_I \cdot \sigma_{I'} + C_3} && (5.4)\\
&= \frac{(2 \cdot \mu_I \cdot \mu_{I'} + C_1) \cdot (2 \cdot \sigma_{II'} + C_2)}{(\mu_I^2 + \mu_{I'}^2 + C_1) \cdot (\sigma_I^2 + \sigma_{I'}^2 + C_2)} && (5.5)
\end{aligned}$$

where C is a constant influenced by the dynamic range for pixel values. This holds if we assume $\alpha = \beta = \gamma = 1$ and $C_3 = C_2/2$. In our work, we've set $C_1 = 0.01^2$ and $C_2 = 0.03^2$.

5.2.2. Learned Perceptual Image Patch Similarity

Despite the advancements SSIM [111] brought to the image quality assessment over MAE, SSIM is still a relatively shallow function that fails to capture all the nuances of human perception. In recent years, we've seen a surge of training losses based on feature representation of images obtained from deep models such as Feature Matching Loss [9] and Perceptual Loss [10]. In 2018, Zhang et al. set out to assess the quality of these deep features as a perceptual metric [112]. They used a collection of images from multiple different datasets, took an image, and then performed two degrees of distortion. The task was to then answer which of these two images is closer to the original image. The authors found out that on both traditional and CNN-based distortions, deep feature-based assessments outperformed shallow metrics such as SSIM.

Besides assessing the quality of deep embeddings of images for image quality assessment, the authors proposed their own architecture. They obtain the deep features of two images from the penultimate layer of the pre-trained neural network model. However, instead of computing differences between the two embeddings directly, they added a small linear model that computes the similarity of the images based on the embeddings. This constitutes a "perceptual calibration" of a few parameters in an existing feature

space. The system outputs a value in the [0, 1] range. The lower the value, the better.

In our work, we've used AlexNet [113] pre-trained on ImageNet for an image classification task as the image feature extractor. We've pre-trained linear model provided by Zhang et al. [112][1].

5.2.3. Fréchet Inception Distance

Fréchet Inception Distance [100] or FID for short is a metric for the evaluation of the performance of GANs. It tries to capture the quality and variety of generated images by estimating the similarity to the real image samples. It was introduced as a successor to the popular Inception score metric.

Inception score attempts to capture two properties of the image generation [9]. To measure the quality of generated images as well as the variety of generated images. Given a generated image of an object class o, the pre-trained Inception classifier should have the output classification probability as close as possible to 1. On the other hand, given a set of generated images, the marginal probability of pre-trained Inception classifier probability output should be as close to uniform distribution as possible. The authors capture both properties with the Inception score by computing KL divergence between the individual (label) distribution and the marginal distribution. The more these distributions differ, the better. The score has however several disadvantages such as being limited by what the Inception classifier can detect or being sensitive to small changes in network weights [114].

FID [100] is an improvement over the Inception score as it uses statistics of real-world samples. It compares the distribution of the generated and real-world samples. Like the Inception score, the FID score uses the Inception model. Specifically, it first runs an image through the Inception model and extracts the last layer features before the output classification layer. It collects these features for both sets of real and generated images. Then, it summarises these features as a multivariate Gaussian by calculating the mean and covariance of the sets. The distance between these two distributions is then calculated using the Frechet distance [115], also called the Wasserstein-2 distance [116] depicted in eq. (5.6). The closer the distance, the better.

$$\text{FID} = \|\mu_{F(I')} - \mu_{F(x)}\|_2^2 + tr(\Sigma_{F(I')} + \Sigma_{F(I)} - 2 \cdot (\Sigma_{F(I')} \cdot \Sigma_{F(I)})^{1/2}) \tag{5.6}$$

where $\mu_{F(I')}$ corresponds to the mean of features from generated images, $\mu_{F(I)}$ to the mean of features from ground-truth images, $\Sigma_{F(I')}$ to the covariance matrix of generated features, and $\Sigma_{F(I)}$ to the covariance matrix of ground-truth features. In our work, we've used Pytorch implementation by Maximilian Seitzer [117].

[1] Available at https://github.com/richzhang/PerceptualSimilarity

5.3. Experimental Setup

The training of the system was done by reconstructing 32 images per batch. We used a loss weight of 1 for the L1 pixel loss and the Hinge Loss, a weight of 5 for the Perceptual Loss and Feature Matching Loss, a weight of 50 for the bounding box L2 Loss, and a weight of 0.1 for the auxiliary classification loss [104]. For the CLEVR [109] dataset, we typically performed 100 thousand iterations to fully train the system. For the VG [88] dataset, we allowed for up to 300 thousand iterations, with early stopping typically done at around $100 - 150$ thousand iterations. This typically meant that training the system on the CLEVR dataset took around $4 - 6$ days while training the system on the VG dataset took about $6 - 9$ days depending on the trained architecture. All experiments were performed with images normalized to a uniform 64×64 dimension.

Training Hardware

All the training alongside the subsequent experiments was performed on hardware provided by an affiliate member of the Technical University of Munich – Leibniz Supercomputing Centre (LRZ). The training was done on an NVIDIA DGX-1, which is an integrated system for deep learning containing NVIDIA Tesla V100 GPUs that have 16 GB of RAM per GPU. In addition, we used the DGX A100 system, which contains NVIDIA A100 GPUs that have 80 GB of RAM per GPU.

5.4. Quantitative Results

This section focuses on the quantitative results of the image reconstruction experiments obtained from the two publicly available datasets CLEVR [109] and Visual Genome [88].

5.4.1. Comparison to Previous State-of-the-art

As we can see in table 5.1, both the two-headed approach and the progressive approach of the PRISM system significantly outperform the baseline SIMSG [12] architecture in all metrics on the CLEVR dataset. They outperform the baseline approach by a significant margin in both metrics that capture pixel differences (MAE) as well as in metrics that capture high-level image differences (LPIPS [112], FID [100]). Furthermore, the experiment results show a boost in performance in terms of image reconstruction when combining the two architectures.

The findings are further supported by results on a more challenging VG dataset, which can be seen depicted in table 5.2. PRISM outperforms the previous state-of-the-art methods [80, 64, 12] by a significant margin and outperforms SIMSG [12] in all image restoration metrics.

In table 5.3 we can see that all versions of PRISM significantly outperform the SIMSG [12] baseline in the image restoration task.

Method	MAE ↓	SSIM ↑	LPIPS ↓	FID ↓
SIMSG [12]	5.048	0.982	0.015	3.575
PRISM + **M** (Ours)	4.273	0.985	0.011	**2.629**
PRISM + **P** (Ours)	4.454	**0.987**	**0.008**	2.757
PRISM + **PM** (Ours)	**4.101**	0.986	0.010	2.634

Table 5.1.: Image reconstruction results on the CLEVR dataset compared to the SOTA. **P**: Progressive, **M**: Multi-task

	MAE ↓	SSIM ↑	LPIPS ↓	FID ↓
ISG [80]	46.44	0.281	0.32	58.73
Cond SG2Im [64]	14.25	0.844	0.081	13.40
SIMSG [12]	10.812	0.861	0.065	6.544
PRISM + **PM** (Ours)	**9.908**	**0.864**	**0.064**	**5.988**

Table 5.2.: Image reconstruction results on the VG compared to the SOTA. **P**: Progressive, **M**: Multi-task

5.4.2. Ablation study – Residual Connection

As we can see in fig. 4.3, the second head, which generates a detailed reconstruction of the masked input, sends the feature representation of the object as an input to the first head which generates the entire image. Intuitively, this allows for an easier propagation of the object's visual detail information between the heads. We've done an ablation study measuring the effects of utilizing the connection which can be seen in fig. 5.5. As we can see, the results obtained from the two publicly available datasets confirm the benefits of having such a connection between the heads.

5.4.3. Two-headed Approach – Delayed Alternating Optimization

We've experimented with delaying the alternating optimization for the training of the two-headed architecture. Since pre-training is a common practice in computer

Connection	MAE ↓	SSIM ↑	LPIPS ↓	FID ↓
CLEVR				
-	4.888	0.985	0.011	2.702
✓	**4.273**	0.985	0.011	**2.629**
Visual Genome				
-	9.9	0.856	0.076	7.728
✓	**9.677**	**0.863**	**0.062**	**6.766**

Figure 5.5.: Ablation study between multi-task approach with and without connection between the heads on the CLEVR and VG datasets.

Method	MAE ↓	SSIM ↑	LPIPS ↓	FID ↓
PRISM + **M** (Ours)	9.802	**0.866**	0.065	6.302
PRISM + **P** (Ours)	**9.677**	0.863	**0.062**	6.766
PRISM + **PM** (Ours)	9.908	0.864	0.064	**5.988**

Table 5.3.: Ablation study of different components on the VG. **P**: Progressive, **M**: Multi-task

vision, we've experimented with pre-training the first head of the two-headed approach, essentially mimicking the original SIMSG training, before starting the join optimization. We've experimented with starting the multi-task approach after 10 and 20 thousand epochs. However, saw no benefits over starting directly with the joint optimization. These results support the finding of Zoph et al. who argue that the joint-training is more adaptive and beneficial compared to pre-training [118].

5.4.4. Two-headed Approach – Different Window-sizes

As described earlier in section 4.2.2, in the two-headed approach one head reconstructs the entire image while the second head focuses on reconstructing a single object. To be specific, we want the second head to reconstruct a masked object region of size $l \times l$, $l = w$ if $w >= h$ otherwise $l = h$, where h and w represent the height and width of the object to be reconstructed respectively. In this experiment, we've investigated the effects of different window sizes (i.e. $l = l \cdot \delta$) on the ability of the first head to reconstruct the entire image. As we can see from table 5.4, having the smallest window proved to be the most beneficial. These results are intuitive as the smallest window corresponds to the highest level of detail reconstructed by the second head and thus offers the most additional information. What's surprising are the results obtained with $\delta = 1.5$, these results can be explained by an implementation detail that backtracks to $\delta = 1.0$ if the cropped window extended beyond the image borders, which happens more often with increasing the window size.

δ	MAE ↓	SSIM ↑	LPIPS ↓	FID ↓
1.0	4.068	0.987	0.010	**2.506**
1.1	4.273	0.986	0.010	2.629
1.2	4.699	0.984	0.011	3.274
1.3	4.413	0.985	0.016	2.842
1.5	**3.925**	0.987	0.011	2.592

Table 5.4.: Effects of different window-sizes of the second head of the two-headed approach evaluated on CLEVR dataset

5.4.5. Image Manipulation Survey

To quantitatively evaluate the image manipulation capabilities of PRISM compared to SIMSG [12], we've created an online survey [2]. The assessment is made up of 10 questions. Each question consists of an original unmodified image, a description of the desired image manipulation, and an image pair generated by PRISM and SIMSG according to the desired manipulation task. For the evaluation, people were asked to pick which image of the pair looked more realistic and in accordance with the manipulation task specification. The survey contains all the image manipulation modes: "object deletion", "object replacement", and "object repositioning" (change of relationship). The survey is done in a single-blind study format, in which the survey participants are unaware of which image of the pair belongs to which manipulation method. The order of generated images in the image pair was selected at random for each image pair.

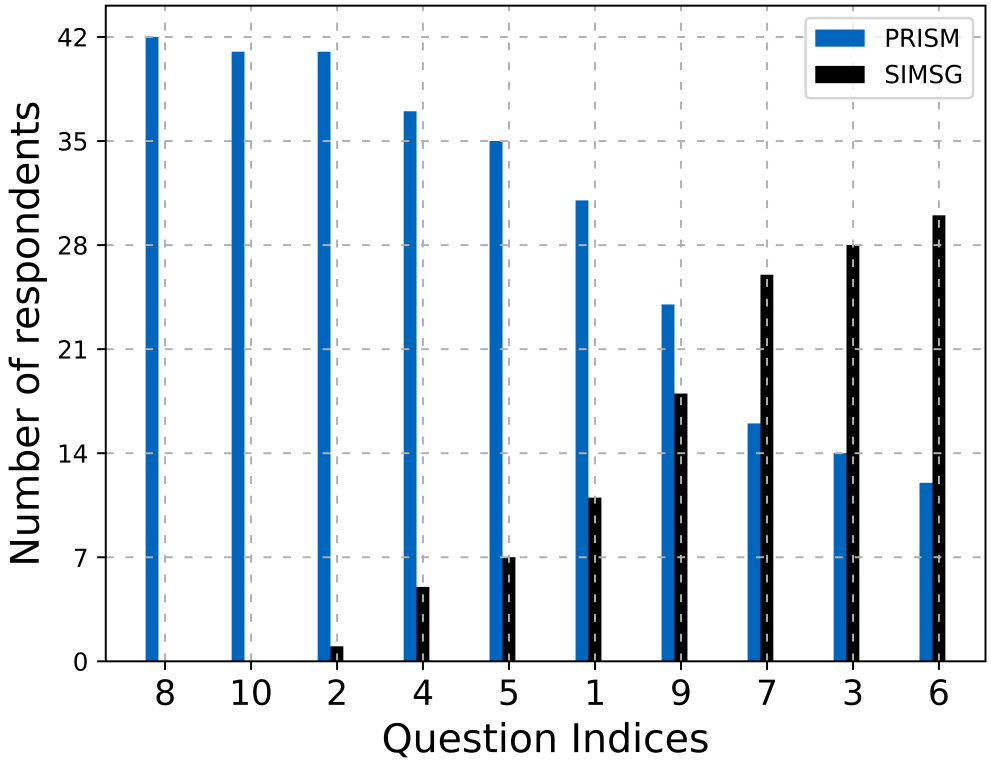


Figure 5.6.: A bar chart depicting how often survey respondents selected PRISM **(blue)** and SIMSG **(black)** as generating more realistic and accurate images according to the image manipulation specification per each question image-pair

The survey was completed by 42 participants. Each volunteer filled out the form only once. The results are shown in fig. 5.6 as a bar chart of the number of times each

[2]Available at bit.ly/jahoda_image_manipulation

system was selected as the one producing more realistic images per each image pair. The images generated by PRISM were considered more realistic in 70% image-pair cases than their SIMSG [12] counterparts. Furthermore, PRISM-generated images were picked on average in 69.9% of the selections made by the participants of the study. Out of all the survey respondents, 90.5% preferred images generated by PRISM (i.e. they selected images generated by PRISM in more than 50% of the cases) and 0% preferred images generated by SIMSG. These results further indicate that PRISM achieves superior image quality in the image manipulation task.

5.4.6. Progressive Diffusion

As discussed in section 4.2.4 we progressively mask the generated image during the diffusion process. Specifically, we mask a square region of a width equal to $w = w_{total} \cdot t/(t_{total} \cdot 2)$, where t is the current timestamp, t_{total} is a total number of diffusion steps and w_{total} is the total width of the entire image. We do the progressive masking 50% of the time and the rest of the time apply the standard diffusion process. We performed an image generation experiment in which two models, the baseline diffusion model, and the progressive diffusion model, were trained to generate images according to the CLEVR dataset training subset. Specifically, we've modified an open-source implementation [3] of the work by Ho et al. [45]. Afterward, both models were asked to generate N_{test} number of samples, where N_{test} is equal to the number of testing samples in the CLEVR dataset. Finally, the generated sets were compared with the ground-truth testing set to calculate the FID score described in section 5.2.3. The progressive diffusion model achieved a significantly lower FID score of 39.516 compared to the baseline diffusion model which achieved FID equal to 65.023. These early experiment results suggest a potential avenue for future research.

5.5. Qualitative Results

This section depicts and compares the generation quality of PRISM and its variants compared to SIMSG [12] by showcasing examples of image restoration and image manipulation on the CLEVR [109] and Visual Genome [88] datasets.

5.5.1. Image Reconstruction

In addition to generating an entire image, the two-headed approach also reconstructs a single image region containing a previously masked object. Generating this zoomed-in detailed reconstruction of the masked object helps the network obtain information about small objects that would otherwise be lost. We can see an example of a ground truth image from a CLEVR [109] dataset together with the output from both heads of the two-headed network in fig. 5.7. We can observe that due to the guidance of

[3] Available at https://github.com/lucidrains/denoising-diffusion-pytorch

the detail-capturing second head, the first head was able to accurately reconstruct the ground-truth image, completely matching the shape, color, texture, and details (such as shadow) of the reconstructed object.

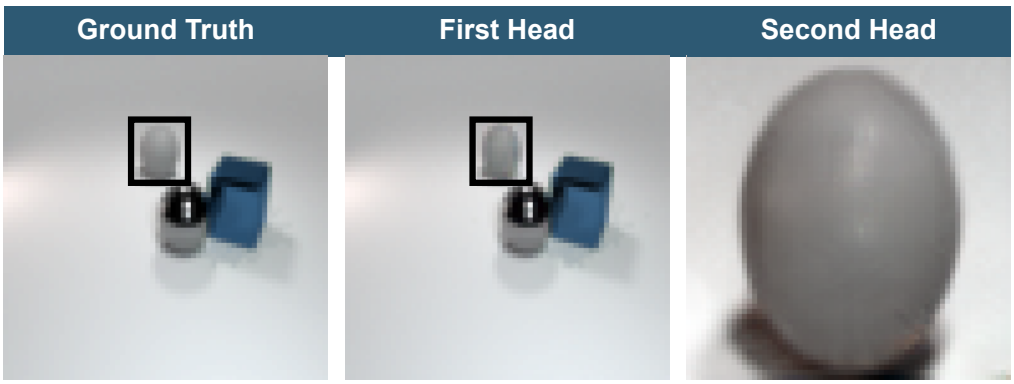


Figure 5.7.: An example of a ground-truth CLEVR image sample **(left)**, entire image reconstruction prediction from the first head of the two-headed approach **(middle)**, and a detailed reconstruction of the previously masked image by the second head **(right)**

Furthermore, image reconstruction from both the two-headed and the SIMSG approach is depicted in fig. 5.8. We can see that both approaches were for the most part successful in generating the correct shape and color of the reconstructed objects. However, the SIMSG approach sometimes fails to capture the fine details of the object's texture. The two-headed approach is significantly more successful in capturing these details.

5.5.2. Image Manipulation

In fig. 5.9 we see an example of all four image manipulation modes (removal, addition, replacement, and repositioning) realized by both SIMSG and PRISM. We can see that similarly to image restoration, both methods were for the most part successful in generating the correct shape and color of the generated objects when performing image manipulation. And again PRISM exhibits better detail-capturing capabilities of the manipulated objects compared to SIMSG on the CLEVR dataset. These findings are further supported by a qualitative study performed on the VG dataset depicted in fig. 5.10. The generated images demonstrate that PRISM exhibits superior performance in capturing the underlying structure of the image compared to SIMSG. In addition, these images show that PRISM outperforms SIMSG in completing the desired manipulation task, further establishing its robustness. For more image manipulation examples, see appendix A.2 in the appendix.

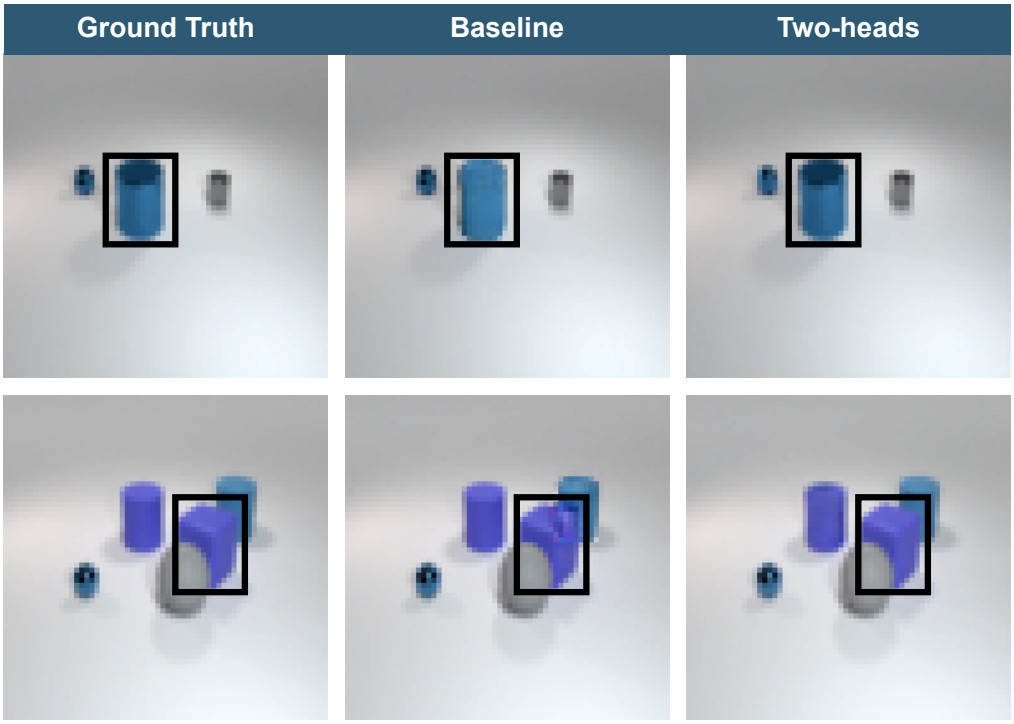


Figure 5.8.: Qualitative comparison between baseline and two-headed image reconstruction on the CLEVR dataset

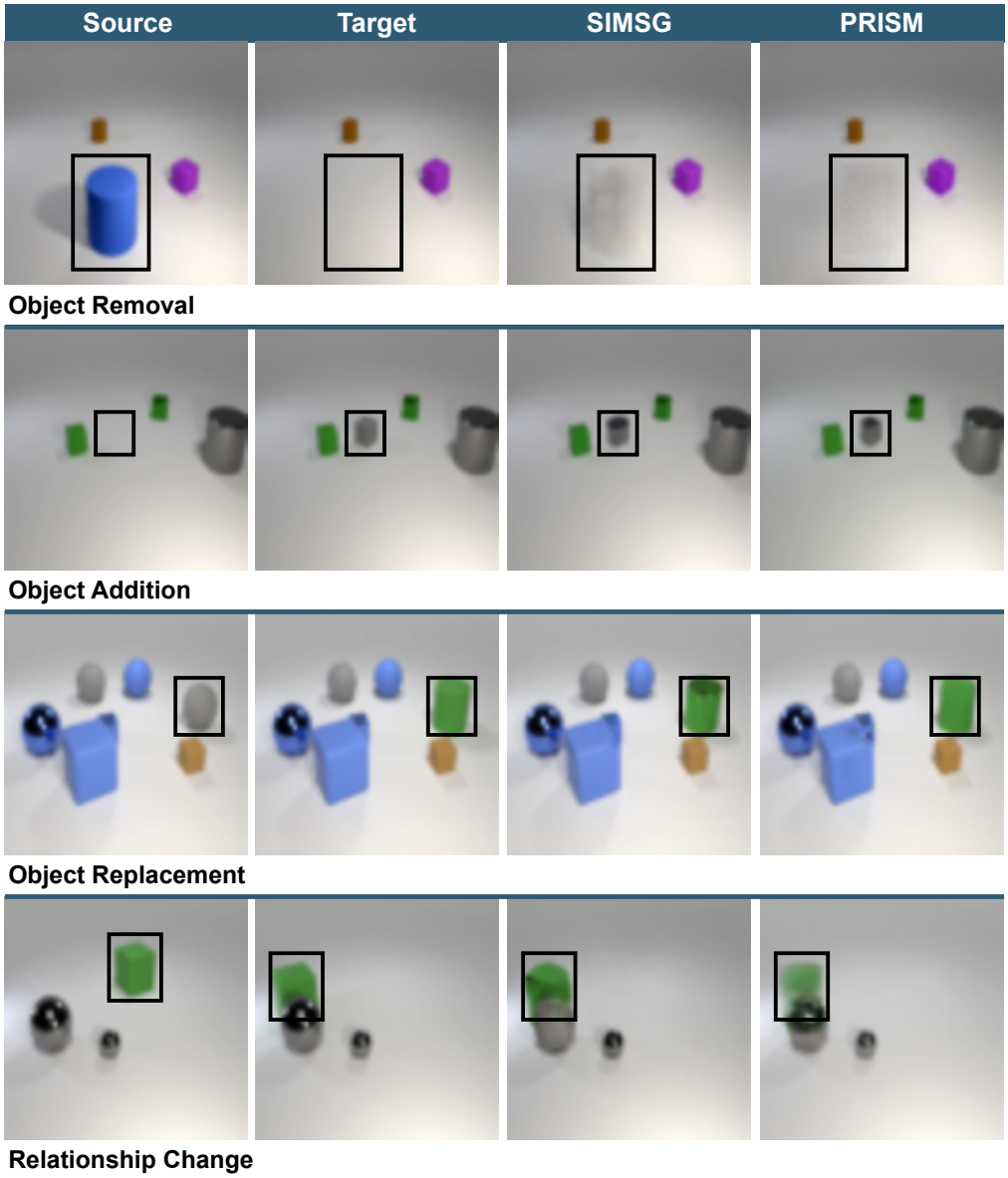


Figure 5.9.: Qualitative comparison between SIMSG [12] and PRISM in terms of all four image manipulation modes (removal, addition, replacement, and reposition) on the CLEVR [109] dataset. Since CLEVR images are generated in a controlled environment, we have access to the target images in addition to the source images and therefore, a direct comparison becomes possible after the change.

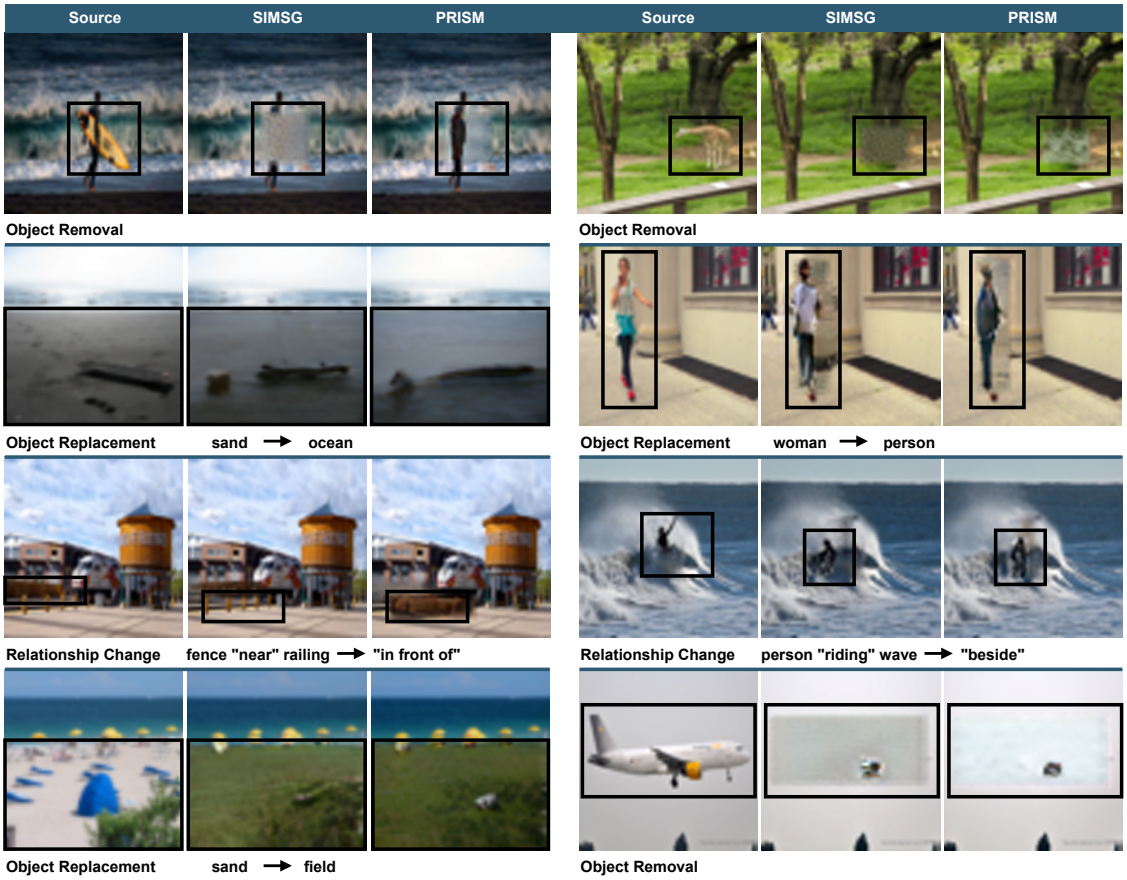


Figure 5.10.: Qualitative comparison between SIMSG [12] and PRISM in terms of object removal, replacement, and relationship change on the VG [88] dataset. Our model achieves better image manipulation performance compared to SIMSG in different manipulation modes.

6. Conclusion

Our proposed multi-task semantic image manipulation framework called PRISM is a novel approach that utilizes a two-headed generator and progressive image inpainting techniques to improve the quality of manipulated images. Through rigorous experimentation on two public benchmarks, our method has demonstrated superior performance compared to previous works in both image reconstruction and image manipulation tasks. These results were further supported by human evaluation survey, where 90.5% of survey respondents preferred images generated by PRISM compared to the previous state-of-the-art methods. The effectiveness of our method highlights the significant potential of these techniques for future research in the field of semantic image manipulation. Improved image manipulation techniques can have far-reaching applications in diverse fields such as image editing, computer graphics, and computer vision.

Moreover, we've modified the progressive approach for diffusion processes and showed that diffusion benefits from using these techniques for image generation. Exploring progressive and multi-task approaches in combination with diffusion processes for image manipulation remains a rich research endeavor and will be explored further in future work. Overall, the thesis presents a significant step forward in the field of semantic image manipulation and opens new avenues for future research.

A. Appendix

A.1. Architecture Details

A.2. Additional Qualitative Results

Name	Type	In Ch.	Out Ch.	Filter	Stride	Pad.	Output	Param
Inp_img	DataLayer	-	4	-	-	-	64×64	-
conv_inp	Convolution	4	32	3×3	1	0	64×64	160
Scale_I	Interpolate	32	32	-	-	-	4×4	-
Layout	DataLayer	-	384	-	-	-	64×64	-
Scale_L	Interpolate	384	384	-	-	-	4×4	-
Noise	DataLayer	-	1	-	-	-	4×4	-
SBlk_0_0	SPADE	384	1	-	-	-	4×4	444K
SBlk_0_1	SPADE	384	1	-	-	-	4×4	444K
SBlk_0_s	SPADE	384	1	-	-	-	4×4	444K
SBlk_0_n0	Convolution	33	1	3	1	1	4×4	298
SBlk_0_n1	Convolution	1	1024	3	1	1	4×4	10K
SBlk_0_ns	Convolution	1	1024	1	1	0	4×4	1K
SBlk_1_0	SPADE	384	1024	-	-	-	8×8	2M
SBlk_1_1	SPADE	384	512	-	-	-	8×8	1M
SBlk_1_s	SPADE	384	1024	-	-	-	8×8	2M
SBlk_1_n0	Convolution	1056	512	3	1	1	8×8	5M
SBlk_1_n1	Convolution	512	512	3	1	1	8×8	2M
SBlk_1_ns	Convolution	1024	512	1	1	0	8×8	500K
SBlk_2_0	SPADE	384	512	-	-	-	16×16	2M
SBlk_2_1	SPADE	384	256	-	-	-	16×16	1M
SBlk_2_s	SPADE	384	512	-	-	-	16×16	2M
SBlk_2_n0	Convolution	544	256	3	1	1	16×16	1M
SBlk_2_n1	Convolution	256	256	3	1	1	16×16	590K
SBlk_2_ns	Convolution	512	256	1	1	0	16×16	131K
SBlk_3_0	SPADE	384	256	-	-	-	32×32	1M
SBlk_3_1	SPADE	384	128	-	-	-	32×32	737K
SBlk_3_s	SPADE	384	256	-	-	-	32×32	1M
SBlk_3_n0	Convolution	288	128	3	1	1	32×32	331K
SBlk_3_n1	Convolution	128	128	3	1	1	32×32	147K
SBlk_3_ns	Convolution	256	128	1	1	0	32×32	33K
SBlk_4_0	SPADE	384	128	-	-	-	64×64	737K
SBlk_4_1	SPADE	384	64	-	-	-	64×64	590K
SBlk_4_s	SPADE	384	128	-	-	-	64×64	737K
SBlk_4_n0	Convolution	160	64	3	1	1	64×64	92K
SBlk_4_n1	Convolution	64	64	3	1	1	64×64	37K
SBlk_4_ns	Convolution	128	64	1	1	0	64×64	8K
out_conv	Convolution	64	64	3×3	1	1	64×64	37K
out_conv	LeakyReLu	-	-	-	-	-	64×64	-0.2
out_conv	Convolution	64	3	1×1	1	0	64×64	195
out_conv2	Convolution	67	64	3×3	1	1	64×64	39K
out_conv2	LeakyReLu	-	-	-	-	-	64×64	-0.2
out_conv2	Convolution	64	3	1×1	1	0	64×64	195

Table A.1.: Detailed description of the decoder architecture parameters. **Param**: describes either the number of parameters of the layer or a function parameter depending on the type of the layer, **SBlk**: Spade Resnet block, **Ch.**: Channel, **Pad.**: Padding

Figure A.1.: Qualitative comparison between SIMSG [12] and PRISM on VG [88] dataset

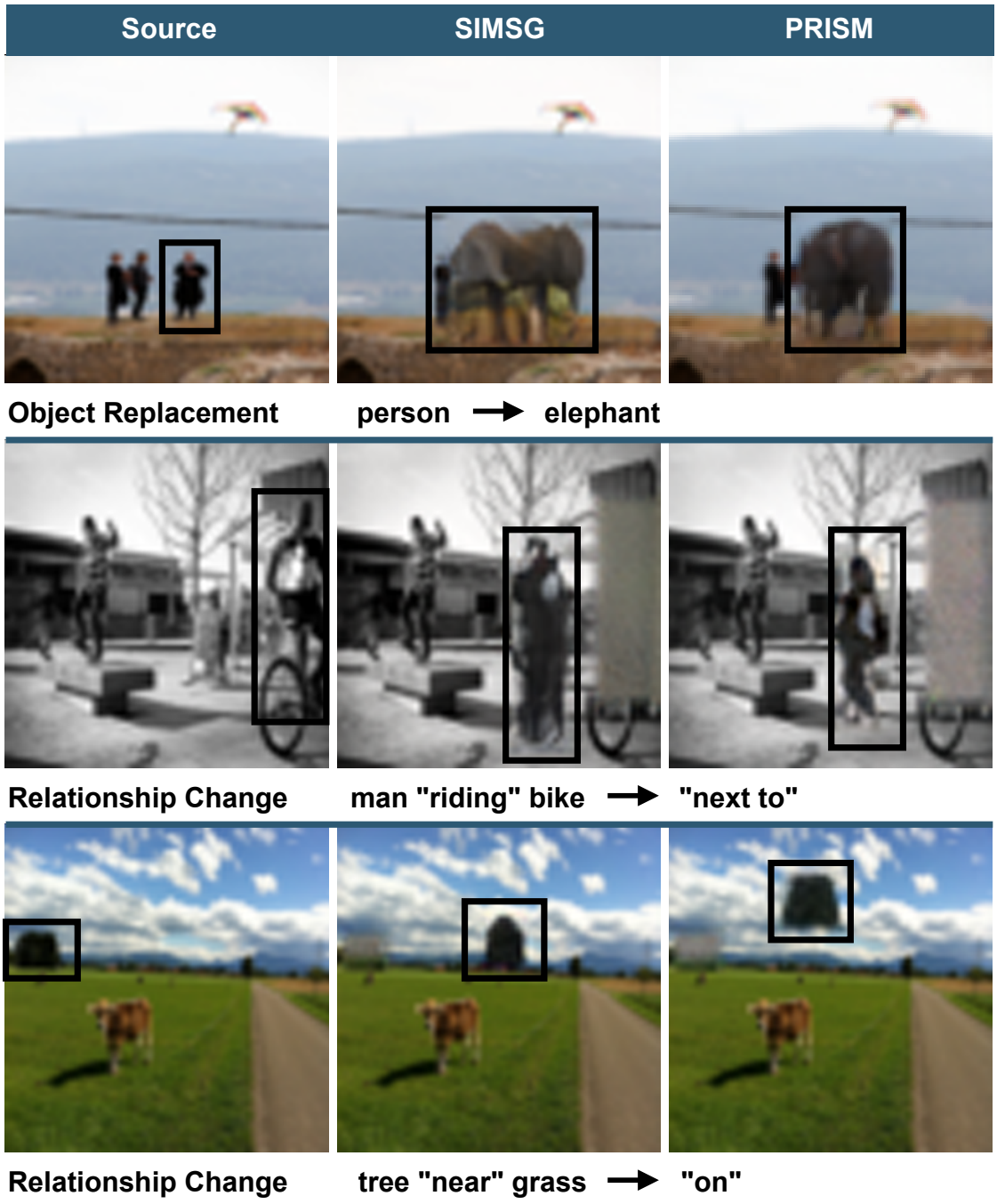


Figure A.2.: Qualitative comparison between SIMSG [12] and PRISM on VG [88] dataset

List of Figures

List of Tables

Bibliography

[1] A. Rosenfeld. "Picture Processing by Computer". In: *ACM Comput. Surv.* 1.3 (July 1969), pp. 147–176. ISSN: 0360-0300. DOI: 10.1145/356551.356554. URL: https://doi.org/10.1145/356551.356554.

[2] Y. LeCun, Y. Bengio, and G. Hinton. "Deep learning". In: *Nature* 521.7553 (May 2015), pp. 436–444. DOI: 10.1038/nature14539. URL: https://doi.org/10.1038/nature14539.

[3] Y. Bengio. "Learning Deep Architectures for AI". In: *Found. Trends Mach. Learn.* 2.1 (Jan. 2009), pp. 1–127. ISSN: 1935-8237. DOI: 10.1561/2200000006. URL: https://doi.org/10.1561/2200000006.

[4] E. G. Tabak and E. Vanden-Eijnden. "Density estimation by dual ascent of the log-likelihood". In: *Communications in Mathematical Sciences* 8.1 (2010), pp. 217–233. DOI: 10.4310/cms.2010.v8.n1.a11. URL: https://doi.org/10.4310/cms.2010.v8.n1.a11.

[5] D. P. Kingma and M. Welling. "Auto-Encoding Variational Bayes". In: *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. Ed. by Y. Bengio and Y. LeCun. 2014. URL: http://arxiv.org/abs/1312.6114.

[6] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. "Generative adversarial networks". In: *Communications of the ACM* 63.11 (2020), pp. 139–144.

[7] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. "Deep Unsupervised Learning using Nonequilibrium Thermodynamics". In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by F. Bach and D. Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, July 2015, pp. 2256–2265.

[8] A. Radford, L. Metz, and S. Chintala. *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*. 2015. eprint: arXiv:1511.06434.

[9] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. "Improved techniques for training gans". In: *Advances in neural information processing systems* 29 (2016).

[10] J. Johnson, A. Alahi, and L. Fei-Fei. "Perceptual losses for real-time style transfer and super-resolution". In: *European conference on computer vision*. Springer. 2016, pp. 694–711.

[11] S. Hong, X. Yan, T. S. Huang, and H. Lee. "Learning Hierarchical Semantic Image Manipulation through Structured Representations". In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper/2018/file/602d1305678a8d5fdb372271e980da6a-Paper.pdf.

[12] H. Dhamo, A. Farshad, I. Laina, N. Navab, G. D. Hager, F. Tombari, and C. Rupprecht. "Semantic Image Manipulation Using Scene Graphs". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[13] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. "Context encoders: Feature learning by inpainting". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2536–2544.

[14] H. Dhamo, K. Tateno, I. Laina, N. Navab, and F. Tombari. "Peeking behind objects: Layered depth prediction from a single image". In: *Pattern Recognition Letters* 125 (2019), pp. 333–340.

[15] C.-A. Yang, C.-Y. Tan, W.-C. Fan, C.-F. Yang, M.-L. Wu, and Y.-C. F. Wang. "Scene Graph Expansion for Semantics-Guided Image Outpainting". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2022, pp. 15617–15626.

[16] X. Wang and A. Gupta. "Generative Image Modeling Using Style and Structure Adversarial Networks". In: *Computer Vision – ECCV 2016*. Springer International Publishing, 2016, pp. 318–335. DOI: 10.1007/978-3-319-46493-0_20. URL: https://doi.org/10.1007/978-3-319-46493-0_20.

[17] E. Ntavelis, A. Romero, I. Kastanis, L. V. Gool, and R. Timofte. "SESAME: Semantic Editing of Scenes by Adding, Manipulating or Erasing Objects". In: *Computer Vision – ECCV 2020*. Springer International Publishing, 2020, pp. 394–411. DOI: 10.1007/978-3-030-58542-6_24. URL: https://doi.org/10.1007/978-3-030-58542-6_24.

[18] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen. "Hierarchical text-conditional image generation with clip latents". In: *arXiv preprint arXiv:2204.06125* (2022).

[19] A. Farshad, Y. Yeganeh, H. Dhamo, F. Tombari, and N. Navab. *DisPositioNet: Disentangled Pose and Identity in Semantic Image Manipulation*. 2022. DOI: 10.48550/ARXIV.2211.05499. URL: https://arxiv.org/abs/2211.05499.

[20] J. Johnson, R. Krishna, M. Stark, L.-J. Li, D. Shamma, M. Bernstein, and L. Fei-Fei. "Image retrieval using scene graphs". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3668–3678.

[21] X. Chang, P. Ren, P. Xu, Z. Li, X. Chen, and A. G. Hauptmann. "A Comprehensive Survey of Scene Graphs: Generation and Application". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021), pp. 1–1. DOI: 10.1109/TPAMI.2021.3137605.

[22] P. Jahoda, A. Vobecky, J. Cech, and J. Matas. "Detecting Decision Ambiguity from Facial Images". In: *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*. 2018, pp. 499–503. DOI: 10.1109/FG.2018.00080.

[23] M. Sulc, L. Picek, and J. Matas. "Plant Recognition by Inception Networks with Test-time Class Prior Estimation." In: *CLEF (Working Notes)*. 2018.

[24] T. Mikolov, K. Chen, G. Corrado, and J. Dean. "Efficient Estimation of Word Representations in Vector Space". In: *Proceedings of Workshop at ICLR* 2013 (Jan. 2013).

[25] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. In: *Proceedings of the 2019 Conference of the North*. Association for Computational Linguistics, 2019. DOI: 10.18653/v1/n19-1423. URL: https://doi.org/10.18653/v1/n19-1423.

[26] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli. "wav2vec 2.0: A framework for self-supervised learning of speech representations". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 12449–12460.

[27] P. Bachman, R. D. Hjelm, and W. Buchwalter. "Learning representations by maximizing mutual information across views". In: *Advances in neural information processing systems* 32 (2019).

[28] D. Stavens and S. Thrun. "A Self-Supervised Terrain Roughness Estimator for Off-Road Autonomous Driving". In: (June 2012).

[29] A. Karpathy, P. Abbeel, G. Brockman, R. Duan, V. Cheung, P. Chen, I. Goodfellow, D. Kingma, J. Ho, R. Houthooft, T. Salimans, J. Schulman, I. Sutskever, and W. Zaremba. *Generative Models*. 2016. URL: https://openai.com/blog/generative-models/.

[30] D. Rezende and S. Mohamed. "Variational Inference with Normalizing Flows". In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by F. Bach and D. Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, July 2015, pp. 1530–1538. URL: https://proceedings.mlr.press/v37/rezende15.html.

[31] J. P. Agnelli, M. Cadeiras, E. G. Tabak, C. V. Turner, and E. Vanden-Eijnden. "Clustering and Classification through Normalizing Flows in Feature Space". In: *Multiscale Modeling and Simulation* 8.5 (Jan. 2010), pp. 1784–1802. DOI: 10.1137/100783522. URL: https://doi.org/10.1137/100783522.

[32] I. Kobyzev, S. J. Prince, and M. A. Brubaker. "Normalizing Flows: An Introduction and Review of Current Methods". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.11 (Nov. 2021), pp. 3964–3979. DOI: 10.1109/tpami.2020.2992934. URL: https://doi.org/10.1109/tpami.2020.2992934.

[33] D. P. Kingma and P. Dhariwal. "Glow: Generative Flow with Invertible 1x1 Convolutions". In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper/2018/file/d139db6a236200b21cc7f752979132d0-Paper.pdf.

[34] J. Ho, X. Chen, A. Srinivas, Y. Duan, and P. Abbeel. "Flow++: Improving Flow-Based Generative Models with Variational Dequantization and Architecture Design". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by K. Chaudhuri and R. Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, June 2019, pp. 2722–2730.

[35] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks". In: *Computer Vision (ICCV), 2017 IEEE International Conference on*. 2017.

[36] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. "Generative Adversarial Text to Image Synthesis". In: *Proceedings of The 33rd International Conference on Machine Learning*. Ed. by M. F. Balcan and K. Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, June 2016, pp. 1060–1069. URL: https://proceedings.mlr.press/v48/reed16.html.

[37] S. E. Reed, Z. Akata, S. Mohan, S. Tenka, B. Schiele, and H. Lee. "Learning What and Where to Draw". In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett. Vol. 29. Curran Associates, Inc., 2016. URL: https://proceedings.neurips.cc/paper/2016/file/a8f15eda80c50adb0e71943adc8015cf-Paper.pdf.

[38] D. Pathak, P. Krähenbühl, J. Donahue, T. Darrell, and A. Efros. "Context Encoders: Feature Learning by Inpainting". In: *Computer Vision and Pattern Recognition (CVPR)*. 2016.

[39] E. Denton, S. Gross, and R. Fergus. *Semi-Supervised Learning with Context-Conditional Generative Adversarial Networks*. 2016. DOI: 10.48550/ARXIV.1611.06430. URL: https://arxiv.org/abs/1611.06430.

[40] H. Wu, S. Zheng, J. Zhang, and K. Huang. "GP-GAN". In: *Proceedings of the 27th ACM International Conference on Multimedia*. ACM, Oct. 2019. DOI: 10.1145/3343031.3350944. URL: https://doi.org/10.1145/3343031.3350944.

[41] C. Saharia, W. Chan, H. Chang, C. Lee, J. Ho, T. Salimans, D. Fleet, and M. Norouzi. "Palette: Image-to-Image Diffusion Models". In: *Special Interest Group on Computer Graphics and Interactive Techniques Conference Proceedings*. ACM, Aug. 2022. DOI: 10.1145/3528233.3530757. URL: https://doi.org/10.1145/3528233.3530757.

[42] M. Mirza and S. Osindero. *Conditional Generative Adversarial Nets*. 2014. eprint: arXiv:1411.1784.

[43] M. D. Zeiler, G. W. Taylor, and R. Fergus. "Adaptive deconvolutional networks for mid and high level feature learning". In: *2011 International Conference on Computer Vision*. IEEE, Nov. 2011. DOI: 10.1109/iccv.2011.6126474. URL: https://doi.org/10.1109/iccv.2011.6126474.

[44] F.-A. Croitoru, V. Hondru, R. T. Ionescu, and M. Shah. "Diffusion models in vision: A survey". In: *arXiv preprint arXiv:2209.04747* (2022).

[45] J. Ho, A. Jain, and P. Abbeel. "Denoising Diffusion Probabilistic Models". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 6840–6851. URL: https://proceedings.neurips.cc/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf.

[46] W. Feller. "On the theory of stochastic processes, with particular reference to applications". In: *Berkeley Symposium on Mathematical Statistics and Probability*. 1949.

[47] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. Denton, S. K. S. Ghasemipour, B. K. Ayan, S. S. Mahdavi, R. G. Lopes, et al. "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding". In: *Advances in Neural Information Processing Systems*. 2022.

[48] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever. "Zero-shot text-to-image generation". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 8821–8831.

[49] A. Lugmayr, M. Danelljan, A. Romero, F. Yu, R. Timofte, and L. Van Gool. "RePaint: Inpainting Using Denoising Diffusion Probabilistic Models". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2022, pp. 11461–11471.

[50] A. Nichol, P. Dhariwal, A. Ramesh, P. Shyam, P. Mishkin, B. McGrew, I. Sutskever, and M. Chen. "Glide: Towards photorealistic image generation and editing with text-guided diffusion models". In: *International Conference on Machine Learning*. 2022.

[51] T. N. Kipf and M. Welling. "Semi-Supervised Classification with Graph Convolutional Networks". In: *International Conference on Learning Representations*. 2017. URL: https://openreview.net/forum?id=SJU4ayYgl.

[52] J. Bruna, W. Zaremba, A. Szlam, and Y. Lecun. "Spectral Networks and Locally Connected Networks on Graphs". In: *International Conference on Learning Representations*. 2014.

[53] M. Defferrard, X. Bresson, and P. Vandergheynst. "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering". In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett. Vol. 29. Curran Associates, Inc., 2016. URL: https://proceedings.

neurips.cc/paper/2016/file/04df4d434d481c5bb723be1b6df1ee65-Paper.pdf.

[54] T. N. Kipf and M. Welling. "Semi-Supervised Classification with Graph Convolutional Networks". In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL: https://openreview.net/forum?id=SJU4ayYgl.

[55] D. Marcheggiani and I. Titov. "Encoding Sentences with Graph Convolutional Networks for Semantic Role Labeling". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2017. DOI: 10.18653/v1/d17-1159. URL: https://doi.org/10.18653/v1/d17-1159.

[56] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu. "A Comprehensive Survey on Graph Neural Networks". In: *IEEE Transactions on Neural Networks and Learning Systems* 32.1 (2021), pp. 4–24. DOI: 10.1109/TNNLS.2020.2978386.

[57] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. "Neural Message Passing for Quantum Chemistry". In: ICML'17. Sydney, NSW, Australia: JMLR.org, 2017, pp. 1263–1272.

[58] D. Chen, Y. Lin, W. Li, P. Li, J. Zhou, and X. Sun. "Measuring and Relieving the Over-Smoothing Problem for Graph Neural Networks from the Topological View". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.04 (Apr. 2020), pp. 3438–3445. DOI: 10.1609/aaai.v34i04.5747. URL: https://doi.org/10.1609/aaai.v34i04.5747.

[59] U. Alon and E. Yahav. "On the Bottleneck of Graph Neural Networks and its Practical Implications". In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=i80OPhOCVH2.

[60] J. Gasteiger, A. Bojchevski, and S. Günnemann. "Combining Neural Networks with Personalized PageRank for Classification on Graphs". In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=H1gL-2A9Ym.

[61] S. Schuster, R. Krishna, A. Chang, L. Fei-Fei, and C. D. Manning. "Generating semantically precise scene graphs from textual descriptions for improved image retrieval". In: *Proceedings of the fourth workshop on vision and language*. 2015, pp. 70–80.

[62] J. Wald, H. Dhamo, N. Navab, and F. Tombari. "Learning 3D Semantic Scene Graphs From 3D Indoor Reconstructions". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[63] R. Wang, Z. Wei, P. Li, Q. Zhang, and X. Huang. "Storytelling from an Image Stream Using Scene Graphs". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.05 (Apr. 2020), pp. 9185–9192. DOI: 10.1609/aaai.v34i05.6455. URL: https://doi.org/10.1609/aaai.v34i05.6455.

[64] J. Johnson, A. Gupta, and L. Fei-Fei. "Image generation from scene graphs". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 1219–1228.

[65] S. Garg, H. Dhamo, A. Farshad, S. Musatian, N. Navab, and F. Tombari. "Unconditional Scene Graph Generation". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2021, pp. 16362–16371.

[66] S. Hong, D. Yang, J. Choi, and H. Lee. "Inferring semantic layout for hierarchical text-to-image synthesis". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 7986–7994.

[67] J. Ma, P. Cui, K. Kuang, X. Wang, and W. Zhu. "Disentangled graph convolutional networks". In: *International conference on machine learning*. PMLR. 2019, pp. 4212–4221.

[68] Y. Yang, Z. Feng, M. Song, and X. Wang. "Factorizable Graph Convolutional Networks". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 20286–20296. URL: https://proceedings.neurips.cc/paper/2020/file/ea3502c3594588f0e9d5142f99c66627-Paper.pdf.

[69] H. Zhang, Z. Hu, C. Luo, W. Zuo, and M. Wang. "Semantic image inpainting with progressive generative networks". In: *Proceedings of the 26th ACM international conference on Multimedia*. 2018, pp. 1939–1947.

[70] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. "Learning transferable visual models from natural language supervision". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 8748–8763.

[71] F. Yu and V. Koltun. "Multi-Scale Context Aggregation by Dilated Convolutions". In: *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. Ed. by Y. Bengio and Y. LeCun. 2016. URL: http://arxiv.org/abs/1511.07122.

[72] T. Park, M.-Y. Liu, T.-C. Wang, and J.-Y. Zhu. "Semantic Image Synthesis With Spatially-Adaptive Normalization". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[73] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro. "High-resolution image synthesis and semantic manipulation with conditional gans". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 8798–8807.

[74] S. Woo, D. Kim, D. Cho, and I. S. Kweon. "LinkNet: Relational Embedding for Scene Graph". In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper/2018/file/58238e9ae2dd305d79c2ebc8c1883422-Paper.pdf.

[75] D. Xu, Y. Zhu, C. B. Choy, and L. Fei-Fei. "Scene Graph Generation by Iterative Message Passing". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017.

[76] H. Liu, N. Yan, M. Mortazavi, and B. Bhanu. "Fully Convolutional Scene Graph Generation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, pp. 11546–11556.

[77] R. Herzig, A. Bar, H. Xu, G. Chechik, T. Darrell, and A. Globerson. "Learning canonical representations for scene graph to image generation". In: *European Conference on Computer Vision*. Springer. 2020, pp. 210–227.

[78] Q. Chen and V. Koltun. "Photographic image synthesis with cascaded refinement networks". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 1511–1520.

[79] B. Zhao, L. Meng, W. Yin, and L. Sigal. "Image generation from layout". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 8584–8593.

[80] G. Mittal, S. Agrawal, A. Agarwal, S. Mehta, and T. Marwah. "Interactive Image Generation Using Scene Graphs". In: *CoRR* abs/1905.03743 (2019). URL: http://arxiv.org/abs/1905.03743.

[81] S. Tripathi, A. Bhiwandiwalla, A. Bastidas, and H. Tang. "Using scene graph context to improve image generation". In: *arXiv preprint arXiv:1901.03762* (2019).

[82] Y. Li, T. Ma, Y. Bai, N. Duan, S. Wei, and X. Wang. "Pastegan: A semi-parametric method to generate image from scene graph". In: *Advances in Neural Information Processing Systems* 32 (2019).

[83] A. Farshad, S. Musatian, H. Dhamo, and N. Navab. "MIGS: Meta Image Generation from Scene Graphs". In: *BMVC*. 2021.

[84] Y. Li, W. Ouyang, B. Zhou, J. Shi, C. Zhang, and X. Wang. "Factorizable Net: An Efficient Subgraph-based Framework for Scene Graph Generation". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Sept. 2018.

[85] K. Simonyan and A. Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *arXiv 1409.1556* (Sept. 2014).

[86] Y. Cong, W. Liao, H. Ackermann, B. Rosenhahn, and M. Y. Yang. "Spatial-temporal transformer for dynamic scene graph generation". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 16372–16382.

[87] 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: *European conference on computer vision*. Springer. 2014, pp. 740–755.

[88] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D. A. Shamma, et al. "Visual genome: Connecting language and vision using crowdsourced dense image annotations". In: *International journal of computer vision* 123.1 (2017), pp. 32–73.

[89] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester. "Image inpainting". In: *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. 2000, pp. 417–424.

[90] O. Elharrouss, N. Almaadeed, S. Al-Maadeed, and Y. Akbari. "Image Inpainting: A Review". In: *Neural Processing Letters* 51.2 (Dec. 2019), pp. 2007–2028. DOI: 10.1007/s11063-019-10163-0. URL: https://doi.org/10.1007/s11063-019-10163-0.

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

[92] J. Shade, S. Gortler, L.-w. He, and R. Szeliski. "Layered depth images". In: *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*. 1998, pp. 231–242.

[93] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. "Image-to-image translation with conditional adversarial networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1125–1134.

[94] Z. Guo, Z. Chen, T. Yu, J. Chen, and S. Liu. "Progressive image inpainting with full-resolution residual network". In: *Proceedings of the 27th acm international conference on multimedia*. 2019, pp. 2496–2504.

[95] A. Li, A. Jabri, A. Joulin, and L. Van Der Maaten. "Learning visual n-grams from web data". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 4183–4192.

[96] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

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

[98] P. Dhariwal and A. Nichol. "Diffusion Models Beat GANs on Image Synthesis". In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan. Vol. 34. Curran Associates, Inc., 2021, pp. 8780–8794. URL: https://proceedings.neurips.cc/paper/2021/file/49ad23d1ec9fa4bd8d77d02681df5cfa-Paper.pdf.

[99] J. Ho and T. Salimans. "Classifier-Free Diffusion Guidance". In: *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*. 2021. URL: https://openreview.net/forum?id=qw8AKxfYbI.

[100] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. "Gans trained by a two time-scale update rule converge to a local nash equilibrium". In: *Advances in neural information processing systems* 30 (2017).

[101] S. Ren, K. He, R. Girshick, and J. Sun. "Faster r-cnn: Towards real-time object detection with region proposal networks". In: *Advances in neural information processing systems* 28 (2015).

[102] S. Ioffe and C. Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *International conference on machine learning*. PMLR. 2015, pp. 448–456.

[103] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro. "High-resolution image synthesis and semantic manipulation with conditional gans". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 8798–8807.

[104] A. Odena, C. Olah, and J. Shlens. "Conditional image synthesis with auxiliary classifier gans". In: *International conference on machine learning*. PMLR. 2017, pp. 2642–2651.

[105] L. I. Rudin, S. Osher, and E. Fatemi. "Nonlinear total variation based noise removal algorithms". In: *Physica D: nonlinear phenomena* 60.1-4 (1992), pp. 259–268.

[106] R. Caruana. "Multitask learning". In: *Machine learning* 28.1 (1997), pp. 41–75.

[107] Y. Zhang and Q. Yang. "An overview of multi-task learning". In: *National Science Review* 5.1 (2018), pp. 30–43.

[108] T. Martyniuk. "Multi-task learning for image restoration". MA thesis. Ukraine: Ukrainian Catholic University, 2019.

[109] J. Johnson, B. Hariharan, L. Van Der Maaten, L. Fei-Fei, C. Lawrence Zitnick, and R. Girshick. "Clevr: A diagnostic dataset for compositional language and elementary visual reasoning". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2901–2910.

[110] 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: *European conference on computer vision*. Springer. 2014, pp. 740–755.

[111] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. "Image quality assessment: from error visibility to structural similarity". In: *IEEE transactions on image processing* 13.4 (2004), pp. 600–612.

[112] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. "The Unreasonable Effectiveness of Deep Features as a Perceptual Metric". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018.

[113] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "Imagenet classification with deep convolutional neural networks". In: *Communications of the ACM* 60.6 (2017), pp. 84–90.

[114] S. Barratt and R. Sharma. "A note on the inception score". In: *arXiv preprint arXiv:1801.01973* (2018).

[115] M. Fréchet. "Sur la distance de deux lois de probabilité". In: *Comptes Rendus Hebdomadaires des Seances de L Academie des Sciences* 244.6 (1957), pp. 689–692.

[116] L. N. Vaserstein. "Markov processes over denumerable products of spaces, describing large systems of automata". In: *Problemy Peredachi Informatsii* 5.3 (1969), pp. 64–72.

[117] M. Seitzer. *pytorch-fid: FID Score for PyTorch*. `https://github.com/mseitzer/pytorch-fid`. Version 0.2.1. Aug. 2020.

[118] B. Zoph, G. Ghiasi, T.-Y. Lin, Y. Cui, H. Liu, E. D. Cubuk, and Q. Le. "Rethinking pre-training and self-training". In: *Advances in neural information processing systems* 33 (2020), pp. 3833–3845.