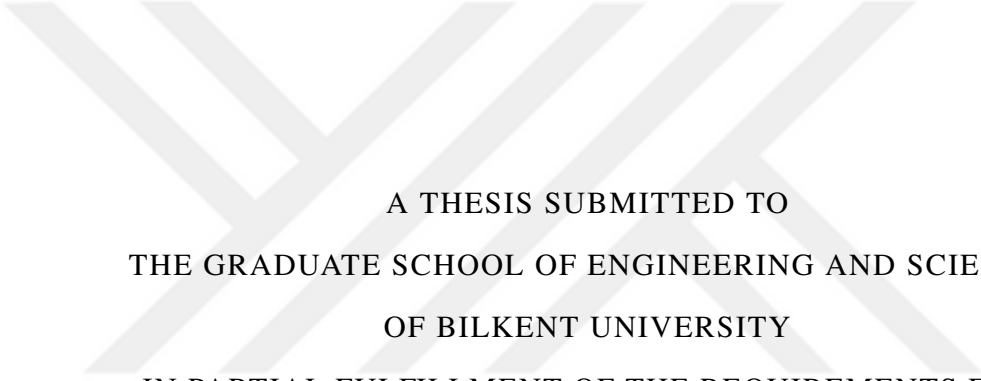


IMAGE INPAINTING WITH DIFFUSION MODELS AND GENERATIVE ADVERSARIAL NETWORKS



A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF
MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

By
Ahmet Burak Yıldırım
May 2024

Image Inpainting with Diffusion Models and Generative Adversarial
Networks

By Ahmet Burak Yıldırım

May 2024

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Ayşegül Dünder Boral (Advisor)

Hamdi Dibekliolu

Mehmet Erkut Erdem

Approved for the Graduate School of Engineering and Science:

Orhan Arıkan
Director of the Graduate School

ABSTRACT

IMAGE INPAINTING WITH DIFFUSION MODELS AND GENERATIVE ADVERSARIAL NETWORKS

Ahmet Burak Yıldırım

M.S. in Computer Engineering

Advisor: Ayşegül Dünder Boral

May 2024

We present two novel approaches to image inpainting, a task that involves erasing unwanted pixels from images and filling them in a semantically consistent and realistic way. The first approach uses natural language input to determine which object to remove from an image. We construct a dataset named GQA-Inpaint for this task and train a diffusion-based inpainting model on it, which can remove objects from images based on text prompts. The second approach tackles the challenging task of inverting erased images into StyleGAN’s latent space for realistic inpainting and editing. For this task, we propose learning an encoder and a mixing network to combine encoded features of erased images with StyleGAN’s mapped features from random samples. To achieve diverse inpainting results for the same erased image, we combine the encoded features and randomly sampled style vectors via the mixing network. We compare our methods with different evaluation metrics that measure the quality of the models and show significant quantitative and qualitative improvements.

Keywords: Image inpainting, diffusion models, generative adversarial networks, instruction-based inpainting, image editing.

ÖZET

DİFÜZYON MODELLERİ VE ÇEKİŞMELİ ÜRETİCİ AĞLAR İLE GÖRÜNTÜ TAMAMLAMA

Ahmet Burak Yıldırım

Bilgisayar Mühendisliği, Yüksek Lisans

Tez Danışmanı: Ayşegül Dünder Boral

Mayıs 2024

Görüntü tamamlama, görüntülerden istenmeyen pikselleri silmeyi ve bunları anlamsal olarak tutarlı ve gerçekçi bir şekilde doldurmayı içeren bir görevdir. Bu göreve yönelik iki yeni yaklaşım sunuyoruz. İlk yaklaşım, bir görüntüden hangi nesnenin çıkarılacağını belirlemek için doğal dil girdisi kullanıyor. Bu görev için GQA-Inpaint adlı bir veri seti oluşturuyor ve bu veri seti üzerinde metin komutuna dayalı olarak görüntülerden nesneleri çıkarabilen bir difüzyon tabanlı görüntü tamamlama modeli eğitiyoruz. İkinci yaklaşım, gerçekçi görüntü tamamlama ve düzenleme amacıyla silinmiş görüntüleri StyleGAN'ın gizli uzayına geri döndürmek gibi zorlu bir görevi başarıyor. Bu görev için, kodlanmış olan silinmiş görüntü özelliklerini rastgele örneklerden elde edilen StyleGAN'ın stil vektörleri ile birleştirmek amacıyla bir kodlayıcı ve bir karıştırma ağı öğrenmeyi öneriyoruz. Aynı silinmiş resim için çeşitli görüntü tamamlama sonuçları elde etmek adına, kodlanmış özellikleri ve rastgele örneklenen stil vektörlerini karıştırma ağı aracılığıyla birleştiriyoruz. Yöntemlerimizi modellerin kalitesini ölçen farklı değerlendirme metrikleriyle karşılaştırıyoruz ve önemli niceliksel ve niteliksel iyileştirmeler gösteriyoruz.

Anahtar sözcükler: Görüntü tamamlama, difüzyon modelleri, çekişmeli üretici ağılar, talimata dayalı görüntü tamamlama, görüntü düzenleme.

Acknowledgement

First and foremost, I would like to express my deepest gratitude to my advisor, Asst. Prof. Ayşegül Dünder Boral, for always being available to help whenever I needed it. Her continuous support and vast knowledge allowed me to overcome the hard times in my research process. She proved to us that we can achieve success in publishing papers in top-tier conferences and journals. Inspired by her dedication, I will strive to follow her path and aim to publish high-impact papers in the future.

Additionally, I want to thank Asst. Prof. Hamdi Dibekliolu and Prof. Mehmet Erkut Erdem for accepting to be on my thesis jury. Their time, effort, and valuable feedback are greatly appreciated.

My sincere thanks also go to my parents, Bülent and Engin Yıldırım, for their support throughout my life. They have always believed in me and encouraged me to pursue my dreams. This achievement would not have been possible without them.

Furthermore, I am grateful to Prof. Mehmet Erkut Erdem, Prof. İbrahim Aykut Erdem, and Dr. Duygu Ceylan who, along with Asst. Prof. Ayşegül Dünder, provided me with the opportunity to join their projects and offered invaluable advice and support during my research.

Last but not least, I would like to thank the members of Generative DLR Lab, Hamza Pehlivan, Mustafa Utku Aydoğdu, Yiğit Ekin, Yusuf Dalva, and Bahri Batuhan Bilecen. Their friendship, feedback, and support made this journey enjoyable and fulfilling.

Contents

1	Introduction	1
1.1	Inst-Inpaint	1
1.2	DivInversion	4
2	Related Work	7
2.1	Image Inpainting	7
2.2	Diverse Image Inpainting	8
2.3	Generative Adversarial Network	8
2.4	StyleGAN	9
2.5	Finding Edit Directions in W Space	11
2.6	Real Image Editing via StyleGAN Inversion	13
2.7	Denoising Diffusion Probabilistic Model	15
2.8	Latent Diffusion Model	16
2.9	Diffusion-based Generative Models	17

3	Inst-Inpaint: Instructing to Remove Objects with Diffusion Models	18
3.1	Dataset Generation	18
3.2	Method	25
3.3	Experiments	28
3.3.1	Baselines	28
3.3.2	Datasets	29
3.3.3	Evaluation Metrics	30
3.3.4	Experimental Analysis	32
3.3.5	Limitations	39
4	DivInversion: Diverse Inpainting and Editing with GAN Inversion	40
4.1	Method	40
4.1.1	First Stage: Diverse Results	40
4.1.2	Second Stage: High-Fidelity Reconstruction	44
4.2	Experiments	45
5	Conclusion	55

List of Figures

1.1	Model teaser of Inst-Inpaint	3
1.2	Framework teaser of DivInversion	5
2.1	The architecture of Generative Adversarial Network	9
2.2	Simplified representation of the StyleGAN architecture	10
2.3	Visualization of feature disentanglement in the style space	10
2.4	Image attribute editing in the style space	12
2.5	StyleGAN inversion with pSp model	14
2.6	Forward and backward diffusion processes of DDPM	15
2.7	Architecture of Latent Diffusion Model	16
3.1	Architecture of the Inst-Inpaint model	19
3.2	A sample image from the GQA dataset and the corresponding scene graph	20
3.3	Distribution of the relation types exist in the GQA-Inpaint dataset	22

3.4	Comparison of mask extraction methods	23
3.5	Intermediate steps of constructing GQA-Inpaint	24
3.6	Distribution of the object classes exist in the GQA-Inpaint dataset	25
3.7	CLIP-based image inpainting scores	30
3.8	Qualitative results of Inst-Inpaint and the competing methods on the GQA-Inpaint dataset	32
3.9	Image inpainting results of Inst-Inpaint along with the average cross-attention maps	34
3.10	Qualitative results of segmentation analysis on Inst-Inpaint and the competing methods	36
3.11	Improvement on Inst-Inpaint reconstruction via weight optimization	36
3.12	Iterative object removal results of Inst-Inpaint	37
3.13	Qualitative results of Inst-Inpaint and the competing methods on the CLEVR dataset	38
4.1	Architecture of the DivInversion model	41
4.2	Two-path loss proposed for training the DivInversion model	42
4.3	FID plots of DivInversion and the competing methods on the FFHQ dataset	48
4.4	Qualitative results of DivInversion and the competing methods on the FFHQ dataset	50
4.5	Qualitative results of DivInversion on AFHQ-Cat, AFHQ-Dog, and Stanford Cars datasets	51

4.6	Qualitative results of DivInversion ablation study	52
4.7	Editing results of DivInversion on the FFHQ dataset	54



List of Tables

3.1	Hyperparameters for the Inst-Inpaint model	27
3.2	Quantitative results of Inst-Inpaint and competing methods	33
3.3	Segmentation scores (IoU) of Inst-Inpaint mask prediction and competing methods	35
3.4	Quantitative results of Inst-Inpaint and competing methods on the CLEVR dataset	37
4.1	Quantitative results of DivInversion and the competing methods on the FFHQ dataset	47
4.2	Quantitative results of DivInversion and the competing methods on AFHQ-Dog, AFHQ-Cat, and Stanford Cars datasets	49
4.3	Quantitative results of DivInversion ablation study on the FFHQ dataset	53

Chapter 1

Introduction

Image inpainting refers to the task of removing unwanted objects and/or filling in missing regions within an image [1, 2, 3]. We propose two novel models named Inst-Inpaint and DivInversion that focus on different goals of the image inpainting task. Inst-Inpaint is a diffusion-based object removal model that is conditioned with a text prompt instead of a traditional object mask. It finds the object referred to by the text prompt and removes it from the scene. In contrast, DivInversion is a GAN-based method that inpaints an image via GAN inversion. With this approach, the inpainted areas can be edited by manipulating the style vector found in StyleGAN’s organized latent space. Furthermore, our model can inpaint the erased parts of the image diversely. In the following sections, we detail the objectives and contributions of these proposed models.

1.1 Inst-Inpaint

Image inpainting task is considered highly challenging as the inpainted regions in the resulting images should be in harmony with the rest of the images. Inpainting approaches should take into account the global image context while filling in the missing pixels. Traditionally, existing studies use binary masks to highlight the missing regions. As in many image enhancement tasks, the recent image inpainting approaches

utilize deep-learning-based frameworks equipped with generative priors and formulate the inpainting task as a supervised learning problem [4, 5, 6]. That is, training data includes pairs of input and erased images acting as source and target data, respectively, and models are generally trained using a reconstruction-based objective. Although we have seen continuous improvements in the performance of inpainting approaches over the past years, these methods still require user-specified masks provided at inference time. These masks can be provided in terms of thin or thick brush strokes or squares, but drawing these masks is often tedious and prone to errors.

Recently, text-based image generation and editing have gained a lot of attention [7, 8]. Especially when trained on large-scale image-text data, models like DALL·E2 [9], Stable Diffusion [10], Imagen [11] and Parti [12] show exceptional generative capabilities while effectively capturing the compositional structure of images. Moreover, using text enables a more natural and user-friendly control over the images. The researchers further showcased the versatility of these methodologies by showcasing their applicability to image inpainting. This was achieved through additional conditioning of the models on binary masks, thereby instructing them on what to fill within the masked regions. However, it’s worth noting that while this approach excels in manipulating object attributes or introducing new elements into a scene, its efficacy in object removal may be limited. For a concrete example, refer to the Blended Latent Diffusion model [13].

In this work, we introduce a dataset and framework for a task that we refer to as *instructional image inpainting*, in which the unwanted objects are specified only via textual instructions, without any need for binary masks (see Fig. 1.1).



Figure 1.1: **Instructional Image Inpainting.** We propose a new image inpainting method, **Inst-Inpaint**, which takes an image and a textual instruction as input and is able to automatically remove objects mentioned in the text, without any need for a user-supplied binary mask to locate the region of interest, as in traditional inpainting approaches. We show sample results of our method trained and tested on a new real image inpainting benchmark dataset, **GQA-Inpaint**, which we created for the proposed instructional image inpainting task.

To tackle this problem, by building upon the GQA dataset [14], we first use a data generation pipeline to create a new benchmark dataset, which we call GQA-Inpaint. Second, we design a single-stage deep inpainting network, Inst-Inpaint, that can remove objects from images using instructions given as text prompts. Compared to existing inpainting methods, our model does not require masks, and neither do we predict masks explicitly. In summary, our contributions are as follows:

- We propose an end-to-end image inpainting framework, Inst-Inpaint, that can perform the removal of an object in a given image based solely on textual instructions. Some example outputs of our method are shown in Fig. 1.1.
- We generate a real image dataset, GQA-Inpaint, to train and evaluate models for the proposed instructional image inpainting task. Here, we use the GQA dataset [14] originally proposed for visual reasoning, and exploit the provided scene graphs to generate paired training data by utilizing state-of-the-art instance segmentation and inpainting methods.

- We conduct extensive experiments to show the effectiveness of our framework. To compare the results, we evaluate various baselines using several metrics, which include a novel CLIP [15]-based inpainting score. We achieve significant improvements over the state-of-the-art text-based image inpainting methods.

1.2 DivInversion

Generative Adversarial Networks (GANs) achieve impressive results on unconditional [16, 17, 18, 19] and conditional image generation [20, 21, 5], inpainting [4, 22, 23], and image editing tasks [24, 25, 26]. Traditionally, each of these tasks has been explored with a dedicated network and training pipeline. However, recently it has been shown that high-quality image editing can be achieved with well-trained GAN models and especially by StyleGAN networks [16, 18] that are trained to generate images without a condition [27, 28]. This approach achieves numerous edits via semantically rich feature representations of well-trained GANs [29, 30, 31, 25]. In this work, we are interested in taking this direction one step further and learning an encoder for a pretrained GAN that can achieve high-quality image inversion, diverse inpainting, and editing under one framework.

To benefit from GAN’s image editing capabilities, extensive research is conducted on image inversion algorithms to find the corresponding latent codes that will generate a particular real image [32, 33, 34, 27]. It is observed that there exists a trade-off between image reconstruction fidelity and image editing quality [27]. Studies show that low-rate latent spaces (z , W , or W^+ for StyleGANs) are limited in their expressiveness power, and not every image can be inverted with high-fidelity reconstruction to GAN’s natural low-rate latent space [27]. When images are not projected to GAN’s natural latent space, even when an image is reconstructed with high fidelity, it will not be editable. Methods are proposed to skip spatially higher resolution features (higher-rate latent codes) from an encoder to a GAN generator for better reconstruction and editing properties [28, 35]. In this work, we are interested in a more challenging scenario; learning an image encoder that can project an erased image to a well-trained GAN’s natural latent space. This framework provides new capabilities for image editing, as shown in Fig. 1.2.

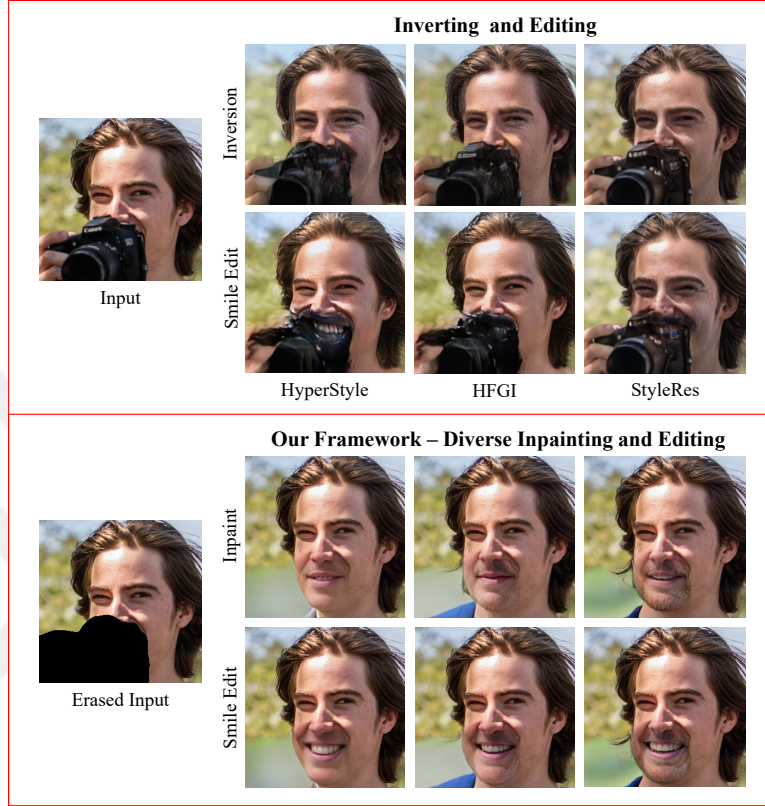


Figure 1.2: Our framework achieves diverse inpainting and editing with GAN inversion. Compared to other works that are proposed to invert and edit images with StyleGAN (HyperStyle [35], HFGI [28], StyleRes [36]), our framework has advantages, as in the presented example.

Recently, GAN inversion-based image inpainting methods have been proposed [6, 37]. These methods optimize an image encoder and learn skip connections to the pretrained StyleGAN model. Even though promising results are achieved, these methods model image inpainting in a deterministic way. They are trained to reconstruct the original image from the erased ones without any stochasticity. They are not regularized to project the image into GAN’s natural latent space and do not enjoy the editing capabilities of them.

In this work, we propose a framework that achieves high-quality image inversion, diverse inpainting, and editing simultaneously. We design an encoder architecture that takes an erased input image and encodes latent codes for the visible parts. We also

design a mixing network that combines randomly sampled latent codes with encoded ones. This setup allows the model to output diverse results. However, we find that the framework learns to ignore the randomly sampled latent codes and has the tendency to output deterministic results. To achieve diversity, we propose to train the framework with a novel design. Our design includes generating images with W^+ and erasing a part of it. During training, sometimes the same W^+ that generated the input image is fed to the mixing network for the whole input reconstruction, and sometimes a newly sampled W^+ is used to reconstruct only the visible part. This way, the network is regularized to use both inputs; erased image and sampled W^+ . Secondly, to achieve high fidelity inversion of unerased pixels and to prevent the color discrepancy between the unerased and erased parts, we learn higher dimension latent codes. In summary, our main contributions are as follows:

- We propose a novel framework for image inpainting with GAN inversion. Our framework includes an encoder to embed images and a mixing network to combine them with randomly sampled latent codes to achieve diversity. The mixing network has a gating mechanism that improves the results.
- To achieve diversity, we propose a novel set-up to train the networks. We use GAN-generated images and train the network with full image reconstruction and valid pixel image reconstruction depending on if we feed the same latent code to the mixing network that is used to generate the input image or not.
- We conduct extensive experiments to show the effectiveness of our framework and achieve significant improvements over state-of-the-art models for image inpainting. Additionally, we show our framework can achieve diverse inpainting and editing under one framework, as shown in Fig. 1.2.

Chapter 2

Related Work

2.1 Image Inpainting

The common set-up for the image inpainting task includes a binary mask that defines the erased pixels [4, 5]. The original image pixels are removed and new ones are generated based on those masks. This domain was previously dominated by Generative Adversarial Networks (GANs) [2, 3, 4, 22, 5, 38]. GAN-based models mostly output deterministic results as these models are trained with reconstruction losses as well for stability [4, 5, 6]. There are also recent methods that achieve diversity with GANs [23, 39]. However, they are trained on single object datasets such as Faces. They are not extended to inpaint diverse scenes because GANs are notoriously difficult to train.

Recently, diffusion-based models also entered this domain with promising results [40, 7]. It is shown that a pretrained unconditional diffusion model can inpaint images by Repaint model [40]. Repaint modifies the denoising process to condition generation on the unerased image content. There are other diffusion-based inpainting works but they are designed to add objects to a scene instead of removing objects such as GLIDE [41] and SmartBrush [42].

2.2 Diverse Image Inpainting

Inpainting missing pixels, especially large-scale holes, receives a significant amount of attention from both the research community and industry. This task requires filling in missing pixels in a semantically meaningful and multi-modal way. However, most of the proposed methods are deterministic and struggle with large-scale holes [2, 3, 4, 22, 5]. Recently, CoModGAN [23] is proposed with an architecture and training pipeline similar to StyleGAN but with skip connections from the encoder to the generator to pass the valid pixels to the generator without a bottleneck. CoModGAN achieves diverse results. Diffusion models have also shown great success in diverse image inpainting [40, 7, 43]. Nevertheless, none of these methods can achieve diverse inpainting and enable user-controlled editing of images with one network simultaneously, which is the goal of our proposed model, DivInversion.

2.3 Generative Adversarial Network

Generative Adversarial Network (GAN) [44] is a neural network architecture that learns how to map randomly sampled data z from a pre-defined distribution $P(Z)$ (commonly selected as Gaussian distribution) to the real data distribution $P(X)$. The architecture of GAN is shown in Figure 2.1. It consists of two neural networks named generator G and discriminator D that train each other during the training phase. This adversarial training process uses the objective L_{GAN} defined in Equation 2.1.

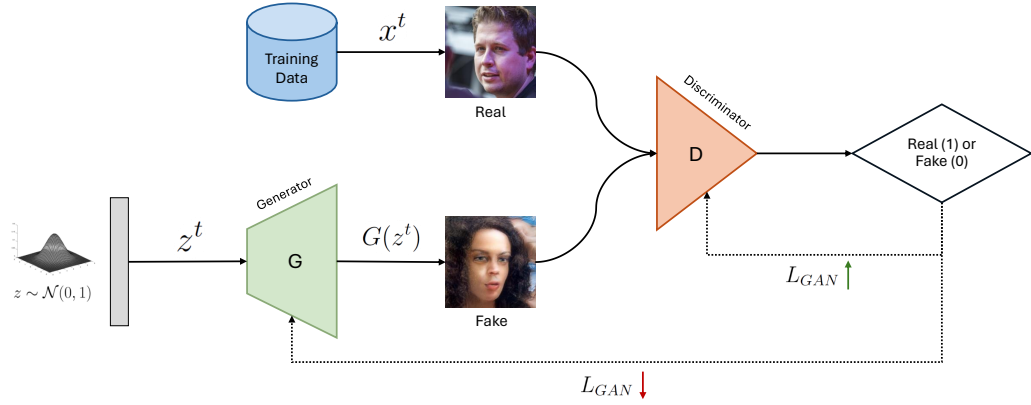


Figure 2.1: The architecture of Generative Adversarial Network.

$$L_{GAN} = \sum_{x^t \in X} \log(D(x^t)) + \sum_{z^t \in P(z)} \log(1 - D(G(z^t))) \quad (2.1)$$

These two networks are trained simultaneously during the training process. While the discriminator network learns how to classify $D(x)$ real and fake images by increasing the objective function, the generator network learns how to generate fake data $G(z)$ closer to the real data distribution, to make them look realistic, by decreasing the objective function. In other words, the discriminator network acts like a learnable loss function that trains the generator network. According to these objectives, the parameters of these networks are updated simultaneously during the training process. After finalizing training, the generator network can be used to generate fake data by mapping the randomly sampled data from the pre-defined distribution to realistic data.

2.4 StyleGAN

StyleGAN [16] modifies the vanilla GAN architecture to condition the generator network on style vectors w for controlling the image generation process. It first maps the Gaussian distribution into a style space W , also called the latent space, learned by the mapping network M . The mapped style vectors are used to modify the activations

of the generator network by using the Adaptive Instance Normalization (AdaIN) approach. This method first normalizes the activations of the network channel-wise, then scales and shifts them according to the style vector. The simplified representation of the architecture is shown in Figure 2.2.

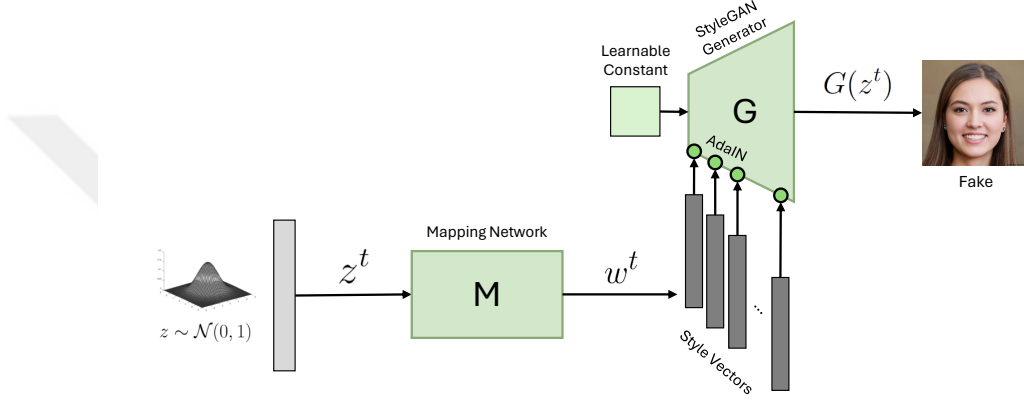


Figure 2.2: Simplified representation of the StyleGAN architecture.

The authors of the paper empirically observed that the W space improved the disentanglement of the features, which is visualized in Figure 2.3.

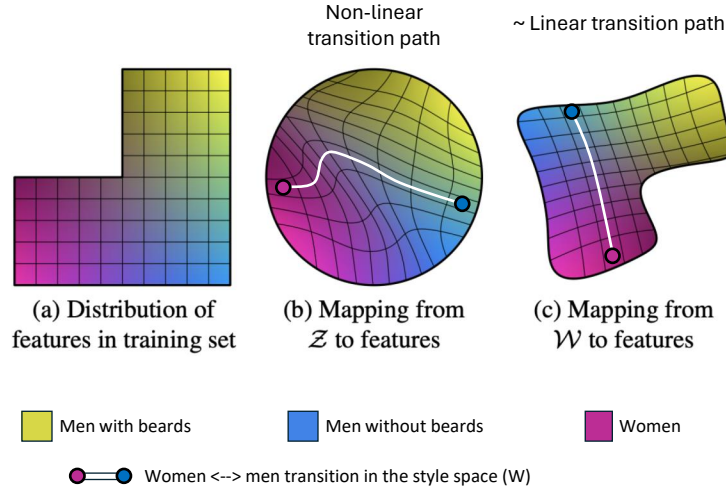


Figure 2.3: Visualization of feature disentanglement in the style space. The figure is adapted from [45]

In the figure, the style space W is shown in a simplified way by using 2D representation. The yellow, blue, and pink areas represent the distribution of men with beards, men without beards, and women, respectively. When we learn a mapping from a Gaussian distribution $P(Z)$ to the real distribution, the transition between features becomes complex, i.e. non-linear. The mapping network disentangles these features by mapping the Z space to the style space W via a deep neural network. This mapping makes the transition between features simpler and close to linear. Therefore, manipulating images in the style space such as interpolation between two style vectors becomes easier compared to Z space.

2.5 Finding Edit Directions in W Space

The disentangled style space W found by the mapping network M makes image editing possible for us. It is observed that a smooth attribute transition occurs between two randomly sampled fake images when their style vectors are interpolated in the style space. This observation made researchers work on finding such edit directions that can change a specific attribute of the images without a need for a reference image as in the interpolation method. In Figure 2.4, an example of this edit direction and the resulting images are shown. The line in the figure represents a possible edit direction for adding beard and the circles correspond to the style vectors w of their corresponding images.

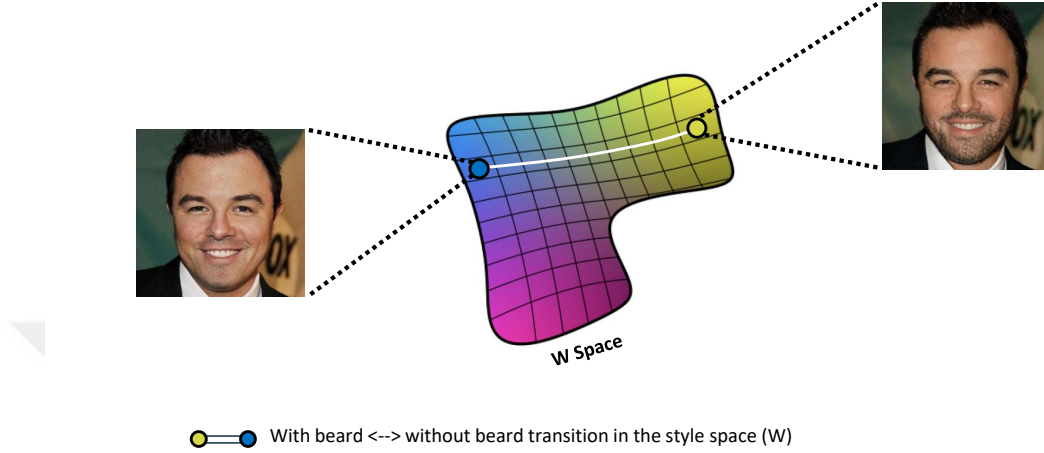


Figure 2.4: Image attribute editing in the style space. The figure is adapted from [45]

InterFaceGAN [29] is one of these editing methods [31, 30, 46] that benefits from the disentangled latent space of StyleGAN. This method finds edit directions by using a pretrained image attribute classifier, such as a beard classifier. To find the direction of the beard edit, the authors first sample 500,000 different random vectors in the Z space and pass them through the mapping network and the StyleGAN generator consecutively to find their corresponding style vectors and the fake images generated by these vectors. Later, they classify these generated fake images via the pre-trained attribute classifier. According to the attribute score found by the classifier, they eliminate the ones having weak classification scores by keeping the images assigned to the top 10,000 highest scores for both negative and positive attribute directions. For the beard attribute example, positive and negative directions represent people with and without beards, respectively. After elimination, they train a linear Support Vector Machine (SVM) in the W space to find the edit direction for the selected attribute. Having the edit direction, the attribute level of a generated image can be manipulated by moving in that direction. The coefficient of the direction can be changed to control the edit strength. For example, by adding the edit vector to the generated image in the positive beard attribute direction, we can add a beard to the image, and the coefficient of this vector controls the strength of the edit, i.e., the thickness of the beard.

2.6 Real Image Editing via StyleGAN Inversion

Recently, extensive research has been conducted on image editing tasks both via end-to-end trained image translation networks [47, 48, 49, 50, 51, 26] and latent space manipulation of pretrained GANs [32, 33, 34, 27, 28]. Pretrained GANs, especially StyleGAN-based methods [16, 18], are shown to organize their latent space in a semantically meaningful way with disentangled representations. This feature enables them to achieve various edits that are beyond what annotated datasets offer [46, 52, 31, 30, 25].

However, the challenge in using these pretrained GANs for real image editing is that one needs to invert images to the natural latent space of GANs and there is known to be a trade-off between reconstruction fidelity of input images and their editability [34, 53, 27]. Different encoders and training schemes are proposed to tackle the inversion problem with great improvements and the inversion problem is still an active research area [28, 54, 35, 36].

Our proposed method, DivInversion, aims to solve a more challenging inversion problem with input images that have missing pixels. This task is referred to as image inpainting or image completion and to the best of our knowledge, our work is the first to explore it with pretrained StyleGAN with the high reconstruction and editability goal.

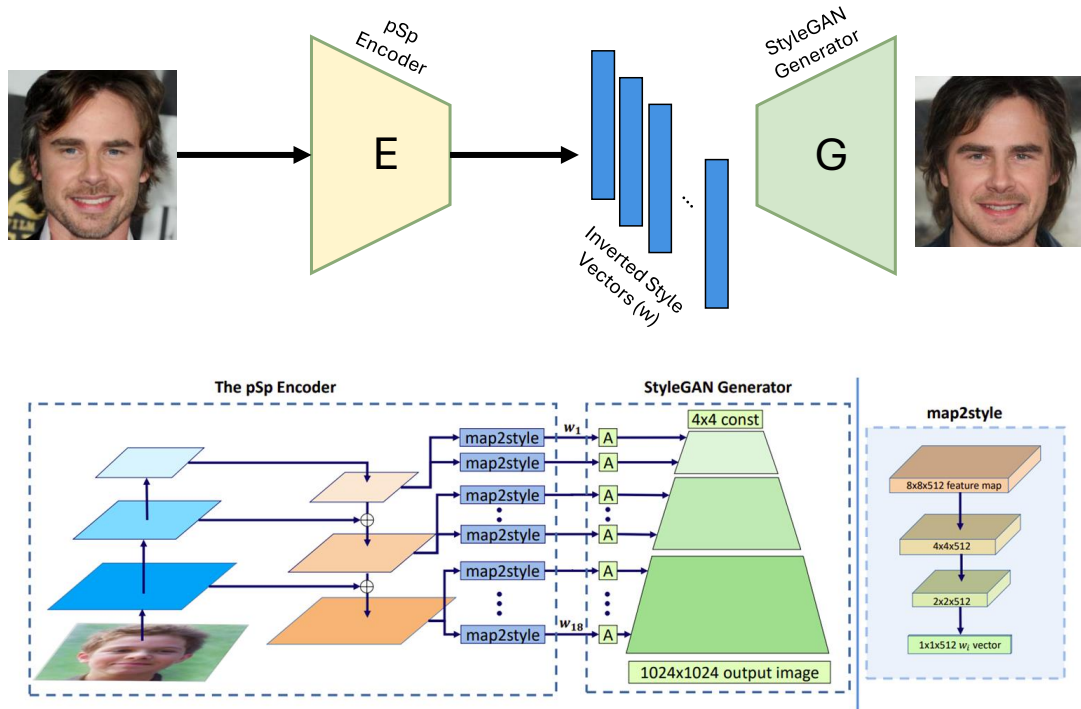


Figure 2.5: StyleGAN inversion with pSp model. The figure is adapted from [53]

Among these inversion models, one of the initial attempts to train an encoder for this task is the pSp [53] model, which is also included in the DivInversion architecture. The authors design an encoder architecture, shown in Figure 2.5, that first extracts the features of the image. As the image passes through deeper layers, the model extracts high-level features, which include more abstract and semantic information about the image. Since StyleGAN’s initial layers determine more abstract features of the generated image, the style vectors of the given image are found by considering the encoder activations at the corresponding levels of abstraction. For this process, they use convolutional network components called `map2style`, also known as style blocks, for each style vector. Additionally, to preserve the information in the next layers, they cumulatively sum the activations of the encoder in reverse order before passing them through the style blocks. Using this encoder network allows us to find the corresponding style vector of a real image and edit it in the W space before feeding it to the generator network. However, as mentioned, pSp is one of the simplest inversion models and cannot

reconstruct some details of the image, such as the background, due to the limitations of StyleGAN’s generator manifold. Nevertheless, it succeeds in editing images using the organized latent space of StyleGAN.

2.7 Denoising Diffusion Probabilistic Model

Denoising Diffusion Probabilistic Model (DDPM) [55] learns how to denoise randomly sampled data from a Gaussian distribution iteratively, resulting in realistic fake images. For this purpose, the authors utilize a modified version of the U-Net architecture [56] to implement the diffusion model. The training process consists of two main phases: forward and backward diffusion. Additionally, they define a scheduler used in training, which has various hyperparameters, such as the number of timesteps T and the variation in noise variance across these timesteps.

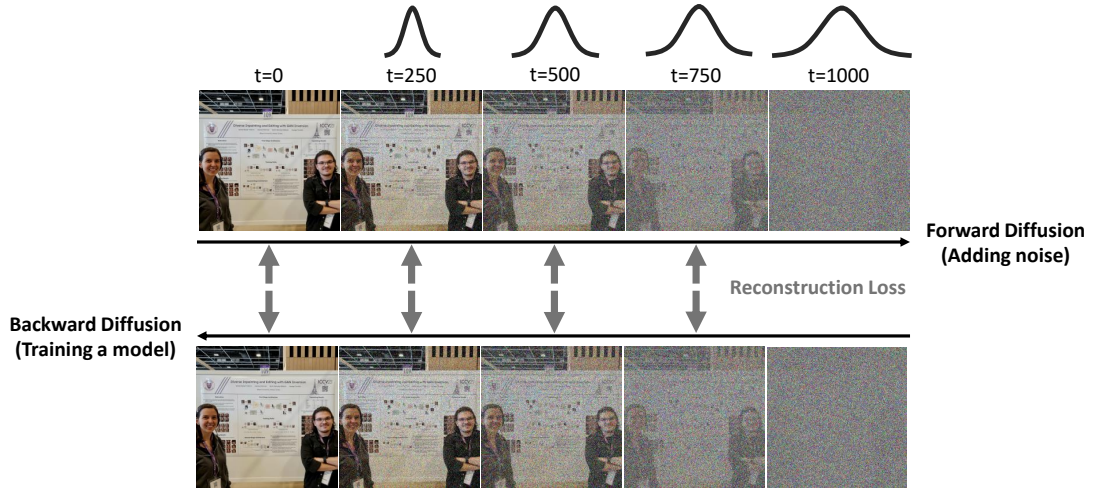


Figure 2.6: Forward and backward diffusion processes of Denoising Diffusion Probabilistic Model.

The training process is visualized in Figure 2.6. In forward diffusion, noise is added to the image incrementally up to a randomly selected timestep t , where the timestep determines the strength of the added noise. As the timestep increases, the mean value

of the pixels gets closer to 0 and the variance approaches 1, which corresponds to the unit Gaussian distribution. In the forward diffusion process, there are no learnable parameters used. After obtaining the noisy version of the image at timestep t , it is passed through the diffusion model, and the reconstruction loss is backpropagated to the network to learn how to predict the noisy image from one step before the timestep at which the forward diffusion was applied. For inference, the process starts with a randomly sampled noise and the model iteratively denoises it until reaching $t = 0$, resulting in the generated fake image.

2.8 Latent Diffusion Model

Instead of working in the pixel space as DDPM does, Latent Diffusion Model (LDM) [7] applies the forward and backward diffusion processes in the latent space of a pre-trained autoencoder. This approach increases the training and inference speeds significantly. Moreover, the authors train multiple LDM models conditioned on various types of conditioning data. Two different conditioning mechanisms are proposed in the paper. The first one is concatenating the conditioning data with the input of the diffusion model. The second one is to apply cross-attention in the layers of the diffusion model. The architecture of LDM is shown in Figure 2.7

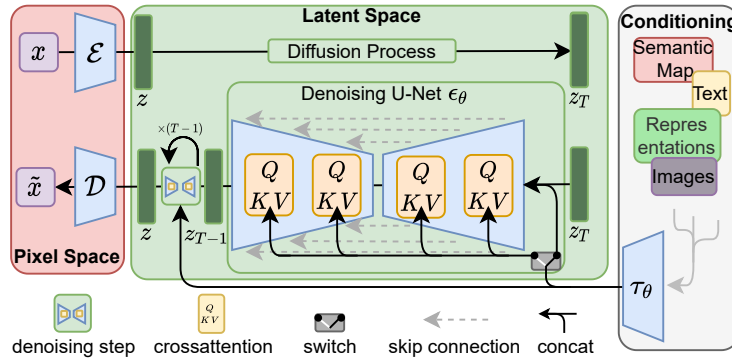


Figure 2.7: Architecture of Latent Diffusion Model. The figure is retrieved from [7].

2.9 Diffusion-based Generative Models

Diffusion models empowered with large-scale text-image paired datasets achieve impressive results on image synthesis [57, 11, 9, 58, 59]. The image synthesis capabilities of diffusion models are extended to video generation [60, 61] and image editing [62, 63, 58, 64, 65, 66] tasks.

Our proposed method, Inst-Inpaint, tackles the inpainting task given text-based instructions and is more related to the editing tasks. It utilizes the LDM architecture conditioned on both a textual prompt and an input image for object removal. These conditionings are applied using the cross-attention and concatenation methods, respectively.

SDEdit is one of the first works that edits real images with diffusion models [62]. It applies noise to input images modified with brush strokes and denoises them. Content preservation is not strictly achieved given the noising-denoising steps. Prompt-to-prompt editing [63] shows via operations applied on the attention mechanisms, generated images can be edited with text prompts. InstructPix2Pix [65] generates a dataset with Prompt-to-prompt and fine-tunes a diffusion model to edit images with paired data.

Chapter 3

Inst-Inpaint: Instructing to Remove Objects with Diffusion Models

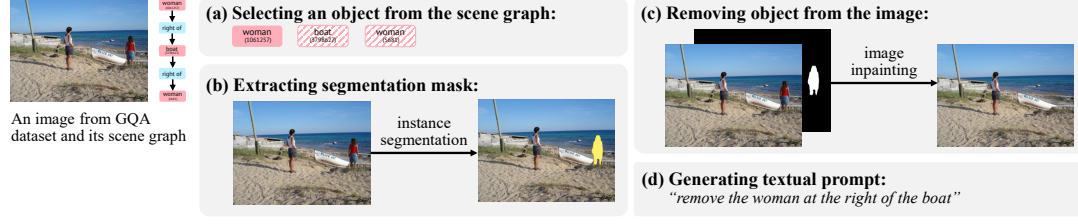
3.1 Dataset Generation

Our goal is to compose a real-image dataset to train and evaluate models for instruction-based image inpainting. There exist some datasets that can be adapted for this task like the CoDraw [67] and the CLEVR [68] datasets. However, the images in these datasets all depict synthetic scenes composed of either geometric objects or clip arts. Although these relatively simple images can be regarded as great tools for building models, the models trained on them are not directly applicable to real image editing.

To alleviate this drawback of the existing datasets, in this work, we propose to build a new image inpainting dataset that involves complex real images without using any human annotation. We refer to this dataset as *GQA-Inpaint* as we opt for building it on top of the GQA Dataset [14]. In particular, the GQA dataset consists of 85K real-world images with their corresponding scene graphs. Scene graphs provide simplified representations of images by representing them in terms of objects, attributes, and relations. Each node in a scene graph denotes an object, with its position and size

specified by a bounding box. Moreover, each object is associated with a number of attributes encoding its color, shape, or material. The edges connecting the nodes and thus objects generally represent the spatial relations between these objects.

Training Data Generation for GQA-Inpaint



Training Inst-Inpaint for Instructional Image Inpainting

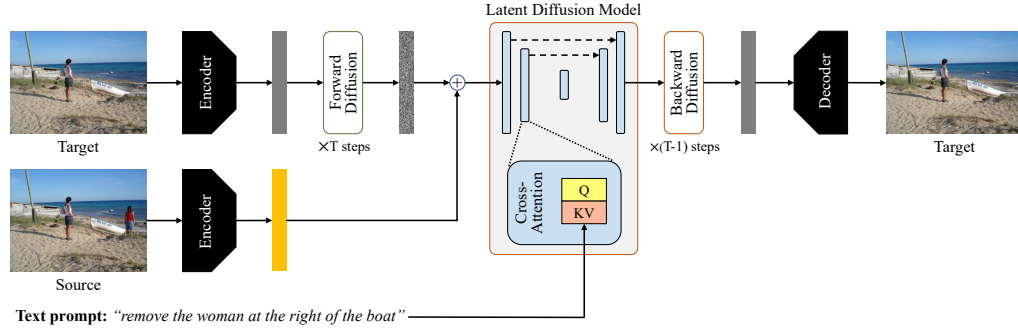


Figure 3.1: **The proposed GQA-Inpaint dataset and our Inst-Inpaint method.** Our work involves initially generating a dataset for the proposed *instructional image inpainting* task. To create input/output pairs, we utilize the images and their scene graphs that exist in the GQA dataset [14]. (a) We first select an object of interest. (b) We perform instance segmentation to locate the object in the image. (c) We apply a state-of-the-art image inpainting method to erase the object. (d) Finally, we create a template-based textual prompt to describe the removal operation. As a result, our **GQA-Inpaint** dataset includes a total of **147165** unique images and **41407** different instructions. Trained on this dataset, our **Inst-Inpaint** model is a text-based image inpainting method based on a conditioned Latent Diffusion Model [7] which does not require any user-specified binary mask and performs object removal in a single step without predicting a mask, as in similar works.

Our data generation pipeline for the proposed dataset shown in Fig. 3.1 heavily depends on the scene graph representation. Specifically, the objects for the inpainting

task are selected from the nodes of the scene graphs. Although the bounding boxes of the objects are already available in the GQA dataset, we observe that directly using them as masks for performing inpainting leads to unsatisfactory results. Hence, to obtain higher-quality target images, we extract segmentation masks of each object and use them to remove objects from images. Additionally, we generate the textual prompts used as the instructions from the provided scene graphs. In the following, we describe these steps.

Selecting Objects from the Scene Graphs. In generating text prompts, we select among the objects included in the scene graphs in the GQA [14] dataset mainly based on the following two questions: “*Is it reasonable to remove this object from the scene?*” and “*Would the removal operation result in an unambiguous scene?*”. Following these questions, we can divide objects in the scene graphs into two broad categories: (i) objects that can be removed and can be uniquely referred to in relation to other objects such as man, boat, and kite, (ii) objects that can appear in referring expressions, but removing them does not make much sense e.g. wall, sky. We manually annotate scene graphs with an attribute to define these categories with a “bidirectional” flag.

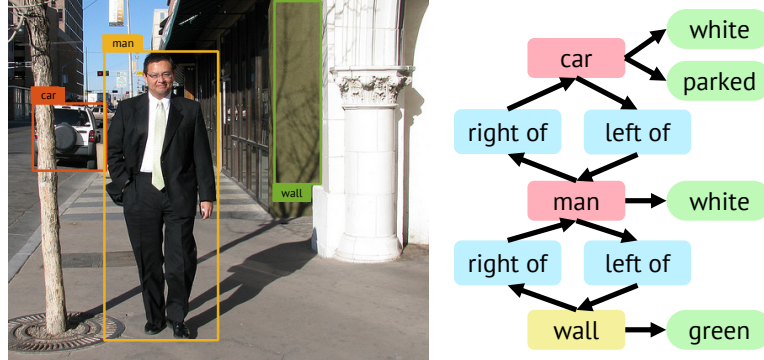


Figure 3.2: A sample image from the GQA dataset [14] and the corresponding scene graph.

From a scene graph point of view, it can be said that the objects in the first group have bidirectional relationships, while those in the second group have unidirectional relationships. For example, consider the image and its scene graph shown in Fig. 3.2. While the *wall* may appear in the expression when referring to the *man*, e.g. ‘*the man*

to the left of the wall’, it cannot be removed from the scene. On the other hand, man and car can be removed from the scene because their bidirectional flag is set to True.

Some of the scene graphs contain nodes that correspond not to a single object but to multiple objects from the same class, e.g. “*apples*”, “*bikes*”, “*windows*”, etc. We manually detect them and opt for using these nodes not as target objects but as objects that can appear in a referring expression, identifying the object of interest.

We also filter out certain instances of the objects based on their sizes and their part-whole relations with other objects. Our criteria are as follows:

- **Object Size.** We do not consider the objects that have an area greater than half of the whole image area as the objects to be removed. We notice that removing those large objects that occupy too much space from an image results in improbable inpainting results. Additionally, we also discard the objects that are too small, which correspond to the objects covering an area smaller than 0.0025% of the whole image. We observe that inpainting such small objects results in scenes that are very similar to the original scene. We decide these threshold values empirically based on our visual inspections.
- **Parts vs. Objects.** We exclude the object classes that can be regarded as parts of some other objects from the list of objects that can be removed. In particular, these objects can be categorized into two as follows:
 - *Implicit parts of an object.* For example, *leg*, *arm*, and *eye* for a human body, *wheel* for a car, and *tail* for a cat belong to this category. Inpainting these implicit parts from images results in non-realistic and visually poor scenes.
 - *Items worn by an object.* For example, *jacket*, *pants*, *shirt*, *jeans*, *shoes* worn by people belong to this category. Like the implicit parts, inpainting these wearable items gives implausible results, while also raising some ethical considerations.

Selecting Relations from the Scene Graphs. Instead of using all the relations that

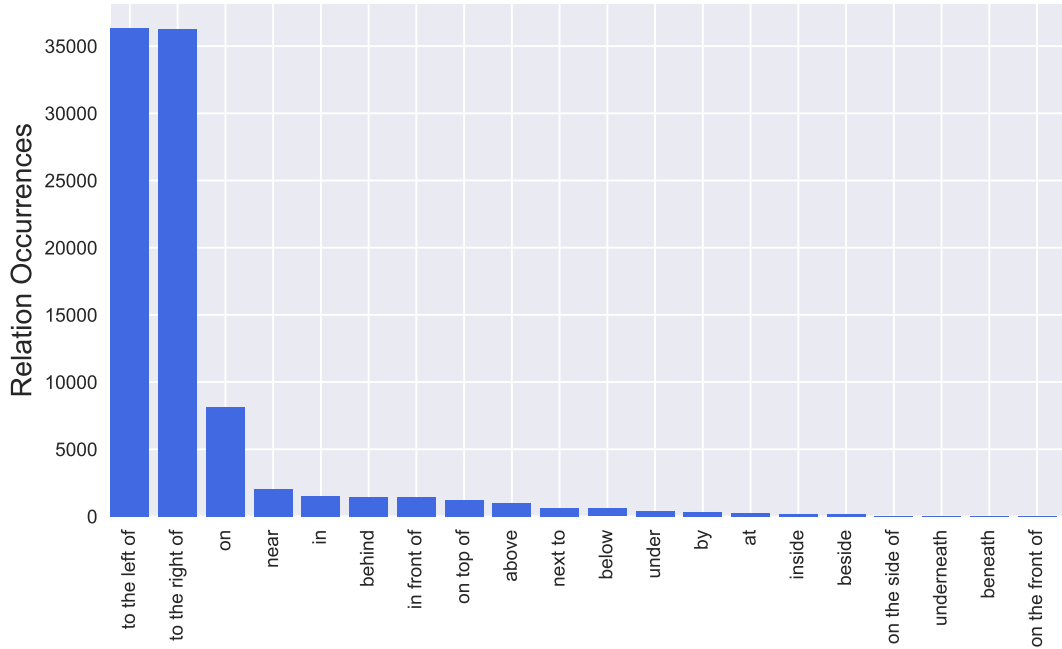


Figure 3.3: Distribution of the relation types exist in the proposed GQA-Inpaint dataset (sorted by their number of occurrences).

exist in the scene graphs, we remove the rarely seen relations that have a frequency rate lower than 0.0001. After this filtering operation, we end up with the relations provided in Fig. 3.3. Additionally, some objects in the scene graph do not have any defined relation with any other object. We follow a simple rule for generating relations for this kind of objects by using their spatial location. Specifically, we split an image into three equal sections along the x-axis, and name these sections as “*left*”, “*center*”, and “*right*”, following the natural naming. We generate the relation descriptions in the form of “*at the [location]*” with the [location] referring to one of the aforementioned sections that largely covers the object of interest.

Extracting Segmentation Masks. We obtain segmentation masks of objects with Detectron2 [69] and Detic [70] frameworks. To cover a wide range of object categories, we segment objects with multiple instance segmentation models that are trained on COCO [71] and LVIS dataset [72].

- **Detectron2** [69]. We extract the segmentation masks using the models available in Detectron2 Model Zoo, namely Mask R-CNN [73], Cascade R-CNN [74],

and Panoptic FPN [75]. The class labels that these models trained on are, in general, different from the class labels from the GQA dataset. To find the proper class label of an object, we compare the Intersection over Union (IoU) scores of the prediction bounding box with all of the bounding boxes of objects from the scene graph, and label the object with the class label with the highest IoU score.

- **Detic** [70]. We use SwinB-21K [76] model trained on LVIS [72], COCO [71] and ImageNet-21K[77] datasets to extract the segmentation masks. In addition to an input image, the Detic framework also expects a set of possible class labels to predict the segmentation masks. For that, we consider the list of object classes available in the COCO dataset along with our own custom-made vocabularies extracted by making use of the attributes of the objects along with the object names (e.g., “*red chair*”), and the object names alone (e.g., “*chair*”). To retrieve segmentation masks whose vocabulary differs from ours, we follow the same methodology explained for Detectron2 reviewed above.

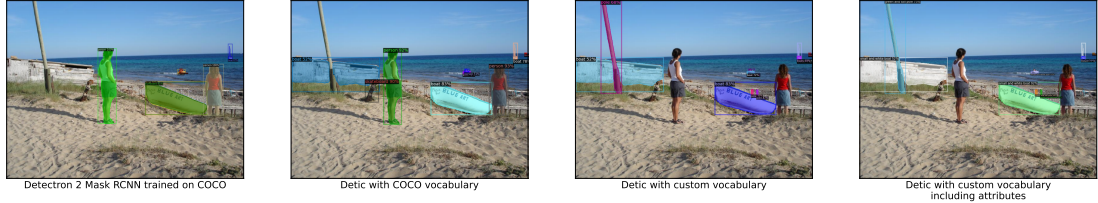


Figure 3.4: Comparison of mask extraction methods. The vocabulary of the COCO dataset is different from the vocabulary of the GQA dataset. To extract the correct segmentation masks, we make use of a combination of pretrained models available in Detectron2’s Model Zoo and use the vocabulary generated based on the scenes to predict the corresponding class labels using the Detic2 framework. Different setups provide us with more options to pick the most accurate instance segmentation mask.

Fig. 3.4 shows the instance segmentation masks predicted by the aforementioned methods and setups. By combining the results of different models, we obtain more accurate segmentation masks for a notable number of object classes. Among these predictions by the multiple methods and setups, we pick the instance segmentation masks having higher IoU between ground-truth and detection bounding boxes as the final object mask.

Removing Objects From Images. Our goal in this step is to remove the objects from the image. For that purpose, we use three of the state-of-the-art image inpainting methods, CRFill [78], LaMa [79], and MAT [39] given their computational efficiency and high-quality results. We generate three datasets based on those three inpainting methods and conduct experiments on each one. Before performing inpainting, we first apply CascadePSP [80] to refine the predicted masks and then use morphological dilation with a structuring element of 11×11 pixels to expand the segmentation masks at the edges, to make the masks better capture the objects. These steps are demonstrated in Fig. 3.5.

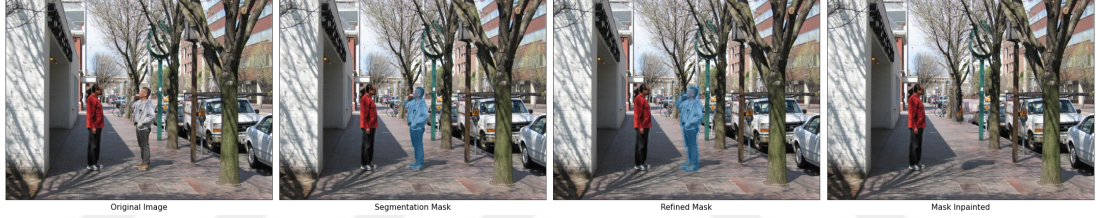


Figure 3.5: An illustration of the intermediate steps of the object removal operation that we considered while constructing GQA-Inpaint. While the first column depicts the original image, the second row shows the segmentation mask detected by a pretrained instance segmentation model. The third column demonstrates the object mask used for inpainting, which is first refined by the CascadePSP [80] and then expanded by a dilation operation. The fourth column presents the final inpainting result obtained by CR-Fill [78].

Generating Textual Prompts. The GQA dataset contains relational representations between objects in scene graphs as shown in Fig. 3.1. If the selected object is a single instance of its category, we do not use any relations and simply create the prompts as “*remove the [object]*”. If multiple instances of the same object category are present in the image, we construct the prompt based on the relations from the scene graph to specify a specific instance. For the object to be removed, we use all of its relations if they are available. Otherwise, we use the image relative position. Multiple relations are combined into a single relation by concatenating all of them with “*and*”. For example, relations “*to the left of the man*”, and “*on the table*” combined into a single relation “*to the left of the man and on the table*”. In addition to the relations, we also make use of

the object attributes both for the object to be removed and for objects that participate in the relations. Unlike relations, we do not combine all the available attributes. Instead, we randomly choose an attribute as a data augmentation method. We provide the object statistics in our dataset in Fig. 3.6.

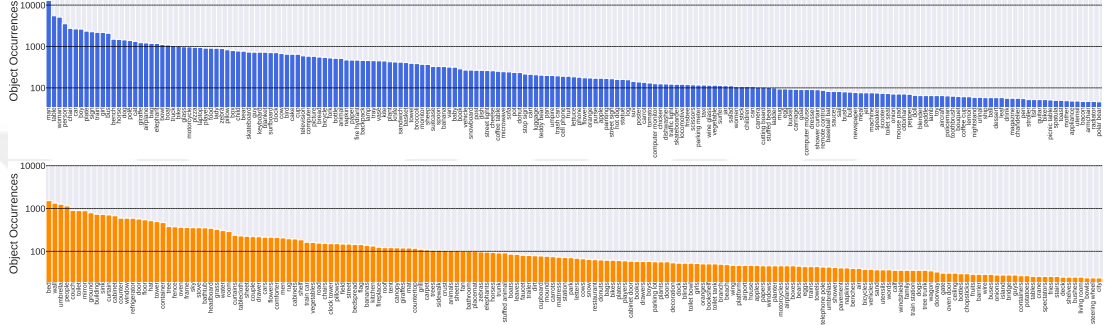


Figure 3.6: Distribution of the object classes exist in the proposed GQA-Inpaint dataset (a subset is given), sorted by their number of occurrences and categorized based on their roles. Objects that can be removed and can be referred to in relation to other objects are shown in blue color, while the objects that can appear in referring expressions are given in orange color. Please zoom in for text.

3.2 Method

To solve the proposed instructional image inpainting task, we develop a novel conditional diffusion model named *Inst-Inpaint* which takes an image and a textual instruction as inputs. Training this model requires a paired dataset that includes source and target images with the corresponding text prompts.

We build our model based on the latent diffusion model [7] due to its computational efficiency. Specifically, the latent diffusion model includes an encoder E , and decoder D to project images to a lower latent dimension and to reconstruct the images back, respectively. It is a variational autoencoder and is trained separately as the first step.

The diffusion process [55] takes place on the latent codes, $z_0 = E(x)$, with x denoting the input image. Noise is gradually added to z_0 at each time step t such

that after T number of steps, $z_T \sim N(0, I)$. Diffusion models learn denoising auto-encoders which are trained to reverse the process and predict a denoised variant of their input, z_t where z_t is a noisy version of z_0 . This objective that operated on the latent codes is as follows:

$$L_{LDM} := \mathbb{E}_{E(x), \epsilon \sim N(0,1), t} [||\epsilon - \epsilon_\theta(z_t, t)||] \quad (3.1)$$

with t sampled from $1, \dots, T$ and $\epsilon_\theta(., t)$ is a neural network that predicts the added noise conditioning on the time-step t . Similar to previous works [7], we use a U-net architecture.

As stated, our model for instructional image inpainting can be interpreted as a conditional diffusion model. That is, we train our model with two additional conditioning; source image and text prompt. During training, the target image goes through the encoder to obtain z_0 as shown in Fig. 3.1. The source image, s , is also encoded with the same pre-trained encoder from the first stage. The encoded features, $E(s)$ are concatenated with z_t at each time step. For the text conditioning, we use the cross-attention mechanism [81]. The objective of the instruction-based object removal is given as follows:

$$L_{LDM} := \mathbb{E}_{E(x), \epsilon \sim N(0,1), t} [||\epsilon - \epsilon_\theta(z_t, t, E(s), T(i))||] \quad (3.2)$$

Here, i refers to the text instruction and T , the transformer. The encoder, E , is fixed during training whereas ϵ_θ and T are optimized together.

Implementation and Training Details. We train two Inst-Inpaint models; one on the GQA-Inpaint and one on the CLEVR datasets. The reason for us to train a model on the CLEVR dataset is to compare the Inst-Inpaint model with GAN-based works. Because we were not able to train the previous instruction-based image-erasing works that are built on GANs successfully on the real GQA-Inpaint dataset. Therefore, we compare with them on the CLEVR dataset. More discussion on this comparison is given in Sec. 3.3.1. In Table 3.1, we provide an overview of the hyperparameters. While the GQA-Inpaint model uses a pretrained VQGAN model¹ as the first stage encoder, we train an autoencoder with KL regularization for the CLEVR model. For both stages of the training, center cropping followed by resizing is applied to both datasets.

¹<https://github.com/CompVis/taming-transformers>

	CLEVR	GQA-Inpaint
First Stage Encoder		
Regularization Type	KL	VQ
z -shape	$16 \times 16 \times 4$	$32 \times 32 \times 4$
$ \mathcal{Z} $	—	16384
Epochs	400	Pretrained
Self-attention Res.	16	32
Conditional LDM		
Diffusion Steps	1000	1000
Noise Schedule	linear	linear
Channels	64	128
Depth	2	2
Channel Multiplier	1, 2, 4	1, 2, 3, 4
Number of Heads	4	8
Batch Size	16	32
Epochs	1000	300
Learning Rate	$2.e-6$	$2.e-6$
Text Cond. Cross Attn		
Resolution	32, 64, 128	32, 64, 128
Transformer Depth	1	1
Text Cond. Embedding		
Dimension	512	512
Transformer Depth	8	16

Table 3.1: Hyperparameters for the proposed Inst-Inpaint model trained on the CLEVR and the GQA-Inpaint datasets.

Our model has two types of conditioning in the second stage, which are source image and text. For source image conditioning, we concatenate the latent representation of the source image with the noisy image at the current timestep. The latent representation is obtained by using the first stage encoder, which is frozen during the training phase. Therefore, the U-Net architectures have doubled latent channel sizes for the input data. For text conditioning, we apply cross-attention between the text embeddings and the specific U-Net activation layers. For this purpose, as in the text-to-image model

of LDM, we train a Transformer model [81] from scratch together with the U-Net by feeding BERT [82] tokens as input to extract in-domain semantic representations from the inpainting instructions.

Since we use LDM as the base architecture, which applies diffusion steps in the latent space instead of the pixel space itself, in complex datasets such as GQA-Inpaint, some scenes with complex patterns especially texts can be poorly reconstructed by the autoencoder. To achieve perfect reconstruction of the background, we use the per-sample weight optimization to the decoder of the autoencoder on the input image [13].

3.3 Experiments

3.3.1 Baselines

We compare Inst-Inpaint against diffusion and GAN-based models with single-stage and two-stage pipelines. Two stages pipelines follow segment and inpainting paradigm where we use Instruct X-Decoder [83] and CLIPSeg [84] for mask segmentation and GAN-based MAT [39], CR-Fill [78], LaMa [79] and diffusion-based BlendedDiffusion [13] and SDInpaint [7] for inpainting tasks. We experiment with the released application of Instruct X-Decoder² which combines X-Decoder for image understanding and GPT-3 for language understanding. It is important to state that all these components are trained on large-scale datasets which are much larger than our generated GQA-Inpaint dataset. InstPix2Pix [65] is similar to our method which is single-stage and trained end-to-end for translating images based on instructions. It is trained on a paired image dataset that is generated by the authors. The paired images based on the instructions are generated with prompt-to-prompt method [63]. We use the 256×256 resolution in our experiments for all methods.

We also try comparing our model with end-to-end trained GAN-based models such TIM-GAN [85] and cMANIGAN [86] for text-based object removal. TIM-GAN learns

²<https://huggingface.co/spaces/xdecoder/Instruct-X-Decoder>

a mask for object removal by encoding the input image and text. Encoded features are multiplied with the mask in a pixel-wise manner and an image is reconstructed via a decoder. The network is trained with adversarial losses. Additionally, reconstruction loss is applied to the mask prediction. cMANIGAN also predicts a mask and learns to edit images with adversarial losses and reconstruction loss on the mask. cMANIGAN proposes to use a cycle consistency loss by editing images twice; first with the given instruction and second with the reverse of it. These models are tested on synthetic datasets. We train these models on our dataset but do not achieve any meaningful results. For the completeness of our study, we compare our method with these methods on a synthetic dataset; namely the CLEVR dataset.

3.3.2 Datasets

GQA-Inpaint Dataset. The construction of this dataset is extensively explained in Section 3.1. We train the models on unique 44500 source images and unique 88369 target images. Because from source images we pick different objects and generate instructions, training includes 173715 unique source-target-prompt pairs. We test the models on 4811 unique source images, 9485 unique target images, and 18883 unique source-target-prompt pairs.

CLEVR Dataset. This dataset is generated by CLEVR toolkit [87, 68]. The dataset contains 3-D synthesized simple shapes with different colors and sizes. Each training pair includes a source image, a target image, and a text specifying the modification from the source image to the target image. There are three types of text; add, remove, and change an object. We train our model with the remove instructions. To increase the size of the dataset, we use the pairs with both remove and add instructions. We flip the source-target pairs of add instructions and modify the text accordingly for the training dataset. This subset includes 5404 training and 2362 test images.

3.3.3 Evaluation Metrics

FID and CMMD. We report the Frechet Inception Distance (FID) [88] and the CLIP Maximum Mean Discrepancy (CMMD) metrics [89] for assessing the photorealism of the generated images by comparing the target image distribution and edited image distributions. These metrics, however, do not evaluate whether the correct object has been removed. Since methods that inpaint only a few pixels are likely to achieve better scores, we complement these with CLIP-based measures and RelSim to more accurately assess the success of the removal operation.

CLIP Distance. The goal of CLIP distance is to evaluate how well the target object in the instruction is removed. We use the bounding boxes linked to the nodes of the scene graphs for this metric. Specifically, our proposed GQA-Inpaint dataset inherently includes information regarding the bounding box of objects that are instructed to be removed and their semantic category labels. We extract the image regions as suggested by these bounding boxes from both the source and the inpainted images. We then estimate the CLIP similarities [15]. between these extracted regions and the text prompts including semantic labels as shown in Fig. 3.7.

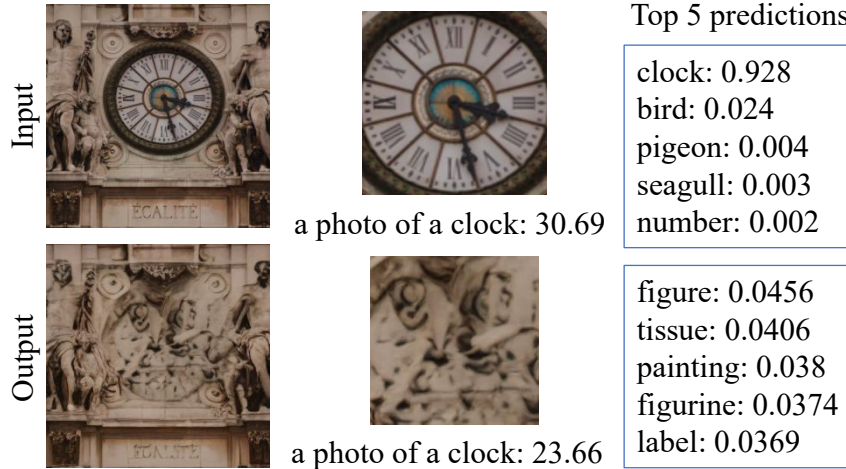


Figure 3.7: **CLIP-based image inpainting scores.** We use the image-text similarity scores and zero-shot classification scores in our evaluations. Please refer to Section 3.3.3 for score descriptions.

As in the example, we use the text prompt “*a photo of a clock*” to extract the object information. In the example, these similarity scores are estimated as 30.68 and 23.66. If the similarity value decreases, this is considered a success as in the example and we report the percentage of it.

CLIP Accuracy. For this evaluation measure, we utilize CLIP as a zero-shot classifier. In particular, we find out the most likely semantic label of the image region extracted from the source image using the bounding box of the target object, by considering the object classes that exist in our dataset. Next, we perform another prediction this time for the image region extracted from the inpainted image. We present the Top5 predictions as an example in Fig. 3.7. We expect the class prediction to change after performing the object removal operation as suggested by the instruction. We mainly look at the Top-1 and Top-5 predictions. If the predicted class based on the source image is not in the Top1 and Top5 predictions of the inpainted image, that scenario is considered a success. We report the percentage of successes.

RelSim. We additionally use the relational similarity score proposed by GeNeVa [90] which is commonly used on CLEVR dataset. This metric compares the arrangement of objects qualitatively via an object detector and localizer. We use the object detector released by GeNeVa fine-tuned on the CLEVR dataset. The detector constructs the scene graphs of inpainted and ground-truth images. The metric computes how many of the ground-truth relations are present in the inpainted image. We use this score for our evaluations on CLEVR following the previous works [85]. However, we find that this evaluation method is difficult to use on real images because it requires highly effective object detectors. For our experiments on GQA-Inpaint, we use the CLIP-based measures explained below.

User Study. We conduct a user study with the first 30 samples from our validation set with 20 users. To include a diverse and successful set of methods, we add InstPix2Pix, X-Decoder-LaMa, and CLIPSeg-LaMa to the user study.

3.3.4 Experimental Analysis

Results on GQA-Inpaint Dataset. We provide the quantitative and qualitative results on GQA-Inpaint dataset in Table 3.2 and Fig. 3.8, respectively.

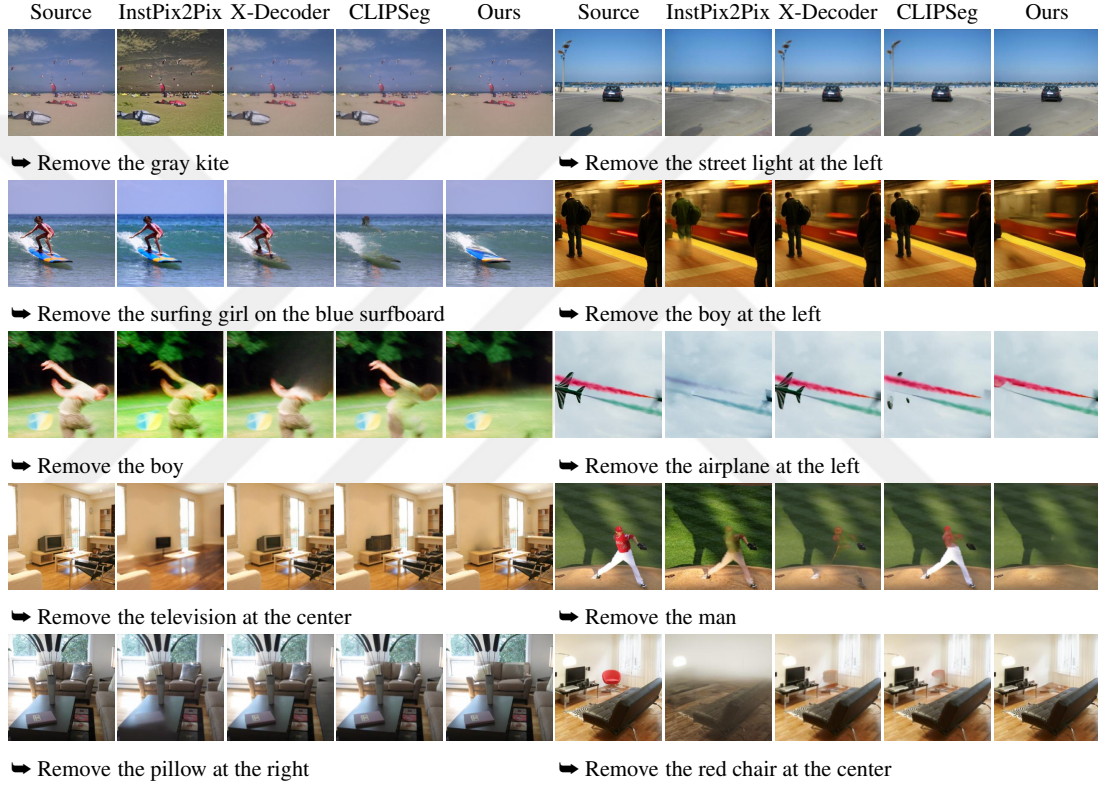


Figure 3.8: Qualitative results of our approach and the competing methods on the GQA-Inpaint dataset. Other methods sometimes struggle to detect the correct object and to erase the object completely. On the other hand, ours achieve significantly better results.

Methods	FID ↓	CMMD ↓	CLIP Dist. ↑	CLIP Acc. ↑	CLIP ↑ Acc. (top 5)	User Study ↑
InstPix2Pix [65]	19.860	0.104	0.385	0.392	0.156	2.5%
X-Decoder [83] SD [10]	21.194	0.134	0.583	0.636	0.409	-
X-Decoder [83] CR-Fill [78]	23.495	0.150	0.598	0.632	0.425	-
X-Decoder [83] MAT [39]	21.296	0.119	0.583	0.616	0.395	-
X-Decoder [83] LaMa [79]	22.819	0.110	0.569	0.617	0.398	<u>23.2%</u>
CLIPSeg [84] SD [10]	21.300	0.137	0.574	0.636	0.392	-
CLIPSeg [84] CR-Fill [78]	23.788	0.175	0.609	0.648	0.426	-
CLIPSeg [84] MAT [39]	<u>21.172</u>	0.128	0.573	0.613	0.374	-
CLIPSeg [84] LaMa [79]	22.755	0.122	0.537	0.604	0.366	15.9 %
Inst-Inpaint - Pseudo-GT with CR-Fill	23.552	0.145	0.694	<u>0.690</u>	<u>0.498</u>	-
Inst-Inpaint - Pseudo-GT with MAT	22.990	0.163	<u>0.693</u>	0.709	0.504	-
Inst-Inpaint - Pseudo-GT with LaMa	22.835	<u>0.106</u>	0.681	0.678	0.492	58.4%

Table 3.2: Quantitative results of our method and competing methods on GQA-Inpaint dataset.

Our method achieves improved scores with the integration of superior inpainting techniques in the ground-truth generations. We select Inst-Inpaint with LaMa ground-truth generations for inpaintings as our final model and show its visual results in Fig. 3.8. We achieve comparable FID and better CMMD scores than other competing methods. Note that, InstPix2Pix achieves better CMMD result than ours however that is because InstPix2Pix makes minimal and sometimes no changes to the images as can also be seen in CLIP scores. When images are not modified, since the real pixels stay in place, those results tend to have better FID and CMMD scores. For CLIP scores, that measure the accuracy of the inpaintings, our methods achieve significantly better scores than others. As seen in Fig. 3.8, InstPix2Pix struggles to achieve the inpainting in many examples. In the first one, instead of removing the gray kite, the model makes the image more gray. In some examples, it successfully erases the objects and in some, it does not remove the objects at all or remove them partially. Instruct X-decoder struggles to find the correct object to remove and does not perform any editing in many examples such as the first row-second example. In the first example, it correctly finds the object to be removed but generates artifacts in the final inpainted image. On the other hand, when no edit is applied to the image as in many examples of X-Decoder and ClipSeg, that brings an advantage for the FID and CMMD scores. Our

Inst-Inpaint model successfully erases objects in challenging scenarios. These visual results are also consistent with the User Study result reported in Table 3.2 where our model is preferred compared to CLIPSeg-LaMa, X-Decoder-LaMa, and InstPix2Pix by a large margin.

Analysis. We visualize the average cross-attention masks in Fig. 3.9 for samples of GQA-Inpaint images. We plot the pixels the prompt attends to.

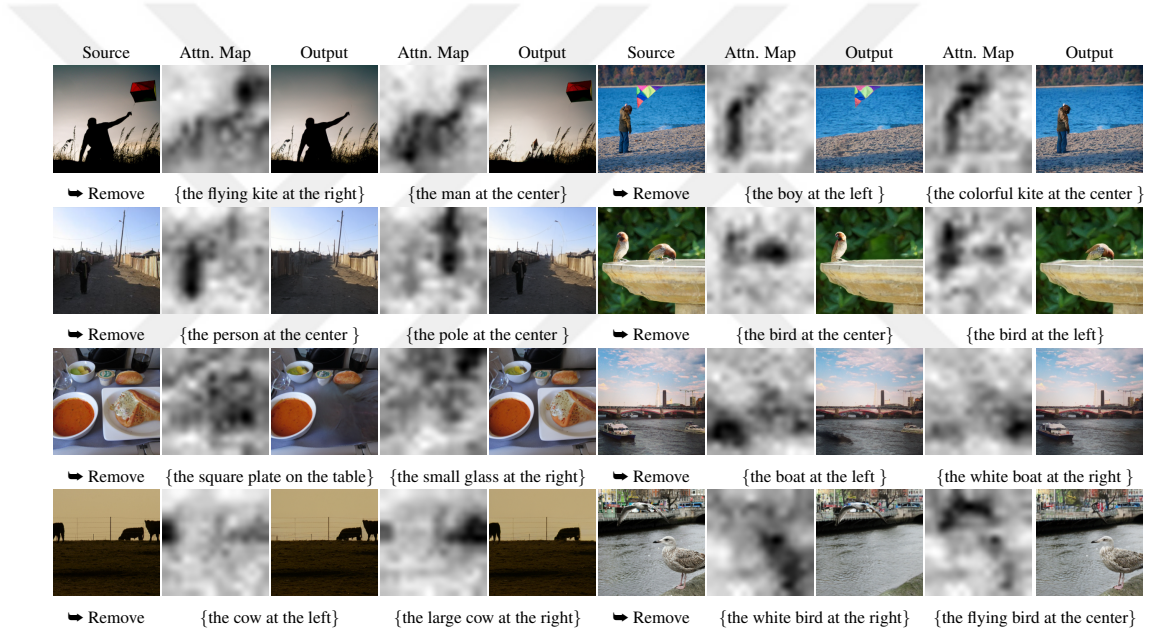


Figure 3.9: Additional instruction-based image inpainting results on the GQA-Inpaint dataset along with the average cross-attention maps of our method. For the same input image, we show two different inpainting results and their attention maps based on two different instructions.

Because ours is a removal task, we find the attention masks to focus on the objects that will be erased. As shown in Fig. 3.9, for the same source image, our model can erase different objects based on the prompts and we see that attention maps focus on those objects that we want to remove.

Next, we analyze the accuracy of attention maps in terms of their segmentation of the prompted object. In our method, we do not explicitly predict masks but inpaint objects in an end-to-end framework. However, to erase an object, the network

needs to detect the pixels to be erased implicitly and we see in Fig. 3.9 that attention maps detect where the objects that we want to erase are. We would like to compare the accuracy of these detections with other methods that predict masks based on prompts such as CLIPSeg [84] and X-Decoder [83]. To analyze that, we train a simple UNet that takes attention maps and predicts a mask. The UNet only takes the attention maps, it does not take prompts or image features as inputs. This UNet is trained separately with the sole purpose of analyzing the mask predictions of a pre-trained Inst-Inpaint. The attention maps are resized to 32×32 resolution from 16×16 , 8×8 , and 4×4 resolutions by using nearest-neighbor interpolation. We simply concatenate attention maps from all time-frames which brings a channel of 1280 inputs. The network has 4 downsampling and 4 upsampling residual block layers with channels going as $\{32, 64, 96, 128, 128, 96, 64, 32\}$. In the end, we have a bilinear upsampling layer which brings the resolution to 256×256 . This is a very simple set-up as we expect the attention maps to already generate well-defined predictions. We compare the mask accuracies in Table 3.3. Our method achieves better scores than ClipSeg and X-Decoder.

CLIPSeg [84]	X-Decoder [83]	Inst-Inpaint (Ours)
40.6	48.7	53.1

Table 3.3: Segmentation scores (IoU) of mask predictions on GQA-Inpaint datasets. Our method does not explicitly detect a semantic map. We train a simple network that operates on our attention maps to evaluate the accuracy of detections.

We visualize the mask predictions in Fig. 3.10 of X-Decoder, ClipSeg, and ours that come from the UNet model which takes the attention maps as inputs. Our model is better at predicting the correct masks as was also expected based on the inpainting results. Note that, we set the U-Net to work on 32×32 resolution and only add a bilinear upsampling in the end, better scores with higher details on the mask can be obtained with a more powerful decoder. We set the segmentation network simple for our analysis.

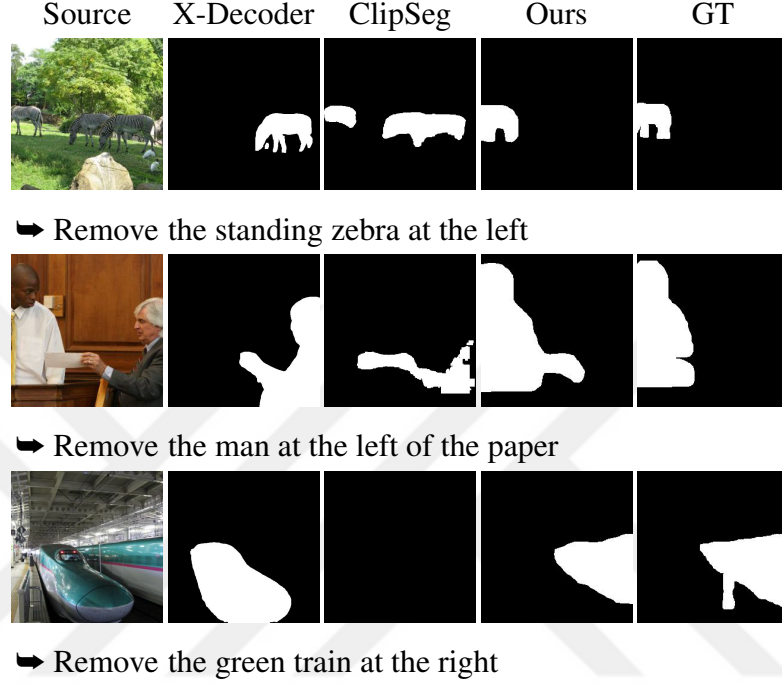


Figure 3.10: Qualitative results of our segmentation analysis and the mask predictions of the competing methods on the GQA-Inpaint dataset.



Figure 3.11: The VAE that is used in LDM does not preserve image details in the reconstruction. To achieve the perfect reconstruction, we use weight optimization in the decoder [13]. When we use the optimized decoder with our method, it can reconstruct challenging texts and achieve inpaintings.

In Fig. 3.11, we show the inpainting results with and without decoder optimization. The VAE that is used in LDM does not preserve image details in the reconstruction

as shown in the second image. To achieve the perfect reconstruction, we use weight optimization in the decoder [13]. When we use the optimized decoder with our method, it can reconstruct challenging texts and achieve inpaintings as shown in the last image. In Fig. 3.12, we show the iterative inpainting results of our framework.



Figure 3.12: Our framework can iteratively remove objects from a scene.

Comparisons with GAN-based models on CLEVR Dataset. Next, we run experiments with the more simple CLEVR dataset and compare our method with GAN-based methods on this dataset. We provide the quantitative and qualitative results on the CLEVR dataset in Table 3.4 and Fig. 3.13, respectively. We train TIM-GAN with their released implementation. TIM-GAN, in the author’s paper, is extensively tested on the CLEVR dataset and there are provided pre-trained models. However, we train their model again since we are only interested in the remove instruction. For cMANI-GAN, the authors provide us with the output images based on the text and input images we send them.

Methods	FID ↓	RelSim ↑
cManiGAN [86]	27.61	0.613
TIM-GAN [85]	0.558	0.919
Inst-Inpaint (Ours)	0.453	0.914

Table 3.4: Quantitative results of our method and competing methods on synthetic CLEVR dataset for the instruction-based inpainting task.

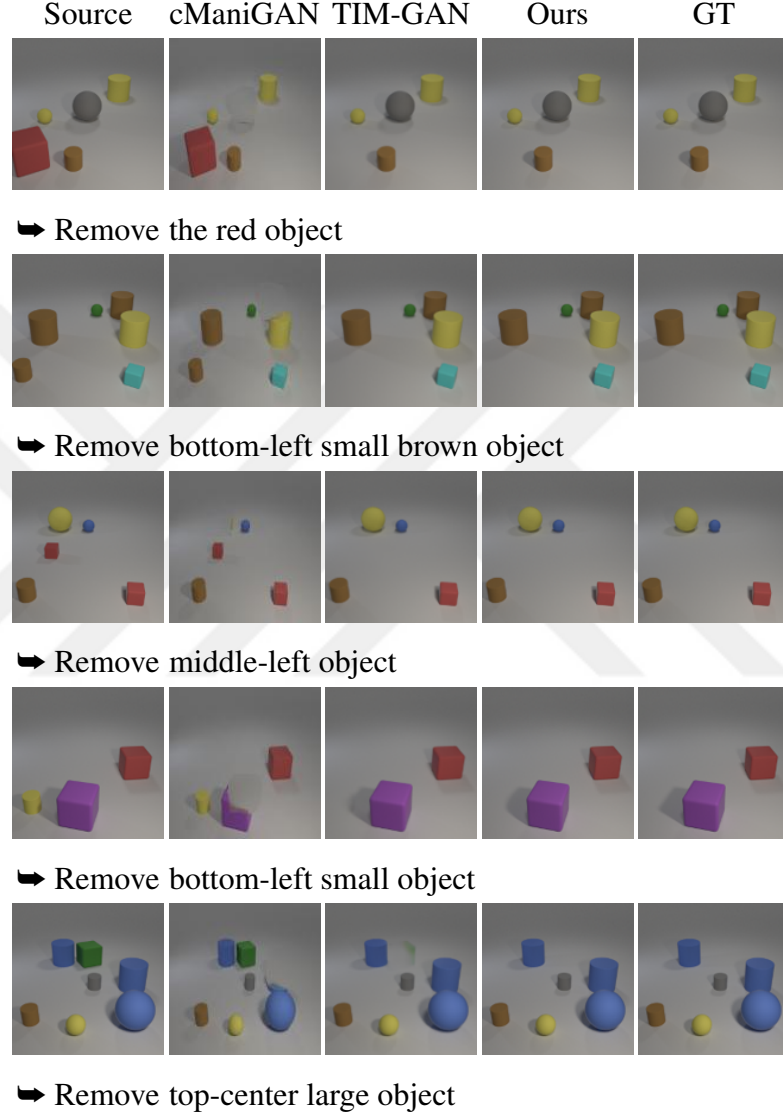


Figure 3.13: Qualitative results of our approach and the competing methods on synthetic CLEVR dataset.

cManiGAN is trained on the CLEVR dataset but not by us. The authors mention that the caption templates they use differ from ours and might result in poor performance. The visual and quantitative results of cManiGAN are indeed poor in our samples. The object to be removed is not correctly detected in many examples and furthermore, the objects are not inpainted with high-fidelity. There is a ghosting effect in the first example and objects are partially removed in some examples (last 4 examples). We

achieve a better FID score and a comparable RelSim score to those of TIM-GAN. Visually, inpainting results obtained by TIM-GAN include artifacts as it is not always able to fully remove the objects. Our diffusion-based Inst-Inpaint model, on the other hand, does include fewer artifacts on this simple dataset. Note that this is a quite easy set-up compared to the GQA-Inpaint dataset. We share these results to compare with GAN-based models that are built on this dataset and to show that Inst-Inpaint model set-up achieves better results than the proposed GAN-based models. Because other diffusion-based models, X-Decoder, CLIPSeg, and InstPix2Pix are trained on a large number of real images, we do not include them in this comparison.

3.3.5 Limitations

In this study, we introduce a model and dataset for instruction-based image inpainting that eliminates the need for masks during inference. However, our model relies on paired data during training, necessitating mask detection for training purposes in the dataset generation. Moreover, due to the challenge of acquiring paired data for inpainting, we had to resort to imperfect inpainting methods. Investigating future approaches that eliminate the requirement for paired data holds significant promise.

Chapter 4

DivInversion: Diverse Inpainting and Editing with GAN Inversion

4.1 Method

We propose a two-stage training pipeline. In the first stage, we learn a base encoder and mixing network to achieve diverse results as presented in Section 4.1.1. In the second stage, Section 4.1.2, we learn skip connections to the generator to achieve high-fidelity reconstructions and seamless transitions between unerased and erased pixels.

4.1.1 First Stage: Diverse Results

Our method utilizes an architecture that includes an image encoder (E), mixing network (Mi), and StyleGAN2’s mapping (Ma) and generator (G) networks [18]. We freeze the StyleGAN’s mapping and generator networks, as shown in Fig. 4.1.

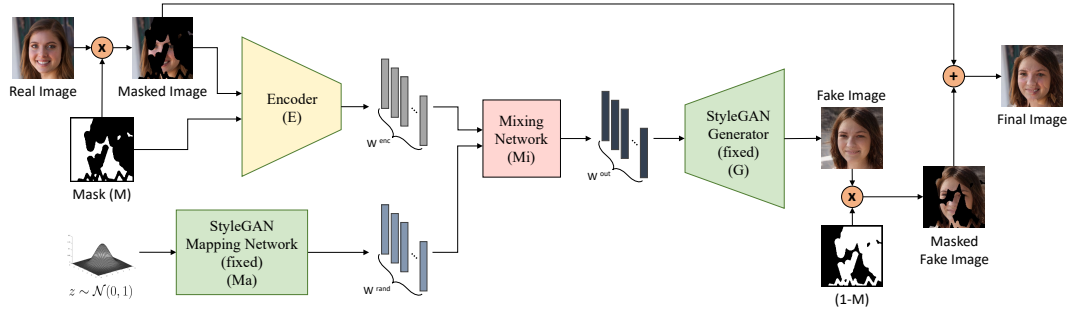


Figure 4.1: First stage framework includes trainable image encoder and mixing network, and frozen StyleGAN’s mapping and generator networks. Our encoder takes an erased image and binary mask to embed the image into StyleGAN’s latent space. We also sample a latent code via the mapping network to achieve stochasticity. The mixing network combines the available information of the erased image from the encoder and the missing part from the mapping network. The mixed encoded latent representations are fed to the Generator via the instance normalization layers to output the fake image. There is a final step where the input image and the fake image are combined based on the mask.

We obtain erased images by multiplying them with a binary mask M that defines the valid and erased pixels; $I^e = M \odot I$. Before feeding the erased image to the encoder, we concatenate the binary mask M with it. We set a simple encoder [53] to project our erased images. Our encoder embeds erased images into W^+ latent space. Our goal is to achieve diversity in our inpainting results. For that, we set a second pathway that utilizes StyleGAN’s mapping network to sample random z ’s and obtains W^+ codes that lie in the natural space of StyleGAN. The encoded, W^{enc} , and mapped, W^{rand} , latent codes are fed to the mixing network. We expect the mixing network to take the information of what is visible from the masked image and combine the missing information with the mapped latent codes. We equip the mixing network with a neural network and gating mechanism as follows:

$$\begin{aligned}
 W^{comb}, g &= \text{NN}(W^{enc}, W^{rand}) \\
 W^{out} &= \sigma(g) \cdot W^{comb} + (1 - \sigma(g)) \cdot W^{enc}
 \end{aligned} \tag{4.1}$$

where $\sigma(\cdot)$ is the sigmoid function. W^{out} goes to the instance normalization layers of StyleGAN generator.

In our experiments, we observe that the mixing network learns to ignore the W^{rand} . To achieve diversity, we propose a training pipeline with GAN-generated data as shown in Fig. 4.2.

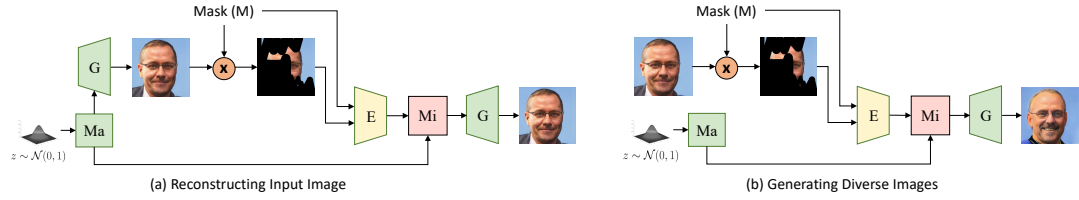


Figure 4.2: During training, we generate images with the StyleGAN generator. If we feed the mixing network (Mi) with the sampled z that generated the input image, we expect the output to match the input. The reconstruction losses are applied for both valid and invalid pixels. If we sample a new z as in (b), we expect to output diverse results and only apply image reconstruction loss for valid pixels and a GAN loss for the overall image.

We first generate an image with z_g , and feed it to the mapping network to obtain W_g^{rand} . We synthesize the output image and erase a part of the image with a randomly generated mask, $I_g^e = M \odot I_g$. Our encoder encodes the erased image to W_g^{enc} . With these encodings, we generate two images, one that combines W_g^{enc} with mapping of z_g and another one with mapping of a randomly sampled different z_r as given in Fig. 4.2 (a) and (b), respectively. The generator outputs images I_g^o and I_r^o from these two different paths. The generation of I_g^o has access to the full information that generated the image and therefore, if one can combine the encoded embedding with the mapping, one correctly can generate the input image. On the other hand, the second path, I_r^o image has only access to the information of unerased pixels and is not expected to reconstruct the input image faithfully for the erased parts. To achieve this objective, we set the following losses to train the framework.

Reconstruction Losses. For reconstructing the pixels correctly, we use L_2 and

perceptual losses. We use perceptual losses from VGG (Φ) at different feature layers (j) between images. We output two images, I_g^o and I_r^o from the input image (I_g) that is erased, I_g^e . I_g^o has access to the correct mapping as well as unerased image pixels, therefore this training pipeline is expected to generate the original input image as follows:

$$\mathcal{L}_{rg} = ||I_g^o - I_g||_2 + |\Phi_j(I_g^o) - \Phi_j(I_g)||_2 \quad (4.2)$$

On the other hand, I_r^o has only access to the unerased pixels of the original image and is expected to fill the missing information with a randomly sampled latent code. Therefore, we only expect pixel-wise matching on the unerased images. We find the reconstruction losses between the masked output image and erased input image as follows:

$$\mathcal{L}_{rr} = ||(M \odot I_r^o) - I_g^e||_2 + |\Phi_j(M \odot I_r^o) - \Phi_j(I_g^e)||_2 \quad (4.3)$$

If the network is trained with Eq. 4.2 alone, the mixing network can ignore the input image and encoded features coming from it since full information is provided by the mapping network. Similarly, if the network is trained with only Eq. 4.3, the mixing network ignores the mapping network. By using these reconstruction losses together, our framework learns to use the information encoded from both paths. Additionally, this training regularizes the encoder and mixing network to output latent codes that are in GAN’s natural space which provides editable inversions.

Adversarial Losses. We additionally output our final images as follows:

$$I^f = M \odot I + (1-M) \odot I^o \quad (4.4)$$

This step guarantees that the valid pixels are not modified as given in Fig. 4.1. We expect these final images to look realistic and use adversarial guidance on the final

images for both I_g^f and I_r^f . We load the pretrained discriminator from StyleGAN training, D , and train the discriminator together with the encoder and mixing network.

$$L_{adv} = 2 \log D(I_g) + \log (1 - D(I_g^f)) + \log (1 - D(I_r^f)) \quad (4.5)$$

Full Objective. We use the overall objectives given below to optimize the parameters of the Encoder (E) and the mixing network (Mi).

$$\min_{E, Mi} \max_D \lambda_a \mathcal{L}_{adv} + \lambda_{r1} \mathcal{L}_{rg} + \lambda_{r2} \mathcal{L}_{rr} \quad (4.6)$$

For both of the training stages, we use the same following hyperparameters, $\lambda_a = 8 \times 10^{-2}$, $\lambda_{r1} = 1$, and $\lambda_{r2} = 1$. We set the pixel-wise and VGG reconstruction loss coefficients as 1 and 5×10^{-5} , respectively for both L_{rg} and L_{gg} .

4.1.2 Second Stage: High-Fidelity Reconstruction

Previous works for image inversion show that low-rate latent codes, $W+$, do not have enough capacity for high-fidelity image reconstruction [28, 35]. There is too much information loss due to the bottleneck of projecting high-dimensional images to low-rate latent codes and this bottleneck limits the high-fidelity reconstructions. The same is observed by inpainting with image inversion works [37, 6] as well. The high-fidelity reconstruction of visible pixels is even more crucial in the inpainting task because we combine the erased and generated pixels as given in Eq. 4.4. When the generator is not able to reconstruct the valid pixels with high fidelity, the color discrepancies become very obvious in the final images. Previous methods [37, 6, 28, 35] suggest encoding information to higher-rate latent codes as well. This is achieved by skip connections from the encoder to the generator on higher-resolution feature maps which are sometimes referred to as F^+ space.

The balance of encoding information between W^+ and F^+ spaces is extensively studied for image inversion methods [28, 35, 36]. That is because by encoding most of the information to F^+ space, one can guarantee a high-fidelity reconstruction. However, since the image is not projected to W^+ properly, it will not be editable which is the main goal of many inversion methods. That is the reason we adopt the two-stage training pipeline similar to state-of-the-art image inversion methods [28, 35, 36]. In the first stage, our method tries to reconstruct and inpaint images only by the encoded low-rate codes, W^+ . Later, we reduce the information bottleneck by letting higher rate codes skipping from the input image to the generator.

The second-stage training takes the final output of the first-stage model that has gone through the masking as given in Eq. 4.4. We train an encoder with skip connections to the StyleGAN generator. The encoder also takes the binary mask as input and can detect inconsistencies in the pixel level. The same encoded W^+ is used for both generations. However, in the first generation, there are no skip connections (F^+), and in the second stage, there are. We also limit the number of skip connection layers and only feed to the feature maps with resolutions of 32×32 , 64×64 , and 128×128 . We fine-tune the models with the same objective from the first stage, given in Eq. 4.6. We provide the architectural details in Supplementary.

4.2 Experiments

Baselines. We first compare our method with state-of-the-art image inversion methods pSp [53], e4e [27], HFGI [28], ReStyle [54], and HyperStyle [35]. We use the author’s released code and train those models for inpainting tasks with an additional channel in the input for masks. Note that this is a different comparison than the display in Fig. 1.2 since we train these models for inpainting. pSp model outputs W^+ predictions for the generation. HFGI and HyperStyle methods use two-stage training. First, they rely on an encoder to output W^+ predictions, then a second encoder takes the input image and StyleGAN generated image with W^+ predictions. The goal is to encode the missed information to higher-rate latent codes. We feed erased images and masks to these inversion models and train them to reconstruct the visible pixels faithfully and output

realistic images overall.

Next, we experiment with state-of-the-art image inpainting models for our comparisons. We run inferences with CoModGAN’s pretrained models [23]. CoModGAN proposes to train a StyleGAN-like model from scratch but with co-modulation and skip connections for the inpainting task. InvertFill [6] and DualPath [37] models are built on pretrained StyleGAN models. InvertFill’s authors provide us with their inference results with the masks we provide. We implement DualPath following the set-up provided in their paper. DualPath trains a generator that also takes StyleGAN features.

Evaluation. We report Frechet Inception Distance (FID) metric [88], which looks at realism by comparing the target image distribution and reconstructed images. We also look at the diversity scores with Learned Perceptual Image Patch Similarity (LPIPS) [91]. That is for the models that provide diverse images, we generate two images per input image-mask pair and find the distance between these two images. We additionally use U-IDS and P-IDS metrics that are proposed by CoModGAN [23]. They measure unpaired and paired inception discriminative scores, respectively by measuring the linear separability in a pre-trained feature space. Our goal is to output images that are not separable from real images. SVM is fitted to the data and we expect a higher error rate.

Datasets. We use FFHQ [16], AFHQ-Cat, AFHQ-Dog [92], and Stanford Cars [93] image datasets. We use the StyleGAN2 pre-trained models [18] and use their train and validation splits. We build our extensive comparison with other works on FFHQ human face dataset following previous works. For the other datasets, we train the best single-stage (pSp) and two-stage encoders (HFGI, HyperStyle) for the comparisons.

Training Details. We train the first stage for 500k iterations with batch size of 8 on two GPUs. We use learning rate of 1×10^{-4} for all networks. We halve the learning rates at each 50k iterations. We use the overall objectives given below to optimize the parameters of the Encoder (E) and the mixing network (M_i) as was also given in the main paper. We also use the same objective to optimize the parameters of S in the second stage.

Quantitative Results. We provide the quantitative results in Table 4.1 for FFHQ evaluations.

Models	FID ↓	LPIPS ↑	U-IDS ↑	P-IDS ↑
e4e [27]	11.41	-	11.96	4.84
pSp [53]	8.23	-	14.05	6.60
ReStyle [54]	8.07	-	16.46	8.61
HFGI [28]	7.66	-	15.63	7.76
HyperStyle [35]	7.46	-	16.66	8.55
CoModGAN [23]	7.35	0.0790	9.78	3.79
InvertFill [6]	7.45	-	15.83	7.69
DualPath [37]	17.60	-	4.43	0.77
Ours	5.92	0.2026	18.43	10.75

Table 4.1: Quantitative results of our and the competing methods on the FFHQ validation dataset for mask ranges between 0-1.

We provide the LPIPS score for the models that provide diversity. The inversion methods specifically pSp, HFGI, and HyperStyle, that are trained for inpainting task achieves reasonable scores. Next, we compare with image inpainting methods. CoModGAN trains a network from scratch for inpainting whereas InvertFill uses a pre-trained StyleGAN similar to our work. CoModGAN has a StyleGAN-like training set-up, and layers are co-modulated with sampling and encoded image features. CoModGAN achieves diversity but is lower than our model when measured with the LPIPS score. InvertFill, on the other hand, does not have a mechanism to provide any stochastic at all. They achieve similar FID scores but InvertFill achieves significantly better U-IDS and P-IDS scores. We also compare with DualPath which is also built on pretrained StyleGAN. DualPath trains a new generator that takes input from StyleGAN feature maps. We follow the setup provided in their paper and try our best on parameter tuning. However, we do not achieve a good FID score with DualPath. Specifically, the generated images look smooth and semantically consistent, but they do not look sharp. DualPath also does not provide diverse results. We achieve significantly better results than previous methods and also achieve diversity.

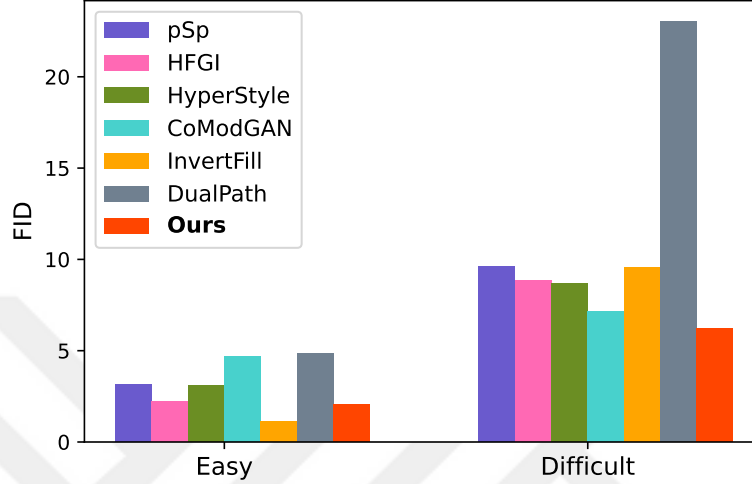


Figure 4.3: FID plots of ours and the competing methods for Easy (0.0-0.4) and Difficult (0.4-1.0) mask ranges on the FFHQ validation dataset.

We analyze how different methods score with different difficulties of masks in Fig. 4.3. We set two difficulty levels, an easy one where the erased mask ratio is between (0, 0.4) and a difficult scenario where the erased mask ratio is between (0.4-1.0). When the erased area only contains a small portion, deterministic models also achieve good results since most of the information can be encoded from the unerased parts. InvertFill achieves even a better score than ours and they all achieve better than CoModGAN. However, when the difficulty level increases, deterministic models struggle. CoModGAN, the other model that can output diverse results achieves the second-best results. InvertFill’s FID score significantly increase. Our model achieves good scores in both scenarios.

We also compare with pSp, HFGI, and HyperStyle on AFHQ-Dog, AFHQ-Cat, and Stanford Cars datasets as shown in Table 4.2. Our method achieves better results than those encoders.

	AFHQ-Dog	AFHQ-Cat	Stanford Cars
Models	FID ↓	FID ↓	FID ↓
pSp [53]	26.46	14.26	11.77
HFGI [28]	20.58	17.01	8.55
HyperStyle [35]	23.92	13.43	10.97
Ours	18.89	12.28	8.39

Table 4.2: Quantitative results of our and the competing methods on AFHQ-Dog, AFHQ-Cat, and Stanford Cars validation datasets for mask ranges between 0-1.

Qualitative Results. Fig. 4.4 shows the results of our and competing methods on the FFHQ dataset. As we mention in the Quantitative Results section, DualPath outputs blurry parts on the inpainted areas in our trainings. The pSp model achieves good semantic consistency. However, the boundaries between the inpainted and original pixels are quite apparent. HFGI and HyperStyle output artifacts on many of the examples. CoModGAN and InvertFill achieve plausible predictions. An interesting observation we make is that all other methods output similar identities for given inputs. For example, the last two row predictions of CoModGAN and InvertFill are similar to each other. On the other hand, since our model samples diverse predictions (more diverse than CoModGAN), our results look quite different. Our results better capture the data distribution as is also measured by FID scores. We also present the inpainting results of our method on AFHQ cat and dog images and Stanford car images in Fig. 4.5. Our method achieves successful inpaints on these image domains as well.

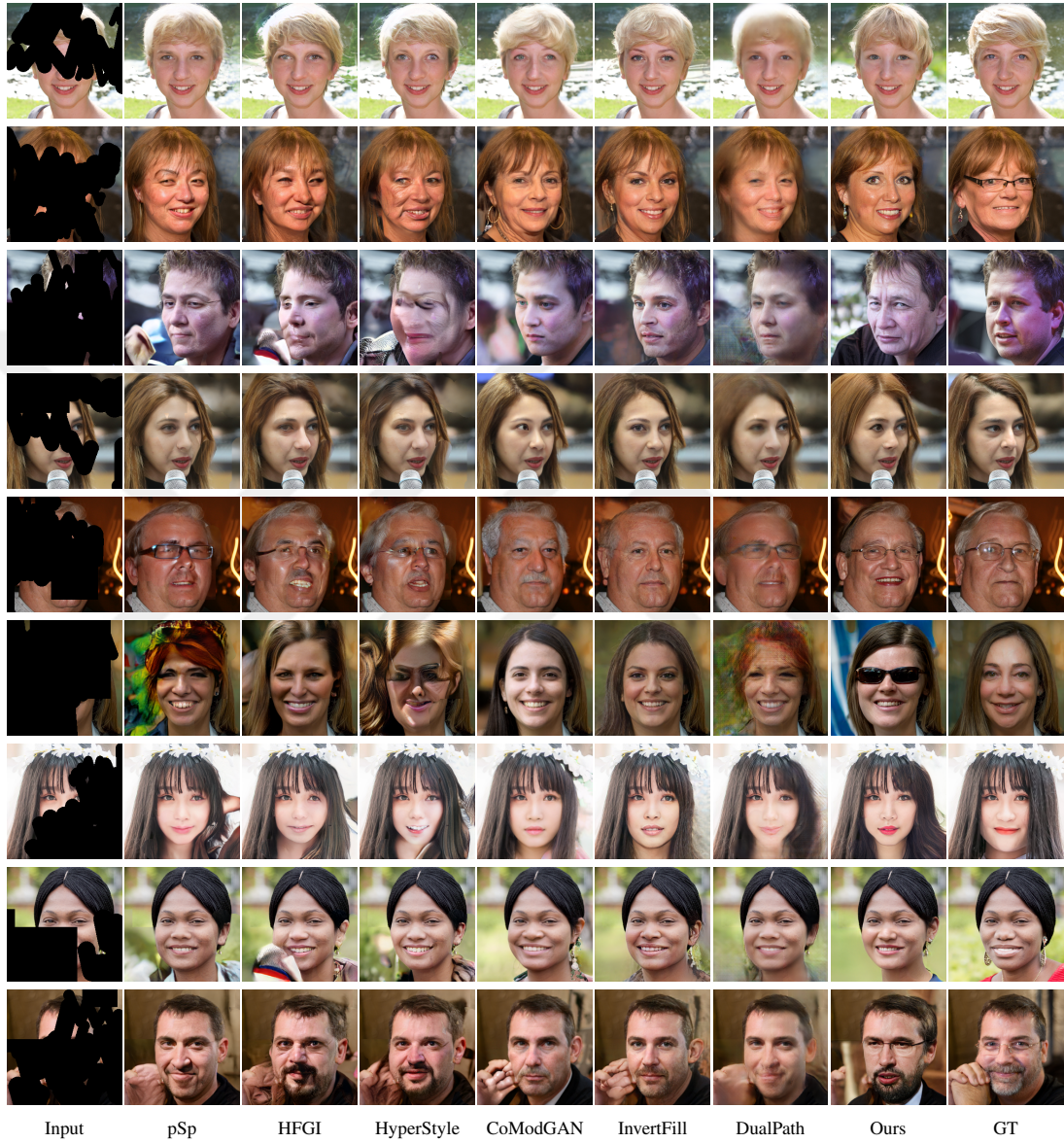


Figure 4.4: Qualitative results of our and the competing methods on the FFHQ dataset. GT refers to original images.

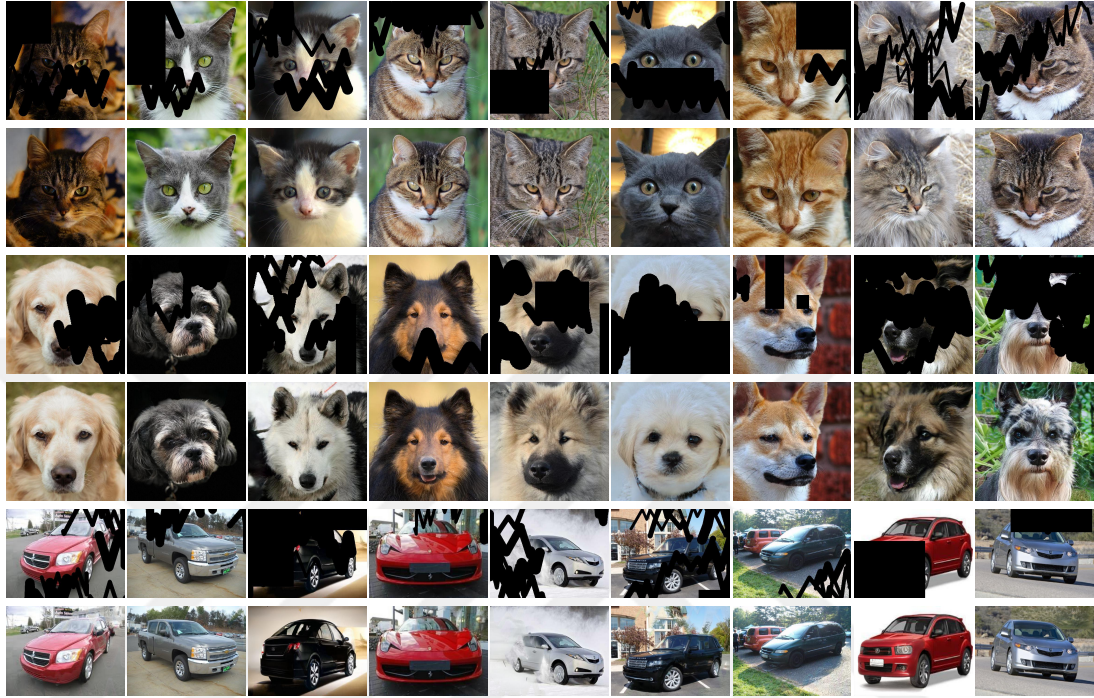


Figure 4.5: Qualitative results of AFHQ-Cat, AFHQ-Dog, and Stanford Cars dataset trainings.

Ablation Study. We present the quantitative and qualitative results of our ablation study in Table 4.3 and Fig. 4.6, respectively. During the development of our model, we track the diversity (LPIPS) and quality (FID) of our results. LPIPS score alone does not reflect the quality of our results because random images can be generated by StyleGAN that are semantically inconsistent with valid pixels of the input image. High LPIPS scores can be obtained from those images. In fact, during training, the LPIPS score starts from a high point and decreases as the framework learns to output consistent pixels with the unerased parts. Therefore, it is important to achieve a reasonable FID and LPIPS together.

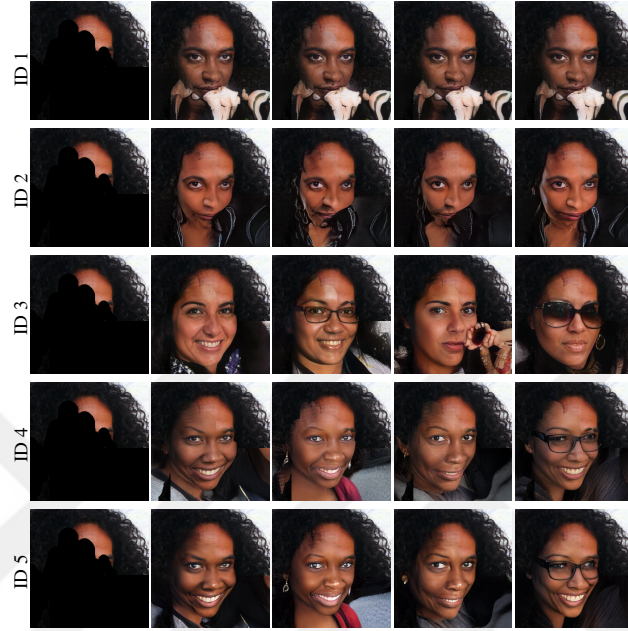


Figure 4.6: Qualitative results of the ablation study. Please refer to Table 4.3 for the training-setting of model IDs.

We assign IDs to each model in Table 4.3 to easily refer to them in the text and figures. Model 1 is trained with real images to reconstruct the unerased parts together with adversarial loss. The objective does not have the full-reconstruction image loss based on the sampled $W+$. The model does not achieve any diversity even though the network architecture takes sampled $W+$ via the mixing network. The mixing network ignores the stochastic input as can be seen from Fig. 4.6. In Model 2, we also add the second stage training which improves the FID results because it removes the color inconsistencies. The diversity is negligible as can be seen from both Fig. 4.6 and Table 4.3.

ID	Full Recons	Gated Mi	Second Stage	FID	LPIPS
1		✓		10.92	0.0073
2		✓	✓	7.87	0.0855
3	✓			20.91	0.2095
4	✓	✓		16.65	0.1440
5	✓	✓	✓	5.92	0.2026

Table 4.3: Quantitative results of our ablation study on the FFHQ validation dataset for the mask of range 0-1. We assign model IDs to easily refer to the model in the text and Fig. 4.6.

Next, we compare methods that are trained with our proposed losses and training pipeline. Model 3 does not have the gated mixing network, instead, it employs a fully connected layer to combine encoded and sampled latent codes. Model 4 has the gated mixing network as given in Eq. 4.1.

In our experiments, we find that the proposed mixing network does a better job of combining features. Model 3 sometimes generates inconsistent results as can be seen in Fig. 4.6 and achieves worse FID scores. It has a higher LPIPS score but this diversity result in semantic inconsistencies. As mentioned before, StyleGAN model that randomly outputs images will achieve the best diversity, but will not output consistent pixels with the erased part. For example, the input image may have half-erased eyeglasses and this network may not output eyeglasses for the visible parts. The gated mixing network achieves better in those scenarios.

Finally, we add the second stage of training and train skip connections to obtain our final model. The color discrepancies disappear after this stage of training. Interestingly, our not-diverse model - Model 1 achieves better scores than our diverse model - Model 4. However, when second-stage training is added, the diverse model achieves better results. That is because the diverse model in the first stage has more difficulties outputting perfectly semantically consistent pixels but it outputs meaningful and diverse parts. Especially when the erased parts are large, the diverse model does a better job as was discussed for Fig. 4.3 because it augments the encoded features with sampled ones. The color discrepancies are resolved with the second stage of training

bringing the diverse model to a better score than the non-diverse model.

Editing Results. For our editing results, we first encode the erased image and randomly sample a $W+$ to our mixing network. We edit the output of the mixing network with InterfaceGAN directions [29]. Editing results are shown in Fig. 4.7.

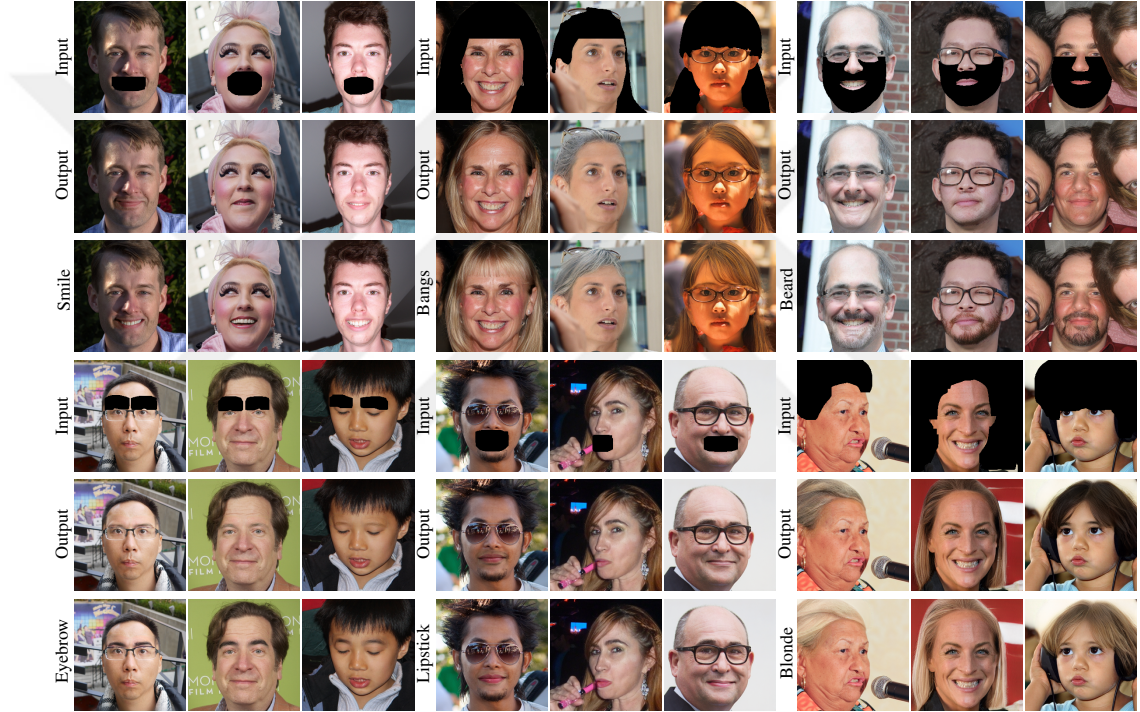


Figure 4.7: Editing results of our method. Our method achieves diverse inpainting and editing under one framework. First rows show the erased inputs. Output rows show the randomly sampled $W+$ results and the next rows show after adding InterfaceGAN directions for different attributes.

First, we show the erased input. Output rows show the randomly sampled $W+$ results and the next rows show after adding InterfaceGAN directions for different attributes. We show smile, bangs, and beard addition results as well as image edits with blush eyebrows, blonde hair, and lipstick addition. Our model can successfully both inpaint and edit erased images thanks to the semantically rich feature representations of the StyleGAN model bringing more capabilities under one framework.

Chapter 5

Conclusion

We introduced two novel approaches, Inst-Inpaint and DivInversion, to image inpainting. Inst-Inpaint is a diffusion-based inpainting model that can remove objects from images based on textual prompts without the need for binary masks. Inpainting with a text prompt is more user-friendly than drawing masks due to its intuitive nature, allowing users to describe changes using natural language. This method enhances efficiency, as generating text prompts is faster than creating detailed masks, especially for complex images. To achieve this task, we created a new benchmark dataset, GQA-Inpaint, consisting of source and target images and textual removal prompts referring to the object to be removed according to the relations around it in the scene. We trained latent diffusion models on this paired dataset and observed that the model can learn which object to remove based on a given text prompt. Additionally, we suggested novel evaluation metrics that assessed whether objects were correctly removed, which proved valuable when combined with other metrics that solely measure the quality of the output images. Our work has demonstrated the potential of using textual instructions as a more user-friendly and natural way of controlling image inpainting.

Our second work, DivInversion, is a GAN-based inpainting model that can achieve diverse image inpainting and editing with GAN inversion. In this work, we revert to traditional masking to outline the pixels to be removed. However, we have enhanced this approach by adding capabilities for diversity and editability. To the best of our

knowledge, our framework is the first to achieve these capabilities simultaneously. Applying GAN inversion in the inpainting task allows us to edit the inpainted areas via the editing methods proposed for StyleGAN. Additionally, we present a two-path loss to achieve diverse inpainting results. We show the effectiveness of our contributions with an extensive ablation study. Our comparisons with competing methods demonstrate that we achieved significantly better scores than previous works in terms of both diversity and quality.



Bibliography

- [1] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, “Patchmatch: A randomized correspondence algorithm for structural image editing,” *ACM Trans. Graph.*, vol. 28, no. 3, p. 24, 2009.
- [2] 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*, pp. 2536–2544, 2016.
- [3] G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao, and B. Catanzaro, “Image inpainting for irregular holes using partial convolutions,” in *Proceedings of the European conference on computer vision (ECCV)*, pp. 85–100, 2018.
- [4] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang, “Free-form image inpainting with gated convolution,” in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 4471–4480, 2019.
- [5] G. Liu, A. Dundar, K. J. Shih, T.-C. Wang, F. A. Reda, K. Sapra, Z. Yu, X. Yang, A. Tao, and B. Catanzaro, “Partial convolution for padding, inpainting, and image synthesis,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.
- [6] Y. Yu, L. Zhang, H. Fan, and T. Luo, “High-fidelity image inpainting with gan inversion,” in *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XVI*, pp. 242–258, Springer, 2022.
- [7] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, “High-resolution image synthesis with latent diffusion models,” in *Proceedings of the IEEE/CVF*

- Conference on Computer Vision and Pattern Recognition*, pp. 10684–10695, 2022.
- [8] O. Bar-Tal, D. Ofri-Amar, R. Fridman, Y. Kasten, and T. Dekel, “Text2live: Text-driven layered image and video editing,” in *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XV*, pp. 707–723, Springer, 2022.
 - [9] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, “Hierarchical text-conditional image generation with clip latents,” *arXiv preprint arXiv:2204.06125*, 2022.
 - [10] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, “High-resolution image synthesis with latent diffusion models,” 2021.
 - [11] 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,” *arXiv preprint arXiv:2205.11487*, 2022.
 - [12] J. Yu, Y. Xu, J. Y. Koh, T. Luong, G. Baid, Z. Wang, V. Vasudevan, A. Ku, Y. Yang, B. K. Ayan, B. Hutchinson, W. Han, Z. Parekh, X. Li, H. Zhang, J. Baldridge, and Y. Wu, “Scaling autoregressive models for content-rich text-to-image generation,” *Transactions on Machine Learning Research*, 2022. Featured Certification.
 - [13] O. Avrahami, D. Lischinski, and O. Fried, “Blended diffusion for text-driven editing of natural images,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18208–18218, 2022.
 - [14] D. A. Hudson and C. D. Manning, “Gqa: A new dataset for real-world visual reasoning and compositional question answering,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6700–6709, 2019.
 - [15] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, “Learning transferable visual models from natural language supervision,” in *Proceedings of the*

- 38th International Conference on Machine Learning* (M. Meila and T. Zhang, eds.), vol. 139 of *Proceedings of Machine Learning Research*, pp. 8748–8763, PMLR, 18–24 Jul 2021.
- [16] T. Karras, S. Laine, and T. Aila, “A style-based generator architecture for generative adversarial networks,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4401–4410, 2019.
 - [17] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, “Self-attention generative adversarial networks,” in *International conference on machine learning*, pp. 7354–7363, PMLR, 2019.
 - [18] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, “Analyzing and improving the image quality of stylegan,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8110–8119, 2020.
 - [19] N. Yu, G. Liu, A. Dundar, A. Tao, B. Catanzaro, L. S. Davis, and M. Fritz, “Dual contrastive loss and attention for gans,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6731–6742, 2021.
 - [20] 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*, pp. 8798–8807, 2018.
 - [21] A. Dundar, K. Sapra, G. Liu, A. Tao, and B. Catanzaro, “Panoptic-based image synthesis,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8070–8079, 2020.
 - [22] J. Li, N. Wang, L. Zhang, B. Du, and D. Tao, “Recurrent feature reasoning for image inpainting,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7760–7768, 2020.
 - [23] S. Zhao, J. Cui, Y. Sheng, Y. Dong, X. Liang, E. I. Chang, and Y. Xu, “Large scale image completion via co-modulated generative adversarial networks,” *arXiv preprint arXiv:2103.10428*, 2021.

- [24] R. Abdal, P. Zhu, N. J. Mitra, and P. Wonka, “Styleflow: Attribute-conditioned exploration of stylegan-generated images using conditional continuous normalizing flows,” *ACM Transactions on Graphics (ToG)*, vol. 40, no. 3, pp. 1–21, 2021.
- [25] Z. Chen, R. Jiang, B. Duke, H. Zhao, and P. Aarabi, “Exploring gradient-based multi-directional controls in gans,” in *European Conference on Computer Vision*, pp. 104–119, Springer, 2022.
- [26] Y. Dalva, S. F. Altındış, and A. Dundar, “Vecgan: Image-to-image translation with interpretable latent directions,” in *European Conference on Computer Vision*, pp. 153–169, Springer, 2022.
- [27] O. Tov, Y. Alaluf, Y. Nitzan, O. Patashnik, and D. Cohen-Or, “Designing an encoder for stylegan image manipulation,” *ACM Transactions on Graphics (TOG)*, vol. 40, no. 4, pp. 1–14, 2021.
- [28] T. Wang, Y. Zhang, Y. Fan, J. Wang, and Q. Chen, “High-fidelity gan inversion for image attribute editing,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11379–11388, 2022.
- [29] Y. Shen, J. Gu, X. Tang, and B. Zhou, “Interpreting the latent space of gans for semantic face editing,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9243–9252, 2020.
- [30] E. Härkönen, A. Hertzmann, J. Lehtinen, and S. Paris, “Ganspace: Discovering interpretable gan controls,” *Advances in Neural Information Processing Systems*, vol. 33, 2020.
- [31] O. Patashnik, Z. Wu, E. Shechtman, D. Cohen-Or, and D. Lischinski, “Styleclip: Text-driven manipulation of stylegan imagery,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2085–2094, 2021.
- [32] A. Creswell and A. A. Bharath, “Inverting the generator of a generative adversarial network,” *IEEE transactions on neural networks and learning systems*, vol. 30, no. 7, pp. 1967–1974, 2018.

- [33] D. Roich, R. Mokady, A. H. Bermano, and D. Cohen-Or, “Pivotal tuning for latent-based editing of real images,” *ACM Transactions on Graphics (TOG)*, vol. 42, no. 1, pp. 1–13, 2022.
- [34] J. Zhu, Y. Shen, D. Zhao, and B. Zhou, “In-domain gan inversion for real image editing,” in *European conference on computer vision*, pp. 592–608, Springer, 2020.
- [35] Y. Alaluf, O. Tov, R. Mokady, R. Gal, and A. Bermano, “Hyperstyle: Style-gan inversion with hypernetworks for real image editing,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18511–18521, 2022.
- [36] H. Pehlivan, Y. Dalva, and A. Dundar, “Styleres: Transforming the residuals for real image editing with stylegan,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1828–1837, 2023.
- [37] W. Wang, L. Niu, J. Zhang, X. Yang, and L. Zhang, “Dual-path image inpainting with auxiliary gan inversion,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11421–11430, 2022.
- [38] A. B. Yildirim, H. Pehlivan, B. B. Bilecen, and A. Dundar, “Diverse inpainting and editing with gan inversion,” *arXiv preprint arXiv:2307.15033*, 2023.
- [39] W. Li, Z. Lin, K. Zhou, L. Qi, Y. Wang, and J. Jia, “Mat: Mask-aware transformer for large hole image inpainting,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10758–10768, 2022.
- [40] 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*, pp. 11461–11471, 2022.
- [41] A. Q. 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*, pp. 16784–16804, PMLR, 2022.

- [42] S. Xie, Z. Zhang, Z. Lin, T. Hinz, and K. Zhang, “Smartbrush: Text and shape guided object inpainting with diffusion model,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22428–22437, 2023.
- [43] A. B. Yildirim, V. Baday, E. Erdem, A. Erdem, and A. Dundar, “Inst-inpaint: Instructing to remove objects with diffusion models,” *arXiv preprint arXiv:2304.03246*, 2023.
- [44] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial networks,” *arXiv preprint arXiv:1406.2661*, 2014.
- [45] A. Kammoun, R. Slama, H. Tabia, T. Ouni, and M. Abid, “Generative adversarial networks for face generation: A survey,” *ACM Computing Surveys*, vol. 55, no. 5, pp. 1–37, 2022.
- [46] A. Voynov and A. Babenko, “Unsupervised discovery of interpretable directions in the gan latent space,” in *International Conference on Machine Learning*, pp. 9786–9796, PMLR, 2020.
- [47] Y. Choi, Y. Uh, J. Yoo, and J.-W. Ha, “Stargan v2: Diverse image synthesis for multiple domains,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.
- [48] T. Xiao, J. Hong, and J. Ma, “Elegant: Exchanging latent encodings with gan for transferring multiple face attributes,” in *Proceedings of the European conference on computer vision (ECCV)*, pp. 168–184, 2018.
- [49] G. Zhang, M. Kan, S. Shan, and X. Chen, “Generative adversarial network with spatial attention for face attribute editing,” in *Proceedings of the European conference on computer vision (ECCV)*, pp. 417–432, 2018.
- [50] X. Li, S. Zhang, J. Hu, L. Cao, X. Hong, X. Mao, F. Huang, Y. Wu, and R. Ji, “Image-to-image translation via hierarchical style disentanglement,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8639–8648, 2021.

- [51] Y. Gao, F. Wei, J. Bao, S. Gu, D. Chen, F. Wen, and Z. Lian, “High-fidelity and arbitrary face editing,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 16115–16124, 2021.
- [52] Z. Wu, D. Lischinski, and E. Shechtman, “Stylespace analysis: Disentangled controls for stylegan image generation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12863–12872, 2021.
- [53] E. Richardson, Y. Alaluf, O. Patashnik, Y. Nitzan, Y. Azar, S. Shapiro, and D. Cohen-Or, “Encoding in style: a stylegan encoder for image-to-image translation,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2287–2296, 2021.
- [54] Y. Alaluf, O. Patashnik, and D. Cohen-Or, “Restyle: A residual-based stylegan encoder via iterative refinement,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6711–6720, 2021.
- [55] J. Ho, A. Jain, and P. Abbeel, “Denoising diffusion probabilistic models,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 6840–6851, 2020.
- [56] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *Medical image computing and computer-assisted intervention—MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pp. 234–241, Springer, 2015.
- [57] 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,” *arXiv preprint arXiv:2112.10741*, 2021.
- [58] O. Avrahami, T. Hayes, O. Gafni, S. Gupta, Y. Taigman, D. Parikh, D. Lischinski, O. Fried, and X. Yin, “Spatext: Spatio-textual representation for controllable image generation,” *arXiv preprint arXiv:2211.14305*, 2022.
- [59] Y. Balaji, S. Nah, X. Huang, A. Vahdat, J. Song, K. Kreis, M. Aittala, T. Aila, S. Laine, B. Catanzaro, *et al.*, “ediffi: Text-to-image diffusion models with an ensemble of expert denoisers,” *arXiv preprint arXiv:2211.01324*, 2022.

- [60] J. Z. Wu, Y. Ge, X. Wang, W. Lei, Y. Gu, W. Hsu, Y. Shan, X. Qie, and M. Z. Shou, “Tune-a-video: One-shot tuning of image diffusion models for text-to-video generation,” *arXiv preprint arXiv:2212.11565*, 2022.
- [61] U. Singer, A. Polyak, T. Hayes, X. Yin, J. An, S. Zhang, Q. Hu, H. Yang, O. Ashual, O. Gafni, *et al.*, “Make-a-video: Text-to-video generation without text-video data,” *arXiv preprint arXiv:2209.14792*, 2022.
- [62] C. Meng, Y. He, Y. Song, J. Song, J. Wu, J.-Y. Zhu, and S. Ermon, “Sdedit: Guided image synthesis and editing with stochastic differential equations,” in *International Conference on Learning Representations*, 2021.
- [63] A. Hertz, R. Mokady, J. Tenenbaum, K. Aberman, Y. Pritch, and D. Cohen-Or, “Prompt-to-prompt image editing with cross attention control,” *arXiv preprint arXiv:2208.01626*, 2022.
- [64] Y. Song, Z. Zhang, Z. Lin, S. Cohen, B. Price, J. Zhang, S. Y. Kim, and D. Aliaga, “Objectstitch: Generative object compositing,” *arXiv preprint arXiv:2212.00932*, 2022.
- [65] T. Brooks, A. Holynski, and A. A. Efros, “Instructpix2pix: Learning to follow image editing instructions,” *arXiv preprint arXiv:2211.09800*, 2022.
- [66] J. Singh, S. Gould, and L. Zheng, “High-fidelity guided image synthesis with latent diffusion models,” *arXiv preprint arXiv:2211.17084*, 2022.
- [67] J.-H. Kim, N. Kitaev, X. Chen, M. Rohrbach, Y. Tian, D. Batra, and D. Parikh, “CoDraw: Collaborative Drawing as a Testbed for Grounded Goal-driven Communication,” *arXiv preprint arXiv:1712.05558*, 2019.
- [68] 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*, pp. 2901–2910, 2017.
- [69] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, “Detectron2.” <https://github.com/facebookresearch/detectron2>, 2019.

- [70] X. Zhou, R. Girdhar, A. Joulin, P. Krähenbühl, and I. Misra, “Detecting twenty-thousand classes using image-level supervision,” in *ECCV*, 2022.
- [71] 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 *Computer Vision—ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6–12, 2014, Proceedings, Part V 13*, pp. 740–755, Springer, 2014.
- [72] A. Gupta, P. Dollar, and R. Girshick, “LVIS: A dataset for large vocabulary instance segmentation,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 5356–5364, 2019.
- [73] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask r-cnn,” in *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.
- [74] Z. Cai and N. Vasconcelos, “Cascade r-cnn: high quality object detection and instance segmentation,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 5, pp. 1483–1498, 2019.
- [75] A. Kirillov, R. Girshick, K. He, and P. Dollár, “Panoptic feature pyramid networks,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6399–6408, 2019.
- [76] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, “Swin transformer: Hierarchical vision transformer using shifted windows,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.
- [77] 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*, pp. 248–255, 2009.
- [78] Y. Zeng, Z. Lin, H. Lu, and V. M. Patel, “Cr-fill: Generative image inpainting with auxiliary contextual reconstruction,” in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 14164–14173, 2021.
- [79] R. Suvorov, E. Logacheva, A. Mashikhin, A. Remizova, A. Ashukha, A. Silvestrov, N. Kong, H. Goka, K. Park, and V. Lempitsky, “Resolution-robust large

- mask inpainting with fourier convolutions,” in *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pp. 2149–2159, 2022.
- [80] H. K. Cheng, J. Chung, Y.-W. Tai, and C.-K. Tang, “CascadePSP: Toward class-agnostic and very high-resolution segmentation via global and local refinement,” in *CVPR*, 2020.
- [81] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” *Advances in neural information processing systems*, vol. 30, 2017.
- [82] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.
- [83] X. Zou, Z.-Y. Dou, J. Yang, Z. Gan, L. Li, C. Li, X. Dai, H. Behl, J. Wang, L. Yuan, *et al.*, “Generalized decoding for pixel, image, and language,” *arXiv preprint arXiv:2212.11270*, 2022.
- [84] T. Lüddecke and A. Ecker, “Image segmentation using text and image prompts,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7086–7096, 2022.
- [85] T. Zhang, H.-Y. Tseng, L. Jiang, W. Yang, H. Lee, and I. Essa, “Text as neural operator: Image manipulation by text instruction,” in *Proceedings of the 29th ACM International Conference on Multimedia*, pp. 1893–1902, 2021.
- [86] W.-C. Fan, C.-F. Yang, C.-A. Yang, and Y.-C. F. Wang, “Target-free text-guided image manipulation,” in *AAAI*, 2023.
- [87] N. Vo, L. Jiang, C. Sun, K. Murphy, L.-J. Li, L. Fei-Fei, and J. Hays, “Composing text and image for image retrieval-an empirical odyssey,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6439–6448, 2019.
- [88] 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,” *Advances in neural information processing systems*, vol. 30, 2017.

- [89] S. Jayasumana, S. Ramalingam, A. Veit, D. Glasner, A. Chakrabarti, and S. Kumar, “Rethinking fid: Towards a better evaluation metric for image generation,” *arXiv preprint arXiv:2401.09603*, 2023.
- [90] A. El-Nouby, S. Sharma, H. Schulz, D. Hjelm, L. E. Asri, S. E. Kahou, Y. Bengio, and G. W. Taylor, “Tell, draw, and repeat: Generating and modifying images based on continual linguistic instruction,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10304–10312, 2019.
- [91] 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*, pp. 586–595, 2018.
- [92] Y. Choi, Y. Uh, J. Yoo, and J.-W. Ha, “Stargan v2: Diverse image synthesis for multiple domains,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8188–8197, 2020.
- [93] J. Krause, M. Stark, J. Deng, and L. Fei-Fei, “3d object representations for fine-grained categorization,” in *Proceedings of the IEEE international conference on computer vision workshops*, pp. 554–561, 2013.