BACHELORARBEIT

Conditioning Diffusion Models for image manipulation

vorgelegt am 19. Februar 2024

Dennis Räk

Erstprüferin: Prof. Dr. Larissa Putzar
Zweitprüfer: Thorben Ortmann

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Department Medientechnik
Finkenau 35
22081 Hamburg

Abstract

This bachelor thesis evaluates Conditional Diffusion Models, a rapidly emerging class of Generative Models. Although Conditional Diffusion Models have a wide range of applications, this thesis focuses on analyzing their usage in the realm of image manipulation. It accomplishes this by reviewing the current state of the Unconditional and Conditional Diffusion Model. Through various image manipulation experiments, the findings on Conditional Diffusion Models are examined. In the final discussion, all discoveries are put into a broader context, drawing a conclusion about the capabilities, limitations and implications of Conditional Diffusion Models.

Abstrakt

Diese Bachelorarbeit analysiert Konditionelle Diffusionsmodelle, eine immer mehr an Popularität gewinnende Klasse Generativer Modelle. Obwohl Konditionelle Diffusionsmodelle in einer Vielzahl von Bereichen angewendet werden können, befasst sich diese Bachelorarbeit speziell mit dem Bereich der Bildbearbeitung. Dies wird realisiert, indem der aktuelle Stand der Unkonditionellen und Konditionellen Diffusionsmodelle untersucht wird. Anhand verschiedener Bildmanipulationsexperimente werden die Erkenntnisse des Konditionellen Diffusionsmodells auf die Probe gestellt. In der abschließenden Diskussion werden alle Ergebnisse in einen umfassenden Kontext eingeordnet und Schlussfolgerungen über die Möglichkeiten, Grenzen sowie Implikationen von Konditionellen Diffusionsmodellen gezogen.

Contents

List of Figures IV

1 Introduction 1
- 1.1 Aim and Motivation . . . 1
- 1.2 Outline . . . 1
- 1.3 Notation . . . 2

2 Unconditional Diffusion Models 3
- 2.1 What are Generative Models? . . . 3
  - 2.1.1 Implicit Generative Models . . . 3
  - 2.1.2 Explicit Generative Models . . . 4
- 2.2 Diffusion Model – A high-level approach . . . 4
- 2.3 Score-Based Generative Models . . . 5
  - 2.3.1 Score Matching . . . 5
  - 2.3.2 Langevin Dynamics . . . 7
- 2.4 Denoising Diffusion Probabilistic Models . . . 7
  - 2.4.1 Forward Process . . . 7
  - 2.4.2 Reverse Process . . . 8
  - 2.4.3 Sampling . . . 9
- 2.5 Score-Based Generative Modeling through Stochastic Differential Equations 9
  - 2.5.1 Equivalence of NCSN and DDPM . . . 9
  - 2.5.2 Stochastic Differential Equation . . . 10
  - 2.5.3 SDE and ODE solver . . . 10
- 2.6 U-Net – The Backbone of Diffusion Models . . . 11
  - 2.6.1 Architecture . . . 11
  - 2.6.2 Residual Blocks . . . 12
  - 2.6.3 Sinusoidal Position Embeddings . . . 12
  - 2.6.4 Attention Module . . . 13

3 Conditional Diffusion Model **14**
3.1 Fundamentals of the Conditional Diffusion Model . . . 14
3.1.1 Mathematical Foundations . . . 14
3.1.2 SDEdit: Guided Image Synthesis and Editing with Stochastic Differential Equations . . . 16
3.1.3 ControlNet – Advanced Conditioning . . . 17
3.2 Architectural Variations and Commercial Models . . . 18
3.2.1 DALL-E – Hierarchical Text-Conditional Image Generation with CLIP Latents . . . 18
3.2.2 CDM – Cascaded Diffusion Models . . . 20
3.2.3 LDM – Latent Diffusion Model . . . 21
3.3 Personalization through Fine-Tuning . . . 25
3.3.1 DreamBooth . . . 25
3.3.2 LoRA – Low-Rank Adaptation . . . 26
3.3.3 Textual Inversion . . . 28

4 Experiments **30**
4.1 Tools . . . 30
4.1.1 Google Colab . . . 30
4.1.2 HuggingFace . . . 31
4.2 Experiment 1: Turning a sketch into an image . . . 32
4.2.1 Methodology . . . 32
4.2.2 SDEdit . . . 33
4.2.3 Classifier-Free Guidance . . . 33
4.2.4 ControlNet . . . 34
4.2.5 Evaluation . . . 36
4.3 Experiment 2: Inpainting . . . 37
4.3.1 Methodology . . . 37
4.3.2 DALL-E 2 . . . 38
4.3.3 Cascaded Diffusion Model . . . 38
4.3.4 LDM – Latent Diffusion Model . . . 42
4.3.5 Evaluation . . . 43
4.4 Experiment 3: Personalization . . . 45
4.4.1 Methodology . . . 45
4.4.2 Dreambooth . . . 45
4.4.3 Low-Rank Adaptation . . . 47
4.4.4 Textual Inversion . . . 48

4.4.5 Evaluation 49

5 Resolution **50**

5.1 Discussion 50

5.1.1 The Generative Learning Trilemma 50

5.1.2 Photorealism 52

5.1.3 Societal Impact 54

5.2 Conclusion 56

Bibliography **57**

Appendix **61**

.1 Notation notes 61

List of Figures

2.1 Score Model, Perturbed Scores (Y. Song, 2021) . . . 6
2.2 DDPM, Noise Schedule (Linear, Cosine) (Nichol and Dhariwal, 2021) . . . 7
2.3 U-Net, Architecture (Ronneberger et al., 2015) . . . 11
2.4 U-Net, Residual Block (He et al., 2015) . . . 12

3.1 SDEdit, Mechanism (Meng et al., 2022) . . . 16
3.2 SDEdit, Examples (Meng et al., 2022) . . . 16
3.3 ControlNet, Model (Zhang et al., 2023) . . . 17
3.4 ControlNet, Examples (Zhang et al., 2023) . . . 17
3.5 DALL-E 2, Model (Ramesh et al., 2022) . . . 19
3.6 CDM, Model (Ho et al., 2021) . . . 21
3.7 LDM, Model (Rombach et al., 2022) . . . 22
3.8 LDM, Stable Diffusion XL (Podell et al., 2023) . . . 24
3.9 DreamBooth, Loss (Ruiz et al., 2023) . . . 25
3.10 DreamBooth, Synthesis (Ruiz et al., 2023) . . . 25
3.11 LoRA, Model (Hu et al., 2021) . . . 27
3.12 LoRA, Example (Cuenca and Sayak, 2023) . . . 28
3.13 Textual Inversion, Process (Gal et al., 2022) . . . 29
3.14 Textual Inversion, Samples (Gal et al., 2022) . . . 29

4.1 Experiment 1, Raphael's 'Young Woman on a Balcony' (Foundation, 2024) . . . 32
4.2 Experiment 1, SDEdit $t_0 = [0.0, 0.2, 0.4, 0.6, 0.8, 1]$ . . . 33

4.3 Experiment 1, Classifier-Free Guidance $\gamma = [1, 2.5, 5.0, 10.0, 20.0, 40.0]$ . . . 34
4.4 Experiment 1, ControlNet (Pre-Processing) . . . 35
4.5 Experiment 1, ControlNet (Results) . . . 35
4.6 Experiment 2, Images . . . 37
4.7 Experiment 2, Masks . . . 37
4.8 Experiment 2, DALL-E 2 . . . 38
4.9 Experiment 2, Deep Floyd IF Schematic (Shonenkov et al., 2024) . . . 39
4.10 Experiment 2, Cascaded Diffusion Model, Stage 1 . . . 40
4.11 Experiment 2, Cascaded Diffusion Model, Stage 2 . . . 40
4.12 Experiment 2, Cascaded Diffusion Model, Stage 3 . . . 41
4.13 Experiment 2, Cascaded Diffusion Model, Composite . . . 41
4.14 Experiment 2, Latent Diffusion Model . . . 42
4.15 Experiment 2, Autoencoder Artefacts . . . 42
4.16 Experiment 2, Latent Diffusion Model, Composite . . . 43
4.17 Experiment 3, Dataset . . . 45
4.18 Experiment 3, DreamBooth . . . 46
4.19 Experiment 3, LoRA . . . 47
4.20 Experment 3, Textual Inversion . . . 48

5.1 The Generative Learning Trilemma (Xiao et al., 2022) . . . 50
5.2 Pixel-based (Y. Song and Ermon, 2020) . . . 53
5.3 (a) Noise (b) No Noise (Karras et al., 2019) . . . 53
5.4 Deep Fake, Pope Franziskus (Huang, 2023) . . . 54

1 Introduction

This introductory chapter provides an overview of the bachelor thesis. It offers insights into its motivation, outline and notation providing the reader with a quick summary of all topics explored within the main body.

1.1 Aim and Motivation

Generative Models have been increasing in popularity which is largely attributed to advancements in Diffusion Models. Since their introduction by Sohl-Dickstein et al., 2015, they have evolved into one of the most important classes of Generative Models.

Understanding how they can be conditioned in the realm of image manipulation can offer significant insights for the creative industry, society and more importantly future research. Therefore, this bachelor thesis focuses on the capabilities and limitations of the Conditional Diffusion Model. It does so by explaining the theoretical foundations of the Unconditional and Conditional Diffusion Model. Following this, these theories and concepts are evaluated through various experiments. Last but not least, a final resolution discusses the limitations and implications of all findings and brings them into a wider context culminating in a final conclusion.

All in all, this thesis aims to investigate the capabilities of the Conditional Diffusion Model and uses these findings to evaluate the implications and limitations thereof.

1.2 Outline

The thesis begins with the Unconditional Diffusion Model, starting with an introduction to Generative Models and a high-level explanation of the diffusion architecture. It then dives into the two primary discrete formulations: Score-Based Generative Models and Denoising

Diffusion Probabilistic Models. Subsequently, these formulations are translated into continuous space using Stochastic Differential Equations. Last but not least, the key elements of the U-Net backbone, that build the foundation of the Diffusion Model, are explained.

Having explained the Unconditional Diffusion Model, the next chapter shifts to the Conditional Diffusion Model. The first section starts by illustrating the fundamentals of the Conditional Diffusion Model, showing how the score function can be rewritten to model a conditional data distribution either through Classifier Guidance or Classifier-free Guidance. Besides that, the conditioning techniques of SDEdit and ControlNet are explained. In the next section, different architectural variations of commercial Diffusion Models are explained showing performance optimization techniques (e.g. Latent Diffusion, Cascading Diffusion) and conditioning methods (e.g. CLIP Guidance, Cross-Attention). Finally, the last section of this chapter presents Fine-Tuning methods such as DreamBooth, Low-Rank Adaptation and Textual Inversion, which offer different strategies for implementing personalized concepts.

In the following chapter, the theoretical foundation of the Conditional Diffusion Model is evaluated using various experimental designs. Each section of the Conditional Diffusion Model has a specific experiment. The first experiment turns a sketch into an image using the fundamentals of the Conditional Diffusion Model. In the next experiment, various commercial architectures are tested on an Inpainting job. Finally, different fine-tuning techniques are analyzed using a personal dataset.

Last but not least, the final chapter puts everything into a broader perspective looking at the limitations and implications of all findings. It analyses the role of Diffusion Models in the Generative Trilemma, what needs to be done to achieve photorealism and what societal impact all of this could have. With every aspect being discussed, this ends in a conclusion.

1.3 Notation

As Diffusion Models can be formulated in a wide variety of ways and each of their architectures employs a unique notational style, this thesis includes additional notational information about all used symbols inside the appendix.

Overall, in this thesis, all vectors are represented in bold (e.g., Image Vector $\mathbf{x}$) and all scalars are presented in a plain typeface (e.g., Timestamps T).

2 Unconditional Diffusion Models

The Conditional Diffusion Model cannot be explained without explaining the Unconditional Model first. Therefore, this chapter quickly explains Generative Models themselves and examines the different formulations of the Diffusion Model. Finally, the most important building blocks of the U-Net backbone are explained.

2.1 What are Generative Models?

A Diffusion Model is a specific type of Generative Model. Before explaining the underlying mechanisms of Diffusion Models, it is necessary to understand the main goal of Generative Modeling.

In Generative Modeling, there is a dataset $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$ which is assumed to be from an unknown data distribution $p_{data}(\mathbf{x})$. The goal is to train a Generative Model $p_\theta(\mathbf{x})$ that fits the data distribution $p_\theta(\mathbf{x}) \approx p_{data}(\mathbf{x})$ and is able to synthesize new samples from it (Luo, 2022).

2.1.1 Implicit Generative Models

A popular method for achieving this goal is the usage of Implicit Generative Models, where the probability density function is implicitly represented through a model of its sampling process (Y. Song, 2021).

An example of such models are Generative Adversarial Networks (GANs) which were developed by Goodfellow et al., 2014. A GAN consists of two interconnected neural networks: a Generator G and a Discriminator D. The Generator creates data (e.g. images) and the Discriminator evaluates if these images are authentic or generated. Using the loss of the Discriminator the Generator and Discriminator are updated creating a zero-sum game.

This type of adversarial approach has yielded impressive results, as demonstrated in works like Karras et al., 2019. However, it also comes with many challenges, such as a difficult training process troubled by mode collapse and catastrophic forgetting (Thanh-Tung and Tran, 2020).

2.1.2 Explicit Generative Models

Explicit Generative Models, as opposed to Implicit ones, directly model the data distribution $p_{data}(\mathbf{x})$. These so-called Likelihood-based Models attempt to model the probability density function (Y. Song and Kingma, 2021) which is defined as:

$$p_\theta(\mathbf{x}) = \frac{e^{-f_\theta(\mathbf{x})}}{Z_\theta} \tag{2.1}$$

Here (see equation 2.1), the function $f_\theta(\mathbf{x})$ is a real-valued function $f_\theta(\mathbf{x}) \in \mathbb{R}$ parameterized by a learnable parameter θ and represents the output of a neural network. Since a probability density function needs to be non-negative everywhere, the neural output is exponentiated by $e^{-f_\theta(\mathbf{x})}$. Additionally, a normalizing constant Z_θ is required, because the integral of a a probability density function must satisfy $\int p_\theta(\mathbf{x})\mathrm{dx} = 1$ to ensure that the area under the entire curve equals one (Y. Song, 2021).

While this normalizing constant is easily traceable for a simple Gaussian Distribution it is intractable for higher dimensions (Hyvärinen, 2005). This untraceability makes it impossible to calculate and maximize the log-likelihood of the data:

$$\max_\theta \sum_{i=1}^{N} \log p_\theta(\mathbf{x}_i). \tag{2.2}$$

To address this, Likelihood-based models either restrict their architectures (e.g. causal convolutions in Autoregressive Models) to make Z_θ traceable or approximate it (e.g. variational inference in Variational Autoencoders), which can be a computationally expensive endeavor (Y. Song, 2021).

Compared to Implicit Models, Explicit Models are generally easier to train, but their sampling quality is often times only moderate (Xiao et al., 2022).

2.2 Diffusion Model – A high-level approach

Before explaining the complex mathematics of Diffusion Models in Generative Modeling, let's start with a high-level explanation.

Diffusion models, discovered by Sohl-Dickstein et al., 2015, draw inspiration from techniques of non-equilibrium thermodynamics. A good way of understanding them is to use dye spreading in water as an analogy: In the beginning, the dye appears in the water representative of a complex probability distribution. As time goes by, it diffuses until it results in a uniform distribution. Diffusion models take advantage of the theoretical reversibility of this process, step-wise transforming a uniform distribution back into a complex one.

In the context of Generative Modeling, this process starts with an image $\mathbf{x}_0$ that has noise added gradually. This forward process step-wise transforms the original data into a uniform distribution. Subsequently, a reverse process is initiated, where a neural network, often a U-Net, step-wise tries to remove the noise. This is done by minimizing the difference between generated samples and the original data until the original data is restored. Once training is completed, it can generate an image belonging to a complex data distribution from simple Gaussian noise (Ho et al., 2020).

This diffusion process can be formulated in numerous ways. Some approaches use a continuous noise schedule modeled by Stochastic Differential Equations (Y. Song et al., 2021), while others use a discrete noise schedule (Ho et al., 2020). Such a discrete noise schedule can be formulated by either predicting the score function of a probability distribution using a Noise Conditional Score Network (Y. Song and Ermon, 2020) or by predicting the amount of noise in a noisy image at a given timestamp, as it is done in Denoising Diffuson Probalistic Models (Ho et al., 2020).

2.3 Score-Based Generative Models

As mentioned in the previous paragraph, one way to formulate Diffusion Models is as a Score-Based Generative Model. This type of model employs an annealed variant of Score Matching and Langevin Dynamics to learn how to generate new samples belonging to a data distribution. This framework is proposed in Y. Song and Ermon, 2020.

2.3.1 Score Matching

Looking back at the Explicit Generative Model, one way of realizing a Generative Model is to directly model the probability density function of an unknown data distribution (see equation 2.1). A Score Based Generative Model tackles the problem of the intractable normalizing

constant Z_θ by modeling the score function $\nabla_{\mathbf{x}} \log p(\mathbf{x})$ and utilizing a neural network to approximate the gradient of the logarithmic probability density:

$$\mathbf{s}_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p(\mathbf{x}) \tag{2.3}$$

As the gradient of a constant is equal to zero, the model is independent of the Z_θ normalizing constant, which removes all model restrictions. Training a Score Based Generative Model can be done by comparing the vector fields of the actual and estimated score functions. This is done using the Fisher divergence objective, which is calculated between the model $\mathbf{s}_\theta$ and the $p(\mathbf{x})$ data distribution.

$$\mathbb{E}_{p(\mathbf{x})}[\|\nabla_{\mathbf{x}} \log p(\mathbf{x}) - \mathbf{s}_\theta(\mathbf{x})\|_2^2] \tag{2.4}$$

Unfortunately, the $p(\mathbf{x})$ data distribution is unknown making it impossible to directly use this objective. A solution for this was discovered in Hyvärinen, 2005. It works by calculating the Jacobian of $\mathbf{s}_\theta(\mathbf{x})$:

$$\mathbb{E}_{p_{\text{data}}(\mathbf{x})}\left[\text{tr}(\nabla_{\mathbf{x}} s_\theta(\mathbf{x})) + \frac{1}{2}\|s_\theta(\mathbf{x})\|_2^2\right] \tag{2.5}$$

If one wants to use this objective for a deep neural network there is a scalability issue, because calculating the $\text{tr}(\nabla_{\mathbf{x}} \mathbf{s}_\theta(\mathbf{x}))$ would require too many backpropagations Y. Song and Ermon, 2020. Furthermore, this technique has a problem in accurately estimating the score function in areas with a low data density. This problem is solved using a Denosing Score Matching objective:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{N} \sigma_i^2 \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{p_{\sigma_i}(\tilde{\mathbf{x}}|\mathbf{x})}\left[\left\|\mathbf{s}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}, \sigma_i) - \nabla_{\tilde{\mathbf{x}}} \log p_{\sigma_i}(\tilde{\mathbf{x}} \mid \mathbf{x})\right\|_2^2\right] \tag{2.6}$$

In this method a noise schedule, increasingly perturbing the data, is applied. The neural network $\mathbf{s}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}, \sigma_i)$ is expanded, making it recognize the noise level of a given sample to accurately estimate the noise function. In their paper, Y. Song and Ermon, 2020 call this type of neural network Noise Conditional Score Network (NCSN). If trained correctly, it delivers an accurate score estimation in both high and low data density regions (2.1)

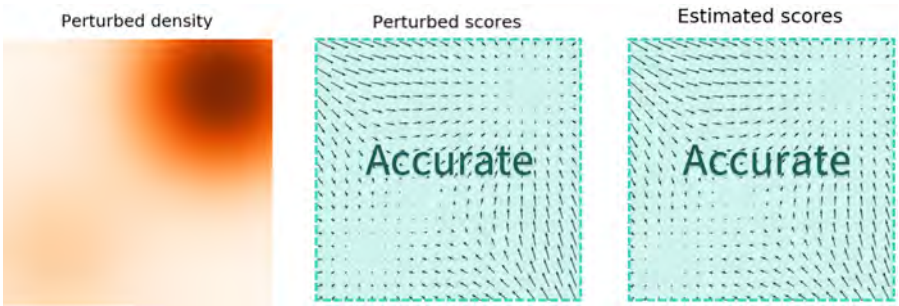

Figure 2.1: Score Model, Perturbed Scores (Y. Song, 2021)

2.3.2 Langevin Dynamics

In a Noise Conditional Score Network, sampling is done using a Markov Chain Monte Carlo Algorithm called Langevin Dynamics. The regular equation is initialized from an initial value $\tilde{\mathbf{x}}_0 \sim \pi(\mathbf{x})$, where π is characterized by a prior distribution (e.g., a Gaussian distribution), and employing a fixed step size $\epsilon > 0$. The regular Langevin equation is defined as:

$$\tilde{\mathbf{x}}_t = \tilde{\mathbf{x}}_{t-1} + \frac{\epsilon}{2}\nabla_{\mathbf{x}} \log p(\tilde{\mathbf{x}}_{t-1}) + \sqrt{\epsilon}\mathbf{z}_t, \tag{2.7}$$

Here, the distribution of $\tilde{\mathbf{x}}_t$ equals $p(\mathbf{x})$ when $T \to \infty$ and $\epsilon \to 0$. To adapt this for approximation by a Noise Conditional Score Network, the data distribution $p(\tilde{\mathbf{x}}_{t-1})$ is substituted by a conditional neural network $\mathbf{s}_\theta(\tilde{\mathbf{x}}_{t-1}, \sigma_i)$. Furthermore, the method is annealed by gradually tuning down the step size ϵ and multiplying it with a noise schedule σ_i^2/σ_L^2 resulting in the final equation:

$$\tilde{\mathbf{x}}_t \leftarrow \tilde{\mathbf{x}}_{t-1} + \frac{\alpha_i}{2}\mathbf{s}_\theta(\tilde{\mathbf{x}}_{t-1}, \sigma_i) + \sqrt{\alpha_i}\mathbf{z}_t \tag{2.8}$$

2.4 Denoising Diffusion Probabilistic Models

Besides Score-Bases Generative Models, another discrete formulation of Diffusion Models was invented during a similar time frame. These models are called Denoising Diffusion Probabilistic Models. While the main idea was outlined in Sohl-Dickstein et al., 2015, it was further advanced by Ho et al., 2020. Essentially, these models are noise predictors, predicting the noise in a generated image during the reverse process.

2.4.1 Forward Process

Figure 2.2: DDPM, Noise Schedule (Linear, Cosine) (Nichol and Dhariwal, 2021)

Their forward process starts with a data point $\mathbf{x}_0$ (e.g., an image vector) that step-wise gets Gaussian noise added. When $T \to \infty$, the $\mathbf{x}_T$ approaches a Gaussian distribution. This is done

by applying a noise schedule β_t, which can be a simple linear function (Ho et al., 2020) or a more complex cosine function (Nichol and Dhariwal, 2021). Utilizing a cosine function has the advantage that noise is injected at a much slower rate. All in all, the step-wise forward process is captured in the following equations:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \quad q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}) \tag{2.9}$$

These equations describe the unparametrized forward process. One issue with this formulation is that each $\mathbf{x}_t$ depends on the previous step $\mathbf{x}_{t-1}$, causing the forward process to become increasingly costly for a huge amount of timestamps T. To solve this problem, Ho et al., 2020 utilizes the reparametrization trick. With this trick, the equations are reduced to the following:

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}) \tag{2.10}$$

Here, every $\mathbf{x}_t$ depends only on the starting data point $\mathbf{x}_0$, thereby avoiding a costly Markov chain. The new variable $\bar{\alpha}_t$ is calculated in the following manner:

$$\alpha_t := 1 - \beta_t \quad \bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s \tag{2.11}$$

2.4.2 Reverse Process

During the reverse process, a neural network is trained to recover the original data. This process also utilizes a Markov chain, starting with a $\mathbf{x}_t$ sampled from a Gaussian distribution. It is defined as:

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)) \quad p_\theta(\mathbf{x}_{0:T}) := p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t), \tag{2.12}$$

To achieve this reversal process, a neural network is added that has the goal to predict the amount of noise added to the image at a specific timestamp t. This network is trained after the following objective:

$$\mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\right\|^2\right] \tag{2.13}$$

As the term $\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$ equals an arbitrary $\mathbf{x}_t$, the model essentially receives a noised sample $\mathbf{x}_t$ at a timestamp t, from which it tries to predict the amount of noise $\boldsymbol{\epsilon}_\theta$. This is then compared to the actual $\boldsymbol{\epsilon}$, making the model a noise predictor.

2.4.3 Sampling

After the training procedure is completed, it is possible to sample from a Denoising Diffusion Probabilistic Model using a Markov Chain similar to that of Langevin Dynamics:

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \sigma_t \mathbf{z} \tag{2.14}$$

As this forward process is slow and computationally intensive, especially with a high number of T timestamps, J. Song et al., 2022 proposed the technique of Denoising Diffusion Implicit Models (DDIM). They allow sampling using a non-Markovian chain that allows to skip steps which in practice offers a good trade-off between sampling speed and image quality. Finally, this equation is defined as follows:

$$\mathbf{x}_{t-1} = \sqrt{\alpha_{t-1}} \underbrace{\left(\frac{\mathbf{x}_t - \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_\theta^{(t)}(\mathbf{x}_t)}{\sqrt{\alpha_t}}\right)}_{\text{“predicted } \mathbf{x}_0\text{”}} + \underbrace{\sqrt{1-\alpha_{t-1}-\sigma_t^2}\cdot\boldsymbol{\epsilon}_\theta^{(t)}(\mathbf{x}_t)}_{\text{“direction pointing to } \mathbf{x}_t\text{”}} + \underbrace{\sigma_t\boldsymbol{\epsilon}_t}_{\text{random noise}} \tag{2.15}$$

2.5 Score-Based Generative Modeling through Stochastic Differential Equations

2.5.1 Equivalence of NCSN and DDPM

In their paper, Ho et al., 2020 states that the DDPM model can be formulated either as an image $\mathbf{x}$ or noise $\boldsymbol{\epsilon}$ predictor, with the latter only delivering better results. Additionally, there exists a third possible option that can be derived from combining Tweedie's formula with the reparameterization trick, as done in Luo, 2022. This results in the following equation proving the equality between the DDPM and NCSN models:

$$\nabla \log p(\mathbf{x}_t) = -\frac{1}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon} \tag{2.16}$$

Using this equivalence, it is possible to rewrite the objective of the DDPM in a way that underlines its resemblance to the Score Matching approach (Y. Song et al., 2021):

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{N} (1-\alpha_i)\mathbb{E}_{p_{\text{data}}(\mathbf{x})}\mathbb{E}_{p_{\alpha_i}(\tilde{\mathbf{x}}|\mathbf{x})}[\left\|\mathbf{s}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}, i) - \nabla_{\tilde{\mathbf{x}}} \log p_{\alpha_i}(\tilde{\mathbf{x}} \mid \mathbf{x})\right\|_2^2] \tag{2.17}$$

2.5.2 Stochastic Differential Equation

These strong similarities motivated Y. Song et al., 2021 to generalize both formulations under a unified framework using a continuous noise schedule. This causes both discrete formulations to become Stochastic Differential Equations (SDEs) following the form of:

$$\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t + g(t)\mathrm{d}\mathbf{w} \tag{2.18}$$

In general, SDEs are mathematical models describing systems influenced by randomness and noise. They consist of a drift and a diffusion term. The drift term is a vector-valued function $\mathbf{f}(\cdot, t) : \mathbb{R}^d \to \mathbb{R}^d$ that represents the deterministic component of the system that predictably evolves over time. On the other hand, the diffusion term is a scalar-valued function $g(\cdot) : \mathbb{R} \to \mathbb{R}$ that is multiplied by a Wiener Process. This Wiener Process is a mathematical model of Brownian motion that models the random fluctuations of the system.

$$\mathrm{d}\mathbf{x} = [\mathbf{f}(\mathbf{x}, t) - g^2(t)\nabla_{\mathbf{x}} \log p_t(\mathbf{x})]\mathrm{d}t + g(t)\mathrm{d}\bar{\mathbf{w}} \tag{2.19}$$

According to Anderson, 1982 there exists a corresponding reverse SDE for every forward SDE. In this reverse SDE (see equation 2.19) $\mathrm{d}t$ represents a negative infinitesimal timestep from $t = T$ until $t = 0$. Furthermore, there is a reappearance of the score function $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ that is approximated by the diffusion model. Finally, this results in an updated training objective that includes the continuous noise schedule:

$$\mathbb{E}_{t\in\mathcal{U}(0,T)}\mathbb{E}_{p_t(\mathbf{x})}[\lambda(t)\|\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) - \mathbf{s}_\theta(\mathbf{x}, t)\|_2^2] \tag{2.20}$$

During their analysis Y. Song et al., 2021 show that the Noise Conditional Score Network as well as the Denoising Diffusion Probabilistic Model can be seen as a discretization of a SDE. For both models, they provide a corresponding continuous representation:

$$\text{[NCSN]} \quad \mathrm{d}\mathbf{x} = \sqrt{\frac{\mathrm{d}\left[\sigma^2(t)\right]}{\mathrm{d}t}}\mathrm{d}\mathbf{w} \qquad \text{[DDPM]} \quad \mathrm{d}\mathbf{x} = -\frac{1}{2}\beta(t)\mathbf{x}\mathrm{d}t + \sqrt{\beta(t)}\mathrm{d}\mathbf{w} \tag{2.21}$$

2.5.3 SDE and ODE solver

Modeling a Diffusion Model through the use of SDEs enables the usage of various approximation techniques. For example, it is possible to integrate numerical methods such as Euler-Maruyama or Runge-Kutta into the Diffusion Model (Y. Song et al., 2021).

Furthermore, there is the option to rewrite the SDE into an Ordinary Differential Equation (ODE). This opens up Diffusion Models to a broad design space, using approximation techniques like Euler's or Heun's methods, which offer a wide range of optimization techniques in terms of performance and image quality (Karras et al., 2022).

2.6 U-Net – The Backbone of Diffusion Models

The architecture of choice for working with Diffusion Models is a U-Net, which was developed by Ronneberger et al., 2015 for the task of biomedical image segmentation. Attempting to describe all possible U-Net variations used in Diffusion Models would go beyond the scope of this paper. Therefore, the following paragraph will focus only on the building blocks used in Ho et al., 2020 and Rombach et al., 2022.

2.6.1 Architecture

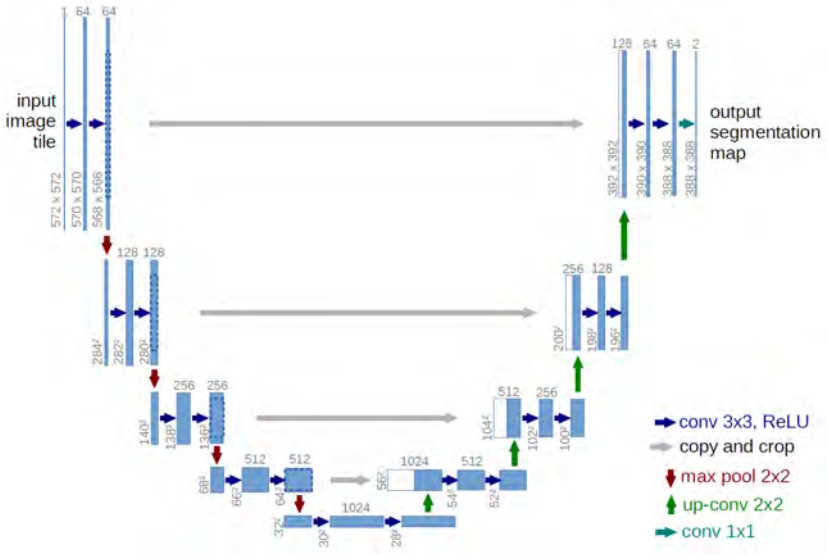

Figure 2.3: U-Net, Architecture (Ronneberger et al., 2015)

As the name suggests, the architecture (see Figure 2.3) is shaped in the form of a "U", consisting of a contracting path and an expansive path.

In the contracting path, the model follows a typical convolutional network structure, with two Convolutions followed by a ReLU activation and a max pooling operation. During this contraction, the feature information increases while the spatial information is reduced, enabling the model to capture the contextual information in the image.

The expansive path replaces the max pooling operator with an upsampling convolution. Additionally, the cropped feature maps of the corresponding contracting paths are concatenated. This allows the model to precisely localize elements within the image.

2.6.2 Residual Blocks

The first important building block of the Diffusion U-Net Model is the Residual Block. First mentioned in He et al., 2015, Residual Blocks address a significant issue in deep neural networks. After reaching a certain depth, many models encounter a degradation problem where, although their depth increases, they stop improving and may start having a reduced accuracy.

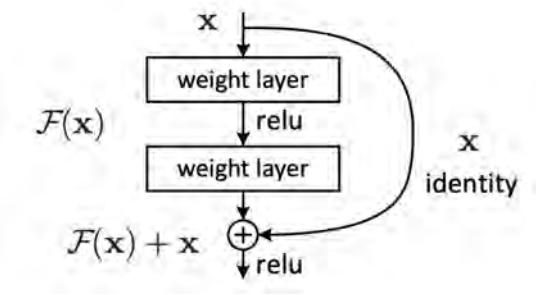

Figure 2.4: U-Net, Residual Block (He et al., 2015)

One solution to this degradation problem is the use of Residual Blocks (see Figure 2.4). These blocks function by adding the input back into the network after skipping a certain amount of steps. This allows for deeper networks by enabling deeper layers to access features learned in shallower ones.

2.6.3 Sinusoidal Position Embeddings

Besides the performance-oriented Residual Blocks, the U-Net needs a way to encode the t timestamps so it can learn at which noise level it is operating. One way to integrate these timestamps is the usage of a technique called Sinusoidal Positional Embeddings. It has its origins in the famous "Attentions is all you need" paper by Vaswani et al., 2023.

$$\begin{aligned} \mathbf{PE}_{(pos,2i)} &= \sin(pos/10000^{2i/d_{\text{model}}}) \\ \mathbf{PE}_{(pos,2i+1)} &= \cos(pos/10000^{2i/d_{\text{model}}}) \end{aligned} \tag{2.22}$$

In theory, there are many possible ways of adding a positional embedding, but the use of Sinusoidal Embeddings (see Figure 2.22) offers some clear advantages. This is best to be understood by looking at alternative methods and their drawbacks.

A simple way of adding Positional Embeddings would be the use of growing integers. However, these are not a good option, because for high values they add a huge amount of distortion by placing the tensors into too distant places.

Furthermore, neural networks operate best with normalized data. A second solution could be the use of fractions ($\frac{1}{1-N}$). While these fulfill the criteria of being bound to [-1, 1], they change according to the sequence length N. This could result in the network not working properly when fed data of different sequential lengths.

As Sinusoidal Embeddings solve both these problems, they are chosen as the preferred method for Positional Embeddings in a Transformer Model, as well as for Timestamp Embeddings in a Diffusion Model (Ho et al., 2020).

2.6.4 Attention Module

Last but not least, the Attention Module is another important technique borrowed from the Transformer architecture (Vaswani et al., 2023). The way this module operates is best understood by examining the following equation:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^{\mathrm{T}}}{\sqrt{d_k}}\right)\mathbf{V} \tag{2.23}$$

It starts with creating $\mathbf{Q}$ Query, $\mathbf{K}$ Key and $\mathbf{V}$ Value matrices by multiplying the input tensor with the corresponding weights $\mathbf{W}_{\mathbf{Q},\mathbf{K},\mathbf{V}}$. These weight matrices are updated during training, enabling the attention techanism. They are essential in determining the relevance of different parts of input data. In the next step, the attention scores $\mathbf{Q}\mathbf{K}^T$ are calculated. These are the dot product of the Query and the transposed Key Matrix. Subsequently, a scaling factor $\frac{1}{\sqrt{d_k}}$ is used to counteract the effects of vanishing gradients due to a large dot product. Finally, the softmax is calculated and the results are multiplied by the $\mathbf{V}$ value vector.

Within Diffusion Models, these Attention Modules are inserted between the convolutional blocks of the expanding and contracting paths, optimizing the performance of the U-Net model (Ho et al., 2020).

3 Conditional Diffusion Model

After providing a comprehensive overview of the Unconditional Diffusion Model, this leads to an in-depth exploration of the Conditional Diffusion Model. This chapter is dedicated to the fundamental techniques, alongside an investigation into the various architectural variations and commercial models. Finally, different methods of Fine-Tuning are examined.

3.1 Fundamentals of the Conditional Diffusion Model

This section explains the fundamentals of the Conditional Diffusion Model. It explains how the score function can be modified to model a conditional distribution $p(\mathbf{x}|\mathbf{y})$, thereby enabling guidance through either Classifier Guidance or Classifier-Free Guidance. Additionally, it delves into the supplementary conditioning techniques of SDEdit and ControlNet.

3.1.1 Mathematical Foundations

An Unconditional Diffusion Model models a data distribution $p(\mathbf{x})$. On the other hand, a Conditional Diffusion Model has a conditional distribution $p(\mathbf{x}|\mathbf{y})$, where the diffusion process is guided by additional conditioning information $\mathbf{y}$.

$$p(\mathbf{x}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{x}) \cdot p(\mathbf{x})}{p(\mathbf{y})} \tag{3.1}$$

Using the Bayes' rule (see equation 3.1) it is possible to split the score function into a conditional and unconditional component (Y. Song, 2021). Applying this rule to the score function results in the following equation:

$$\nabla_{\mathbf{x}} \log p(\mathbf{x} \mid \mathbf{y}) = \nabla_{\mathbf{x}} \log p(\mathbf{y} \mid \mathbf{x}) + \nabla_{\mathbf{x}} \log p(\mathbf{x}) - \nabla_{\mathbf{x}} \log p(\mathbf{y}) \tag{3.2}$$

Due to the fact that the gradient $\mathbf{x}$ of $p(\mathbf{y})$ equals zero, the conditional score function is just the sum of the unconditional score function and a conditioning term.

$$\nabla_{\mathbf{x}} \log p(\mathbf{x} \mid \mathbf{y}) = \nabla_{\mathbf{x}} \log p(\mathbf{y} \mid \mathbf{x}) + \nabla_{\mathbf{x}} \log p(\mathbf{x}) \tag{3.3}$$

Classifier Guidance

There are many possible ways to obtain the conditioning term $\nabla_{\mathbf{x}} \log p(\mathbf{y} \mid \mathbf{x})$. Dhariwal and Nichol, 2021, in their paper "Diffusion Models Beat GANs on Image Synthesis", proposes a method for obtaining the conditioning gradient using a classifier:

$$\nabla_{\mathbf{x}} \log p(\mathbf{x} \mid \mathbf{y}) = \nabla_{\mathbf{x}} \log p(\mathbf{x}) + \gamma \nabla_{\mathbf{x}} \log p(\mathbf{y} \mid \mathbf{x}) \tag{3.4}$$

To implement this, a classifier is trained on noisy images $\mathbf{x}$ to provide accurate class predictions $\mathbf{y}$. Additionally, the variable γ serves as a guidance scale hyperparameter, allowing for the adjustment of the conditioning signal's influence on the generated output.

Classifier-Free Guidance

As Classifier Guidance requires training an additional classifier on noisy images, this adds a training step and strongly resembles an adversarial technique similar to GANs. These disadvantages inspired Ho and Salimans, 2022 to develop a method of Classifier-Free Guidance.

$$p(\mathbf{y} \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid \mathbf{y}) \cdot p(\mathbf{y})}{p(\mathbf{x})} \tag{3.5}$$

Similar to Classifier Guidance, it can be derived by applying Bayes' rule (see equation 3.5) to the score function, but this time in the opposite direction $p(\mathbf{y}|\mathbf{x})$:

$$\nabla_{\mathbf{x}} \log p(\mathbf{y} \mid \mathbf{x}) = \nabla_{\mathbf{x}} \log p(\mathbf{x} \mid \mathbf{y}) + \nabla_{\mathbf{x}} \log p(\mathbf{y}) - \nabla_{\mathbf{x}} \log p(\mathbf{x}) \tag{3.6}$$

This function can be further simplified by excluding $\nabla_x \log p(\mathbf{y}) = 0$ and substituting $\nabla_{\mathbf{x}} \log p(\mathbf{y} \mid \mathbf{x})$ into the Classifier Guidance. Finally, this causes the conditional equation to only rely on the conditional and unconditional score function which is provided entirely by the Diffusion Model:

$$\nabla_{\mathbf{x}} \log p(\mathbf{y} \mid \mathbf{x}) = \nabla_{\mathbf{x}} \log p(\mathbf{x} \mid \mathbf{y}) - \nabla_{\mathbf{x}} \log p(\mathbf{x}) \tag{3.7}$$

Because training a conditional and unconditional model separately would be expensive, Classifier Free Guidance trains both at the same time. This is done by replacing the conditioning information with fixed constants (e.g., zeros) for a certain percentage during training (Luo, 2022).

$$\nabla_{\mathbf{x}} \log p(\mathbf{x} \mid \mathbf{y}) = (1 - \gamma) \nabla_{\mathbf{x}} \log p(\mathbf{x}) + \gamma \nabla_{\mathbf{x}} \log p(\mathbf{x} \mid \mathbf{y}) \tag{3.8}$$

Finally, adding a γ scaling factor similar to that of Classifier Guidance completes the equation. With this scaling factor, the model is unconditional when $\gamma = 0$ and conditional when $\gamma = 1$. Additionally, if the value exceeds $\gamma > 1$ there is a strong increase in adherence to the guidance combined with a diminishing diversity.

3.1.2 SDEdit: Guided Image Synthesis and Editing with Stochastic Differential Equations

The architecture of Diffusion Models allows for a special way of conditioning. This technique, described in Meng et al., 2022, works by hijacking the generative process. It allows the Diffusion Model to perform various Image-to-Image tasks.

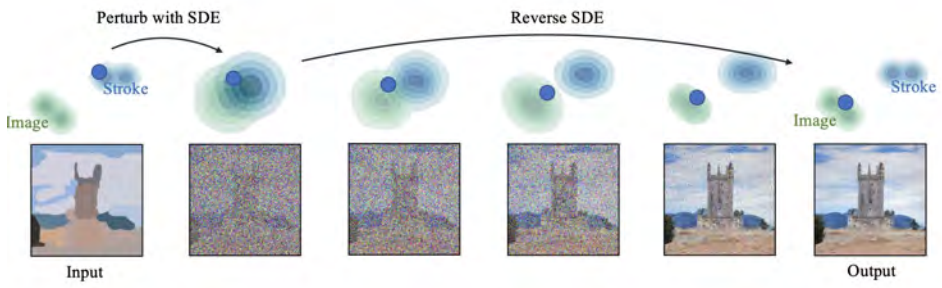

Figure 3.1: SDEdit, Mechanism (Meng et al., 2022)

Exploring the mechanism behind SDEdit, as shown in Figure 3.1, reveals that the Diffusion Model is guided using a perturbed input image. Depending on the amount of perturbation termed denoising strength $t_0 \in (0, 1)$, the resulting image can be more or less faithful to the input. At the extremes if $t_0 = 0$ is used the image remains unchanged and conversely, if $t_0 = 1$ the produced image is completely independent of the input. This allows for a trade-off between realism and fidelity to the input. According to Meng et al., 2022 the sweet spot for an optimal output seems to lie between $t_0 = 0.3$ and $t_0 = 0.6$.

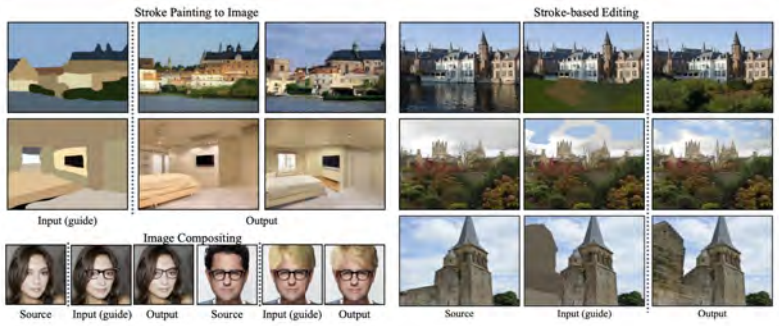

Figure 3.2: SDEdit, Examples (Meng et al., 2022)

As mentioned in the introduction, SDEdit allows for multiple Image-to-Image tasks (see

Figure 3.2). For example, it is possible to use a sketch to create an image. Furthermore, it is possible to Inpaint a certain area of an image and refill it with something else.

3.1.3 ControlNet – Advanced Conditioning

Lastly, another powerful way of adding additional conditioning to a Diffusion Model is the usage of a ControlNet which was proposed in Zhang et al., 2023.

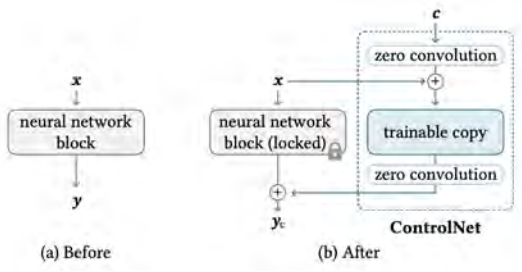

Figure 3.3: ControlNet, Model (Zhang et al., 2023)

Figure 3.4: ControlNet, Examples (Zhang et al., 2023)

Here (see Figure 3.3), the main Diffusion Model is locked while a trainable copy is created. This trainable copy could be the whole model or a smaller version. For example, only the encoder block of the U-Net could be used largely reducing the required training resources. Furthermore, training only a copy leaves the original model unaffected by overfitting and catastrophic forgetting.

Subsequently, the ControlNet model receives the original image $\mathbf{x}$ and a $\mathbf{c}$ conditioning term as input. At the end of the process, both the output of the ControlNet and the Diffusion Model are combined, creating a $\mathbf{y_c}$ conditioned image.

In their paper, Zhang et al., 2023 demonstrates the various conditional methods (see Figure 3.4). For example, one could use an edge detection mechanism like Canny Edge or a human pose detection model to guide the diffusion process. It is also possible to mix multiple ControlNets allowing to heavily control the image synthesis of Diffusion Models.

3.2 Architectural Variations and Commercial Models

This second section of the Conditional Diffusion Model explores various architectural variations, building upon the insights gained from the Unconditional Diffusion Model and the fundamentals of the Conditional Diffusion Model. Here, additional conditioning methods and performance optimizations are explored. Overall this section includes an overview of the three most significant models: DALL-E, Imagen and Stable Diffusion focusing on their origins and contributions.

3.2.1 DALL-E – Hierarchical Text-Conditional Image Generation with CLIP Latents

DALL-E, or more specifically DALL-E 2, is one of the first diffusion-based Text-to-Image models released by Open AI. Its research is a continuation of the findings from the GLIDE model (Nichol et al., 2022).

GLIDE – Guided Language to Image Diffusion for Generation and Editing

Building on the idea of using class labels $\mathbf{y}$ as guidance for Diffusion Models, the GLIDE model uses $\mathbf{c}$ text inputs as conditioning. In their paper "GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models", Nichol et al., 2022 evaluates the usage of Classifier Guidance and Classifier Free-Guidance for Text-to-Image generation.

CLIP Guidance

While any type of classifier could technically be used for Guidance, GLIDE uses the Contrastive Language-Image Pre-Training (CLIP) model which was constructed in Radford et al., 2021. CLIP is a model that consists of an image encoder $\mathbf{f}(\mathbf{x})$ and a text encoder $\mathbf{g}(\mathbf{c})$. During training, the model optimizes a cross-entropy loss that encourages a high dot product $\mathbf{f}(\mathbf{x}) \cdot \mathbf{g}(\mathbf{c})$. This causes the model to learn how close an image is to a caption because when trained properly, an input image that matches a given text prompt returns a high dot product. Within the GLIDE model, CLIP is used as a classifier to guide the diffusion process. For this purpose, the CLIP model is retrained on noisy images to optimize Guidance.

Classifier Free Guidance

The second approach chosen in the GLIDE paper is text conditioning using Classifier-Free Guidance only. This procedure is done the same way as normal Classifier-Free Guidance, only substituting the class labels $\mathbf{y}$ with $\mathbf{c}$ text captions and using an $\emptyset$ empty sequence for 20% of the training:

$$\hat{\epsilon}_\theta(\mathbf{x_t}|\mathbf{c}) = \epsilon_\theta(\mathbf{x_t}|\emptyset) + s \cdot (\epsilon_\theta(\mathbf{x_t}|\mathbf{c}) - \epsilon_\theta(\mathbf{x_t}|\emptyset)) \tag{3.9}$$

During experimentation, this method was superior to Classifier Guidance (CLIP Guidance), with the s hyperparameter giving good control over balancing diversity against image quality. This matches with the results of Ho and Salimans, 2022.

DALL-E 2

While the first DALL-E model uses a discrete Variational Autoencoder in combination with an Autoregressive Transformer (Ramesh et al., 2021), its successor DALL-E 2, also utilizes a two-stage approach (Ramesh et al., 2022). Here, OpenAI uses their findings in GLIDE to propose a new model combining the powers of CLIP and Classifier-Free Guidance.

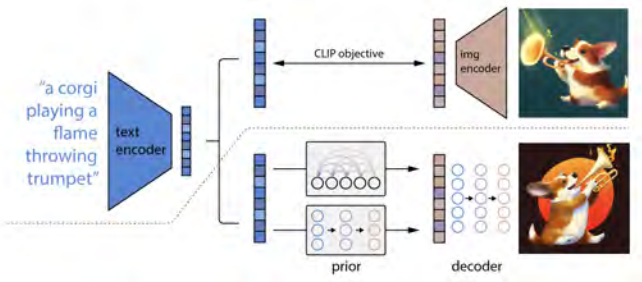

Figure 3.5: DALL-E 2, Model (Ramesh et al., 2022)

The process illustrated in Figure 3.5 starts with an input text being converted into a CLIP text embedding. During the prior stage, a Diffusion or Autoregressive Model creates a CLIP image embedding from the CLIP text embedding. Afterward, in the decoding step, a Diffusion Model creates an image out of the CLIP image embedding.

As the decoder is able to create images out of CLIP image embeddings, DALL-E 2 has the ability to create image variations. An input image is encoded using CLIP and used as decoder input. This creates a new image that preserves the semantic information while varying the non-essential details.

Currently, DALL-E 3 is the latest version of DALL-E (Betker et al., 2023). It is integrated into ChatGPT, which is built on top of the multimodal Large Language Model GPT-4. Unfortunately, most aspects of the DALL-E 3 architecture are not publicly available, making a concrete evaluation impossible.

3.2.2 CDM – Cascaded Diffusion Models

While the DALL-E 2 model from OpenAI relies on a two-stage approach, Imagen from Google Research uses a Cascaded Diffusion Model instead. This architecture was first described in a Super Resolution model (Saharia et al., 2021), tested on an ImageNet benchmark (Ho et al., 2021) and finally combined with a text encoder (Saharia et al., 2022).

SR3 – Super Resolution

The main idea of Cascading Diffusion Models stems from "Image Super-Resolution via Iterative Refinement". Here, Saharia et al., 2021 examines the use of Conditional Diffusion Models for the task of Super-Resolution.

$$\mathbb{E}_{(\mathbf{x},\mathbf{y})}\mathbb{E}_{\boldsymbol{\epsilon},\gamma}\left\| f_\theta(\mathbf{x}, \underbrace{\sqrt{\gamma}\mathbf{y}_0 + \sqrt{1-\gamma}\boldsymbol{\epsilon}}_{\tilde{\mathbf{y}}}, \gamma) - \boldsymbol{\epsilon} \right\|_p^p \tag{3.10}$$

They achieve this by building a Conditional Diffusion Model that learns to upscale images. Adapting the DDPM algorithm (see equation 3.10), the model receives high-resolution $\mathbf{y}_0$ images during the forward process. In the reverse process, a noisy image $\tilde{\mathbf{y}}$ is conditioned with a low-resolution image $\mathbf{x}$ that is concatenated around the channel dimension. With this training procedure, the model learns to upscale low-resolution images into high-resolution images.

A key finding of this paper is that these upscaling models can be used in a cascading manner. Starting with an Unconditional model that creates the ground image, consecutive Conditional Diffusion Models can upscale it to the target resolution.

Cascading Diffusion Models

Building upon the idea of Cascaded Diffusion Models in SR3, Ho et al., 2021 examines the Cascaded Diffusion Model on an ImageNet benchmark. In their study, the first model (see Figure 3.6) creates an image according to an ImageNet class label with a resolution of 32x32

pixels. This image, along with the class label, serves as the input for the second model, which creates a 64x64 image. Finally, a third model, conditioned on the 64x64 image and the class label, creates a 256x256 image. Using this technique, Ho et al., 2021 achieved good results at the ImageNet generation benchmark.

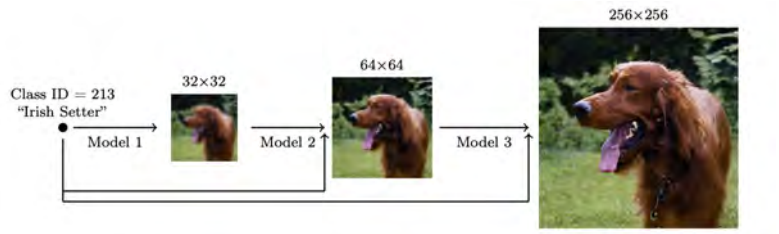

Figure 3.6: CDM, Model (Ho et al., 2021)

Imagen

This Cascading Diffusion Model is further combined with a text encoder to create the Text-to-Image model Imagen (Saharia et al., 2022).

Similar to CLIP (Radford et al., 2021), Imagen uses a text encoder to turn text input into text embeddings. Its main distinction is that rather than using a text encoder trained on text-image pairs, Imagen uses the T5 text encoder belonging to a Large Language Model. This encoder is bigger, but only trained on a text-only corpus. According to the results published by the Google Research Team (Saharia et al., 2022), the sheer usage of a larger text encoder results in a better sample quality as well as image-text alignment.

Overall, the Imagen model starts with a frozen T5 text encoder that converts the input text to a textual embedding. These embeddings are then put into a Conditional Diffusion Model that creates a 64x64 image. Subsequently, two consecutive Super-Resolution models receive the input image as well as the text embedding and upscale it to a 1024x1024 resolution.

3.2.3 LDM – Latent Diffusion Model

A major problem of Diffusion Models is that they require a significant amount of computational resources to create high-resolution images. This is primarily due to the reversal process requiring a certain amount of steps to successfully converge to an image. While methods like the non-Markovian DDIM (J. Song et al., 2022) can help, the creation of high-resolution images remains computationally expensive.

In their paper, "High-Resolution Image Synthesis with Latent Diffusion Models", Rombach et al., 2022 attempts to solve this problem by transforming the diffusion process from pixel space to a lower dimensional latent space.

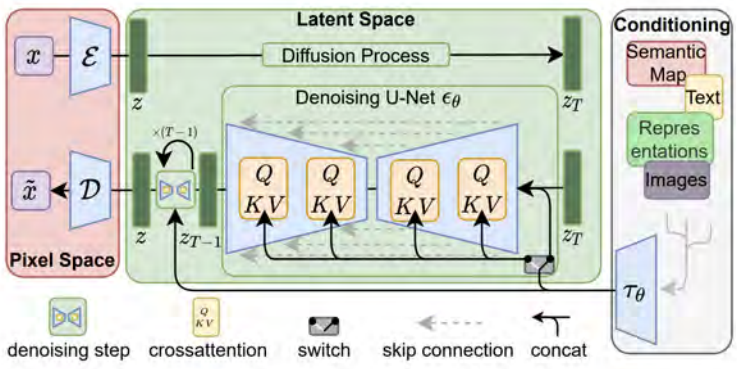

Figure 3.7: LDM, Model (Rombach et al., 2022)

Departure to the Latent Space

This transformation is achieved by utilizing pre-trained Autoencoders. It begins with an image $\mathbf{x} \in \mathbb{R}^{H \times W \times 3}$, which is encoded by an $\mathscr{E}$ Encoder into a downsampled latent representation $\mathbf{z} \in \mathbb{R}^{h \times w \times 3}$. The downsampling factor is $f = H/h = W/w$. After the diffusion process, the image $\tilde{\mathbf{x}}$ is reconstructed by a decoder $\mathscr{D}$.

The paper discusses two types of regularization techniques. One method is the use of KL-divergence, which regularizes a continuous latent representation towards a standard Gaussian distribution. The other method is Vector Quantization during which a discrete latent representation is learned.

Employing an Autoencoder as preprocessing stems from the insight that Likelihood models learn in two stages. Initially, the model engages in perceptual compression, removing high-frequency details. Subsequently, semantic compression is done, focusing on the semantic and conceptual composition of the data. Separating these stages saves computational resources because the diffusion process is only applied to semantic details, while the high frequency and imperceptible details are abstracted away or rather learned by a less computationally expensive architecture.

Additional Conditioning

The Latent Diffusion paper not only presents performance optimizations but also introduces new conditioning techniques. This is achieved by introducing a domain-specific encoder τ_θ, which projects the condition $\mathbf{y}$ into an intermediate representation $\tau_\theta(\mathbf{y}) \in \mathbb{R}^{M \times d_\tau}$.

In the example of text inputs, this intermediate representation is implemented into the U-Net via Cross-Attention Layers. These layers work similarly to the attention module described in the Attention 2.23 equation. Here, the difference is that the query weights are projected to the flattened, intermediate representations, while the key and value weights are projected onto the $\tau_\theta(\mathbf{y})$ intermediate representation:

$$Q = W_Q^{(i)} \cdot \varphi_i(\mathbf{z}_t) \tag{3.11}$$

$$K = W_K^{(i)} \cdot \tau_\theta(\mathbf{y}) \tag{3.12}$$

$$V = W_V^{(i)} \cdot \tau_\theta(\mathbf{y}) \tag{3.13}$$

In the case of Image-to-Image tasks, conditioning is applied by concatenation. Here, the latent vector $\mathbf{z}$ is concatenated, alongside the channel dimension, by the intermediate representation $\tau_\theta(\mathbf{y})$ in a way similar to that of the SR3 Super-Resolution model (Saharia et al., 2021). Additionally, the diffusion process can be hijacked using the SDEdit method (Meng et al., 2022).

Stable Diffusion

The research on the Latent Diffusion Model was supported by Stability AI, which resulted in the creation of a Text-to-Image Latent Diffusion Model called Stable Diffusion. Using the open-source LAION-5B dataset, the first version of Stable Diffusion was trained to create 512x512 images. It uses a frozen version of the CLIP ViT-L/14 text encoder to create the textual embeddings. While Inpainting is possible out of the box, Stable Diffusion offers an additional Inpainting model. This has a U-Net with five additional input channels, from which four are used for the encoded masked image and one for the mask itself. During training the model is explicitly trained to restore images with synthetic masks covering up a quarter of the image.

In November 2022 a second version of the Stable Diffusion model was released. It uses the larger OpenCLIP text encoder and was trained on a larger resolution of 768x768. Besides a specialized Inpainting model, it also comes with a model that turns depth maps into images.

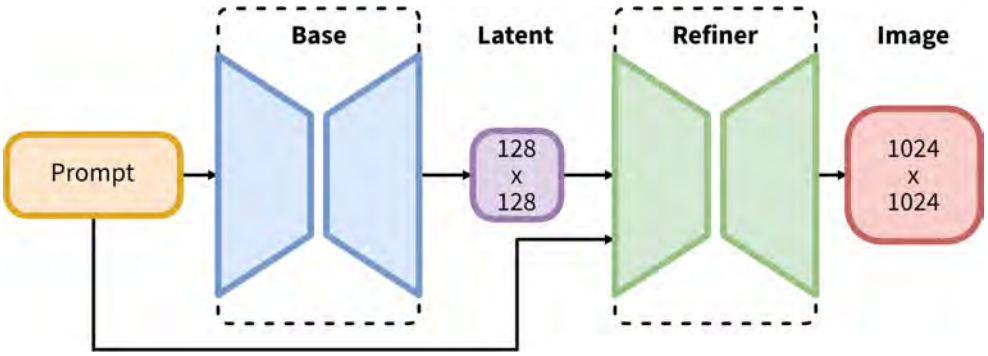

Figure 3.8: LDM, Stable Diffusion XL (Podell et al., 2023)

Last but not least, another series of Latent Diffusion Models Stable Diffusion XL, was released in July 2023. This model follows a two-stage pipeline (see Figure 3.8), where a base model creates a latent image out of a text prompt and a refinement model adds additional detailing by using the latent image combined with the text prompt (Podell et al., 2023).

3.3 Personalization through Fine-Tuning

In this final section, different techniques for incorporating personalized concepts into Diffusion Models are explained.

The direct training of a model could pose several challenges. For example, because the training dataset is often way smaller than the original, this can lead to overfitting and language drift (Ruiz et al., 2023). Additionally, training the model as a whole is computationally expensive. To address these issues, techniques such as DreamBooth, Low Rank Adaptation and Textual Inversion employ different strategies.

3.3.1 DreamBooth

Adding a personalized concept can mean integrating oneself, one's friends or anything else that comes to mind into a Diffusion Model. This is what DreamBooth (Ruiz et al., 2023) attempts to accomplish by updating the language-vision dictionary of the Diffusion Model.

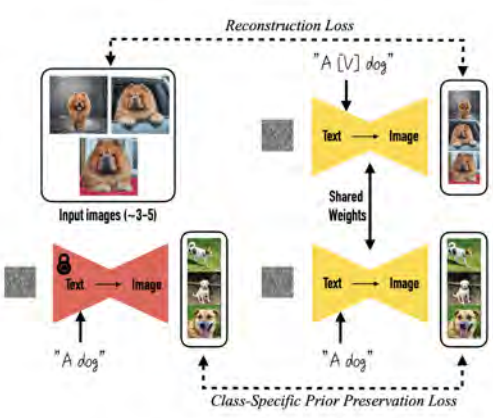

Figure 3.9: DreamBooth, Loss (Ruiz et al., 2023)

Figure 3.10: DreamBooth, Synthesis (Ruiz et al., 2023)

Training a DreamBooth model requires 3-5 images of a subject that should be implemented. Additionally, it is necessary to specify an [identifier] and a [class noun]. The chosen [identifier] should be a word that has a weak prior in the Language- and Diffusion Model. It should

not be a selection of random letters, as the tokenizer could interpret them as single letters, which have a strong prior. Within Ruiz et al., 2023 rare tokens are identified by performing a rare-token lookup and inverting these into text. Alternatively, it is possible to use a word that is not commonly used in the vocabulary.

Besides the [identifier], it is necessary to specify a [class noun] that accurately describes the input images. For example, if trying to integrate a personal dog, the [class noun] should be "dog".

$$L_{Subject} = w_t \|\hat{\mathbf{x}}_\theta(\alpha_t \mathbf{x} + \sigma_t \boldsymbol{\epsilon}, \mathbf{c}) - \mathbf{x}\|_2^2 \tag{3.14}$$

$$L_{Class} = w_{t'} \|\hat{\mathbf{x}}_\theta(\alpha_{t'} \mathbf{x_{pr}} + \sigma_{t'} \boldsymbol{\epsilon'}, \mathbf{c_{pr}}) - \mathbf{x_{pr}}\|_2^2 \tag{3.15}$$

$$\mathbb{E}_{\mathbf{x},\mathbf{c},\boldsymbol{\epsilon},\boldsymbol{\epsilon'},t}[L_{Subject} + \lambda * L_{Class}] \tag{3.16}$$

DreamBooth training utilizes a class-specific prior preservation loss. This concept is best understood by examining the loss equation (see equation 3.16). While the loss equation may seem complex, it can be divided into two parts. Additionally, ω_t is a weighting term specific to the Imagen model and can be ignored (Saharia et al., 2022).

The left part (see equation 3.14) compares the $\hat{\mathbf{x}}_\theta$ output of a Diffusion Model, conditioned by an $\mathbf{x}$ input image and the [class noun] [identifier] text prompt, to an input image. On the right side (see equation 3.15), there is another output of a Diffusion Model, conditioned by a class-specific image and the [class noun] text prompt, which is compared to a class-specific image. Both terms are then added together, using λ as a hyperparameter to adjust the influence of the class-specific loss. This process is visualized in Figure 3.9 .

During training, DreamBooth trains the complete model making the training not less computationally expensive and the resulting models match the size of the original. Finally, when the model is trained successfully, the Diffusion Model is capable of synthesizing the provided images in various novel styles. Some examples of these novel styles include art renditions, property modifications or text-guided view synthesis, as demonstrated in Figure 3.10.

3.3.2 LoRA – Low-Rank Adaptation

Taking inspiration from techniques across different fields often proves beneficial, as seen by the adoption of the diffusion process from non-equilibrium thermodynamics (Sohl-Dickstein et al., 2015). Similarly, Low-Rank Adaptation, which was primarily invented for Fine-Tuning Large Language Models, can also be used for Fine-Tuning Diffusion Models.

Background

Diving into Large Language Models, "GPT-3 175B" stands out with its 175 billion trainable parameters, demanding 1.2 TB of VRAM. Training such a model is computationally extremely expensive, but research (Aghajanyan et al., 2020) suggests these models have a low "intrinsic dimension" and are still able to learn while being projected to a smaller subspace.

For example, a model can have a weight matrix $\mathbf{W}_0 \in \mathbb{R}^{d \times k}$, where $d = 50$ and $k = 100$, totaling 5000 trainable parameters ($d * k$). The hypothesis from Aghajanyan et al., 2020 suggests that many rows in this matrix are linearly dependent and thus redundant. This leads to the possibility of using a lower-rank matrix for Fine-Tuning using a method called Low-Rank Adaptations, which was proposed in Hu et al., 2021.

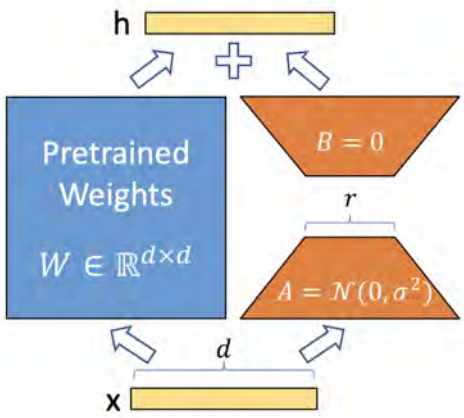

Figure 3.11: LoRA, Model (Hu et al., 2021)

Understanding how this method works can be accomplished very well using an example. Continuing with the values of the introduction, according to the hypothesis in Hu et al., 2021 this 5000-parameter $\mathbf{W}_0$ weight matrix can be projected into a lower rank subspace $\Delta\mathbf{W}$.

$$\mathbf{W}_0 + \Delta\mathbf{W} = \mathbf{W}_0 + \mathbf{BA} \tag{3.17}$$

This $\Delta\mathbf{W}$ matrix consists of the matrix multiplication of a matrix $\mathbf{A} \in \mathbb{R}^{r \times k}$ and a matrix $\mathbf{B} \in \mathbb{R}^{d \times r}$. The r value corresponds to the rank of the matrix and can be any number in $\mathbb{R}$ that fulfills $r \ll \min(d, k)$. For this example, the value is set to $r = 2$ causing matrix $\mathbf{A}$ to have 200 parameters and matrix $\mathbf{B}$ to have 100 parameters, which effectively reduces the amount of learnable parameters to 300.

At the start of the training, the original $\mathbf{W}_0$ weight matrix is locked and the starting values of the training matrix $\mathbf{A}$ are sampled from a normal distribution $\mathcal{N}(0, \sigma^2)$, while the matrix B contains only zeros. Finally, the output of the model will be the addition of $\mathbf{W}_0$ and $\Delta\mathbf{W}$ as described in equation 3.17. Using this technique, only 300 parameters need to be trained, significantly reducing the computational resources required.

Usage in Diffusion Models

Figure 3.12: LoRA, Example (Cuenca and Sayak, 2023)

In the original paper (Hu et al., 2021), this technique is applied to the four weight matrices of the Self-Attention Module in a Large Language Model, but in theory, this technique can be adapted to all kinds of weight matrices.

Therefore, it is also possible to Fine-Tune a Diffusion Model using Low-Rank Adaptation. In an experimental script provided by Cuenca and Sayak, 2023, Low-Rank Adaptation is used on the Attention Layers of the Stable Diffusion U-Net model. These are crucial because they relate the image representations with the text prompts.

As only a small part of the model is getting trained, the resulting file size is only a few megabytes and the hardware requirements are reduced. In an experiment using the "pokemon-blip-captions" dataset (see Figure 3.12), Low-Rank Adaptation successfully integrated a personalized concept into the Diffusion Model.

3.3.3 Textual Inversion

While the last two methods attempted to influence the Diffusion Model, the technique of Textual Inversion takes a different route. Published in the paper "An Image is Worth One Word: Personalizing Text-to-Image Generation using Textual Inversion" by Gal et al., 2022, it attempts to integrate a personalized concept by manipulating the text embedding of the text encoder.

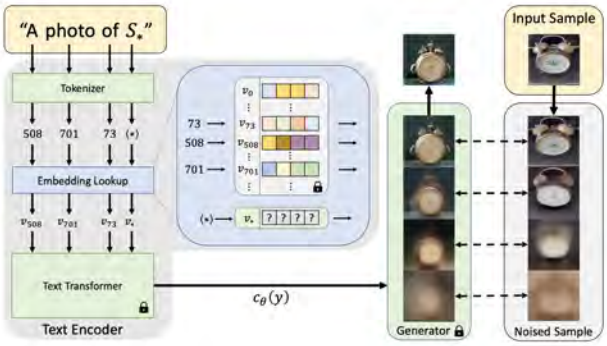

Figure 3.13: Textual Inversion, Process (Gal et al., 2022)

The Textual Inversion process, as seen in Figure 3.13, starts with a prompt containing a user-specified pseudo-Word S_*. In the next step, the text is tokenized and the discrete tokens are converted into a continuous text embedding vector. Here, a special point of interest lies on the embedding vector $\mathbf{v}_*$ of the pseudo-word, because it will be optimized during training:

$$\mathbf{v}_* = \arg\min_{\mathbf{v}} \mathbb{E}_{z\sim\mathcal{E}(x),y,\epsilon\sim\mathcal{N}(0,1),t}\left[\|\epsilon - \epsilon_\theta(\mathbf{z}_t, t, \mathbf{c}_\theta(y))\|_2^2\right] \tag{3.18}$$

Subsequently, the text transformer converts the textual embedding vector into a conditioning vector $\mathbf{c}_\theta(y)$, which conditions the Diffusion Model. While both the text transformer and generator remain locked, the text embedding vector is optimized by minimizing the noise difference between the generated image and the input sample.

Figure 3.14: Textual Inversion, Samples (Gal et al., 2022)

When training is successful, the user-specified word can be used to generate images (see Figure 3.14) containing the newly integrated concept. As only a part of the textual encoder is trained, the resulting model is small and the training does not require many resources.

4 Experiments

In this chapter, the advancements of the Conditional Diffusion Model are put to the test. Using different experimental settings, the methods and models are evaluated. This aims to provide insights into their practical performance.

Each section of the Conditional Diffusion Model is explored in a specialized experiment. During the first experiment, a sketch is turned into an image. This examines the fundamentals of the Conditional Diffusion Model, including techniques like SDEdit, Classifier-Free Guidance and ControlNet. In a subsequent experiment, three architectural variations (DALL-E 2, CDM and LDM) are tested through three Inpainting tasks. Ultimately, the last experiment tries to add a personalized concept using methods of Fine-Tuning (DreamBooth, Low-Rank Adaptation and Textual Inversion).

Overall, these experiments aim to showcase the strengths and limitations of using a Conditional Diffusion Model for image manipulation.

4.1 Tools

These experiments were conducted using a large amount of libraries and tools. However, this section focuses on explaining the two fundamental resources that had a significant impact.

4.1.1 Google Colab

Google Colab (Google Colab, 2024), which is a product of the Google Research Team, allows users to write and execute Python code directly through the browser. It is a popular research tool because of its ease of sharing notebooks.

Additionally, Google Colab provides access to computational resources such as GPUs and TPUs. Utilizing the Google Colab Pro Tier, all experiments are executed on a 16GB T4 Nvidia GPU.

4.1.2 HuggingFace

Being an important platform for sharing models and datasets, HuggingFace is a key player in the field of machine learning. It offers a wide variety of libraries that simplify working with state-of-the-art models. For instance, the Transformers library (Wolf et al., 2020) facilitates working with transformer models, making them easily usable for Natural Language Processing or Computer Vision tasks.

Crucial for carrying out all experiments is the Diffusers (von Platen et al., 2022) library. It provides inference pipelines, training scripts and pre-trained models of various diffusion architectures. All these things are essential when testing the Conditional Diffusion Model on image manipulation.

Finally, a HuggingFace dataset (Lhoest et al., 2021) is used as a repository for all utilized images.

4.2 Experiment 1: Turning a sketch into an image

Many historians speculate that Leonardo da Vinci might have created two versions of the Mona Lisa (Foundation, 2024). Although it is unclear if this theory will ever be confirmed, a Conditional Diffusion Model can be used to turn a sketch (see Figure 4.1) of the supposed second version into a realistic painting.

Figure 4.1: Experiment 1, Raphael's 'Young Woman on a Balcony' (Foundation, 2024)

This can be accomplished using the different methods described in the fundamentals of the Conditional Diffusion Model. The following experiment will evaluate how these can be set up to turn a sketch into an image.

4.2.1 Methodology

In this experiment, the Latent Diffusion Model Stable Diffusion 1.5 is used as the base model. This allows for the integration of a wide variety of ControlNet models and supports Classifier-Free Guidance and SDEdit.

During testing, the Fine-Tuned Realistic Vision V6.0 model (SG161222, 2024) showed superior aesthetic results when compared to the 1.5-base model. Although it was trained on achieving photorealism, it still performs well in turning a sketch into a painting, which is why it was chosen to demonstrate the different methods.

4.2.2 SDEdit

The first part of the experiment evaluates guided image synthesis by hijacking the diffusion process, as described in Meng et al., 2022. By using varying denoising values t_0, starting from 0 and running in 0.2 steps until 1, the sketch is injected into the denoising process. Further guidance is applied using a text prompt "A painting of a woman in the style of a renaissance artist" with a Classifier-Free Guidance scale of 5. According to Meng et al., 2022, increasing the denoising strength should make the result less dependent on the injected sketch.

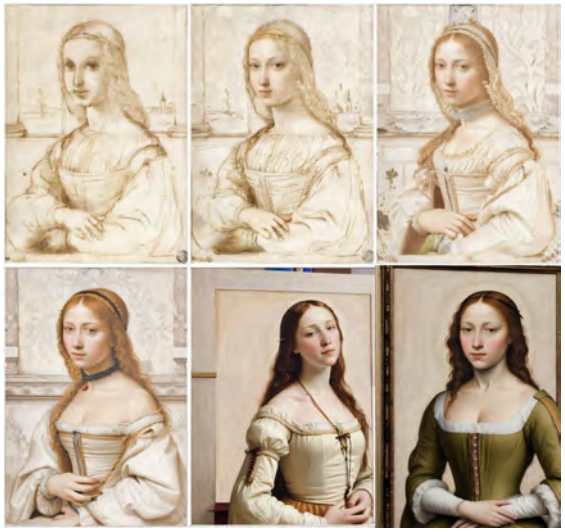

Figure 4.2: Experiment 1, SDEdit
$t_0 = [0.0, 0.2, 0.4, 0.6, 0.8, 1]$

As expected, the higher the denoising value t_0, the more the image (see Figure 4.2) diverges from the initial conditioning. Notably, at the extreme ends, the image is either unaffected or almost completely random. In their paper, Meng et al., 2022 picks the range of $t_0 \in [0.3, 0.6]$ as the sweet spot for SDEdit. Looking at the results, this finding is replicated, with the subjectively best representation occurring at $t_0 = 0.6$. All in all, it is confirmed that hijacking the diffusion process can turn an image into a sketch and that the denoising value t_0 can be used as a trade-off between a more realistic and less faithful result.

4.2.3 Classifier-Free Guidance

In the next step, Classifier-Free Guidance is assessed. Utilizing the same prompt as in the previous experiment, various γ scaling factors of Classifier-Free Guidance are analyzed. This

is done at a constant denoising value of $t_0 = 0.6$. It is expected that at lower γ values, the impact of the text prompt will be moderate, but will increase significantly at higher values (Ho and Salimans, 2022).

Figure 4.3: Experiment 1, Classifier-Free Guidance
$\gamma = [1, 2.5, 5.0, 10.0, 20.0, 40.0]$

Similar to the SDEdit experiment, the expected results (see Figure 4.3) are replicated during the experiment. Using a higher Classifier-Free Guidance causes the image to obtain more details, indicating stronger prompt adherence. At a certain threshold, in this case when $\gamma > 10$, the image starts to break down and exhibit visible artifacts combined with a strong contrast.

4.2.4 ControlNet

For the last part of this experiment, conditioning using ControlNet models is tested. As a wide variety of models exist, testing all of them would go beyond the scope of this thesis. Therefore, this experiment is limited to six conditional methods which are: Canny Edge Detection, Depth Map, Normal Map, OpenPose, LineArt and Soft Edge.

Before setting up the ControlNets, the images need to be pre-processed (see Figure 4.4) to fit the requirements of the specific model. For instance, in the case of a Depth Map, the image is put through the "Intel/dpt-large" neural network. It estimates the depth of all objects in the image and generates a corresponding depth map. Another example is the Canny Edge

detection. Here, a lower and upper threshold is defined, specifying the areas affected by edge detection. Utilizing the Canny() function from the OpenCV library, a pre-processed image is created. A complete breakdown of all pre-processing methods can be found in the complementary source codes of this thesis.

In this experimental setting, all images are created using a diffusion inference pipeline with a single ControlNet matching the created pre-processing. The used text prompt remains the same and the ControlNet image is used as only input. During the diffusion process, the ControlNet is active over all timestamps. It is expected, that the reproduced image matches the selected conditional method.

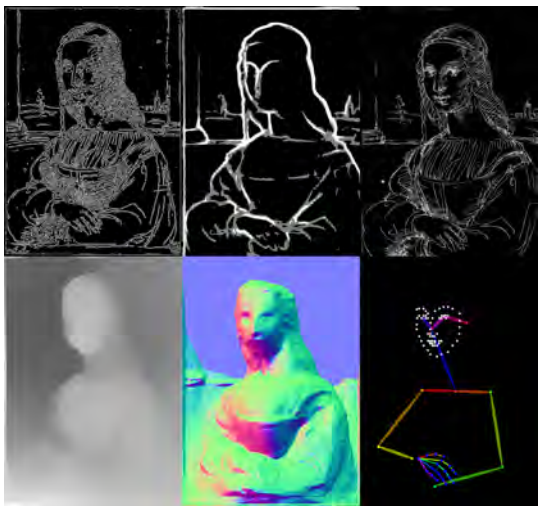

Figure 4.4: Experiment 1, ControlNet (Pre-Processing)

Figure 4.5: Experiment 1, ControlNet (Results)

Judging by the results in Figure 4.5, using ControlNets is a powerful way of conditioning Diffusion Models. Depending on the type of conditioning, their closeness to the input sketch varies. For instance, all edge detection algorithms (images 1-3 in Figure 4.5) create images closely related to the sketch, while OpenPose (image 6 in Figure 4.5) on the other hand, allows for a stronger deviation, especially around the facial features. This is expected because every condition contains only a certain characteristic of the sketch. Overall, the various ways of conditioning make ControlNet another powerful tool to creatively influence the diffusion process.

4.2.5 Evaluation

All in all, the fundamentals of the Conditional Diffusion Model offer a wide variety of possibilities to influence the diffusion process. With their architectural uniqueness, methods like SDEdit are powerful tools for image manipulation. In these experiments, the claims made by the authors of SDEdit (Meng et al., 2022) and Classifier-Free Guidance (Ho and Salimans, 2022) could be replicated.

Last but not least, it has to be noted that the results are pretty good, but not perfect. Especially small contextual details like hands or background elements are often displayed incorrectly, as seen in Figure 4.5. Interestingly, using specialized ControlNets like OpenPose, which contain information about face and hand placements, recreates these details correctly. This suggests that exploring additional ways of conditioning Diffusion Models could help to accurately reproduce small contextual details.

4.3 Experiment 2: Inpainting

In the second experiment, different architectural variations of the Diffusion Model are tested. To do this, the models are conditioned to solve three Inpainting tasks, on which they will be judged according to their performance and quality.

4.3.1 Methodology

Three images (see Figure 4.6) are chosen for this experiment. The first shows a woman wearing a mask, the second shows a man wearing a virtual reality headset and the last one depicts a woman wearing a jacket.

Figure 4.6: Experiment 2, Images

As it is an Inpainting task, three masks (see Figure 4.7) are created. In the first image, the mask of the woman should be removed, therefore this area is Inpainted. The same applies to the virtual reality headset in the second image. Last but not least, the woman wearing a jacket should receive a scarf, which is the reason the area around her jacket is Inpainted. Additionally, this is further supported by using a conditional text prompt “A woman wearing a scarf”.

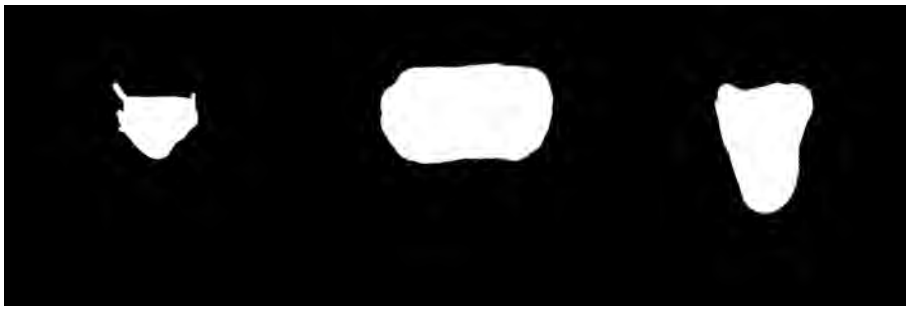

Figure 4.7: Experiment 2, Masks

4.3.2 DALL-E 2

DALL-E 2 is the first model being tested. It is accessible through the OpenAI API. For this, an OpenAI client with a corresponding API key is created.

Before a request can be made, the images undergo pre-processing. First, the mask is adapted by converting it into a RGBA images, incorporating the inverted mask in the alpha channel and retaining the original image in the RGB channels. Subsequently, the mask and the original image are converted into byte format before initiating the request.

Figure 4.8: Experiment 2, DALL-E 2

Looking at the results seen in Figure 4.8, the DALL-E 2 model can provide a contextually correct solution to the Inpainting tasks. The problem with these solutions is that they do not match the overall style of the image. They have a rather soft look that lacks essential high-frequency structures and contains unrealistic structures. Overall, the DALL-E 2 model only achieves a moderately good Inpainting output.

Due to the model being only available through an API request, the performance cannot be evaluated.

4.3.3 Cascaded Diffusion Model

In this second part of the architectural experiment, the Cascaded Diffusion Model is tested. Unfortunately, the original Cascaded Diffusion Model Imagen, developed by the Google Brain Research Team, is not publicly available. As a workaround, the Deep Floyd IF model by Stability AI is used for this experiment. This model comes with a zero-shot Inpainting version and has no model specifically trained in Inpainting.

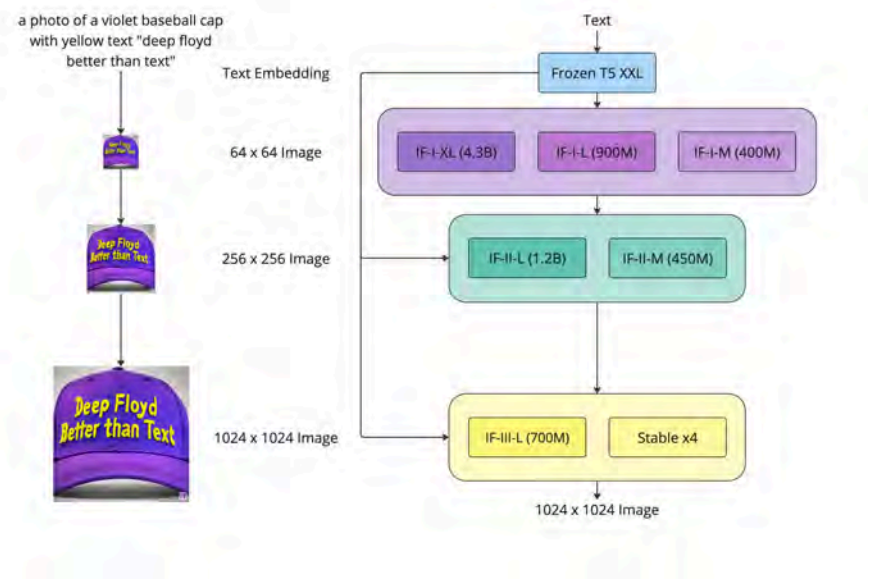

Figure 4.9: Experiment 2, Deep Floyd IF Schematic
(Shonenkov et al., 2024)

Deep Floyd (Shonenkov et al., 2024) is heavily inspired by Google Imagen, utilizing similar building blocks (Saharia et al., 2022). For the creation of text embeddings, it uses a T5 text encoder and the diffusion process consists of a multi-stage approach with a wide variety of differently sized Diffusion Models.

In this experimental design, the first stage uses the "IF-I-XL" model which creates a 64x64 image. The next stage uses the 64x64 image and the original image as input for the "IF-II-L" second stage model. During the second stage, a 256x256 image is created, which itself is used as input for the third stage. At this third stage, the Stable x4 latent diffusion upscaler is required because the pixel-based "IF-III-L" model has not been publicly released. This is unfortunate, as it prohibits testing a completely pixel-based approach. All pipelines are created using the HuggingFace Diffusers library.

Performance is a huge disadvantage of the Deep Floyd IF model. Running the model at a full 32-bit precision would require over 40GB of VRAM (Shonenkov et al., 2023), mainly due to the huge T5 Text Encoder (20GB) and the first stage U-Net (17.2GB). To get this model to run on a T4 GPU, optimization techniques and CUDA memory management are required.

Because of these memory issues, the experiment starts by loading the Text Encoder with an 8-bit precision and creating the textual embeddings. Subsequently, the T5 Text Encoder is deleted from the CUDA memory and the "IF-I-XL" model is initialized using a 16-bit precision.

This model, conditioned on the input images, is then used to create the 64x64 images of the first stage.

Figure 4.10: Experiment 2, Cascaded Diffusion Model, Stage 1

All images of the first stage (see Figure 4.10), excluding the second one, look promising but are still inconclusive regarding the success of the Inpainting task. For the next stage, the "IF-II-L" is loaded using a 16-bit precision and conditioned with the original images and the 64x64 generated images of the first stage.

Figure 4.11: Experiment 2, Cascaded Diffusion Model, Stage 2

In the second stage, 256x256 images (see Figure 4.11) are created, which exhibit an interesting founding. While they are mostly contextually correct, they fail to match the sharpness of the original image. Furthermore, the second image completely fails to produce a contextually sensible solution. Despite these negative findings, the second-stage images are used as the sole input for the Stable x4 Upscaler. For this purpose, the CUDA memory is cleared again and the pipeline is loaded using a 16-bit precision.

Figure 4.12: Experiment 2, Cascaded Diffusion Model, Stage 3

During the third and last stage, 1024x1024 images are created (see Figure 4.12). They exhibit certain misrepresentations outside the Inpainted regions. Without anticipating the analysis of the Latent Diffusion Model, this will be solved by creating a composite of the primary image and the masked area.

Figure 4.13: Experiment 2, Cascaded Diffusion Model, Composite

Even though this fixes artifacts outside the Inpainted area, there is still a lack of sharpness and detail inside the Inpainted area (see Figure 4.13).

Overall, the Deep Floyd IF Cascaded Diffusion model does not perform well on the task of Inpainting. With the last pixel-based third stage not being released and a scientific paper still missing, the model seems unfinished and further investigation is necessary to conclude if the architecture or the conception of the model is at fault. Additionally, the large Text Encoder (20 GB) makes it difficult to run the model on a T4 GPU.

4.3.4 LDM – Latent Diffusion Model

In the last part, the Latent Diffusion Model is evaluated. This is done using a specialized Stable Diffusion 1.5 Inpainting model (Rombach et al., 2022), that comes with a U-Net which has 5 additional input channels. Of these five input channels, four contain the encoded-masked image and one the mask itself. The model is implemented using the Diffusers inference pipeline.

A major advantage of the Latent Diffusion Model is its performance. With only one inference pipeline, it can be loaded on a T4 (16 GB) GPU without requiring any type of optimization.

Figure 4.14: Experiment 2, Latent Diffusion Model

Looking at the results (see Figure 4.14), the Latent Diffusion Model is able to achieve good results for all tasks. In the first image, the Inpainting job looks seamless, but the second and third show minor artifacts. Inside the second image, there exists a structural misrepresentation with the inpainted area not completely matching the rest of the image.

Figure 4.15: Experiment 2, Autoencoder Artefacts

Additionally, the third image contains similar artifacts to those found in the Latent Diffusion stage of the Cascaded Diffusion Model (see Figure 4.12). These artifacts are created due to a reconstruction error by the Autoencoder. This can be proven by creating an Autoencoder pipeline that simply encodes and decodes an image. An example of this can be seen in Figure 4.15.

Figure 4.16: Experiment 2, Latent Diffusion Model, Composite

Again, a workaround for this issue is to create a composite of the masked area and the original image. As seen in Figure 4.16, this seamlessly resolves the issue but remains a disadvantage of Latent Diffusion Models.

Overall, the image quality and performance of the Latent Diffusion Model is really good. Unfortunately, the tasks are not completed perfectly, with structural misrepresentation still present in the second image and faulty reconstruction occurring in the third image.

4.3.5 Evaluation

In the category of performance, Stable Diffusion is the best-performing model. Utilizing a departure into the latent space, it fits easily in the T4 testing GPU. The opposite is true for the Deep Floyd IF model. Here, just the large T5 Text Encoder alone would require 20 GB VRAM on full precision. The DALL-E 2 model is exempt from this category because it is only available via API.

Judging in terms of quality, the Stable Diffusion model comes back on top again. Although there are visible issues (see Figure 4.15), the images created by the Latent Diffusion Model are the closest to a real solution. Interestingly, in the second image, every model fails to accurately reproduce the skin texture of the man. These problems of reproducing a photorealistic level of detail are even more pronounced in DALLE-2 and Deep Floyd IF. Further investigation is

needed to evaluate if these deviations occur because of the lower space representations or the Diffusion Model itself.

Overall, comparing different architectures is complicated and not always a fair endeavor. It could be possible, that using a dedicated Inpainting model for the Cascading Diffusion Model could give better results, Furthermore, the performance of a Cascading architecture could be optimized by using a smaller text encoder. Finally, the fact that DALLE-2 is not publicly available is really unfortunate, because it complicates research.

4.4 Experiment 3: Personalization

The final experiment evaluates the different Fine-Tuning methods of Diffusion Models. In this section, a personalized concept is added to the model.

4.4.1 Methodology

For this experiment, five images from a photo shoot of a woman in different poses and outfits are used (see Figure 4.17). The goal is to enable the Diffusion Model to learn her features, allowing the generation of new images without the need for an additional photoshoot.

Figure 4.17: Experiment 3, Dataset

All three Fine-Tuning methods are implemented using scripts from the HuggingFace Diffusers library. These scripts are run using the Accelerate library (Gugger et al., 2022) to simplify the use of multi-GPU and mixed precision training. The training is done on the "Realistic Vision 6.0" Stable Diffusion 1.5 model (SG161222, 2024), which was designed for photorealism.

After the training of each model, three images are created using the prompts "A photorealistic portrait of a woman", "A painting of a woman in the style of vincent van gogh starry night" and "A photorealistic portrait of a woman in a space suit". The first prompt is expected to create results similar to the dataset images, while the other two are designed to produce more creative outcomes.

4.4.2 Dreambooth

The first tested method is DreamBooth, which requires the specification of an [identifier] and a [class noun] (Ruiz et al., 2023). In the case of the identifier, the rarely used token "sks"

is used and the class noun, describing the overall class of the dataset, is termed “woman”.

Judging from a performance perspective, this method is resource-intensive because it requires training the complete Diffusion Model. Running this model without optimization techniques and in the fastest way possible would require more than 30GB of VRAM. To enable running on a 16GB T4 GPU, an 8-bit Adam optimizer is used.

Training a DreamBooth model carries a risk of overfitting and is sensitive to hyperparameters. It is recommended to use a low learning rate and a high number of training steps (Patil et al., 2022). In this experiment, good results were achieved using the default script parameters, which specify a learning rate of 5e-6 and 400 training steps. Finally, when working with a small dataset, a batch size of one is utilized and all images are adapted to 512x512, matching the resolution of the Stable Diffusion Model.

Figure 4.18: Experiment 3, DreamBooth

While not achieving complete photorealism, the images (see Figure 4.18) created using DreamBooth recognizably display the woman from the training dataset. This works in scenarios closely matching the dataset as well as in scenarios strongly differing from the dataset.

When the model is saved, it has a sizing of a couple of gigabytes matching the size of the original Diffusion Model. This is reasonable considering the whole model was trained. However, this is a major disadvantage making models not easily shareable.

4.4.3 Low-Rank Adaptation

Having achieved good results using the DreamBooth Fine-Tuning method, the next step will test Fine-Tuning using Low-Rank Adaptation (LoRA). Inside the Stable Diffusion Model, this step is applied to the Cross-Attention layers (Cuenca and Sayak, 2023), which are responsible for the relationship between image and text representations.

Low-Rank Adaptation does not require a lot of computational resources. They are easily runnable on as little as 11GB of VRAM without optimization. This is due to the fact that only the Cross-Attention layers are trained.

Training is conducted using 400 steps, matching the DreamBooth experiment. Furthermore, a higher learning rate of 1e-04 is combined with a cosine learning rate scheduler and a batch size of one is used. These values are left untouched from the example experiment of the HuggingFace script (Cuenca and Sayak, 2023).

Before the training can be started, it is necessary to create image captions that describe the images inside the dataset. While this can be done manually, this experiment utilizes the BLIP model (Li et al., 2022) for automatic caption creation. Last but not least, all captions are added the "sks" token which triggers the LoRA after training is completed.

Figure 4.19: Experiment 3, LoRA

The method of Low-Rank Adaptation (LoRA) achieves moderate results. While the facial structure might be reproduced with slightly less accuracy, the model is capable of creating images that resemble the subject of the dataset (see Figure 4.19).

Finally, the saved model is only a couple of megabytes in size and easily shareable.

4.4.4 Textual Inversion

Textual Inversion is the last tested Fine-Tuning method in this experiment. In this method, a new "word" is integrated into the embedding space of the text encoder. It is necessary to specify a placeholder token as well as an initializer token (Gal et al., 2022). The placeholder token is "<sks>" and the initializer token, which describes the object, is called "woman". Additionally, the type of learnable property is defined. As this experiment trains an object instead of a style, the learnable property is an object.

Similar to Low-Rank Adaptation (LoRA), Textual Inversion does not require a lot of computational resources. By training only the textual embeddings of the Text Encoder, it can easily be loaded on a T4 (16GB) GPU without the use of specialized optimization techniques.

For this method a scalable learning rate of 5e-04, matching the values of the original paper (Gal et al., 2022), is used. Furthermore, a batch size of 1 and 400 training steps are used. This corresponds to the values used in previous experiments.

Figure 4.20: Experment 3, Textual Inversion

Judging by the results (see Figure 4.20), Textual Inversion displays only a weak integration of the visual concept. The model is unable to create a convincing representation of the subject, although outlines are recognizable.

Finally, the saved model is only a few kilobytes in size, making it easily shareable.

4.4.5 Evaluation

All methods can integrate the personalized concept to a certain degree. Judging by the results, the best choice appears to be between DreamBooth and Low-Rank Adaptation. In comparison, DreamBooth achieves the best results but requires the most resources and creates the largest models. On the other hand, Low-Rank Adaptations create results of moderate quality, while requiring fewer resources and creating smaller models.

Unfortunately, in this experimental setting, Textual Inversion does not provide good results. Further research has to be done to see if there are ways how this method could be made viable. Possible options include hyperparameter optimization or using a different/larger dataset.

Last but not least, although all created images convincingly depict the woman, they fail to achieve photorealism. This is primarily due to incorrect contextual elements (e.g., hands) and missing structural details (e.g., skin pores).

5 Resolution

This final chapter integrates all findings into a broader context, addressing the problems and implications of the Conditional Diffusion Model leading to a conclusion that revisits the aim of the thesis.

5.1 Discussion

In this discussion, the Conditional Diffusion Model is placed into a broader context, analyzing the findings and their implications. It addresses the role of the Diffusion Model in the Generative Trilemma, how close its quality approaches photorealism and what societal impact this might have.

5.1.1 The Generative Learning Trilemma

In their paper, Xiao et al., 2022 presents the idea of the Generative Learning Trilemma. This Trilemma states that none of the major generative architectures can achieve fast sampling, high quality and diversity at the same time. When examining the Diffusion Model, the Trilemma says that it achieves a high quality and good diversity, but not a fast sampling.

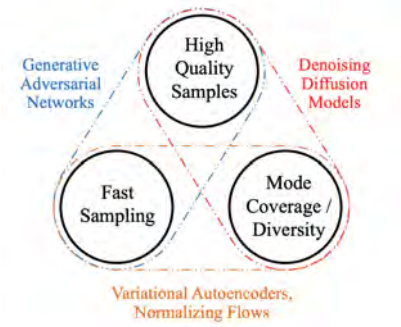

Figure 5.1: The Generative Learning Trilemma (Xiao et al., 2022)

This notion is reproduced within the findings of this thesis. Sampling requires multiple denoising steps and is far from away being fast. Additionally, most Diffusion Models require dedicated GPUs with at least 10GB of VRAM (Rombach et al., 2022) and if the trend towards using larger text encoders (Saharia et al., 2022) continues, this issue will only worsen in the future.

To fix this issue, lots of research is done trying to optimize the performance of Diffusion Models. One approach involves optimizing the architecture of the Diffusion Model, as seen in the case of Latent Diffusion Models (Rombach et al., 2022). Another approach is to enhance the reversal process by employing non-Markovian sampling (J. Song et al., 2022) or approximating SDEs/ODEs through methods like the Eule-Maruyama or Heun's method (Y. Song et al., 2021).

Interestingly, it seems like current approaches can not fix the inherent flaw of the diffusion architecture. They rather trade off performance over quality (J. Song et al., 2022). All in all, finding out if the Generative Learning Trilemma will be solvable is an interesting topic in determining the future of Diffusion Models.

At the writing of this thesis, current research can reduce sampling up to 1-4 reversal steps using techniques like Adversarial Diffusion Distillation (Sauer et al., 2023) or Latent Consistency Models (Y. Song et al., 2023). If these techniques are shown to be successful, future Diffusion Models could be used in real-time applications, but still require dedicated hardware.

5.1.2 Photorealism

While the Generative Learning Trilemma (see Figure 5.1) claims that Diffusion Models can create high-quality samples, they often fail to provide a photorealistic solution. This issue can be split up into contextual and structural problems.

Before beginning this analysis, it is important to highlight that image quality was assessed through simple human evaluation. Several alternative methods exist for evaluating the image quality of Generative Models. For instance, the Fréchet Inception Distance (FID) measures the distance between the distribution of Inception-V3 features of real images and those of generated images (Heusel et al., 2018). A lower FID score indicates better quality. Despite their widespread usage in research, metrics like FID often produce results that differ from human evaluations and solely offer a quantitative analysis (Jayasumana et al., 2024). Considering these factors, human evaluation is used to obtain a more realistic and in-depth analysis.

Contextual

In these contextual problems, Diffusion Models create images that have small details like eyes and hands displayed incorrectly or even completely missing.

One reason for these problems can be found in the second experiment (see Figure 4.15). Here, the departure into the latent space or the usage of an Autoencoder introduces artifacts. As a Latent Diffusion Model was mostly used in all experiments, this explains the occurrence of this problem in almost all experiments.

Interestingly, one way of solving this issue was found when providing the model with more accurate conditioning. For example, during the ControlNet Experiment (see Figure 4.4) OpenPose conditioning gave Information about facial structure and hand placement. This caused the model to reproduce these details more accurately, indicating that conditioning could counter this reconstructional problem.

Further investigations are needed to evaluate if this problem can be solved using additional conditioning information. Besides that, looking into different architectures or Autoencoders improvements could be beneficial as well. Unfortunately, during testing none of the other architectures tested could reach a performance and quality equal to that of Latent Diffusion Models.

Structural

Besides contextual details being displayed incorrectly, many of the generated images are also missing structural details like skin pores. This makes images generated by Diffusion Models look rather soft and artificial. In the Fine-Tuning experiment, this can be seen when comparing the images of the DreamBooth model to the original dataset (see Figure 4.18 and 4.17). While the face is recognisable the produced images do not look like a convincing photograph.

What would come to mind first is that the departure into the latent space might also be responsible for removing these details, but when looking at Figure 5.2, which was created by the pixel-based Score Based Generative Model, this soft look is also visible.

It could be plausible that this problem is inherent to the diffusion architecture itself because fine details like skin textures are not that dissimilar from noise. If a Diffusion Model creates an image by denoising, it runs into the risk of removing these noise-like structures.

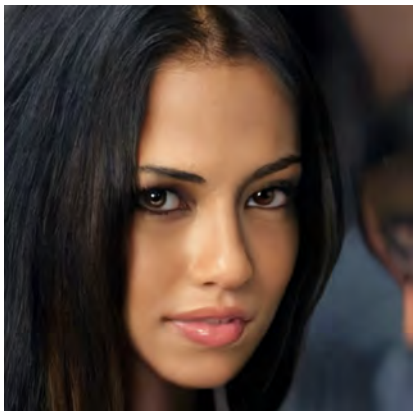

Figure 5.2: Pixel-based
(Y. Song and Ermon, 2020)

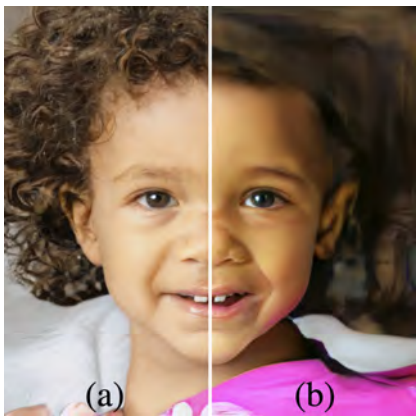

Figure 5.3: (a) Noise (b) No Noise
(Karras et al., 2019)

A good indicator of this can be found in the importance of noise in the generative process of the StyleGAN architecture. Here, Karras et al., 2019 discovers that adding noise into the layers of the synthesis network promotes the creation of finer background details and skin pores (see Figure 5.3). Ablating these noise inputs causes the images to obtain a soft and painterly look that has similar qualities to that found in images generated by Diffusion Models.

Investigating if an additional refining step could add these details or if the diffusion architecture itself can be adjusted could solve these issues.

5.1.3 Societal Impact

While complete photorealism might not be achieved, there already have been images good enough to trick the public. For example, a viral image of Pope Franziskus (see Figure 5.4) in a stylish down jacket made international headlines.

Figure 5.4: Deep Fake, Pope Franziskus (Huang, 2023)

Additionally, the results of the first experiment show how easily diffusion models can recreate the style of an artist (see Figure 4.5) and do a job that would have required many man-hours, in a couple of minutes. This can become a problem concerning copyright violations as well as changes in the labor market.

Copyright

The recreation of an image in the style of a Renaissance painter causes no copyright issues because the creators died a long time ago. However, this does not hold true if a contemporary artist is being copied.

A major problem is that models like Stable Diffusion can recreate many contemporary artists because they are trained on the LAION-5B dataset. According to Schuhmann et al., 2022, the LAION-5B dataset is a reference dataset that indexes to the internet and completely lacks any kind of copyright surveillance.

This has already caused a lawsuit between Stability AI and Getty Images (Brittain, 2023). Here, Getty Images claims that Stability AI violated copyright laws by training Stable Diffusion on

their images without having a license. Trying to avoid further legal troubles, other models are already adapting. For example, OpenAI is limiting DALL-E 3's (Betker et al., 2023) ability to generate images by following a strict content policy.

Precedent cases and governmental regularisation like the European Artificial Intelligence Act (Satariano, 2023) are needed to obtain a clear understanding of how the Conditional Diffusion Model can be used.

Deep Fakes

Another difficult area is the risk of conditioning Diffusion Models for malicious use, like the creation of Deep Fakes. This has been one of the reasons why the Imagen model has not been publicly released by the Google Research Team (Saharia et al., 2022).

While Deep Fakes are a big threat, there is already a promising Joint Developmental Foundation project trying to stop them. This project called, "Coalition for Content Provenance and Authenticity (C2PA)" is an alliance between big tech companies like Adobe, Microsoft and Intel (C2PA, 2024). It works on a technical standard that ensures that every media contains metadata, showing every editing step that was made. This is done utilizing digital signatures and hashcodes to protect their truthfulness.

If such a standard became the norm, images manipulated by Diffusion Models would easily be recognizable.

Labor Market

The last point concerns the influence Conditional Diffusion Models can have on the labor market.

Turning a sketch into a painting could take a skilled worker many man-hours. Judging by the results of the first experiment (see Figure 4.5), a Diffusion Model could potentially do this in a couple of minutes in the future. Additionally, methods like DreamBooth could be used to substitute photographers for product and portrait photography.

While it is unclear if this will result in a rise in productivity or a loss of jobs, companies and workers in the fields affected by Conditional Diffusion Models should be aware of the potential effects these developments could have.

5.2 Conclusion

Reflecting on the aim of this thesis, it attempted to investigate the capabilities and uses of the Conditional Diffusion Model, leading to its limitations and implications

In image manipulation, the Conditional Diffusion Model is an extremely capable tool. It can turn sketches into paintings, solve Inpainting tasks and integrate personalized concepts. Using the fundamental conditioning techniques (SDEdit, Classifier-Free Guidance, ControlNet) the model can be guided in numerous ways offering a great creative toolkit. For the integration of personalized concepts, the methods of DreamBooth and Low-Rank Adaptation offer a good choice between performance and quality.

Performance is one of the main limitations. Most research has tried to fix this issue by either optimizing the architecture of the Diffusion Model or enhancing the diffusion process. While these techniques have been successful, it seems that they always come with a trade-off between performance and quality.

Quality is another important limitation of the Conditional Diffusion Model. The Diffusion Model often fails to create photorealistic images. As some of the quality issues may stem from the architecture itself, future research is required to see if these obstacles can be overcome.

Despite all these limitations Diffusion Models are good enough to have a societal impact. For all tasks not depending on photorealism Diffusion Models can speed up their creation process. To protect Diffusion Models from being exploited for Deep Fakes and copyright violations precedent cases and new legislation are needed. Furthermore, it is sensible to introduce a system that can track image manipulation caused by Diffusion Models.

Bibliography

Aghajanyan, A., Zettlemoyer, L., & Gupta, S. (2020). *Intrinsic dimensionality explains the effectiveness of language model fine-tuning*. arXiv: 2012.13255 [cs.LG].

Anderson, B. D. (1982). *Reverse-time diffusion equation models*. https://doi.org/10.1016/0304-4149(82)90051-5

Betker, J., Goh, G., Jing, L., Brooks, T., Wang, J., Li, L., Ouyang, L., Zhuang, J., Lee, J., Guo, Y., Manassra, W., Dhariwal, P., Chu, C., Jiao, Y., & Ramesh, A. (2023). *Improving image generation with better captions*. https://cdn.openai.com/papers/dall-e-3.pdf

Brittain, B. (2023, February). *Getty images lawsuit says stability ai misused photos to train ai*. https://www.reuters.com/legal/getty-images-lawsuit-says-stability-ai-misused-photos-train-ai-2023-02-06/

C2PA. (2024). *The coalition for content provenance and authenticity*. https://c2pa.org/

Cuenca, P., & Sayak, P. (2023). *Using lora for efficient stable diffusion fine-tuning*. https://huggingface.co/blog/lora

Dhariwal, P., & Nichol, A. (2021). *Diffusion models beat gans on image synthesis*. arXiv: 2105.05233 [cs.LG].

Foundation, M. L. (2024). *Leonardo da vinci's influence on raphael*. https://monalisa.org/2012/09/07/the-influence-of-leonardo-da-vinci-on-raffaello-sanzio-da-urbino/

Gal, R., Alaluf, Y., Atzmon, Y., Patashnik, O., Bermano, A. H., Chechik, G., & Cohen-Or, D. (2022). *An image is worth one word: Personalizing text-to-image generation using textual inversion*. arXiv: 2208.01618 [cs.CV].

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). *Generative adversarial networks*. arXiv: 1406.2661 [stat.ML].

Google Colab, G. R. (2024). *Google colab*. https://research.google.com/colaboratory/faq.html

Gugger, S., Debut, L., Wolf, T., Schmid, P., Mueller, Z., Mangrulkar, S., Sun, M., & Bossan, B. (2022). *Accelerate: Training and inference at scale made simple, efficient and adaptable.* https://github.com/huggingface/accelerate

He, K., Zhang, X., Ren, S., & Sun, J. (2015). *Deep residual learning for image recognition*. arXiv: 1512.03385 [cs.CV].

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., & Hochreiter, S. (2018). *Gans trained by a two time-scale update rule converge to a local nash equilibrium*. arXiv: 1706.08500 [cs.LG].

Ho, J., Jain, A., & Abbeel, P. (2020). *Denoising diffusion probabilistic models*. arXiv: 2006.11239 [cs.LG].

Ho, J., Saharia, C., Chan, W., Fleet, D. J., Norouzi, M., & Salimans, T. (2021). *Cascaded diffusion models for high fidelity image generation*. arXiv: 2106.15282 [cs.CV].

Ho, J., & Salimans, T. (2022). *Classifier-free diffusion guidance*. arXiv: 2207.12598 [cs.LG].

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., & Chen, W. (2021). *Lora: Low-rank adaptation of large language models*. arXiv: 2106.09685 [cs.CL].

Huang, K. (2023, April). *Why pope francis is the star of a.i.-generated photos*. https://www.nytimes.com/2023/04/08/technology/ai-photos-pope-francis.html

Hyvärinen, A. (2005). *Estimation of non-normalized statistical models by score matching*. http://jmlr.org/papers/v6/hyvarinen05a.html

Jayasumana, S., Ramalingam, S., Veit, A., Glasner, D., Chakrabarti, A., & Kumar, S. (2024). *Rethinking fid: Towards a better evaluation metric for image generation*. arXiv: 2401.09603 [cs.CV].

Karras, T., Aittala, M., Aila, T., & Laine, S. (2022). *Elucidating the design space of diffusion-based generative models*. arXiv: 2206.00364 [cs.CV].

Karras, T., Laine, S., & Aila, T. (2019). *A style-based generator architecture for generative adversarial networks*. arXiv: 1812.04948 [cs.NE].

Lhoest, Q., del Moral, A. V., Jernite, Y., Thakur, A., von Platen, P., Patil, S., Chaumond, J., Drame, M., Plu, J., Tunstall, L., Davison, J., Šaško, M., Chhablani, G., Malik, B., Brandeis, S., Scao, T. L., Sanh, V., Xu, C., Patry, N., … Wolf, T. (2021). *Datasets: A community library for natural language processing*. arXiv: 2109.02846 [cs.CL].

Li, J., Li, D., Xiong, C., & Hoi, S. (2022). *Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation*. arXiv: 2201.12086 [cs.CV].

Luo, C. (2022). *Understanding diffusion models: A unified perspective*. arXiv: 2208.11970 [cs.LG].

Meng, C., He, Y., Song, Y., Song, J., Wu, J., Zhu, J.-Y., & Ermon, S. (2022). *Sdedit: Guided image synthesis and editing with stochastic differential equations*. arXiv: 2108.01073 [cs.CV].

Nichol, A., & Dhariwal, P. (2021). *Improved denoising diffusion probabilistic models*. arXiv: 2102.09672 [cs.LG].

Nichol, A., Dhariwal, P., Ramesh, A., Shyam, P., Mishkin, P., McGrew, B., Sutskever, I., & Chen, M. (2022). *Glide: Towards photorealistic image generation and editing with text-guided diffusion models*. arXiv: 2112.10741 [cs.CV].

Patil, S., Cuenca, P., & Kozin, V. (2022). *Training stable diffusion with dreambooth using diffusers*. https://huggingface.co/blog/dreambooth

Podell, D., English, Z., Lacey, K., Blattmann, A., Dockhorn, T., Müller, J., Penna, J., & Rombach, R. (2023). *Sdxl: Improving latent diffusion models for high-resolution image synthesis*. arXiv: 2307.01952 [cs.CV].

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., & Sutskever, I. (2021). *Learning transferable visual models from natural language supervision*. arXiv: 2103.00020 [cs.CV].

Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., & Chen, M. (2022). *Hierarchical text-conditional image generation with clip latents*. arXiv: 2204.06125 [cs.CV].

Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., & Sutskever, I. (2021). *Zero-shot text-to-image generation*. arXiv: 2102.12092 [cs.CV].

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., & Ommer, B. (2022). *High-resolution image synthesis with latent diffusion models*. arXiv: 2112.10752 [cs.CV].

Ronneberger, O., Fischer, P., & Brox, T. (2015). *U-net: Convolutional networks for biomedical image segmentation*. arXiv: 1505.04597 [cs.CV].

Ruiz, N., Li, Y., Jampani, V., Pritch, Y., Rubinstein, M., & Aberman, K. (2023). *Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation*. arXiv: 2208.12242 [cs.CV].

Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E., Ghasemipour, S. K. S., Ayan, B. K., Mahdavi, S. S., Lopes, R. G., Salimans, T., Ho, J., Fleet, D. J., & Norouzi, M. (2022). *Photorealistic text-to-image diffusion models with deep language understanding*. arXiv: 2205.11487 [cs.CV].

Saharia, C., Ho, J., Chan, W., Salimans, T., Fleet, D. J., & Norouzi, M. (2021). *Image super-resolution via iterative refinement*. arXiv: 2104.07636 [eess.IV].

Satariano, A. (2023, December). *E.u. agrees on landmark artificial intelligence rules*. https://www.nytimes.com/2023/12/08/technology/eu-ai-act-regulation.html

Sauer, A., Lorenz, D., Blattmann, A., & Rombach, R. (2023). *Adversarial diffusion distillation*. arXiv: 2311.17042 [cs.CV].

Schuhmann, C., Beaumont, R., Vencu, R., Gordon, C., Wightman, R., Cherti, M., Coombes, T., Katta, A., Mullis, C., Wortsman, M., Schramowski, P., Kundurthy, S., Crowson, K., Schmidt, L., Kaczmarczyk, R., & Jitsev, J. (2022). *Laion-5b: An open large-scale dataset for training next generation image-text models*. arXiv: 2210.08402 [cs.CV].

SG161222. (2024). *Sg161222/realistic vision v6.0 b1 novae*. https://huggingface.co/SG161222/Realistic_Vision_V6.0_B1_noVAE

Shonenkov, A., Bakshandaeva, D., Konstantinov, M., Berman, W., & von Platen Apolinário, P. (2023). *Running if with diffusers on a free tier google colab*. https://huggingface.co/blog/if

Shonenkov, A., Konstantinov, M., Bakshandaeva, D., Schuhmann, C., Ivanova, K., & Klokova, N. (2024). *Deep-floyd/if*. https://github.com/deep-floyd/IF

Sohl-Dickstein, J., Weiss, E. A., Maheswaranathan, N., & Ganguli, S. (2015). *Deep unsupervised learning using nonequilibrium thermodynamics*. arXiv: 1503.03585 [cs.LG].

Song, J., Meng, C., & Ermon, S. (2022). *Denoising diffusion implicit models*. arXiv: 2010.02502 [cs.LG].

Song, Y. (2021). *Generative modeling by estimating gradients of the data distribution*. https://yang-song.net/blog/2021/score/

Song, Y., Dhariwal, P., Chen, M., & Sutskever, I. (2023). *Consistency models*. arXiv: 2303.01469 [cs.LG].

Song, Y., & Ermon, S. (2020). *Generative modeling by estimating gradients of the data distribution*. arXiv: 1907.05600 [cs.LG].

Song, Y., & Kingma, D. P. (2021). *How to train your energy-based models*. arXiv: 2101.03288 [cs.LG].

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., & Poole, B. (2021). *Score-based generative modeling through stochastic differential equations*. arXiv: 2011.13456 [cs.LG].

Thanh-Tung, H., & Tran, T. (2020). *On catastrophic forgetting and mode collapse in generative adversarial networks*. arXiv: 1807.04015 [cs.LG].

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2023). *Attention is all you need*. arXiv: 1706.03762 [cs.CL].

von Platen, P., Patil, S., Lozhkov, A., Cuenca, P., Lambert, N., Rasul, K., Davaadorj, M., & Wolf, T. (2022). *Diffusers: State-of-the-art diffusion models*.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. L., Gugger, S., … Rush, A. M. (2020). *Huggingface's transformers: State-of-the-art natural language processing*. arXiv: 1910.03771 [cs.CL].

Xiao, Z., Kreis, K., & Vahdat, A. (2022). *Tackling the generative learning trilemma with denoising diffusion gans*. arXiv: 2112.07804 [cs.LG].

Zhang, L., Rao, A., & Agrawala, M. (2023). *Adding conditional control to text-to-image diffusion models*. arXiv: 2302.05543 [cs.CV].

All sources were accessed on February 17, 2024, at 08:00 AM.

Appendix

.1 Notation notes

Unconditional Diffusion Model

$\mathbf{x}$	Image Vector
$p_{data}(\mathbf{x})$	Data Distribution
$p_\theta(\mathbf{x})$	Generative Model
f_θ	Neural Network Output
Z_θ	Normalising Constant

Score Based Generative Models

$\mathbf{x}$	Image Vector
$\tilde{\mathbf{x}}$	Image Vector, Perturbed
$\tilde{\mathbf{x}}_0$	Image Vector, Perturbed, Initial
$\tilde{\mathbf{x}}_{t-1}$	Image Vector, Perturbed, Previous
σ	Noise Schedule
π	Prior Distribution (e.g., Gaussian)
$\nabla_{\mathbf{x}} \log p(\mathbf{x})$	Score Function
$\mathbf{s}_\theta$	Score Based Generative Model
$\mathbf{s}_\theta(\mathbf{x}, \sigma)$	Score Based Generative Model, NCSN
ϵ, α_i	Step Size
tr()	Trace
$\mathbf{z}$	Sample from Gaussian Distribution

Denoising Diffusion Probabilistic Models

$\mathbf{x}_0$	Image Vector, Initial
$\mathbf{x}_t$	Image Vector, Noisy at t
$\mathbf{x}_T$	Image Vector, Noisy at T
$\mathbf{I}$	Identity Matrix
$q(\mathbf{x}_t \vert \mathbf{x}_{t-1})$	Process, Forward
$q(\mathbf{x}_t \vert \mathbf{x}_0)$	Process, Forward, Reparametrization
$p_\theta(\mathbf{x}_{t-1} \vert \mathbf{x}_t)$	Process, Reverse
β_t	Noise Schedule
α_t, $\bar{\alpha}_t$	Noise Schedule, Reparametrization
$\boldsymbol{\epsilon}$	Noise, Target
$\boldsymbol{\epsilon}_\theta(x_t, t)$	Noise, Prediction (Network)
T	Timestamps
z	Sampled from Gaussian Distribution, $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

Score-Based Generative Modeling through Stochastic Differential Equations

$\mathbf{f}(\mathbf{x}, t)$	Drift Term
$g(t)$	Diffusion Term
$\mathrm{d}\mathbf{x}$	Infinitesimal change in $\mathbf{x}$
$\mathrm{d}\mathbf{w}$	Infinitesimal increment of a Wiener process
$\mathrm{d}t$	Infinitesimal increment in time
$\lambda(t)$	Weighting function

Sinusoidal Position Embeddings

d_{model}	Dimensionality
i	Dimension
$\mathbf{PE}_{(pos,2i)}$	Function Sinus, Even Indices
$\mathbf{PE}_{(pos,2i+1)}$	Function Cosine, Odd Indices
pos	Position

Attention Module

d_k	Dimension, Keys
Attention()	Function, Attention
softmax()	Function, Softmax
$\mathbf{Q}$	Matrix, Query
$\mathbf{K}$	Matrix, Key
$\mathbf{V}$	Matrix, Value
$\mathbf{W}_{\mathbf{Q},\mathbf{K},\mathbf{V}}$	Weight Matrices (Query, Keys, Values)

Mathematical Foundations

$p(\mathbf{x})$	Unconditional Data Distribution
$p(\mathbf{x}\|\mathbf{y})$	Conditional Data Distribution
$\nabla_{\mathbf{x}} \log p(\mathbf{x})$	Unconditional Diffusion Model
$\nabla_{\mathbf{x}} \log p(\mathbf{x} \mid \mathbf{y})$	Conditional Diffusion Model
$\nabla_{\mathbf{x}} \log p(\mathbf{y} \mid \mathbf{x})$	Classifier
γ	Scaling Factor

DALL-E – Hierarchical Text-Conditional Image Generation with CLIP Latents

$\epsilon_\theta(\mathbf{x}_t\|\emptyset)$	Diffusion Model, Unconditional
$\epsilon_\theta(\mathbf{x}_t\|\mathbf{c})$	Diffusion Model, Conditional
$\emptyset$	Empty Sequence
$\mathbf{c}$	Conditioning
s	Scaling Factor
ϵ_θ	Diffusion model

CDM – Cascaded Diffusion Models

$\mathbf{x}$	Image Vector, Source Image
$\tilde{\mathbf{y}}$	Image Vector, Target Image, Noisy
$\mathbf{y}_0$	Image Vector, Target Image
f_θ	Denoising model
ϵ	Noise, Target
γ	Noise Schedule

LDM – Latent Diffusion Model

$\mathscr{E}$	Autoencoder, Encoder
$\mathscr{D}$	Autoencoder, Decoder
$\mathbf{Q}, \mathbf{K}, \mathbf{V}$	Cross-Attention, Query, Key, Value Matrices
$\mathbf{W_Q}, \mathbf{W_K}, \mathbf{W_V}$	Cross-Attention, Query, Key, Value Weight Matrices
$\tau_\theta(\mathbf{y})$	Domain Encoder (Conditioning)
$\mathbf{x}$	Image Vector, Pixel Space
$\tilde{\mathbf{x}}$	Image Vector, Pixel Space, Reconstructed
$\mathbf{z_t}$	Image Vector, Latent Space
$\varphi(\mathbf{z_t})$	Intermediate Representation, U-Net
$\boldsymbol{\epsilon}_\theta$	Noise, Denoising Model
$\boldsymbol{\epsilon}$	Noise, Target
H, h	Scaling, Height, Pixel and Latent
W, w	Scaling, Width, Pixel and Latent
f	Scaling, Downsampling factor

DreamBooth

$\mathbf{c}$	Conditioning, [identifier] [class noun]
$\mathbf{c_{pr}}$	Conditioning, [class noun]
$\hat{\mathbf{x}}_\theta$	Image Vector, Denoised, Diffusion Model
$\mathbf{x_{pr}}$	Image Vector„ Class Generated Image
$\mathbf{x}$	Image Vector„ Subject Image
$\boldsymbol{\epsilon}$	Noise
L_{Class}	Loss, Class Generated Images
$L_{Subject}$	Loss, Subject Images
λ	Weighting Term, Class Loss
$\omega_t, \alpha_t, \sigma_t$	Weighting Term, Image Quality and Noise Schedule

LoRA – Low Rank Adaptation

r	Rank
$\mathbf{A}$	Matrix, Normal Distribution
$\mathbf{B}$	Matrix, Zeros
$\mathbf{W}_0$	Weight Matrix, Pre-Trained
$\Delta\mathbf{W}$	Weight Matrix, Low Rank Adaptation

Textual Inversion

$c_\theta(\mathbf{y})$	Conditioning Vector (Textual Embedding)
$\mathbf{z}_t$	Image Vector, Latent
$\mathbf{v}_*$	Text Embedding Vector
S_*	User-specified Pseudo-Word
ϵ_θ	Noise, Denoising Model
ϵ	Noise, Target

Eigenständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit mit dem Titel

Conditioning Diffusion Models for image manipulation

selbstständig und nur mit den angegebenen Hilfsmitteln verfasst habe. Alle Passagen, die ich wörtlich aus der Literatur oder aus anderen Quellen wie z. B. Internetseiten übernommen habe, habe ich deutlich als Zitat mit Angabe der Quelle kenntlich gemacht.

Hamburg, 19. Februar 2024