Eötvös Loránd University
Faculty of Science

Milán Szabó

Diffusion Models and their Applications

MSc Thesis

Supervisor:

András Lukács
Department of Computer Science

Budapest, 2024

Acknowledgement

I would like to express my deepest gratitude to my advisor, András Lukács, for his constructive comments and suggestions, which greatly improved the quality of this work. His expertise and encouragement have been invaluable. I am also grateful to my family, girlfriend, and friends for their endless patience, understanding, and encouragement. Their support has been my foundation throughout this journey.

Contents

Introduction

In recent years, the field of generative modelling has witnessed remarkable advancements, fuelled by the relentless pursuit of understanding and modelling complex data distributions. At the forefront of this endeavour lies the domain of diffusion models, a family of stochastic processes that offer a compelling framework for generating intricate and high-dimensional data distributions.

Diffusion Models were first proposed by Sohl-Dickstein et al. [23] in 2015, inspired by non-equilibrium statistical physics. They are a type of generative model that has been used in several popular deep-learning models, such as DALL-E 2 [17], Stable Diffusion [18], Google Imagen [21], and GLIDE [14]. Not only do they possess the ability to produce diverse and high-quality samples, but they also exhibit flexibility and tractability, rendering them invaluable tools in various domains.

The primary purpose of diffusion models is to map training data to a latent space using a Markov chain. This process gradually adds noise to the data, resulting in an asymptotically transformed image that is Gaussian distributed in nature. Our goal is to learn the reverse of the Markov process, enabling us to generate new data by producing a Gaussian image and traversing the reverse process. Diffusion models have a wide range of applications, including text simplification, question generation, text-to-image generation, paraphrasing, and more.

The purpose of this project is to apply diffusion models to computer vision tasks, namely image segmentation. By leveraging the principles of diffusion models and their inherent capabilities, we aim to explore their efficacy in the context of image segmentation, a fundamental problem in computer vision with numerous real-world applications. The practical application of these models is demonstrated through their implementation on the COVID-QU dataset, a comprehensive collection of medical images specifically curated for COVID-19 research. This dataset provides a unique opportunity to evaluate the performance of diffusion models in real-world medical scenarios, particularly in the segmentation of lung and infection regions, which are critical for accurate diagnosis and treatment planning. Through theoretical investigation, empirical analysis, and practical implementations, this research endeavour seeks to advance our understanding of diffusion models and their applications in computer vision, contributing to the broader landscape of generative modelling and paving the way for innovative solutions to complex data modelling challenges.

Structure of the Thesis

In the first chapter, we provide a detailed overview of different types of diffusion models, starting with the fundamentals of Denoising Diffusion Probabilistic Models (DDPMs) and extending to more complex variants such as Noise Conditioned Score Networks, Stochastic Differential Equation (SDE)-based models, Brownian Bridge Diffusion Models (BBDMs), and Denoising Diffusion Implicit Models (DDIMs). The purpose of this section is to establish the theoretical foundations for the subsequent chapters and offer a comprehensive understanding of the nature

of this class of models.

The next chapter focuses on the practical applications of diffusion models in image synthesis. Specifically, it covers the implementation of these models on the COVID-QU dataset, detailing the process of training DDPMs, discussing the architecture of the models, and presenting the results of image synthesis. This chapter aims to demonstrate the practical viability and effectiveness of diffusion models in generating high-quality images.

In the third chapter, the focus remains on practical applications but shifts to conditional image synthesis. Here, we explore the use of diffusion models for conditional image synthesis, particularly in the context of image segmentation. The chapter thoroughly examines the architecture of the models, segmentation quality metrics, and results for lung and infection segmentation, showcasing the versatility and precision of diffusion models in medical image analysis.

So far, the practical chapters have focused on DDPM models. The fourth chapter specifically addresses the application of Brownian Bridge Diffusion Models (BBDMs) for image segmentation tasks, presenting detailed results for lung and infection segmentation on the COVID-QU dataset.

In the fifth chapter, we shift our focus to latent space models. This chapter delves into models such as VQ-GANs and compares different downsampling rates and the performance of Latent Diffusion Models (LDMs) and Latent Brownian Bridge Models. This exploration into latent space representations aims to enhance the efficiency and scalability of diffusion processes in high-dimensional image generation and segmentation tasks.

The final two sections summarize the key findings, contributions, and potential future directions of this research. Through this comprehensive exploration, this thesis aims to contribute to the growing body of knowledge in the field of generative models, providing valuable insights into their application in medical image analysis and beyond.

1 Diffusion Models

Diffusion Models represent a powerful paradigm within generative modelling, offering a systematic framework for generating complex data distributions. Rooted in principles of stochastic processes, diffusion models have emerged as versatile tools with wide-ranging applications spanning fields such as computer vision, and audio processing. Their aim is to map an unknown data distribution to a known distribution and back. This mapping to the latent distribution is done by gradually adding noise to our original data. Our goal is to learn to reverse this process, this way we can generate images by first sampling from the latent distribution and passing it through the estimate of the reverse process, obtaining a data point from the unknown distribution.

In this chapter, we embark on a journey through the theoretical foundations of three pivotal variations of Diffusion Models: Denoising Diffusion Probabilistic Models, Noise Conditioned Score Networks, and Stochastic Differential Equation Diffusion Models. While these models have proven adept at unconditional data synthesis tasks, the realm of conditional image synthesis poses distinct challenges. Most image-to-image translational computer vision tasks fall under this category, necessitating additional architectural complexities and encountering uncertain convergence when adapting existing frameworks. To address these challenges, Li et al.[11] introduced Brownian Bridge Diffusion Models, offering promising avenues for conditional synthesis tasks, as detailed in Section 1.3. Despite these advancements, one persistent obstacle in diffusion-based models remains their sluggish inference time. In response, two directions have proven to be successful in speeding up sampling. Song et al. [24] introduce Denoising Diffusion Implicit Models, where they define a new kind of non-Markovian inference process that shares the same marginals with DDPMs, while making sampling more efficient, their results are presented in Section 1.4. Rombach et al.[18] pioneered latent diffusion models, which conduct the diffusion process within the latent space of pre-trained autoencoders. Section 1.5 delves into the intricacies of Latent Diffusion Models, shedding light on their innovative approach to expediting inference. Moreover, to provide a comprehensive understanding of Latent Diffusion Models, Subsection 1.5 also elucidates the architecture of the foremost autoencoders employed in conjunction with these models. By unravelling the theoretical frameworks and practical implementations of these diverse diffusion models, we aim to illuminate their potential to revolutionize the landscape of generative modelling, enabling the creation of intricate and lifelike data distributions across various domains.

1.1 Denoising Diffusion Probabilistic Models

This subsection aims to give a theoretical formulation for the Denoising Diffusion Probabilistic Model (DDPMs) framework and builds heavily upon Ho et al. [9]. DDPMs were first introduced by Ho et al. [9] and are the first kind of diffusion models that could produce high-quality samples, while achieving state-of-the-art sample quality results. Although the idea of DMs has existed before, models before had no practical use, therefore most scholars regard Ho et al.'s formulation of diffusion models as the foundation of most DMs today. For this reason, we will present this article's main ideas in detail.

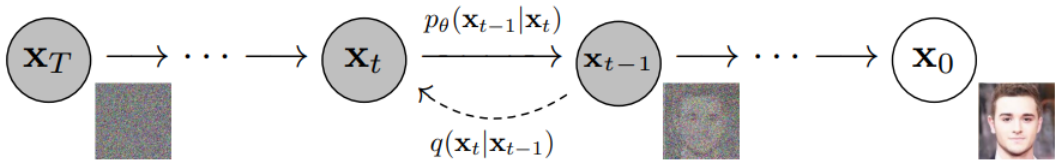

Figure 1: The forward and reverse processes of DDPMs [9]

As seen in Figure 1 DDPMs contain two main parts, the forward (or noising) and reverse process. In the case of DDPMs, the noising process is a Markov chain, which evolves according to

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I). \tag{1}$$

Where x_0 is our original sample, $x_1, \ldots, x_T$ are the latent variables during each step of the inference process, and $\beta_1, \ldots, \beta_T$ is a variance schedule. Under good settings of T and $\beta_1, \ldots, \beta_T$, $q(x_T)$ is nearly Gaussian. This is crucial since our ultimate goal is to learn the translation between the data distribution and the Gaussian distribution.

Not getting into the calculations too much (all ideas in this chapter are discussed in detail by Ho et al. [9]), sampling at a given t can be simplified by writing

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\overline{\alpha_t}}x_0, (1-\overline{\alpha_t})I), \tag{2}$$

$$\text{where } \alpha_t = 1 - \beta_t, \text{ and } \overline{\alpha}_t = \prod_{s=1}^{t} \alpha_t.$$

Using Bayes' theorem it can be proven that the $q(x_{t-1}|x_t, x_0)$ posteriors are also Gaussian [7]:

$$q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}, \tilde{\mu}(x_t, x_0), \tilde{\beta}_t \mathbb{I}),$$

with a mean that depends on the data x_0. This means that the reverse transitions depend on the whole data distribution, and so we must estimate them in order to take samples from the data distribution. So to sample from $q(x_0)$, first we would have to sample from $q(x_T)$ (which is essentially Gaussian) and then using the estimated reverse transitions we can take steps on the reverse process until we reach x_0.

Now that we have laid down the goal of the training process, let us introduce how we model the reverse transitions $q(x_{t-1}|x_t, x_0)$. Let our model that estimates the reverse transitions

$p_\theta(x_{t-1}|x_t)$ be of the following form:

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)),$$

where $\mu_\theta(x_t, t)$ and $\Sigma_\theta(x_t, t)$ are a class of neural networks indexed by θ. The goal of training is to find such weights for these neural networks, which maximize the log-likelihood of our training data. The model is trained by minimizing the negative log-likelihood of the training data.

$$\mathbb{E}_q(-\log p_\theta(x_0)) = \int -\log p_\theta(x^{(0)}) q(x^{(0)}) dx^{(0)}$$

Using this as a loss function is not optimal, so instead we aim to obtain an upper bound that we can easily and efficiently calculate.

$$\begin{aligned}
\int -\log p_\theta(x^{(0)}) q(x^{(0)}) dx^{(0)} &= \int -\log \Big[\int p_\theta(x^{(T)}) \prod_{t=1}^{T} \frac{p_\theta(x^{(t-1)}|x^{(t)})}{q(x^{(t)}|x^{(t-1)})} \cdot \\
&\quad q(x^{(1\ldots T)}|x^{(0)}) dx^{(1\ldots T)} \Big] q(x^{(0)}) dx^{(0)} \\
&\leq \int -\log \Big[p_\theta(x^{(T)}) \prod_{t=1}^{T} \frac{p_\theta(x^{(t-1)}|x^{(t)})}{q(x^{(t)}|x^{(t-1)})} \Big] \cdot \\
&\quad q(x^{(0\ldots T)}) dx^{(0\ldots T)} \qquad (3)
\end{aligned}$$

In line (3) Jensen's inequality was applied. In the end, we got a bound on the log-likelihood, commonly called the Evidence Lower Bound (ELBO). By applying ELBO for the inference and reverse distributions, we get one step closer to obtaining a tractable loss function.

$$\mathbb{E}\left[-\log p_\theta\left(\mathbf{x}_0\right)\right] \leq \mathbb{E}_q\left[-\log \frac{p_\theta\left(\mathbf{x}_{0:T}\right)}{q\left(\mathbf{x}_{1:T} \mid \mathbf{x}_0\right)}\right] = \mathbb{E}_q\left[-\log p\left(\mathbf{x}_T\right) - \sum_{t \geq 1} \log \frac{p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)}{q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}\right)}\right] =: L.$$

Further deconstructing L into a sum of divergences.

$$\begin{aligned}
L &= \mathbb{E}_q\left[-\log \frac{p_\theta\left(\mathbf{x}_{0:T}\right)}{q\left(\mathbf{x}_{1:T} \mid \mathbf{x}_0\right)}\right] \\
&= \mathbb{E}_q\left[-\log p\left(\mathbf{x}_T\right) - \sum_{t \geq 1} \log \frac{p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)}{q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}\right)}\right] \\
&= \mathbb{E}_q\left[-\log p\left(\mathbf{x}_T\right) - \sum_{t > 1} \log \frac{p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)}{q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}\right)} - \log \frac{p_\theta\left(\mathbf{x}_0 \mid \mathbf{x}_1\right)}{q\left(\mathbf{x}_1 \mid \mathbf{x}_0\right)}\right] \\
&= \mathbb{E}_q\left[-\log p\left(\mathbf{x}_T\right) - \sum_{t > 1} \log \frac{p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)}{q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0\right)} \cdot \frac{q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_0\right)}{q\left(\mathbf{x}_t \mid \mathbf{x}_0\right)} - \log \frac{p_\theta\left(\mathbf{x}_0 \mid \mathbf{x}_1\right)}{q\left(\mathbf{x}_1 \mid \mathbf{x}_0\right)}\right] \\
&= \mathbb{E}_q\left[-\log \frac{p\left(\mathbf{x}_T\right)}{q\left(\mathbf{x}_T \mid \mathbf{x}_0\right)} - \sum_{t > 1} \log \frac{p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)}{q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0\right)} - \log p_\theta\left(\mathbf{x}_0 \mid \mathbf{x}_1\right)\right]
\end{aligned}$$

With the derivations above we got a loss function containing terms of Kullback-Leibler (KL) divergences, an asymmetrical distance of probability measures. Fortunately, the KL divergence has a closed form for Gaussian distributions. Also, let us fix $\Sigma_\theta(x_{t-1}, t) = \sigma_t \mathbf{I}$, Ho et al. [9] did

not find any benefit by training $\Sigma_\theta(x_{t-1}, t)$ in their experiments as well as this simplifies the calculation of the objective function.

$$\mathbb{E}_q[\underbrace{D_{\mathrm{KL}}\left(q\left(\mathbf{x}_T \mid \mathbf{x}_0\right) \| p\left(\mathbf{x}_T\right)\right)}_{L_T} + \sum_{t>1} \underbrace{D_{\mathrm{KL}}\left(q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0\right) \| p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)\right)}_{L_{t-1}} \underbrace{-\log p_\theta\left(\mathbf{x}_0 \mid \mathbf{x}_1\right)}_{L_0}]$$

As mentioned before the reverse posteriors depend on the original data, specifically they can be calculated as

$$q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0\right) = \mathcal{N}\left(\mathbf{x}_{t-1}; \tilde{\mu}_t\left(\mathbf{x}_t, \mathbf{x}_0\right), \bar{\beta}_t \mathbf{I}\right),$$
$$\text{where} \quad \tilde{\boldsymbol{\mu}}_t\left(\mathbf{x}_t, \mathbf{x}_0\right) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}\left(1-\bar{\alpha}_{t-1}\right)}{1-\bar{\alpha}_t}\mathbf{x}_t,$$
$$\text{and } \tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t.$$

By applying the closed form of KL divergence for Gaussian distributions, the L_{t-1} term reduces to an easily computable form.

$$L_{t-1} = \mathbb{E}_q\left[\frac{1}{2\sigma_t^2}\left\|\tilde{\boldsymbol{\mu}}_t\left(\mathbf{x}_t, \mathbf{x}_0\right) - \boldsymbol{\mu}_\theta\left(\mathbf{x}_t, t\right)\right\|^2\right] + C.$$

Since $x_t(x_0, \varepsilon) = \sqrt{\overline{\alpha}}x_0 + \sqrt{(1-\overline{\alpha})}\varepsilon$, where $\varepsilon \approx \mathcal{N}(0, I)$, and using the definition of $\tilde{\mu}_t(x_t, x_0)$, we can rewrite the above term as:

$$\begin{aligned} L_{t-1} - C &= \mathbb{E}_{\mathbf{x}_{0,\epsilon}}\left[\frac{1}{2\sigma_t^2}\left\|\tilde{\mu}_t\left(\mathbf{x}_t\left(\mathbf{x}_0, \boldsymbol{\epsilon}\right), \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t\left(\mathbf{x}_0, \boldsymbol{\epsilon}\right) - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}\right)\right) - \mu_\theta\left(\mathbf{x}_t\left(\mathbf{x}_0, \boldsymbol{\epsilon}\right), t\right)\right\|^2\right] \\ &= \mathbb{E}_{\mathbf{x}_{0,\epsilon}}\left[\frac{1}{2\sigma_t^2}\left\|\frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t\left(\mathbf{x}_0, \boldsymbol{\epsilon}\right) - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}\right) - \mu_\theta\left(\mathbf{x}_t\left(\mathbf{x}_0, \boldsymbol{\epsilon}\right), t\right)\right\|^2\right] \end{aligned}$$

We can see that $\mu_\theta(x_t, x_0)$ has to approximate $\frac{1}{\sqrt{\overline{\alpha}}}\Big(x_t(x_0, \varepsilon) - \frac{\beta_t}{\sqrt{1-\overline{\alpha}}}\varepsilon\Big)$. Let's reparametrize $\mu_\theta(x_t, x_0)$ the following way:

$$\mu_\theta\left(\mathbf{x}_t, t\right) = \tilde{\mu}_t\left(\mathbf{x}_t, \frac{1}{\sqrt{\bar{\alpha}_t}}\left(\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\epsilon_\theta\left(\mathbf{x}_t, t\right)\right)\right) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta\left(\mathbf{x}_t, t\right)\right)$$

Now we have a much simpler loss term where our model only needs to approximate the noise.

$$\mathbb{E}_{x_0,\varepsilon}\left[\frac{\beta_t^2}{2\sigma_t^2\alpha_t(1-\overline{\alpha_t})}||\varepsilon - \varepsilon_\theta(\sqrt{\overline{\alpha_t}}x_0 + \sqrt{1-\overline{\alpha_t}}\varepsilon, t)||^2\right].$$

Ho et al.[9] also found that it's more beneficial to modify the weighing term in the previous loss. The final loss function is defined below, where t is sampled according to the uniform distribution on $1, \ldots, T$.

$$L_{\text{simple}}(\theta) = \mathbb{E}_{x_0,\varepsilon,t}\left[||\varepsilon - \varepsilon_\theta(\sqrt{\overline{\alpha_t}}x_0 + \sqrt{1-\overline{\alpha_t}}\varepsilon, t)||^2\right].$$

Finally, they gave a model architecture for $\varepsilon_\theta(x_t, t)$. It is a variation of U-Net, that shares parameters across time. The Transformer sinusoidal position embeddings of t are added at

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
6: **until** converged

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

Figure 2: Training and sampling algorithm for DDPMs by Ho et al. [9]

each layer, thus specifying t to the network. This is defined as follows, where k is the size of the embedding.

$$\text{PE}(t, 2i) = \sin\left(\frac{t}{10000^{2i/k}}\right)$$

$$\text{PE}(t, 2i+1) = \cos\left(\frac{t}{10000^{2i/k}}\right)$$

Additionally, the network contains self-attention at resolution 16×16.

To give some context to this section's dense theoretical results, and to make these ideas more easily digestible, we have included the summarized training and sampling algorithms in Figure 2.

1.2 Noise Conditioned Score Networks and SDE-based Diffusion Models

In addition to the Denoising Diffusion Probabilistic Models presented in the previous section, there are multiple alternative frameworks of diffusion models, which might be worth mentioning. In this short subsection, we delve into the foundational principles of two important variations of DMs. Noise Conditioned Score Networks and SDE based Diffusion Models provide a different interpretation of the meaning of the DDPM framework and hopefully give us a deeper understanding of them.

Noise Conditioned Score Networks (NCSNs) represent a pioneering approach within the realm of generative modelling, leveraging the concept of score matching to learn complex data distributions. Unlike traditional generative models, which directly parameterize the data distribution, NCSNs focus on learning the score function (a gradient of the log-density) of the data distribution. By conditioning this score function on a noise source, NCSNs facilitate the generation of high-quality samples while providing enhanced flexibility and tractability. This paradigm shift not only enables the generation of diverse and realistic data but also allows for efficient inference and training. As mentioned before, Noise Conditioned Score Networks [25] aim to estimate the gradient of the data density. Formally, $\sigma_1 < \sigma_2 < \ldots < \sigma_T$ is a sequence of Gaussian noise scales, $p_{\sigma_1}(x) \sim q(x_0)$ (the data density), $p_{\sigma_T}(x) \sim \mathcal{N}(0, I)$, and $p_{\sigma_t}(x_t|x) \sim \mathcal{N}(x_t; x, \sigma_t I)$. Their aim is to estimate $\nabla_{x_t} p_{\sigma_t}(x_t)$, we know that $\nabla_{x_t} p_{\sigma_t}(x_t|x) = \dfrac{x_t - x}{\sigma_t}$, so they minimize the

following loss function, where $s_\theta(x_t, \sigma_t)$ is a neural network.

$$\frac{1}{T}\sum_{t=1}^{T}\lambda(\sigma_T)E_{q(x)}E_{x_t\sim p_{\sigma_t}(x_t|x)}||s(x_t,\sigma_t)-\frac{x_t-x}{\sigma_t}||$$

The estimated gradient is then used to iteratively denoise a sample from $\mathcal{N}(0, I)$, given the timestep. Namely, the sampling algorithm is the Langevin dynamics algorithm [25] described in Figure 3.

Algorithm 2 Annealed Langevin dynamics

Input:
$\sigma_1, \ldots, \sigma_T$ – a sequence of Gaussian noise scales.
N – the number of Langevin dynamics iterations.
$\gamma_1, \ldots, \gamma_T$ – the update magnitudes for each noise scale.

Output:
x_0^0 – the sampled image.

Computation:
1: $x_T^0 \sim \mathcal{N}(0, \mathbf{I})$
2: **for** $t = T, \ldots, 1$ **do**
3: **for** $i = 1, \ldots, N$ **do**
4: $\omega \sim \mathcal{N}(0, \mathbf{I})$
5: $x_t^i = x_t^{i-1} + \frac{\gamma_t}{2} \cdot s_\theta(x_t^{i-1}, \sigma_t) + \sqrt{\gamma_t} \cdot \omega$
6: $x_{t-1}^0 = x_t^N$

Figure 3: Langevin dynamics sampling algorithm [25]. Figure by Croitoru et al. [5]

Stochastic Differential Equations [26] based data synthesis is another interesting approach. Just like the previous two methods, here the data is also transformed into noise. However, the diffusion process is considered to be continuous, and it is defined to be the solution of an SDE. The SDE describing the diffusion process is the following,

$$\frac{\partial x}{\partial t} = f(x,t) + \sigma(t)\omega_t \iff \partial x = f(x,t)\cdot\partial t + \sigma(t)\cdot\partial\omega,$$

where ω_t are standard normal variables, σ is a time-dependent function that computes the diffusion coefficient, and f computes the drift coefficient. To have a diffusion process as a solution, the drift coefficient should be designed such that x_t gradually becomes pure noise. Now the aim is just like before, to reverse this process. The reverse is defined as,

$$\partial x = [f(x,t) - \sigma(t)^2 \cdot \nabla_x \log p_t(x)]\cdot\partial t + \sigma(t)\cdot\partial\hat{\omega},$$

where $\hat{\omega}$ is the time reversed Brownian motion. The job of our neural network is to learn $\nabla_x \log p_t(x)$, just like before. To this end, we train the following objective, which is a version of the NCSN loss function adapted for the continuous case.

$$\mathcal{L}^*_{dsm} = \mathbb{E}_t\left[\lambda(t)\mathbb{E}_{p(x_0)}\mathbb{E}_{p_t(x_t|x_0)}\|s_\theta(x_t,t) - \nabla_{x_t}\log p_t(x_t \mid x_0)\|_2^2\right]$$

Here $\lambda(t)$ is a weighing function for $t \sim U(0, T)$. It is important to mention that with an affine drift coefficient f the log gradients of the data density can be easily calculated and score

matching can be used, otherwise one can fall back to sliced score matching [27]. For sampling, we can use any numerical SDE solver. One such solver is the Euler-Maruyama algorithm, which is described in Figure 4.

Algorithm 3 Euler-Maruyama sampling method

Input:
$\Delta t < 0$ – a negative step close to 0.
f – a function of x and t that computes the drift coefficient.
σ – a time-dependent function that computes the diffusion coefficient.
$\nabla_x \log p_t(x)$ – the (approximated) score function.
T – the final time step of the forward SDE.

Output:
x – the sampled image.

Computation:
1: $t = T$
2: **while** $t > 0$ **do**
3: $\quad \Delta x = \left[f(x,t) - \sigma(t)^2 \cdot \nabla_x \log p_t(x)\right] \cdot \Delta t + \sigma(t) \cdot \Delta\hat{\omega}$
4: $\quad x = x + \Delta x$
5: $\quad t = t + \Delta t$

Figure 4: The Euler-Maruyama numerical SDE solving algorithm [25]. Figure by Croitoru et al. [5]

Both NCSNs and DDPMs can be regarded as a discretization of an SDE that fits the framework defined before. First, the NCSNs transitions are equivalent to the following Markov chain, where $z \sim \mathcal{N}(0, \mathbf{I})$, $\mathbf{I}$ being the identity matrix, and $\sigma_t \in \{\sigma_i\}_{i=1}^N$.

$$x_t = x_{t-1} + \sqrt{\sigma_t - \sigma_{t-1}}\mathbf{z}_{t-1}$$

Which can be regarded as the discretization of the SDE

$$d\mathbf{x} = \frac{d[\sigma^2(t)]}{d\mathbf{t}} d\omega.$$

For DDPMs, as they evolve according to 2, the corresponding Markov chain is

$$x_t = \sqrt{1 - \beta_t} x_{t-1} + \beta_t \mathbf{z}_{t-1}.$$

And if we let $N \to \infty$, it converges to the following SDE

$$d\mathbf{x} = -\frac{1}{2}\beta(t)\mathbf{x}dt + \sqrt{\beta(t)}d\omega.$$

So far we have introduced three formulations of Diffusion Models and have demonstrated the connections between them. In the remaining part of the thesis we won't deal with NCSN and SDEs, but with the help of them we were able to better understand the nature of DDPMs and other DMs. Also, in the last couple of years, DDPMs have been researched extensively, while other formulations like NCSNs and SDEs have not. Future research on these two underexplored areas might help to overcome the weaknesses of DDPMs. This chapter is heavily built upon

Croituru et al. [5].

1.3 Brownian Bridge Diffusion Models

The usual approach for dealing with image-to-image translational tasks, which is the ultimate goal of this thesis, is to formulate them as a conditional image generation problem. In conditional image synthesis, we aim to generate data from the conditional densities of our dataset. With DDPMs, this is usually achieved by adding the information about the conditioning to the noise estimating model, which will be in the form of $\varepsilon_\theta(x, y, t)$. While some models could achieve relative success by formulating DMs this way, the models generally suffer from poor generalization and there's no theoretical guarantee that the model will truly estimate the desired conditional distributions [11]. The only tasks where these models could achieve success were ones where the conditioning and modelled domain were relatively similar.

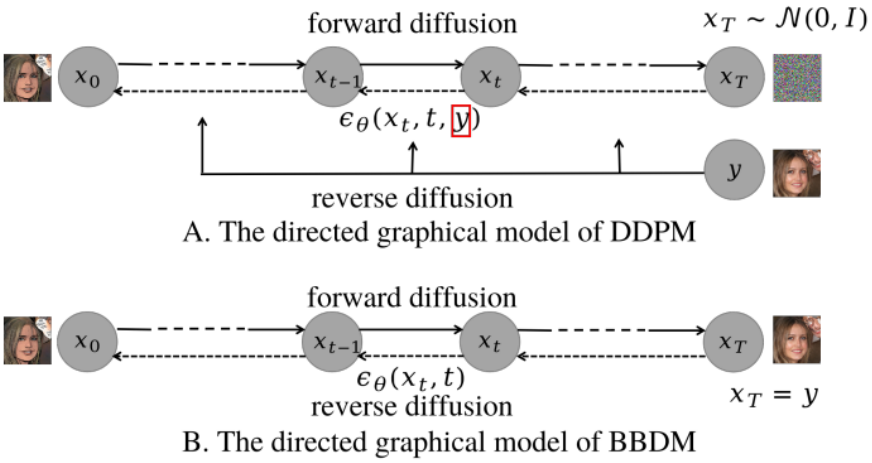

Figure 5: Comparison of BBDM and DDPM architectures by Li et al. [11]

In this section, we present Brownian Bridge Diffusion Models (BBDMs). They were first introduced by Li et al. [11] in 2023 for image-to-image translational tasks, and they aim to learn the mapping between two distinct image domains, A and B. Their name comes from the stochastic processes called Brownian Bridges, which are continuous-time stochastic processes where the transitional distribution during the diffusion process is conditioned on both the starting and ending states. The distribution of the middle steps of a Brownian Bridge process starting from point x_0 sampled from $q_{\text{data}}(x_0)$ at $t = 0$, and ending at point x_T at $t = T$, can be expressed as:

$$p(x_t|x_0, x_T) = \mathcal{N}\left((1 - \frac{t}{T})x_0 + \frac{t}{T}x_T, \frac{t(T-t)}{T}I\right) \tag{4}$$

This formulation illustrates that the process is anchored at both ends with x_0 and x_T, creating a bridge between them. The forward process of the BBDM is a Brownian Bridge, defined as,

$$q_{BB}\left(\boldsymbol{x}_t \mid \boldsymbol{x}_0, \boldsymbol{y}\right) = \mathcal{N}\left(\boldsymbol{x}_t; \left(1 - m_t\right)\boldsymbol{x}_0 + m_t\boldsymbol{y}, \delta_t\boldsymbol{I}\right)$$
$$\boldsymbol{x}_0 = \boldsymbol{x}, \quad m_t = \frac{t}{T},$$

where T is the number of steps, $m_t = \frac{t}{T}$, and $\delta_t = 2s(m_t - m_t^2)$. The schedule δ_t is designed such that the maximum variance doesn't make the model untrainable, the scaling parameter s also serves this purpose. Additionally, x_0 is sampled from distribution A, while y is sampled from distribution B. Applying the previous definition for x_t and x_{t-1}, one can obtain transitional probabilities for q_{BB}, where $\delta_{t|t-1} = \delta_t - \delta_{t-1}\frac{(1-m_t)^2}{(1-m_{t-1})^2}$.

$$\begin{aligned} q_{BB}\left(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}, \boldsymbol{y}\right) = \mathcal{N}\Big(\boldsymbol{x}_t; \tfrac{1-m_t}{1-m_{t-1}}\boldsymbol{x}_{t-1} \\ + \Big(m_t - \tfrac{1-m_t}{1-m_{t-1}}m_{t-1}\Big)\boldsymbol{y}, \delta_{t|t-1}\boldsymbol{I}\Big) \end{aligned}$$

In the case of conventional diffusion models, the reverse process initiates with pure noise drawn from a Gaussian distribution and progressively removes the noise, revealing the underlying clean data distribution. To accommodate the modelling of the conditional distributions, current approaches [18, 1] incorporate the condition as an extra input for the noise estimating neural network during the reverse diffusion process. Different from this approach, Brownian Bridge Diffusion models start directly from the conditional input and iteratively transform it to the desired data. Similar to DDPMs the reverse transitions are estimated in the form of

$$p_\theta(x_{t-1}|x_t, y) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)), \tag{5}$$

where $\mu_\theta(x_t, t)$ and $\Sigma_\theta(x_t, t)$ are both also estimated by some neural network and most importantly $x_T = y$ which is sampled from distribution B. After sampling from distribution B one can sample from distribution A by repeatedly applying the reverse transitions.

To train a BBDM, one must minimize the Evidence Lower Bound(ELBO), which bounds the negative log likelihood. For the BBDM transitions, the ELBO takes the following form, much similar to the DDPM loss function.

$$\begin{aligned} ELBO = &- \mathbb{E}_q(D_{KL}(q_{BB}\left(\boldsymbol{x}_T \mid \boldsymbol{x}_0, \boldsymbol{y}\right)||p\left(\boldsymbol{x}_T \mid \boldsymbol{y}\right)) \\ &+ \sum_{t=2}^{T} D_{KL}(q_{BB}\left(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0, \boldsymbol{y}\right)||p_\theta\left(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{y}\right)) \\ &- \log p_\theta\left(\boldsymbol{x}_0 \mid \boldsymbol{x}_1, \boldsymbol{y}\right)) \end{aligned}$$

By applying Bayes' theorem and the chain rule, one can calculate the $q_{BB}\left(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0, \boldsymbol{y}\right)$ transitions. Which are also Gaussian with some mean dependent on x_0, y and x_t.

$$\begin{aligned} q_{BB}\left(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0, \boldsymbol{y}\right) &= \frac{q_{BB}\left(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}, \boldsymbol{y}\right) q_{BB}\left(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_0, \boldsymbol{y}\right)}{q_{BB}\left(\boldsymbol{x}_t \mid \boldsymbol{x}_0, \boldsymbol{y}\right)} \\ &= \mathcal{N}\left(\boldsymbol{x}_{t-1}; \tilde{\boldsymbol{\mu}}_t\left(\boldsymbol{x}_t, \boldsymbol{x}_0, \boldsymbol{y}\right), \tilde{\delta}_t \boldsymbol{I}\right) \end{aligned}$$

By writing $\tilde{\boldsymbol{\mu}}_t\left(\boldsymbol{x}_t, \boldsymbol{y}, \boldsymbol{x_0}\right)$ in the following form, a possible parametrization of $\boldsymbol{\mu_\theta}\left(\boldsymbol{x}_t, \boldsymbol{y}, \boldsymbol{t}\right)$ becomes apparent.

$$\tilde{\boldsymbol{\mu}}_t\left(\boldsymbol{x}_t, \boldsymbol{y}, x_0\right) = c_{xt}\boldsymbol{x}_t + c_{yt}\boldsymbol{y} + c_{\epsilon t}\left(m_t\left(\boldsymbol{y} - \boldsymbol{x}_0\right) + \sqrt{\delta_t}\boldsymbol{\epsilon}\right)$$

Algorithm 1 Training

repeat
  paired data $\boldsymbol{x}_0 \sim q(\boldsymbol{x}_0)$, $\boldsymbol{y} \sim q(\boldsymbol{y})$
  timestep $t \sim Uniform(1, ..., T)$
  Gaussian noise $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  Forward diffusion $\boldsymbol{x}_t = (1 - m_t)\boldsymbol{x}_0 + m_t\boldsymbol{y} + \sqrt{\delta_t}\epsilon$
  Take gradient descent step on
    $\nabla_\theta ||m_t(\boldsymbol{y} - \boldsymbol{x}_0) + \sqrt{\delta_t}\epsilon - \epsilon_\theta(\boldsymbol{x}_t, t)||^2$
until converged

Algorithm 2 Sampling

sample conditional input $\boldsymbol{x}_T = \boldsymbol{y} \sim q(\boldsymbol{y})$
for $t = T, \ldots, 1$ **do**
  $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\boldsymbol{z} = \mathbf{0}$
  $\boldsymbol{x}_{t-1} = c_{xt}\boldsymbol{x}_t + c_{yt}\boldsymbol{y} - c_{\epsilon t}\epsilon_\theta(\boldsymbol{x}_t, t) + \sqrt{\delta_t}\boldsymbol{z}$
return $\boldsymbol{x}_0$

Figure 6: Summary of the BBDM sampling and training algorithm by Li et al. [11]

$$
\begin{aligned}
c_{xt} &= \frac{\delta_{t-1}}{\delta_t}\frac{1 - m_t}{1 - m_{t-1}} + \frac{\delta_{t|t-1}}{\delta_t}(1 - m_{t-1}) \\
c_{yt} &= m_{t-1} - m_t\frac{1 - m_t}{1 - m_{t-1}}\frac{\delta_{t-1}}{\delta_t} \\
c_{\epsilon t} &= (1 - m_{t-1})\frac{\delta_{t|t-1}}{\delta_t}
\end{aligned}
$$

Now our neural model only needs to estimate $\left(m_t(\boldsymbol{y} - \boldsymbol{x}_0) + \sqrt{\delta_t}\boldsymbol{\epsilon}\right)$ instead of the whole mean.

$$
\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, \boldsymbol{y}, t) = c_{xt}\boldsymbol{x}_t + c_{yt}\boldsymbol{y} + c_{\epsilon t}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t).
$$

Finally, by applying this parametrization to the loss function, it is reduced to the following form

$$
\mathbb{E}_{\boldsymbol{x}_0, \boldsymbol{y}, \boldsymbol{\epsilon}}\left[c_{\epsilon t}\left\|m_t(\boldsymbol{y} - \boldsymbol{x}_0) + \sqrt{\delta_t}\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)\right\|^2\right].
$$

To summarize, we describe the final training and sampling algorithm, which can also be seen in Figure 6 in a consice form. During training, one first samples a pair x_0 and y from both the A and the B distributions. Then with a random t which is uniformly sampled from a given set, we apply the forward process with some ε. Finally, we take a gradient step on the loss defined before. As for the sampling process, we first sample from the B distribution, let's say y. Then we apply our trained model to estimate $\left(m_t(\boldsymbol{y} - \boldsymbol{x}_0) + \sqrt{\delta_t}\boldsymbol{\epsilon}\right)$. With the estimate we can obtain x_{t-1}, we repeat this process until we reach x_0.

1.4 Denoising Diffusion Implicit Models

One drawback of DMs is that they require thousands of denoising steps during sampling, which drains a lot of computational resources and makes them much slower compared to GANs. Song et al. [24] propose a new sampling method for DDPMs, which is 10 to 50 times faster than the standard sampling procedure with minimal losses in sample quality, called Denoising Diffusion Implicit Models (DDIM). Since the DDPM loss function L_{simple} introduced by Ho

et al. [9], only depends on the marginals, this raises the question of whether one can find a family of inference distributions that has an equivalent loss function to DDPMs, but require less steps than them during sampling. Motivated by this, Song et al. [24] define a family of non-Markovian inference processes that have the same marginals and loss function as DDPMs, but the intermediate transitions differ from them. In addition to their theoretical results, their experiments have demonstrated that DDIMs achieve comparable results to DDPMs, but with higher efficiency. Their results presented in the paper [24] have proven to be so impactful, that DDIMs have become the standard sampling algorithm in most applications of DDPMs. In this section, we aim to present the theoretical background of DDIMs, as later in the thesis our models will incorporate them.

As stated previously, DDPMs aim to approximate the inverse of the inference process. Song et al. [24] studied the forward process to reduce the number of iterations demanded by the backwards model. Their primary observation is that the DDPM objective, solely relies on the marginal distributions of the variables at each step (denoted as $q(x_t|x_0)$), rather than directly on the joint distribution $q(x_{1:T}|x_0)$. Given that numerous inference distributions (joint distributions) share identical marginals, they explore alternative inference approaches that deviate from the Markovian framework. These non-Markovian inference methods yield an equivalent objective function to that of DDPMs, that we demonstrate in this section.

First, let's define the non-Markovian inference process introduced by Song et al.[24]. Let

$$q_\sigma\left(\boldsymbol{x}_{1:T} \mid \boldsymbol{x}_0\right) := q_\sigma\left(\boldsymbol{x}_T \mid \boldsymbol{x}_0\right) \prod_{t=2}^{T} q_\sigma\left(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0\right)$$

be the family of inference distribution considered, where $q_\sigma\left(\boldsymbol{x}_T \mid \boldsymbol{x}_0\right) = \mathcal{N}\left(\sqrt{\overline{\alpha_T}}\boldsymbol{x}_0, \left(1-\overline{\alpha_T}\right)\boldsymbol{I}\right)$, $\sigma \in \mathbb{R}^T_{\geq 0}$ and for all $t > 1$,

$$q_\sigma\left(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0\right) = \mathcal{N}\left(\sqrt{\overline{\alpha_{t-1}}}\boldsymbol{x}_0 + \sqrt{1-\overline{\alpha_{t-1}}-\sigma_t^2} \cdot \frac{\boldsymbol{x}_t - \sqrt{\overline{\alpha_t}}\boldsymbol{x}_0}{\sqrt{1-\overline{\alpha_t}}}, \sigma_t^2 \boldsymbol{I}\right). \tag{6}$$

It can be shown, that the $q_\sigma\left(\boldsymbol{x}_t \mid \boldsymbol{x}_0\right) = \mathcal{N}\left(x_t; \sqrt{\overline{\alpha_t}}\boldsymbol{x}_0, \left(1-\overline{\alpha_t}\right)\boldsymbol{I}\right)$ marginals match the DDPM marginals. The forward process can be derived from Bayes' rule:

$$q_\sigma\left(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}, \boldsymbol{x}_0\right) = \frac{q_\sigma\left(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0\right) q_\sigma\left(\boldsymbol{x}_t \mid \boldsymbol{x}_0\right)}{q_\sigma\left(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_0\right)},$$

As we can see the forward transitional densities now depend on x_0, so it is not Markovian. The σ parameter sets the variance of the forward distribution. Just like in the case of DDPMs, we can train a model $\varepsilon_\theta^{(t)}\left(\boldsymbol{x}_t\right)$ that estimates ε_t given x_t in the following equation,

$$x_t = \sqrt{\overline{\alpha_t}}x_0 + (1-\overline{\alpha_t})\varepsilon_t,$$

this relationship derives from the transitional probabilities 2. Therefore, using the trained model, we can get an estimate for x_0 from x_t based on the previous equation.

$$f_\theta^{(t)}\left(\boldsymbol{x}_t\right) := \left(\boldsymbol{x}_t - \sqrt{1-\alpha_t} \cdot \varepsilon_\theta^{(t)}\left(\boldsymbol{x}_t\right)\right) / \sqrt{\overline{\alpha_t}}. \tag{7}$$

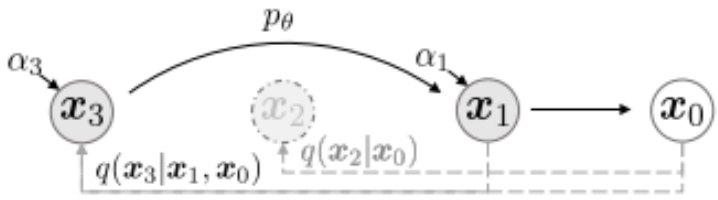

Figure 7: Graphical representation of the accelerated forward and backward process by Song et. al. [24]

Based on this, we can define an estimated forward process, where the initial distribution is $p_\theta(\boldsymbol{x}_T) = \mathcal{N}(\mathbf{0}, \boldsymbol{I})$.

$$p_\theta^{(t)}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t) = \begin{cases} \mathcal{N}\left(f_\theta^{(1)}(\boldsymbol{x}_1), \sigma_1^2 \boldsymbol{I}\right) & \text{if } t = 1 \\ q_\sigma\left(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, f_\theta^{(t)}(\boldsymbol{x}_t)\right) & \text{otherwise,} \end{cases} \tag{8}$$

Just like in the case of DDPMs, we want to minimize the Evidence Lower Bound. With similar calculations, we get the following loss function.

$$\begin{aligned} &J_\sigma(\epsilon_\theta) := \mathbb{E}_{\boldsymbol{x}_{0:T} \sim q_\sigma(\boldsymbol{x}_{0:T})}\left[\log q_\sigma(\boldsymbol{x}_{1:T} \mid \boldsymbol{x}_0) - \log p_\theta(\boldsymbol{x}_{0:T})\right] \\ =&\mathbb{E}_{\boldsymbol{x}_{0:T} \sim q_\sigma(\boldsymbol{x}_{0:T})}\left[\log q_\sigma(\boldsymbol{x}_T \mid \boldsymbol{x}_0) + \sum_{t=2}^{T} \log q_\sigma(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0) - \sum_{t=1}^{T} \log p_\theta^{(t)}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t) - \log p_\theta(\boldsymbol{x}_T)\right] \end{aligned}$$

A major result by Song et al. [24] is that this objective is equivalent (for all σ) with minimizing L_{simple}, the reweighted loss function, defined by Ho et al. [9]. Therefore, by training $\varepsilon_\theta(x_t)$ with L_{simple} we get an estimated forward process for multiple non-Markovian inference processes. From the definition of the reverse transitions 6 and the x_0 estimate 7, we get that x_t evolves according to the following formula during the sampling process.

$$\boldsymbol{x}_{t-1} = \sqrt{\overline{\alpha}_{t-1}} \underbrace{\left(\frac{\boldsymbol{x}_t - \sqrt{1-\overline{\alpha}_t}\epsilon_\theta^{(t)}(\boldsymbol{x}_t)}{\sqrt{\overline{\alpha}_t}}\right)}_{\text{"predicted } \boldsymbol{x}_0\text{"}} + \underbrace{\sqrt{1-\overline{\alpha}_{t-1}-\sigma_t^2} \cdot \epsilon_\theta^{(t)}(\boldsymbol{x}_t)}_{\text{"direction pointing to } \boldsymbol{x}_t\text{"}} + \underbrace{\sigma_t \epsilon_t}_{\text{random noise}} .$$

The special case when σ_t is set to 0 yields a deterministic process, where if we sample x_T from $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ despite the transitions being deterministic (except for $t = 1$), the marginals match the desired distributions, they call this process DDIM. Additionally, this deterministic process can be seen as a mapping between the data and latent spaces, which can be used to interpolate between data points.

A key observation of Song et al. [24] for speeding up the generative process is that L_{simple} only depends on the marginals, what happens during the transitions does not matter. Therefore, by fixing an increasing subset of steps $\tau \subset \{0, 1, \ldots, T\}$, and defining transitions between them like 2, one can obtain a non-Markovian forward process that matches the marginals on $\{0, 1, \ldots, T\}$ (a graphical representation is shown in Figure 7). Formally, the inference

distribution is the following

$$q_\sigma\left(\boldsymbol{x}_{1:T} \mid \boldsymbol{x}_0\right) := q_\sigma\left(\boldsymbol{x}_{\tau_S} \mid \boldsymbol{x}_0\right) \prod_{i=1}^{S} q_\sigma\left(\boldsymbol{x}_{\tau_{i-1}} \mid \boldsymbol{x}_{\tau_i}, \boldsymbol{x}_0\right) \prod_{t \in \overline{\tau}} q_\sigma\left(\boldsymbol{x}_t \mid \boldsymbol{x}_0\right)$$

The corresponding generative process only uses the steps of τ during sampling and the transitions between them defined in 8, therefore greatly reducing the computational demand of the inference procedure.

$$p_\theta\left(\boldsymbol{x}_{1:T}\right) := \underbrace{p_\theta\left(\boldsymbol{x}_T\right) \prod_{i=1}^{S} p_\theta\left(\boldsymbol{x}_{\tau_{i-1}} \mid \boldsymbol{x}_{\tau_i}\right)}_{\text{"used for sampling"}} \prod_{t \in \overline{\tau}} p_\theta\left(\boldsymbol{x}_0 \mid \boldsymbol{x}_t\right).$$

Once again, it can be shown that the ELBO objective is equivalent to L_{simple}. Therefore, by training a model on many T steps with L_{simple}, we obtain the generative process for the above non-Markovian process as well.

In conclusion, Song et al. show that by training a diffusion model on the L_{simple} loss, we can obtain a family of generative processes with the same marginals. With this family of processes, we are able to speed up sampling by 10 to a 100 times. For speeding up BBDM sampling, a similar procedure can also be defined [11], built upon similar ideas.

1.5 Latent Diffusion Models and Latent Brownian Bridge Models

A paper by Rombach et al. [18] presents the current state-of-the-art of Diffusion Models. One major limitation of DMs is their high computational demands. This reason prevented them from being applied to high-resolution images. Latent Diffusion Models [18] solved this problem by training DDPMs in a latent space obtained by a pre-trained autoencoder. This simple method significantly improved the training and sampling efficiency of denoising diffusion models without degrading their quality. Rombach et al. [18] used the DDPM framework, but Li et al. [11] have discovered that Brownian Bridge Diffusion Models also benefit from a latent autoencoder.

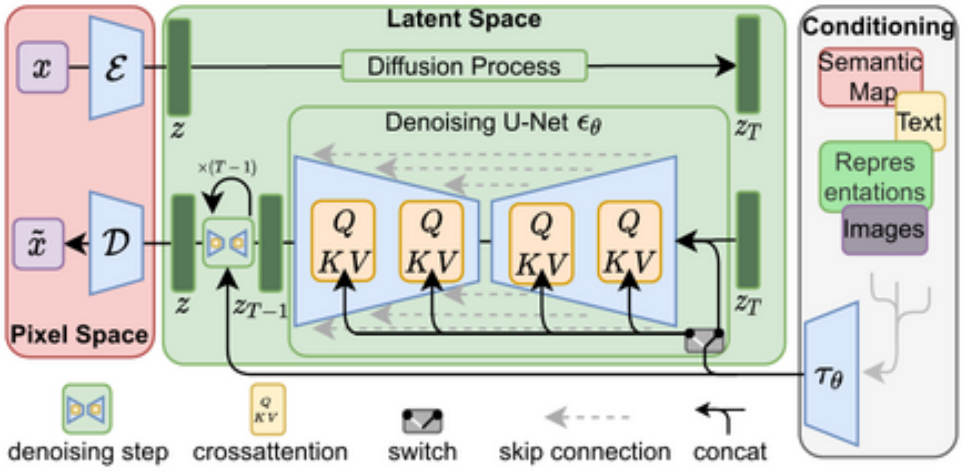

Figure 8: High level architecture of LDMs. Figure by Rombach et al. [18].

Latent Diffusion Models essentially partition the training algorithm into two phases. In the first phase, we train our autoencoders. Here, the high-frequency details are learned by the encoder and removed from the compressed latent space. Thus, during the next phase, the underlying Diffusion Model can focus on learning the semantic and conceptual composition of the data. This partitioning of the training process makes it computationally more efficient while keeping performance the same or potentially improving it.

The models, used for latent compression, are made up from an encoder $\mathcal{E}$ and a decoder $\mathcal{D}$ part. The encoder downsamples the input data while keeping the spatial structure of the image. Previous methods relied on a 1D ordering of the latent space and lost much of the spatial information encoded. The decoder takes an element of the latent space and maps it to the original distribution. Many different kinds of models would fit this general framework, but Rombach et al. [18] found VQGANs and a Kullback-Leiber regularized Variational Autoencoder trained in an adversarial manner to be the most successful. In fact, most works dealing with Latent Diffusion Models exclusively use VQGANs. During our experiments in the latter parts of the thesis, we will only deal with VQGANs, since they seem to be the standard latent model used. Therefore, in the remaining part of the section we will present the concept of Generative Adversarial Networks (GANs), Vector Quantization (VQ) and the VQGAN model used by Rombach et al. [18].

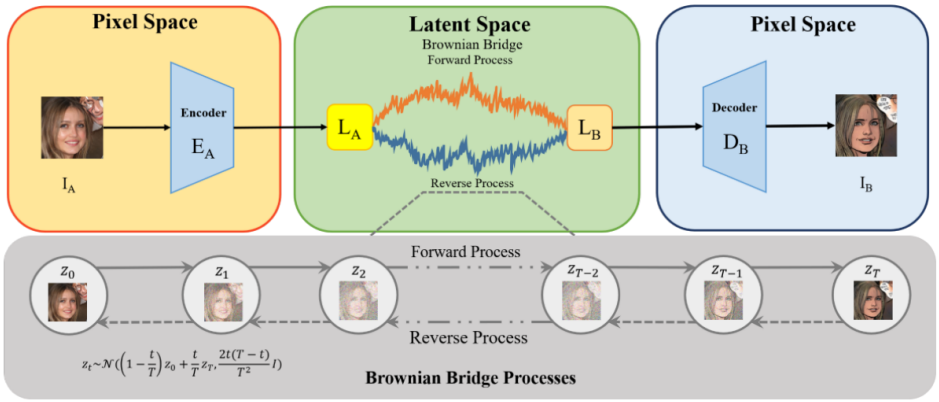

Figure 2. Architecture of BBDM.

Figure 9: High level architecture of BBDMs. Figure by Li et al. [11].

The general Generative Adversarial Networks (GANs) framework was introduced by Ian Goodfellow and his colleagues [8] in 2014. The core idea of GANs revolves around a minimax game between two neural networks: a generator G and a discriminator D. Let x denote real data samples and z denote random noise vectors sampled from a prior distribution $p(z)$. The generator G takes z as input and outputs synthetic samples $G(z)$, attempting to mimic the distribution of real data. The discriminator D receives both real samples x and generated samples $G(z)$, aiming to differentiate between the two by assigning high probabilities to real samples ($D(x) \approx 1$) and low probabilities to fake ones ($D(G(z)) \approx 0$). The training process is

formulated as a minimax optimization problem

$$\min_G \max_D \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z\sim p(z)}[\log(1 - D(G(z)))],$$

where $p_{\text{data}}(x)$ represents the true data distribution. This objective function encourages the generator to produce samples that are indistinguishable from real data while pushing the discriminator to become more accurate in distinguishing between real and fake samples. Of course, our version of GAN is a bit different. Since our aim is not to generate diverse samples but to learn a latent space that is perceptually equivalent to the original image space, the prior distribution $p(x)$ is the latent distribution obtained by passing the original images through the encoder $\mathcal{E}$. As suggested by the definition, GANs have been created for the same kind of tasks as Diffusion Models and were the state-of-the-art generative framework until Nichol et al. [13] could reach results that beat GANs. This result has introduced a paradigm shift to generative models, since GANs have many weaknesses, such as instability during training and mode collapse. Although DMs are the preferred generative model nowadays, GANs have not become obsolete, they are still used to generate highly realistic data samples across various domains, ranging from images to text and beyond, with applications in image synthesis, style transfer, and data augmentation, among others.

The architecture of VQ-GANs integrates Vector Quantization into the traditional GAN setup. In a typical GAN, a generator network creates images from random noise, while a discriminator network evaluates the authenticity of these images. In contrast, VQ-GANs introduce a discrete latent space by quantizing the continuous feature map produced by the encoder. This quantization process involves mapping the continuous vectors to discrete codes from a trained codebook of fixed size, resulting in a more structured representation of the data. The discrete representation facilitates more efficient training and generation processes, leading to faster convergence and improved sample quality. In the model used by Rombach et al., the VQ layer is a part of the decoder. To decode a latent image first, a convolutional layer is applied to it. Here, each pixel is represented by a vector. The VQ layer looks up the nearest neighbour of every such vector in the codebook and replaces it with accordingly, the rest of the architecture is a standard convolutional model with residual connections. The encoder part of the VQGAN is a simple convolutional model with residual connections.

2 Image Synthesis on the COVID-QU dataset

As a first step towards other, more complex computer vision tasks, we have to implement DDPMs for their original function, which is unconditional image synthesis. In this section, we are going to present some technical details about the implementation of DDPMs. As well as introduce the COVID-QU dataset, the dataset analysed in the remaining parts of the thesis. Additionally, a sketch of the model architecture used by us is presented, before the results of the first experiments in image synthesis are shared.

2.1 Implementing Diffusion Models

Throughout our experiments, we used a training pipeline written by the Computer Vision group. The pipeline allows us to easily define hyperparameters and automate the experimentation process. The first task was to implement DDPMs and BBDMs. Our approach was the following. Take the logic written for DDPMs by Rombach et al. [18] and for BBDMs by [11] and implement them into the framework using wrapper classes without changing the base code too much. Since Diffusion Models have a fundamentally different training algorithm compared to classical neural networks, employing wrapper classes was necessary and changing the pipeline couldn't always be avoided. During the making of this thesis, newer and newer functionalities were implemented, like LDMs, measuring the performance of the models, logging sample images, and many more. Our repository can be found here.

2.2 The COVID-QU Segmentation Dataset

In all the experiments, the COVID-QU [2] dataset was used. The dataset contains 33,920 chest X-rays of which 11,956 have COVID-19, 11,263 have non-COVID infections (Viral or Bacterial Pneumonia), and 10,701 are X-rays of normal lungs. Additionally, the whole dataset includes segmentation masks for lungs. The infection dataset is a smaller subset of the data, it contains 1,456 Normal and 1,457 non-COVID-19 chest X-rays with corresponding lung masks, plus 2,913 COVID-19 chest X-rays with corresponding lung mask from the COVID-QU-Ex [2] dataset and corresponding infection masks from the QaTaCov19 [6] dataset.

Since training and sampling diffusion models is a memory-heavy process, all the training and sampling in the experiments were done on images of size $64 \times 64 \times 1$.

2.3 Training DDPMs and results

The first experiments where we tested the image synthesis capabilities were not carried out in the experimentation pipeline. With default settings, we tested and measured the generative capabilities of LDMs and Guided Diffusion by Baranchuck et al. [3]. With these experiments, we got a good idea about the capabilities of such models. Motivated by the success of these experiments, we tested the implemented non-latent DDPMs with the pipeline on the COVID-QU dataset. Some generated images can be seen in Figure 10. Although in image synthesis

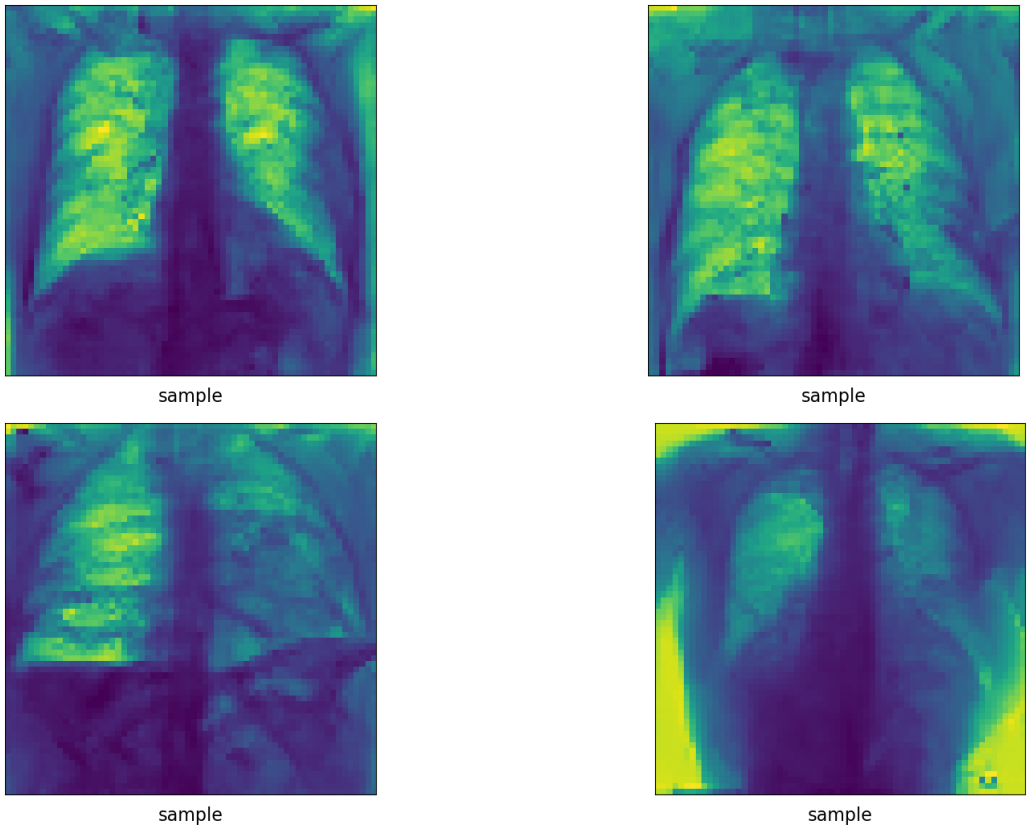

Figure 10: Synthetic Images of Chest X-Rays Generated from a Diffusion Model

measuring the synthesized images' diversity and fidelity is important, it goes beyond the scope of this thesis, therefore they were not measured. Despite this, we obtained valuable experience in training practices and hyperparameter settings through these experiments. Some of these parameters are shared in the next subsection, where we introduce some details about the model architecture used in most of our experiments.

2.4 Model Architecture

During training, the model used 1000 diffusion steps. A linear noising schedule was employed with $\beta_0 = 0.0001$ and $\beta_{1000} = 0.02$, this controls how much noise we add at each diffusion step. These hyperparameters have remained fixed throughout our experiments. During sampling, we used the DDIM [24] approach with 50 steps.

The model architecture described below is based on Rombach et al. [18] with the exception that we didn't use a latent encoder or decoder and that the input and output channels of the U-Net were fit to our dataset. The input of the noise-predicting U-Net model is an image of size 64×64 with one channel and the timestep embedding. The output is a denoised image of size 64×64 with one channel. First, the input image is transformed with a convolutional layer to an image of the same spatial size, but with 192 channels. After the first step, the input passes through the three segments of the U-Net [20], an encoder, a middle, and a decoder part. To define the architecture, one must first describe the two other major building blocks of our U-Net: Residual Blocks and Linear Attention.

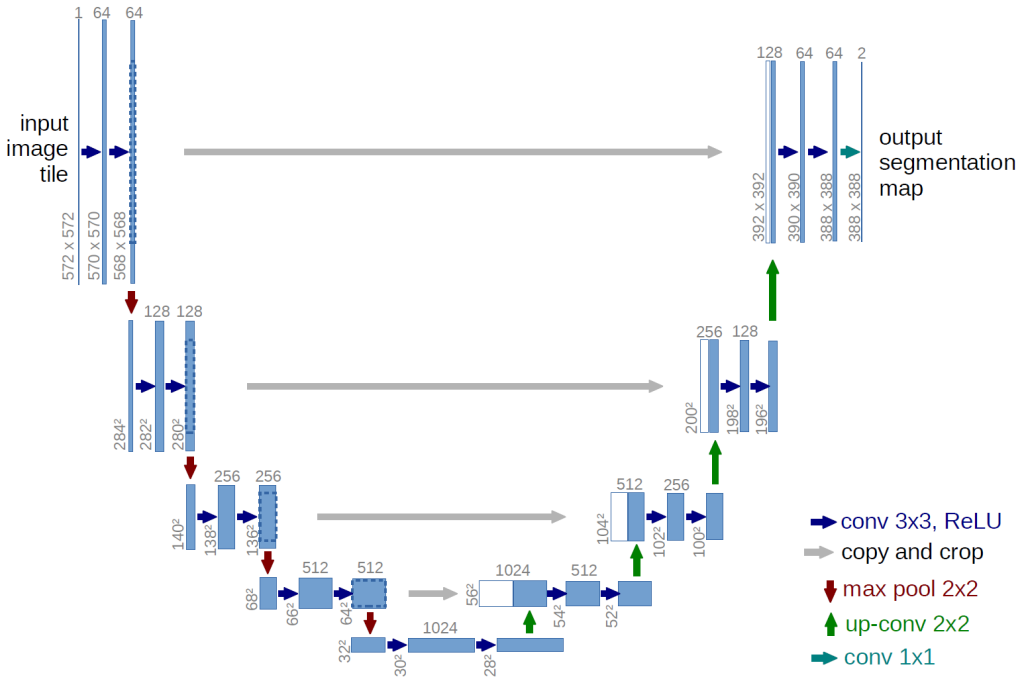

Figure 11: U-Net Architecture, Figure by Ronneberger et al. [20]

A **Residual Block** has two inputs, an image, and the timestep embedding. Given a timestep t, first, a sinusoidal positional embedding is calculated, it is a vector of size 192, and then it is transformed with linear layers and SiLU activation to a vector of size $4 \cdot 192$. Then, the input image is normalized using Group Normalization, and it is passed through a SiLU activation. If the Residual Block is used for down or upsampling, average pooling or nearest interpolation is applied to the output of the previous step. The output is then passed through a convolutional layer. Then the timestep embedding is passed through SiLU and a linear layer and added together with the output of the previous step, the output is then passed through a Group Normalization layer, a SiLU activation, and a convolutional layer. The output of the Residual Block is the sum of the previous step and the original input that was down or upsampled so the shapes fit.

A **Linear Attention Layer** first flattens the spatial dimensions of the input. After normalizing the input, Q, V, K matrices are calculated with a one-dimensional convolutional layer. Then the rows of the matrices are cut up to equal pieces of size 64. After this the following formula is calculated: $softmax(\frac{QK^T}{\sqrt{D}})V$, where Q, V, K are now the submatrices and D is a scaling parameter, the outputs are then concatenated and transformed back to the shape of the input. Finally, the output is given by the sum of the input and the output of the previous step. With this approach, pairwise correlations of different image regions can be modelled.

Now we can define the **Encoder**. The encoder has multiple levels, and each level has two Residual Blocks and a downsampling Residual Block. The first Residual Block grows the image channels, the second keeps the image dimensions the same, and the third RB downsamples the image spatial dimensions by a factor of two. The number of channels on each level is controlled by the *channel_mult* hyperparameter. In our case, the first level has 192, then 2×192, 3×192, and finally 4×192. Additionally, Attention layers are applied at levels of spatial resolution:

$32 \times 32, 16 \times 16, 8 \times 8$, after each non-downsampling Residual Block.

The **Middle** part of the U-Net is made up of two Residual Blocks and a Linear Attention Layer in the middle. These Residual Blocks keep the channel size constant.

The structure of the **Decoder** part matches the encoder, but instead of downsampling on each level, the Residual Block upsamples by a factor of two, and the input of each level is the concatenation of the output of the level of the Encoder with the same spatial resolution and the output of the previous Decoder level. The final output is then obtained by passing the output of the Decoder through a Group Normalization Layer, a SiLU activation, and a convolutional layer which transforms the image back to its original shape.

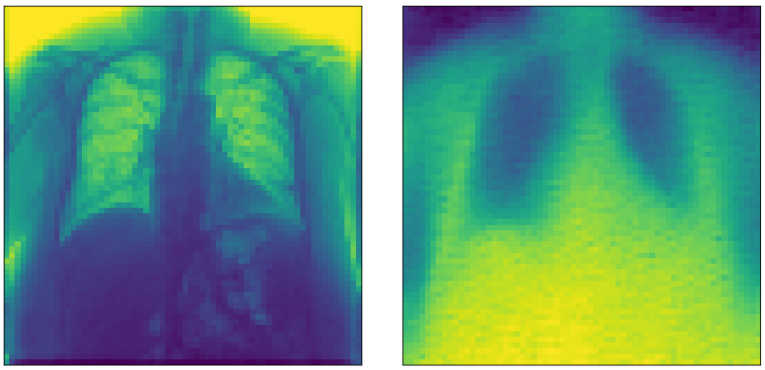

Figure 12: Example of an image and the image after adding noise and passing it through the U-Net. As you can imagine, one U-Net pass has removed some noise. During vanilla sampling, 1000 passes would be completed.

3 Conditional Image Synthesis using DDPMs

In conditional image synthesis, we aim to generate data from the conditional densities of our dataset. With diffusion models, this is usually achieved by adding the information about the conditioning to the noise estimating model, which will be in the form of $\varepsilon_\theta(x, y, t)$. One major architectural question is how we should add information about the conditioning. A usual approach is to simply concatenate, add it to the input of the U-Net, or use cross-attention to add it to intermediate levels of the U-Net. In the latter part of this section, we are going to present the results of our experiments which have compared the effectiveness of the first two methods. This area of Diffusion Model research seems underexplored thus, future experimentation might be useful.

3.1 Diffusion Models in Image Segmentation

Image segmentation is a crucial task in computer vision that involves dividing an image into multiple segments or regions, each representing a different object or part of an object. From a technical perspective, it involves assigning a label to each pixel, such that pixels with the same label belong to the same kind of object. The goal is to simplify or change the representation of an image into something more meaningful and easier to analyse. This technique is widely used in various applications such as medical imaging, autonomous driving, and object detection, where precise delineation of objects is essential. Advanced segmentation methods often employ deep learning algorithms, but classical computer vision approaches like edge detection can be used as well. State-of-the-art methods include U-Net [20], or DeepLab [4] based models, to achieve high accuracy and efficiency, enabling detailed analysis and interpretation of complex visual data.

Diffusion models can also be used for such tasks. Amit et al. [1] tackle image segmentation as a conditional image synthesis task. Here they perform the diffusion steps on the segmentation masks and condition the noise estimating U-Net on the original image. With this method they have been able to achieve state-of-the-art results on the Cityscapes validation set, the Vaihingen building segmentation benchmark, and the MoNuSeg dataset. Baranchuk et al. [3] show that the intermediate activations of the denoising U-Net capture semantic information well. They use a classifier on the upsampled activations to obtain a segmentation mask. Pinaya et al. [16] detect and segment anomalies on MRI data. First, they train a Diffusion Model on a healthy dataset. Using an anomalous image as an input will result in large loss values at anomalous regions. With an appropriate threshold, they can create a segmentation mask.

Our approach resembles Amit et al. [1]. We aim to generate segmentation masks conditioned on the original image. One of the main questions of this approach is how we should supply the original image to the denoising U-Net. In the last part of this section, we are going to present results that compare different ways of conditioning to state-of-the-art results on the COVID-QU segmentation dataset.

3.2 Model Architecture

As mentioned previously, we tested two methods for adding the conditioning to the denoising U-Net. These two are adding conditioning by concatenation, and adding the conditioning to the image input of the U-Net. In both cases the conditioning, in this case the original image, was transformed by a Residual in Residual Dense Block, used by Amit et al. [1].

A **Residual in Residual Dense Block**(RRDB) includes three Residual Dense Blocks (RDB). A Residual Dense Block has 5 convolutional layers, after which each layer follows a Leaky ReLU activation. Each of the 5 convolutional layers has the output of the previous layers as input, finally, the input of the RDB and output of the last layer is added. The output of the RRDB is obtained by passing the input through all the RDBs and combining it with the input. In our case, the RRDB transforms the input from $64 \times 64 \times 1$ to $64 \times 64 \times 16$. According to Amit et al. [1] increasing the number of RRDB blocks doesn't affect the mIoU much, however, not using a condition embedder yields significantly worse results.

The rest of the U-Net architecture used in our segmentation experiments is the same as in Section 2.4 with the only difference that it has 16 or 17 input channels, depending on whether we add the conditioning with addition or concatenation.

3.3 Segmentation Quality Metrics

One of the main disadvantages of diffusion models is their inference speed. Thus, evaluating the performance of diffusion models becomes a slow task. For this reason, we calculate the following metrics on a 10% validation split in each epoch, the random seed used for sampling was the same across all measurements, so it makes sense to compare them. Amit et al. [1] calculated the segmentation mask by inferring multiple times with the same conditioning and averaging the mask. Because of the nondeterministic nature of their sampling method [24], this approach yields more stable and accurate results, however because of the high inference cost we decided to only evaluate each image one time. So we opted to use the DDIM sampling method, which provides a deterministic mapping from the latent space to the mask space for LDMs, increasing stability, and a deterministic mapping between image space and mask space for BBDMs, eliminating any variance.

Image segmentation can be thought of as the classification of each pixel into various categories. In this project, we only concerned ourselves with the case of binary classification, we used a confidence threshold of 0.5 on all of our experiments. We noticed that the models' outputs are concentrated on 0 and 1, therefore exploring different confidence levels, was not necessary. Throughout all the experiments, the following metrics are calculated from the confusion matrix: DICE-index, Jaccard-index, Accuracy, Sensitivity, Specificity, Balanced Accuracy, Precision, and Matthew's correlation coefficient. The metrics are defined as follows.

- DICE-index or F-score: $\frac{2TP}{2TP+FN+FP}$, where TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives, FN is the number of true

negatives. This can be interpreted as two times the intersection of the predicted positives and real positives over the sum of the sizes of these two sets.

- Jaccard-index: $\frac{TP}{TP+FN+FP}$. This can be interpreted as the area of intersection over the union of predicted positives and real positives.
- Accuracy: $\frac{TP+TN}{TP+FN+FP+TN}$ Measures the ratio of correct classifications and the number of total classifications.
- Sensitivity is the true positive rate. The proportion of actual positive values that have been correctly classified.
- Specificity is the true negative rate. The proportion of actual negative values that have been correctly classified.
- Balanced Accuracy is the mean of sensitivity and specificity.
- Precision measures what proportion of the total positive classifications are true positives.
- MCC(Matthew's correlation coefficient) is the Pearson correlation coefficient for the two binary variables, the predictions and the actual values.

3.4 Lung Segmentation on COVID-QU

The previous best result on the lung segmentation COVID-QU dataset achieved by the Computer Vision Group is a 0.99 DICE index with multiple variations of U-Net models.

Our diffusion models were trained for 100 epochs, around 4000 gradient steps, with a learning rate of 0.003 and a batch size of 512. These hyperparameters were carefully optimized since the training process would become unstable otherwise. Table 1 below summarizes our results. The results were not vastly different between the two models, although Amit et al. [1] found that addition should produce better results in most cases. One reason for not noticing a significant difference between the two methods could be, that the complexity of the task is not high enough. Our models also produce comparable results with the U-Net models.

3.5 Infection Segmentation on COVID-QU

The previous best results on the infection segmentation COVID-QU dataset achieved by the Computer Vision Group are 0.86 with a U-Net model and 0.85 by the Res50AttUNet model.

Our diffusion models were trained for 2400 epochs and 8000 gradient steps, with a learning rate of 0.003 and a batch size of 512. Here the produced results are like expected, supplying the conditioning with addition has reached much better results than supplying it with concatenation, see Table 2. The infection segmentation problem is a much more difficult task, therefore differences in model performance are more noticeable. Also, the variance of the metrics was high between batches. Although measuring the metrics is a time-consuming task, in the future I

	Lung	
	Addition	Concatenation
DICE index	0.9529	0.9541
Jaccard index	0.9101	0.9123
Accuracy	0.9792	0.9793
Sensitivity	0.9455	0.9502
Specificity	0.9888	0.9878
Balanced Accuracy	0.9672	0.9793
Precision	0.9604	0.9581
MCC	0.9396	0.9408

Table 1: Metrics measured on the COVID-QU lung segmentation validation dataset at the final epoch using DDPMs.

should let some experiments run for a longer time so we can get a clearer picture of the model's performance and reach a level where it can rival the U-Nets. The reason for the models' subpar performance compared to some U-Nets could be that the original purpose of diffusion models is image synthesis, while we can get some acceptable results in image segmentation further research is required to reach state-of-the-art results on multiple datasets.

	Infection	
	Addition	Concatenation
DICE index	0.6755	0.4708
Jaccard index	0.5105	0.3087
Accuracy	0.9183	0.8967
Sensitivity	0.6687	0.3646
Specificity	0.9550	0.9736
Balanced Accuracy	0.8119	0.6691
Precision	0.6838	0.6674
MCC	0.6293	0.4431

Table 2: Metrics measured on the COVID-QU infection segmentation validation dataset at the final epoch using DDPMs.

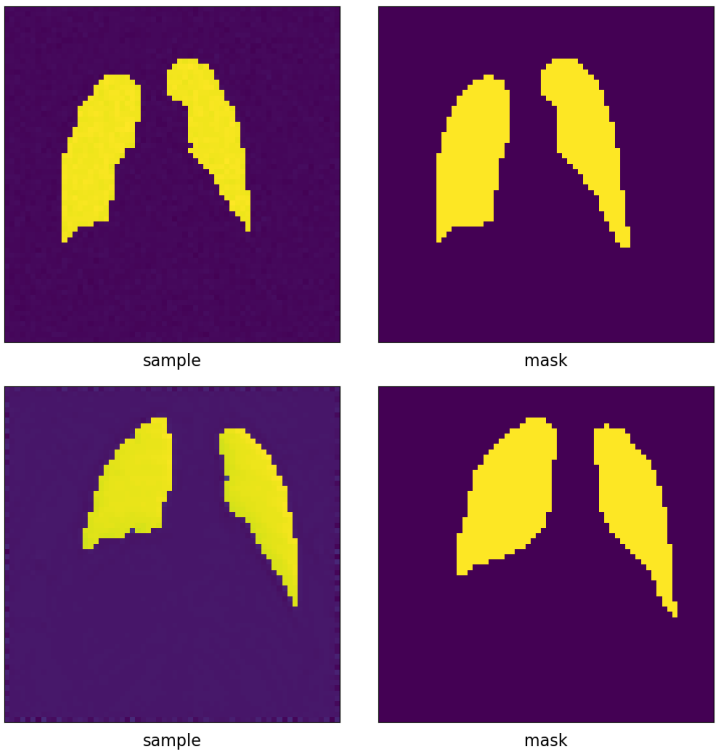

Figure 13: Example of generated lung segmentation mask (left) and the original mask (right), the conditioning was supplemented via concatenation (top) and addition (bottom)

4 Image Segmentation using BBDMs

In earlier sections, Image Synthesis was approached as a conditional image synthesis task, where given an input image (an X-ray), the goal was to model the conditional densities, namely the segmentation masks. However, as discussed in section 1.3, this approach has several limitations. To address these shortcomings, we introduced BBDMs. The primary objective of BBDMs is to model the mapping between two distributions using Brownian Bridges. While BBDMs have been successfully applied to various image-to-image translational tasks in the past, such as semantic synthesis, style transfer, and sketch-to-photo conversion, to our knowledge, they have not been utilized for semantic segmentation tasks.

In this section, we leverage BBDMs to segment images from the COVID-QU dataset, focusing on both lung segmentation and infection segmentation. Similar to DDPMs, implementing BBDMs required extensive hyperparameter optimization and data preprocessing. Without careful tuning, BBDMs failed to produce satisfactory results. Contrary to DDPMs, we found that reducing the learning rate and batch size contributed to improved performance with BBDMs. In contrast, DDPMs typically require a larger batch size to stabilize the learning process. Despite these adjustments, BBDMs still yielded subpar results, prompting the need for more significant changes.

Recognizing that normalizing X-ray images is a standard practice in medical image processing, we explored the use of the Contrast Limited Adaptive Histogram Equalization (CLAHE) algorithm as a preprocessing step. This adjustment was aimed at enhancing the quality and contrast of the X-ray images, potentially improving the performance of BBDMs in segmenting

lung and infection regions accurately. CLAHE (Contrast Limited Adaptive Histogram Equalization) normalization is an image processing technique used to improve the contrast in images. It is a variant of Adaptive Histogram Equalization (AHE), which adjusts the contrast of an image by redistributing the lightness values based on the local neighbourhood of each pixel. CLAHE enhances this method by limiting the contrast amplification to avoid noise amplification, which can be an issue in AHE. By using CLAHE on the X-ray images, the BBDM was able to learn the segmentation masks and produce acceptable results. Also, CLAHE has proven to be so beneficial, that in all the remaining experiments it was employed.

	BBDM	
	Lung	Infection
DICE index	0.8411	0.5152
Jaccard index	0.7268	0.3577
Accuracy	0.9281	0.8851
Sensitivity	0.8459	0.5359
Specificity	0.9522	0.9333
Balanced Accuracy	0.8990	0.7346
Precision	0.8397	0.5341
MCC	0.7959	0.4625

Table 3: Metrics measured on the COVID-QU infection segmentation validation dataset at the final epoch using Brownian Bridge Diffusion Models.

4.1 Lung Segmentation

For segmentation tasks utilizing BBDMs, the noise-estimating U-Net was designed with the same architecture as employed in DDPMs. The U-Net architecture is widely recognized as a versatile model in the field of computer vision, and it has been meticulously optimized for diffusion models by previous researchers [9, 18]. Given this optimization, we determined that altering the architecture was unnecessary. Specifically for lung segmentation, we trained our models for 200 epochs with a batch size of 8. During training, we observed that the loss function often plateaued, prompting us to implement a linear learning rate scheduler. This scheduler began with a learning rate of 10^{-5} and gradually reduced it to 10^{-7} by the end of the training. We utilized 1000 diffusion steps during the training process. In the context of BBDMs, a sampling procedure analogous to the DDIM framework can be defined. For our experiments, we employed this sampling algorithm using 200 steps.

The outcomes of our experiments are summarized in Table 3. When comparing the performance metrics of BBDMs to those of DDPMs, we found that BBDMs consistently underperformed across all measured metrics. Figure 15 illustrates a comparison between a generated segmentation mask and the corresponding original mask.

4.2 Infection Segmentation

For the task of segmenting infections, we largely employed the same hyperparameters and model architecture as described previously. However, there were a couple of key differences in our approach. First, we extended the training duration significantly, training the models for 1500 epochs instead of the earlier configuration. Second, we utilized a different learning rate scheduler for this task. The new scheduler was designed to automatically reduce the learning rate whenever the optimizer detected that the training had reached a plateau, thereby attempting to enhance model performance through finer adjustments in learning rates.

Despite these adjustments, the performance of BBDMs remained inferior to that of DDPMs across all evaluated metrics. Despite these results, it might be worth exploring latent training. Since the U-Net has to estimate more aggressively in the case of BBDMs, latent training in an appropriate latent space might be beneficial.

5 Latent Models

As previously discussed, training within an appropriate latent space can significantly simplify the task of the noise-estimating U-Net. According to Rombach et al. [18], latent encoders such as VQ-GANs effectively capture low-frequency signals from the data, thereby allowing the U-Net to focus on modelling high-frequency details. This insight opens up intriguing possibilities for leveraging this characteristic in both Denoising Diffusion Probabilistic Models (DDPMs) and Brownian Bridge Diffusion Models (BBDMs).

In this section, we delve into the specifics of the training procedures and the implementation of the latent models used, specifically VQ-GANs. Additionally, we present a comparative analysis of Latent Diffusion Models (LDMs) and Latent Brownian Bridge Diffusion Models (LBBDMs), experimenting with various downsampling rates. In this case, the downsampling rate means the ratio between the input spatial dimensions and the latent spatial dimension. This comparison is conducted using the COVID-QU dataset to highlight the potential advantages and differences between the two frameworks.

5.1 Training VQ-GANs

Training LDMs and BBDMs requires the creation of several latent models, specifically VQ-GANs in our case. Both the conditioning input (X-ray images) and the segmentation masks must be mapped into a latent space with the same dimensionality. To achieve this, we train separate models for each data type. Thus, for a fixed downsampling rate, we end up training three distinct models: one for X-ray images, one for lung segmentation masks, and one for infection segmentation masks.

Given that we work with images of size 64×64, we confined our training to downsampling rates of 4 and 8, while growing channel size from 1 to 3 and 4. We hypothesized that further compression would reduce spatial dimensions excessively, compromising the functionality of the diffusion models. Additionally, higher downsampling rates posed significant challenges in training the VQ-GANs.

Integrating GAN training algorithms, which differ fundamentally from traditional deep learning approaches, was the first challenge we addressed. We tackled this by defining wrapper classes to accommodate the unique characteristics of GANs. We then adopted the VQ-GAN implementation from Rombach et al. [18], making slight modifications to integrate it into our pipeline.

For training the VQ-GANs, we utilized two architectures, one for each downsampling level. Both architectures were previously used by Rombach et al. [18] with LDMs. During training, we had to adjust certain hyperparameters related to the loss function and the discriminator models. Achieving equilibrium between the discriminator loss and the reconstruction loss proved challenging, as the training process was volatile and convergence was not guaranteed. Each model required multiple training attempts with different random seeds to achieve convergence. After numerous runs, we successfully obtained models with discretized latent spaces

and satisfactory reconstruction capabilities. Examples of these reconstructions are illustrated in Figure 16.

5.2 Comparison of Downsampling rates

To assess the effects of downsampling on the performance of our segmentation models, we trained LDMs and LBBDMs using the pretrained autoencoders obtained in the previous section. Specifically, we utilized VQ-GANs with 4x and 8x downsampling rates.

For LDMs, the U-Net architecture remained largely unchanged from its original design in DDPMs. However, we had to adjust the model to accommodate the different dimensionalities of the latent spaces, including modifying some parameters of the diffusion process accordingly. Detailed hyperparameters for the trained LDMs, LBBDMs, and VQ-GANs are provided in Appendix 5.3. The results of these experiments are summarized in Table 4. By comparing the outcomes across various downsampling rates with those for no downsampling, as shown in Table 2 and Table 1, it becomes evident that Latent Diffusion Models (LDMs) do not show performance improvements with downsampling. However, downsampling significantly reduces both training and inference times. Additionally, since LDMs require sampling from the standard normal distribution during inference, this introduces some variance in the segmentation performance metrics. To obtain more interpretable results, multiple trials for each experiment and exploration of various downsampling rates would be necessary. Unfortunately, due to the high costs associated with inference, we were unable to conduct these additional trials.

For LBBDMs, the architecture and training parameters were consistent with those used for BBDMs, except for necessary adjustments to match the latent space dimensionality. The outcomes of our experiments with LBBDMs are shown in Table 5. Comparing Table 3 and Table 5, it is evident that 4x downsampling produced the most accurate results for lung segmentation, with 8x downsampling and the vanilla model following. For infection segmentation, the latent models performed comparably, surpassing the vanilla model. This indicates that BBDMs benefit from using latent encoders. However, more aggressive downsampling does not necessarily enhance performance. Further experimentation with larger image sizes and varying downsampling rates is required to explore this concept in greater depth.

5.3 Comparison of LDMs and Latent BBDMs

In this subsection, we compare the performance of the BBDM and DDPM frameworks across various downsampling rates. Figure 17 illustrates this comparison. In the lung segmentation task, both models exhibited similar scaling across the downsampling rates, with LBBDM showing slight improvements at 4x downsampling. However, in the infection segmentation task, the differences are more pronounced. The LDM framework experiences a significant reduction in performance when latent models are applied, whereas LBBDMs improve with the application of latent encoders and show minimal performance degradation even at 8x downsampling. This suggests that BBDMs can utilize the encoded information from latent models more efficiently.

	Lung		Infection	
	LDM-f4	LDM-f8	LDM-f4	LDM-f8
DICE index	0.9533	0.8992	0.3299	0.4696
Jaccard index	0.9115	0.8206	0.2008	0.3133
Accuracy	0.9792	0.9535	0.7357	0.8649
Sensitivity	0.9511	0.9218	0.5039	0.4880
Specificity	0.9873	0.9630	0.6710	0.9260
Balanced Accuracy	0.9424	0.9301	0.6383	0.7070
Precision	0.9558	0.8798	0.2549	0.5003
MCC	0.9400	0.8702	0.2101	0.4087

Table 4: Measured metrics on the COVID-QU segmentation validation dataset from the last epoch.

	Lung		Infection	
	LBBDM-f4	LBBDM-f8	LBBDM-f4	LBBDM-f8
DICE index	0.8921	0.8565	0.5775	0.5768
Jaccard index	0.9115	0.7494	0.4068	0.4074
Accuracy	0.9512	0.9352	0.8963	0.9015
Sensitivity	0.8916	0.8568	0.5311	0.5087
Specificity	0.9686	0.9580	0.7405	0.9613
Balanced Accuracy	0.9301	0.9074	0.7419	0.7350
Precision	0.8927	0.8565	0.6358	0.6759
MCC	0.9400	0.8147	0.5225	0.5313

Table 5: Measured metrics on the COVID-QU segmentation validation dataset from the last epoch.

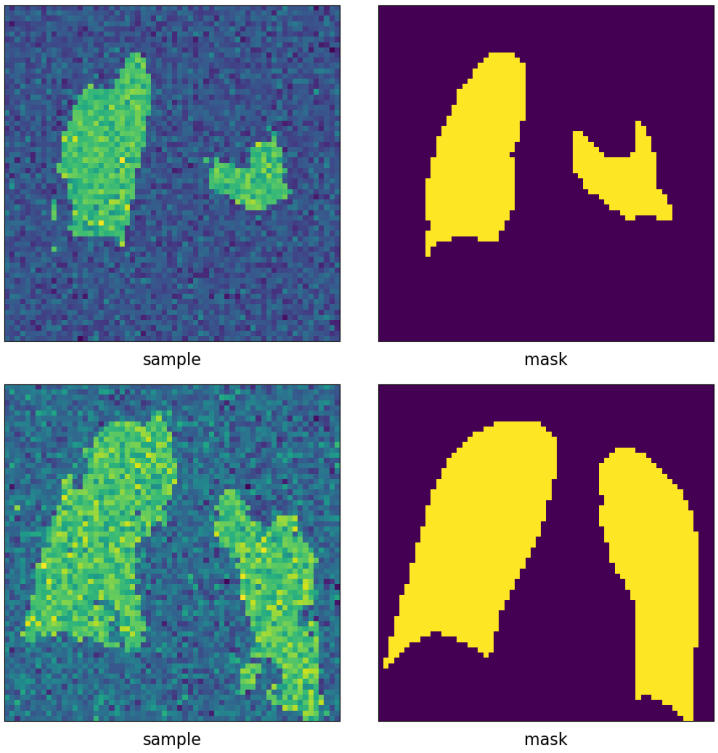

Figure 14: Example of generated infection segmentation mask (left) and the original mask (right), the conditioning was supplemented via addition (top) and concatenation (bottom)

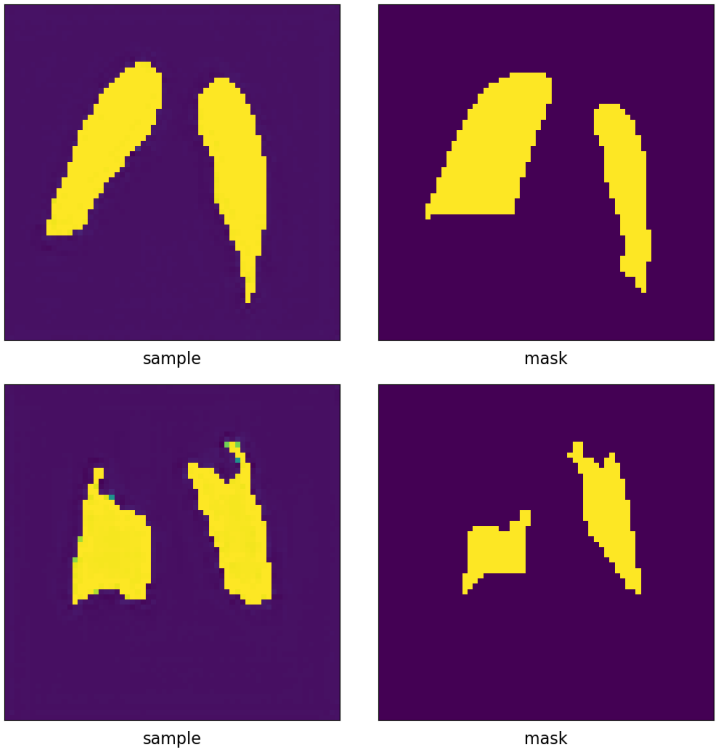

Figure 15: Example of generated segmentation mask (left) and the original mask (right) for lung segmentation (top) and infection segmentation (bottom) via BBDM model

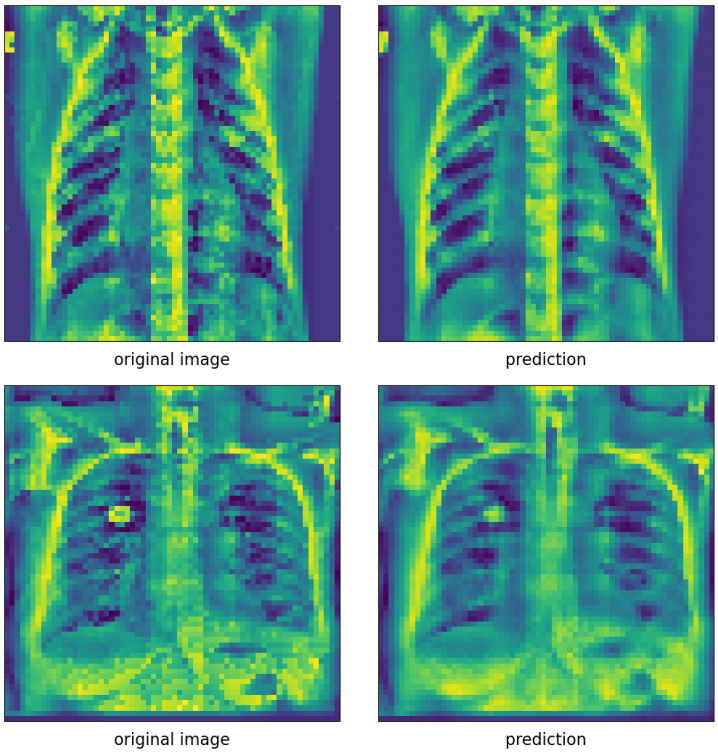

Figure 16: Examples of CLAHE normalized VQ-GAN reconstructions

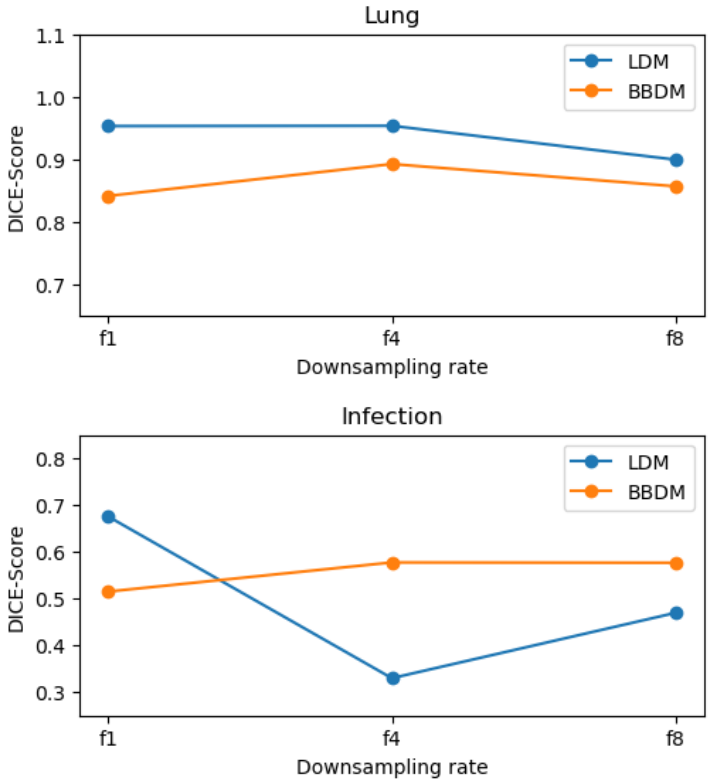

Figure 17: Comparison of LDMs and LBBDMs across multiple downsampling rates.

Conclusion

In this thesis, we have explored the theoretical background, applications, and effectiveness of diffusion models, particularly focusing on image segmentation tasks. Through comprehensive experimentation, we have demonstrated the versatility and potential of these models, specifically Denoising Diffusion Probabilistic Models (DDPMs) and Brownian Bridge Diffusion Models (BBDMs), in handling complex computer vision problems such as lung and infection segmentation in medical images.

The application of DDPMs and BBDMs to the COVID-QU dataset demonstrated their effectiveness in segmenting lung and infection regions with satisfactory accuracy. Although DDPMs generally achieved better performance, this does not diminish the value of BBDMs for image segmentation. BBDMs simplify implementation and optimization by eliminating the need for complex conditioning mechanisms. Furthermore, BBDMs are a relatively new concept compared to LDMs and have not been extensively researched. Investigating their potential is essential for advancing image translation tasks.

Latent Diffusion Models (LDMs) and Latent Brownian Bridge Diffusion Models (LBBDMs) were assessed across various downsampling rates. The results indicate that while downsampling significantly reduces training and inference times, LDMs struggled to efficiently use the information encoded by the latent models and showed decreased performance. At the same time, LBBDMs even showed mild improvements compared to no downsampling, particularly at 4x downsampling. This indicates LBBDMs' superior performance in processing latent representations. Also, the results imply that for some image translation tasks, where inference time is crucial, using LBBDMs is more efficient.

Future Directions

The field of diffusion models has shown remarkable advancements, not only in image synthesis but other various and complex computer vision tasks. However, there remain numerous promising avenues for future research that can enhance the capabilities and broaden the impact of diffusion models. The following directions are proposed that build upon the results of this thesis and other current advancements.

During our experiments, a significant limiting factor was the substantial resources and time required for training diffusion models. On average, a single training run took about 80 hours. Consequently, only a limited number of experiments could be conducted for each instance. Future research should aim to repeat these experiments multiple times to better understand the variance in the results. Additionally, due to technical constraints, we were restricted to operating on images of size 64×64 pixels. Increasing the spatial resolution of the images would enable us to explore a broader range of downsampling rates and derive more meaningful conclusions. Addressing these limitations should be a key focus in the continuation of this thesis.

Another limitation of our experiments was the constraint of using only one dataset and focusing on a single computer vision task. To gain a more comprehensive understanding of the model's capabilities and performance, future research should aim to benchmark multiple datasets across various computer vision tasks. This broader approach will help to understand the differences between DDPMs and BBDMs, as well as the effect of different downsampling rates.

In recent years, Transformers [28] have surged in popularity, originating from natural language processing and becoming integral to many state-of-the-art models such as BERT and GPT. This architecture's success has extended beyond NLP, with numerous computer vision models now incorporating Transformers into their frameworks, achieving remarkable results. Transformers have increasingly supplanted traditional convolutional networks as the leading approach in various computer vision tasks, establishing themselves as versatile, general-purpose models. In the realm of diffusion models, there have been efforts to replace the noise-estimating U-Net with alternative models. Notably, Peebles et al. [15] demonstrated the effectiveness of substituting the U-Net backbone with Transformers in their work on scalable diffusion models. This influential paper has led to the widespread adoption of Transformers in diffusion-based models. Given that BBDMs require learning the mapping between two distributions directly, the backbone must capture richer representations than what is typically required in conditional DDPMs. We propose using a Transformer backbone with BBDMs to leverage the Transformer's capability for learning rich representations. To the best of our knowledge, this approach has not been previously explored, and it holds the potential to enhance the performance of BBDMs significantly.

By pursuing these future directions, researchers can continue to advance the field of diffusion models, uncovering new applications and improving the efficiency of these powerful generative models. The ongoing development and refinement of these models hold great potential for transforming the field of computer vision.

References

[1] Tomer Amit et al. "Segdiff: Image segmentation with diffusion probabilistic models". In: *arXiv preprint arXiv:2112.00390* (2021).

[2] Anas M. Tahir et al. *COVID-QU-Ex Dataset*. 2022. DOI: 10.34740/KAGGLE/DSV/3122958. URL: https://www.kaggle.com/dsv/3122958.

[3] Dmitry Baranchuk et al. "Label-efficient semantic segmentation with diffusion models". In: *arXiv preprint arXiv:2112.03126* (2021).

[4] Liang-Chieh Chen et al. "DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs". In: *CoRR* abs/1606.00915 (2016). arXiv: 1606.00915. URL: http://arxiv.org/abs/1606.00915.

[5] Florinel-Alin Croitoru et al. "Diffusion Models in Vision: A Survey". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2023), pp. 1–20. DOI: 10.1109/tpami.2023.3261988.

[6] Aysen Degerli et al. "Reliable Covid-19 Detection using Chest X-Ray Images". In: *2021 IEEE International Conference on Image Processing (ICIP)*. 2021, pp. 185–189. DOI: 10.1109/ICIP42928.2021.9506442.

[7] Prafulla Dhariwal and Alexander Nichol. "Diffusion models beat gans on image synthesis". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 8780–8794.

[8] Ian J. Goodfellow et al. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML].

[9] Jonathan Ho, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 6840–6851.

[10] Jonathan Ho and Tim Salimans. "Classifier-free diffusion guidance". In: *arXiv preprint arXiv:2207.12598* (2022).

[11] Bo Li et al. *BBDM: Image-to-image Translation with Brownian Bridge Diffusion Models*. 2023. arXiv: 2205.07680 [cs.CV].

[12] Mohammed Ali mnmoustafa. *Tiny ImageNet*. 2017. URL: https://kaggle.com/competitions/tiny-imagenet.

[13] Alex Nichol and Prafulla Dhariwal. *Improved Denoising Diffusion Probabilistic Models*. 2021. arXiv: 2102.09672 [cs.LG].

[14] Alex Nichol et al. *GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models*. 2022. arXiv: 2112.10741 [cs.CV].

[15] William Peebles and Saining Xie. *Scalable Diffusion Models with Transformers*. 2023. arXiv: 2212.09748 [cs.CV].

[16] Walter HL Pinaya et al. "Fast unsupervised brain anomaly detection and segmentation with diffusion models". In: *Medical Image Computing and Computer Assisted Intervention–MICCAI 2022: 25th International Conference, Singapore, September 18–22, 2022, Proceedings, Part VIII*. Springer. 2022, pp. 705–714.

[17] Aditya Ramesh et al. *Hierarchical Text-Conditional Image Generation with CLIP Latents*. 2022. arXiv: 2204.06125 [cs.CV].

[18] Robin Rombach et al. "High-resolution image synthesis with latent diffusion models". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 10684–10695.

[19] O. Ronneberger, P.Fischer, and T. Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Vol. 9351. LNCS. (available on arXiv:1505.04597 [cs.CV]). Springer, 2015, pp. 234–241. URL: http://lmb.informatik.uni-freiburg.de/Publications/2015/RFB15a.

[20] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. arXiv: 1505.04597 [cs.CV].

[21] Chitwan Saharia et al. *Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding*. 2022. arXiv: 2205.11487 [cs.CV].

[22] Yuyang Shi et al. *Diffusion Schrödinger Bridge Matching*. 2023. arXiv: 2303.16852 [stat.ML].

[23] Jascha Sohl-Dickstein et al. "Deep unsupervised learning using nonequilibrium thermodynamics". In: *International Conference on Machine Learning*. PMLR. 2015, pp. 2256–2265.

[24] Jiaming Song, Chenlin Meng, and Stefano Ermon. "Denoising diffusion implicit models". In: *arXiv preprint arXiv:2010.02502* (2020).

[25] Yang Song and Stefano Ermon. "Generative Modeling by Estimating Gradients of the Data Distribution". In: *CoRR* abs/1907.05600 (2019). arXiv: 1907.05600. URL: http://arxiv.org/abs/1907.05600.

[26] Yang Song et al. "Score-Based Generative Modeling through Stochastic Differential Equations". In: *CoRR* abs/2011.13456 (2020). arXiv: 2011.13456. URL: https://arxiv.org/abs/2011.13456.

[27] Yang Song et al. "Sliced Score Matching: A Scalable Approach to Density and Score Estimation". In: *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*. Ed. by Ryan P. Adams and Vibhav Gogate. Vol. 115. Proceedings of Machine Learning Research. PMLR, 22–25 Jul 2020, pp. 574–584. URL: https://proceedings.mlr.press/v115/song20a.html.

[28] Ashish Vaswani et al. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL].

[29] Julia Wolleb et al. *Diffusion Models for Implicit Image Segmentation Ensembles*. 2021. arXiv: 2112.03145 [cs.CV].

[30] Junde Wu et al. *MedSegDiff-V2: Diffusion based Medical Image Segmentation with Transformer*. 2023. arXiv: 2301.11798 [eess.IV].

[31] Junde Wu et al. *MedSegDiff: Medical Image Segmentation with Diffusion Probabilistic Model*. 2023. arXiv: 2211.00611 [cs.CV].

[32] Linqi Zhou et al. *Denoising Diffusion Bridge Models*. 2023. arXiv: 2309.16948 [cs.CV].

Appendix

Hyperparameter	Value
image_size	64
in_channels	1
out_channels	1
model_channels	192
attention_resolutions	(8, 4, 2)
num_res_blocks	2
channel_mult	(1, 2, 3, 4)
use_spatial_transformer	false
num_head_channels	64
resblock_updown	true
dropout	0.0

Table 6: Hyperparameters of the UNetModel

Hyperparameter	Value
num_timesteps	1000
mt_type	linear
max_var	1.0
eta	1.0
sample_type	linear
sample_step	200
objective	grad
image_size	64
in_channels	1
condition_key	nocond

Table 7: Hyperparameters of the BrownianBridgeDiffusionModel

Hyperparameter	Value
timeteps	1000
beta_schedule	linear
image_size	64
channels	1
log_every_t	100
linear_start	0.0001
linear_end	0.02
cosine_s	0.008
original_elbo_weight	0.0
v_posterior	0.0
l_simple_weight	1.0
given_betas	null
parameterization	eps
ddim	true
ddim_num_timesteps	50
is_conditional	true
cond_stage_trainable	true
data_key	mask
concat_mode	true
addition_mode	false
cross_attn_mode	false

Table 8: Hyperparameters of the DDPM Model

Hyperparameter	Value
embed_dim	3
n_embed	8192
double_z	false
z_channels	3
resolution	64
in_channels	1
out_ch	1
ch	128
ch_mult	(1, 2, 4)
num_res_blocks	2
attn_resolutions	null
dropout	0.0
lr_g_factor	1.0
sane_index_shape	false
normalize_latent	false

Table 9: Hyperparameters of the VQModel-F4

Hyperparameter	Value
image_key	mask
embed_dim	4
n_embed	16384
double_z	false
z_channels	4
resolution	64
in_channels	1
out_ch	1
ch	128
ch_mult	(1, 2, 2, 4)
num_res_blocks	2
attn_resolutions	32
dropout	0.0
lr_g_factor	1.0
sane_index_shape	false
normalize_latent	false

Table 10: Hyperparameters of the VQModel-F8

NYILATKOZAT

Név: Szabó Milán

ELTE Természettudományi Kar, szak: Alkalmazott Matematikus MSc

NEPTUN azonosító: GUP46S

Szakdolgozat címe:
Diffuziós modellek és alkalmazásaik

A **szakdolgozat** szerzőjeként fegyelmi felelősségem tudatában kijelentem, hogy a dolgozatom önálló szellemi alkotásom, abban a hivatkozások és idézések standard szabályait következetesen alkalmaztam, mások által írt részeket a megfelelő idézés nélkül nem használtam fel.

Budapest, 2024.05.30.

Szabó M.

a hallgató aláírása