ETH zürich

Master Thesis

Convergence Maps With Denoising Diffusion Probabilistic Models

Author:
Mila Luescher

Supervisors:
Prof. Dr. Alexandre Refregier
Dr. Tilman Troester
Arne Thomsen

March 26, 2023

Abstract

In this thesis we investigate the efficacy of denoising diffusion probabilistic models (DDPM) in generating convergence maps conditional on the cosmological parameters Ω_m and σ_8, and compare it to previous work using the conditional Wasserstein-GAN (CWGAN) by Perraudin et al. 2020. We utilise a number of summary statistics, including the pixel value distribution, power-spectrum, bispectrum, Minkowski functionals as well as the Wasserstein-1 distance to compare the images generated by the DDPM to test data. Additionally, we examine the use of different hyper-parameters as well as sampling methods. We find that our DDPM successfully generates accurate convergence maps which are only marginally outperformed by the CWGAN and thus provide a simpler and more stable alternative. Given more time to fine-tune the DDPM, it is likely that it will be able to surpass the CWGAN.

Contents

Chapter 1

Introduction

Since the dawn of time, humankind has been fascinated by the sky, trying to unveil the mysteries the universe holds. In recent decades, the study of the cosmos has increasingly relied on cosmological N-Body simulations (e.g. Potter, J. Stadel, and Teyssier 2016, Hopkins 2015). By comparing these simulations to observations the cosmological parameters governing our universe can be constrained. Unfortunately, running a single cosmological simulation can take up to several weeks and requires significant computational resources. With the rise of generative machine learning researchers have tried to circumvent these issues by training generative models, which produce outcomes in just a few seconds to minutes, to be used in place of simulations or observations (e.g. Perraudin et al. 2020, Ullmo, Decelle, and Aghanim 2021, Wing Hei Yiu, Fluri, and Kacprzak 2021).

In this thesis we explore the potential of the newly developed denoising diffusion probabilistic model (DDPM) (Ho, Jain, and Abbeel 2020) in generating convergence maps. This is not a completely novel approach, as Mudur and Finkbeiner 2022 have already shown their potential at generating dark matter mass density fields and images of interstellar dust and Smith et al. 2022 have provided evidence for the generation of galaxy images, outperforming other generative models such as generative adversarial networks (GAN) in image sharpness and realism. Our model differs not only in the type of maps generated but also in complexity as we train a conditional DDPM, conditioned on both the matter density Ω_m and the root-mean-square matter fluctuation σ_8. To check the quality of our DDPM we make use of a number of summary statistics such as pixel value distribution histograms and Wasserstein-1 distance which are a simple statistical measure of the distribution of the pixels. We also probe the structure of the maps by calculating the power-spectrum for lower order as well as the bispectrum for higher order structure. Finally, we also make use of Minkowski functionals which give us information about the morphology of the convergence maps. For comparability, we utilise the same training data as in Perraudin et al. 2020 in addition to using their conditional Wasserstein-GAN as a baseline.

The thesis is structured in the following way. First discussing some theoretical concepts, we briefly introduce weak lensing and convergence in Sec. 2.1 as well as the cosmological parameters the DDPM is conditioned on in Sec. 2.2. Next, in Sec. 2.3 we examine two widely used generative models in computer vision, the variational autoencoder and GAN before introducing the DDPMs and its sampling methods. Building upon these theoretical concepts we outline the experiment in Sec. 3 going into detail on the training data, neural network architectures as well as the summary statistics. Finally, we discuss the results in Sec. 4, comparing the best checkpoint of the DDPM to the CWGAN as well investigating the use of different sampling methods and hyper-parameters before concluding the thesis.

Chapter 2

Theory

2.1 Gravitational lensing

The theory of general relativity predicts that massive objects bend the fabric of spacetime (Einstein 1914). When we observe distant objects these wells in spacetime act as lenses, causing the path of light to bend. This leads to the observed objects appearing distorted, magnified and in another position in the sky, an illustration of which can be seen in Fig. 2.1. This phenomenon is known as gravitational lensing and it is an important tool for studying the distribution of matter in the universe. A few examples of objects we can observe acting as lenses include black holes (e.g. Bozza and Mancini 2004), galaxy clusters (e.g. Liu et al. 2023) and dark matter halos (e.g. Herrera-Martín et al. 2019).

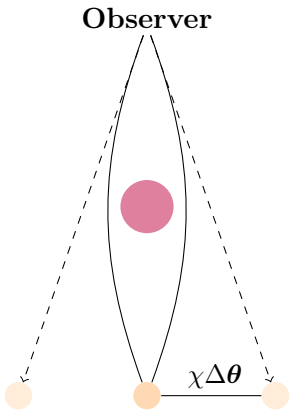

Figure 2.1: The purple circle represents a massive object which acts as the gravitational lens on the observed object in orange. The observed object then appears (for example) on both sides of the lens.

When observing a lensed object, we can describe the deflection angle (Scott Dodelson and Schmidt 2020)[1] of the light emitted from the source as

$$\Delta\theta^i(\boldsymbol{\theta}) = \frac{\partial}{\partial\theta^i}\phi_L(\boldsymbol{\theta}) \tag{2.1}$$

$$\phi_L(\boldsymbol{\theta}) \equiv 2\int_0^\chi \frac{d\chi'}{\chi'}\Phi(\boldsymbol{x}(\boldsymbol{\theta},\chi'))\left(1-\frac{\chi'}{\chi}\right) \tag{2.2}$$

[1]The full derivation can be found in Scott Dodelson and Schmidt 2020 in Chapter 13

where $\phi_L(\boldsymbol{\theta})$ is the lensing potential, $\boldsymbol{\theta}$ the observed position, χ is the co-moving radius, Φ the perturbation of the spatial curvature and $\boldsymbol{x}(\boldsymbol{\theta}, \chi') = (\chi'\theta^1, \chi', \theta^2, \chi')$ the unperturbed photon path.

2.1.1 Weak lensing

Occasionally astrophysical objects can appear strongly lensed, for example if a supermassive black hole lies within their line of sight (e.g. Bozza, Novati, and Mancini 2008), but generally lensing effects are not detectable by visualising a single source and a statistical approach is necessary where multiple sources are averaged over. In this case we introduce the galaxy distribution $W(\chi)$, normalised $\int_0^\infty d\chi W(\chi) = 1$, and implement this into Eq. 2.2, finding the weak lensing potential

$$\phi_L(\boldsymbol{\theta}) = 2\int_0^\infty d\chi W(\chi) \int_0^\chi \frac{d\chi'}{\chi'} \Phi(\boldsymbol{x}(\boldsymbol{\theta}, \chi')) \left(1 - \frac{\chi'}{\chi}\right) \tag{2.3}$$

$$\phi_L(\boldsymbol{\theta}) = 2\int_0^\infty \frac{d\chi'}{\chi'} g_L(\chi') \Phi(\boldsymbol{x}(\boldsymbol{\theta}, \chi')) \tag{2.4}$$

where $g_L(\chi')$ is the lensing kernel

$$g_L(\chi') \equiv \int_{\chi'}^\infty d\chi \left(1 - \frac{\chi'}{\chi}\right) W(\chi) \tag{2.5}$$

which can be used to model the efficiency of gravitational lensing on a distribution of sources. A lensing kernel was also utilised in the creation of the convergence maps used in this thesis (details in Sec. 3.1). The convergence κ is known as the dimensionless surface-mass density

$$\kappa(\boldsymbol{\theta}) = \frac{\Sigma(\boldsymbol{\theta})}{\Sigma_{\text{cr}}} \quad \text{with} \quad \Sigma_{\text{cr}} = \frac{c^2}{4\pi G} \frac{D_S}{D_L D_{LS}} \tag{2.6}$$

with Σ_{cr} being the critical surface-mass density, D_S the distance between the source and observer, D_L the distance between lens and observer and D_{LS} the distance between lens and source. The convergence can also be described as the source of the lensing potential

$$\kappa(\boldsymbol{\theta}) = \frac{1}{2} \nabla^2_{\theta} \phi_L(\boldsymbol{\theta}) \tag{2.7}$$

2.2 Ω_m, σ_8 and the S_8 tension

As mentioned previously, the cosmological parameters on which we condition our generative model are the matter density Ω_m and present root-mean-square matter fluctuations σ_8. A parameter which combines the two is the S_8 parameter $S_8 = \sigma_8 \sqrt{\Omega_m / 0.3}$ and quantifies the tension between the CMB Planck data and lower redshift data at a level of 2-3σ (Martinelli and Tutusaus 2019). Specifically in weak lensing measurements, when aiming to constrain Ω_m and σ_8 the S_8 parameter becomes useful since it breaks some of the degeneracy between the two and allows to constrain them better. Hence N-Body simulations (e.g. Fluri et al. 2019) or generative machine learning models (e.g. Perraudin et al. 2020, Wing Hei Yiu, Fluri, and Kacprzak 2021) in the field of cosmology are often conditioned on these two parameters mostly because they have the largest influence on the evolution of structure formation but also to gather more data on the S_8 tension.

2.3 Generative modelling

In recent years, significant progress has been made in machine learning research with a focus on improving unsupervised learning techniques such as generative models (e.g. Karras, Aittala, et al. 2021, Hong et al. 2023, Xu et al. 2022). Typically, the goal of a generative model is to learn the underlying probability distribution $P_{\text{data}}(\boldsymbol{x})$ of a dataset in order to generate new samples that resemble the original dataset, but are not contained within it (Ng and Jordan 2001). Since this distribution is not known in most cases, such as for images of faces or weak lensing maps, we aim to find an approximate model $P_\theta(\boldsymbol{x}) \approx P_{\text{data}}(\boldsymbol{x})$. This is achieved by training a neural network, which consists of minimising a loss function in order to optimise the trainable parameters θ of the approximate model. In the following section we introduce two popular generative models: the variational autoencoder (Kingma and Welling 2014) and the generative adversarial network (Goodfellow et al. 2014). Building upon these we then direct our attention to denoising diffusion probabilistic models (Ho, Jain, and Abbeel 2020).

2.3.1 Variational autoencoders

Autoencoders

Before we have a look at how variational autoencoders work, we first need to understand the structure of autoencoders (AE). Autoencoders were initially created for dimensionality reduction (Hinton and Salakhutdinov 2006) but soon found use also in generative machine learning (Vincent et al. 2010). The dimensionality reduction is preformed with a so called encoder which encodes the original data onto the lower dimensional latent space $\boldsymbol{x} \mapsto \boldsymbol{e}_\theta(\boldsymbol{x}) = \boldsymbol{z}$ in a nonlinear fashion, where θ are the trainable parameters. For the purpose of training the encoder a decoder is used to revert back to the original data $\boldsymbol{z} \mapsto \boldsymbol{d}_\phi(\boldsymbol{z})$, with trainable parameters ϕ. Training such a network would then consist of minimising a loss function which for example could take the form of an L2-loss

$$L_{\text{AE}}(\theta, \phi) = \frac{1}{2m} \sum_{i=1}^{m} ||\boldsymbol{x}_i - d_\phi(e_\theta(\boldsymbol{x}_i))||_2^2 \tag{2.8}$$

which compares the input data $\boldsymbol{x}_i$ to the output $d_\phi(e_\theta(\boldsymbol{x}_i))$ after passing though a bottleneck. As mentioned earlier, the decoder of an AE can also be seen as a rudimentary version of a generative model by sampling latent vectors from the latent space and feeding

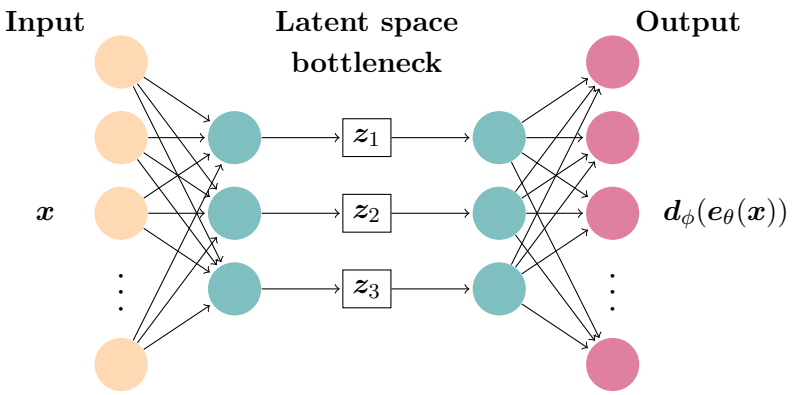

Figure 2.2: Depiction of the network structure of a fully connected autoencoder where the dimensionality of the input vector is reduced to $d = 3$. The encoder $\boldsymbol{e}_\theta$ encodes an input $\boldsymbol{x}$ onto the latent space $\boldsymbol{x} \mapsto \boldsymbol{e}_\theta(\boldsymbol{x}) = \boldsymbol{z}$ and the decoder $\boldsymbol{d}_\phi$ aims at mapping the latent vector back such that $\boldsymbol{z} \mapsto \boldsymbol{d}_\phi(\boldsymbol{z}) \approx \boldsymbol{x}$.

them through the decoder. Even though autoencoders have shown great success in dimensionality reduction (as compared to t-SNE or PCA) (Hinton and Salakhutdinov 2006) they are not a great choice for a generative model. This is because the network only learns to reconstruct the input data and does not span the whole latent space. Nevertheless, there have been some successful attempts at altering the vanilla AE with the purpose of making it generative (Böhm and Seljak 2022, Makhzani et al. 2016).

Variational autoencoders

Variational autoencoders (VAE) were created with the goal of making an AE where the decoder can be used to generate samples from the latent space (Kingma and Welling 2014). As this is a generative model we wish to learn an approximate probability distribution $p_\theta(\boldsymbol{x})$ which mimics the true distribution of the data $p_{\text{data}}(\boldsymbol{x})$. A trainable encoder $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ and decoder $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ are then utilised to map between a simple prior/latent distribution $p_\theta(\boldsymbol{z})$ and the more complex data distribution. This process is visualised in Fig. 2.3.

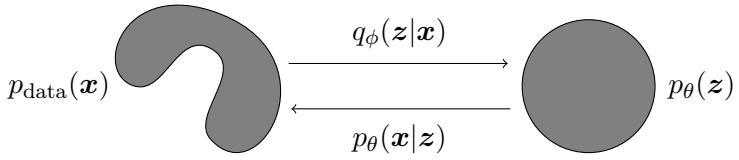

Figure 2.3: Meta-structure of a variational autoencoder, where $p_{\text{data}}(\boldsymbol{x})$ is the unknown data distribution and $p_\theta(\boldsymbol{z})$ is some simple latent distribution, for example a Gaussian. The encoder $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ then maps a point from the data distribution onto the latent distribution and the decoder $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ maps from the latent distribution onto a datapoint.

The model then simultaneously learns the joint distribution $p_\theta(\boldsymbol{x}, \boldsymbol{z}) = p_\theta(\boldsymbol{z})p_\theta(\boldsymbol{x}|\boldsymbol{z})$ and the interference model $q_\phi(\boldsymbol{z}|\boldsymbol{x})$. We can see this more clearly by investigating the log-likelihood (more in Kingma and Welling 2019) of the data

$$\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_\theta(\boldsymbol{x})\right] \tag{2.9}$$

$$= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log\left[\frac{p_\theta(\boldsymbol{x}, \boldsymbol{z})}{p_\theta(\boldsymbol{z} \mid \boldsymbol{x})}\right]\right] \tag{2.10}$$

$$= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log\left[\frac{p_\theta(\boldsymbol{x}, \boldsymbol{z})}{q_\phi(\boldsymbol{z} \mid \boldsymbol{x})}\frac{q_\phi(\boldsymbol{z} \mid \boldsymbol{x})}{p_\theta(\boldsymbol{z} \mid \boldsymbol{x})}\right]\right] \tag{2.11}$$

$$= \underbrace{\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log\left[\frac{p_\theta(\boldsymbol{x}, \boldsymbol{z})}{q_\phi(\boldsymbol{z} \mid \boldsymbol{x})}\right]\right]}_{\substack{= L_{\theta,\phi}(\boldsymbol{x}) \\ \text{(ELBO)}}} + \underbrace{\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log\left[\frac{q_\phi(\boldsymbol{z} \mid \boldsymbol{x})}{p_\theta(\boldsymbol{z} \mid \boldsymbol{x})}\right]\right]}_{=D_{KL}\left(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p_\theta(\boldsymbol{z}|\boldsymbol{x})\right)} \tag{2.12}$$

where D_{KL} is the Kullback-Leibler divergence (Kullback and Leibler 1951) which is a measure of distance between the probability distributions and takes on positive values expect for when $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ matches $p_\theta(\boldsymbol{z}|\boldsymbol{x})$, where it becomes zero. This indicates, that the first term, the *evidence lower bound* (ELBO), creates a lower bound for the log-likelihood of the data distribution

$$L_{\theta,\phi}(\boldsymbol{x}) = \log p_\theta(\boldsymbol{x}) - D_{KL}\left(q_\phi(\boldsymbol{z} \mid \boldsymbol{x})||p_\theta(\boldsymbol{z} \mid \boldsymbol{x})\right) \leq \log p_\theta(\boldsymbol{x}) \tag{2.13}$$

Hence, the training objective becomes maximising the ELBO loss with regards to the training parameters ϕ and θ

$$L_{\theta,\phi}(\boldsymbol{x}) = \log p_\theta(\boldsymbol{x}, \boldsymbol{z}) - \log q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \tag{2.14}$$

A simple implementation of a VAE, where the prior distribution is chosen to be a diagonal multivariate gaussian can be seen in Fig. 2.4.

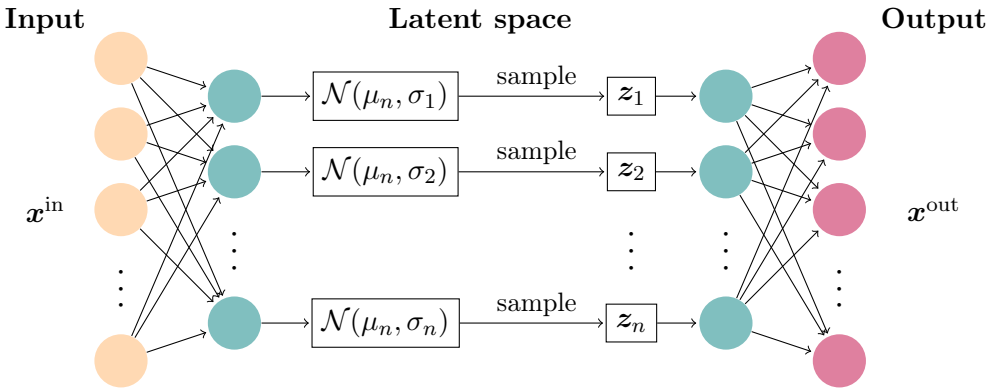

Figure 2.4: The architecture of a variational autoencoder where the input $\boldsymbol{x}$ is mapped onto parameters of a normal distribution from which a latent point $\boldsymbol{z}$ is drawn and decoded back.

2.3.2 Generative adversarial networks

The introduction of generative adversarial networks (GAN) (Goodfellow et al. 2014) marked the start of a new era of image generation (Karras, Laine, and Aila 2019, Brock, Donahue, and Simonyan 2019, Zhu et al. 2020). The way GANs are constructed is as a game with two adversarial players: the generator $G_{\theta_G}(\boldsymbol{z}) : \boldsymbol{z} \to \boldsymbol{x}$, which tries to create data from noise which fools the discriminator, and the discriminator $D_{\theta_D}(\boldsymbol{x}) : \boldsymbol{x} \to$ true/false, which tries to correctly distinguish between generated data and training data. An example of the structure of a GAN can be seen in Fig. 2.5, where the architecture of the generator and discriminator usually consists of convolutional layers when used in computer vision (as in Perraudin et al. 2020). During training the generator and discriminator take turns evaluating their loss function and updating their parameters.

$$L_D(\theta_D; \theta_G) = -\frac{1}{2N_1}\sum_i \log D_{\theta_D}(\boldsymbol{x}_i) - \frac{1}{2N_2}\sum_j \log(1 - D_{\theta_D}(G_{\theta_G}(\boldsymbol{z}_j))) \tag{2.15}$$

$$L_G(\theta_G; \theta_D) = -\frac{1}{2N_2}\sum_j \log D_{\theta_D}(G_{\theta_G}(\boldsymbol{z}_j)) \tag{2.16}$$

In the loss function for the discriminator $\log D_{\theta_D}(\boldsymbol{x}_i)$ is the probability that the discriminator correctly classifies a real image and $\log(1 - D_{\theta_D}(G_{\theta_G}(\boldsymbol{z}_j)))$ the probability that it correctly classifies the image of the generator as fake. The generator on the other hand tries to increase the probability $\log D_{\theta_D}(G_{\theta_G}(\boldsymbol{z}_j))$ of having the discriminator falsely classify its generated image as real, which incentivises the generator to produce more realistic images.

Despite their great success GANs have a few significant shortcomings. They require a large amount of training data in order to generate accurate data, their training dynamics are unstable and they tend be vulnerable to mode collapse, meaning that the generator becomes very good at producing only one class of images which fools the discriminator best (Saad, O'Reilly, and Rehmani 2022). There are a few ways to combat this issue (Wiatrak, Albrecht, and Nystrom 2020), such as allowing the discriminator to evaluate a few images at a time, thus letting it take notice to multiple similar images. Another popular approach to overcome mode collapse is implementing a Wasserstein loss (Arjovsky, Chintala, and Bottou 2017).

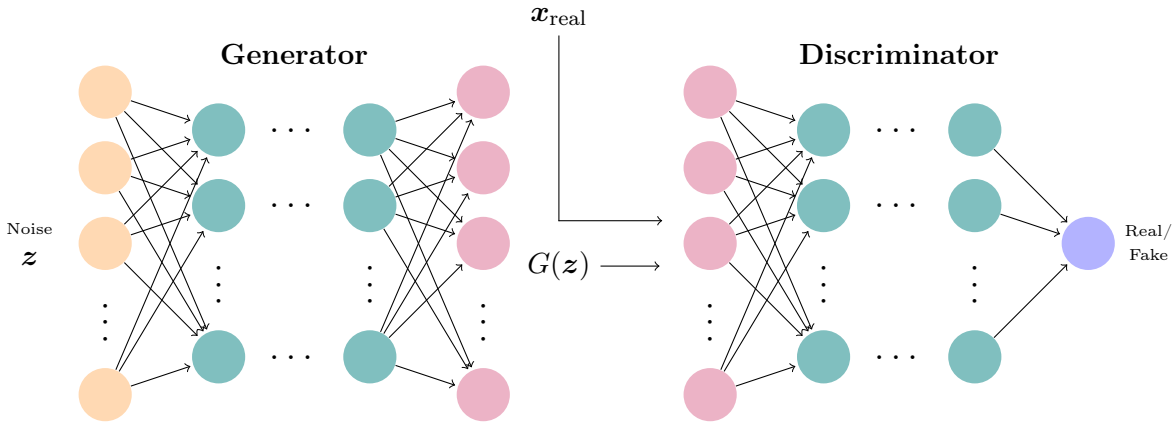

Figure 2.5: The structure of a generative adversarial network where the generator tries to generate images $G(\boldsymbol{z})$ which fool the discriminator $D(G(\boldsymbol{z}))$. The discriminator tries to correctly distinguish between the real $\boldsymbol{x}_{\text{real}}$ and fake images $G(\boldsymbol{z})$.

2.3.3 Denoising diffusion probabilistic models

Both variational autoencoders and GANs are powerful tools for natural image generation. Nevertheless choosing one over the other always leaves room for improvement. GANs are notoriously bad at producing a diverse range of samples, are hard to train and require large amounts of training data (Wiatrak, Albrecht, and Nystrom 2020). On the other hand, VAEs can more easily produce a wider range of images but lack in image quality and resolution (Yacoby, Pan, and Doshi-Velez 2021). In 2015, Sohl-Dickstein et al. n.d. proposed a new architecture, diffusion probabilistic models, that would produce high quality images without sacrificing variety. The idea behind the diffusion model is to train a neural network to convert Gaussian noise into a target sample via a Markov chain. Originally, this method was inspired by work in sequential Monte Carlo (Neal 1998) and non-equilibrium statistical physics (Jarzynski 1997). The implementation investigated in this thesis is the denoising diffusion probabilistic model (DDPM) (Ho, Jain, and Abbeel 2020) which has already seen success in projects like DALL-E 2 or Stable Diffusion where they use a diffusion model for the image generation part of their generative language model. In Fig. 2.6 one can see some examples of the text-to-image generation from DALL-E 2[2]. Furthermore, there have been projects implementing the diffusion process into VAEs (e.g. Pandey et al. 2022) as well as GANs (e.g. Wang et al. 2022).

There are two key aspects to the diffusion model, the forward diffusion process and the reverse diffusion process. Essentially, the former is a controlled Markov chain that adds noise to an image in incremental steps and the latter is trained by a neural network to de-noise the image in an equal amount of steps. In the following section we provide a more detailed description of both.

Forward diffusion process

The forward diffusion process is characterised by adding small amounts of Gaussian noise to the original data point $\boldsymbol{x}_0$ in T amount of steps, where $\boldsymbol{x}_t$ is the noisy image at time-step $t < T$. The amount of noise added in each step is determined by a variance scheduler $\{\beta_t \in (0,1)\}_{t=1}^{T}$. The conditional probability of the image at the next time-step then is

[2]It is important to note, that for these projects DDPMs are only a small part of the whole network and a lot more work went into the encoding of textual and visual information, where a sentence such as "*An oil painting of a corgi playing chess*" is first mapped onto a representation space and later onto an image representation space.

Figure 2.6: Three images generated with DALL-E 2 with the following prompts: left - "*An oil painting of a corgi playing chess*", middle - "*A 3D rendering of a whale floating in space*", right - "*A black hole consuming Earth as digital art*"

described as

$$q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t\mathbf{1}) \tag{2.17}$$

The entire forward diffusion

$$q(\boldsymbol{x}_{0:T}|\boldsymbol{x}_0) = q(\boldsymbol{x}_0)\prod_{t=1}^{T} q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) \tag{2.18}$$

is then the product of all conditional probabilities. These steps are also visualised in Fig. 2.7. For large time-steps T the process of adding noise can become very computationally tedious since the process is recursive. Luckily, there is a way to simplify the problem by defining $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$. Using $\boldsymbol{\epsilon}_{t-1}, \boldsymbol{\epsilon}_{t-2}, \cdots \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$, one can now write

$$\begin{aligned}
\boldsymbol{x}_t &= \sqrt{\alpha_t}\boldsymbol{x}_{t-1} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_{t-1} && (2.19)\\
&= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{\alpha_t}\sqrt{1-\alpha_{t-1}}\boldsymbol{\epsilon}_{t-2} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_{t-1} && (2.20)\\
&= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\bar{\boldsymbol{\epsilon}}_{t-2} && (2.21)
\end{aligned}$$

where $\bar{\boldsymbol{\epsilon}}_{t-2}$ merges two Gaussians which leads to a new standard deviation of

$$\sqrt{\sigma_t^2 + \sigma_{t-1}^2} = \sqrt{\alpha_t(1-\alpha_{t-1}) + (1-\alpha_t)} = \sqrt{1-\alpha_t\alpha_{t-1}} \tag{2.22}$$

This process can be iterated multiple times to find a dependency on the original image $\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$ which implies

$$q(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0, (1-\bar{\alpha}_t)\mathbf{1}) \tag{2.23}$$

Since $\boldsymbol{x}_0$ is known, this means that as long as all $\bar{\alpha}_t$ are calculated once, one can sample any given time-step $\boldsymbol{x}_t$ on the spot without having to iterate through all of the intermediate steps in the Markov chain.

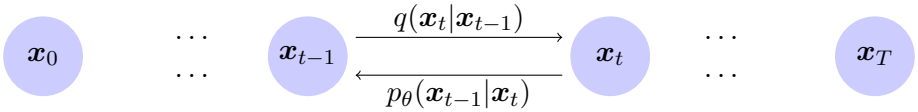

Figure 2.7: Both the forward diffusion process and the reverse diffusion process are shown, where $\boldsymbol{x}_0$ is the original image and $\boldsymbol{x}_t$ is the noised image after t steps.

Reverse diffusion process

Analytically we can not find the reverse diffusion process $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ and hence we need to approximate it with a trainable model p_θ

$$p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_{t-1}; \mu_\theta(\boldsymbol{x}_t, t), \beta_\theta(\boldsymbol{x}_t, t)) \tag{2.24}$$

$$p_\theta(\boldsymbol{x}_{0:T}) = p(\boldsymbol{x}_T) \prod_{t=1}^{T} p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) \tag{2.25}$$

where μ_θ and β_θ depend on the trainable parameters θ. Using Bayes' theorem $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$ and the fact that the reverse conditional probability

$$q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_{t-1}; \tilde{\mu}(\boldsymbol{x}_t, \boldsymbol{x}_0), \tilde{\beta}_t \mathbf{1}) \tag{2.26}$$

is a Normal distribution $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) \propto \exp\left(-\frac{1}{2}(\frac{x_{t-1}-\mu}{\sigma})^2\right)$ we find (Weng 2021)

$$\begin{aligned}
q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) &= \frac{q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{x}_0) q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)}{q(\boldsymbol{x}_t|\boldsymbol{x}_0)} \\
&\propto \exp\left(-\frac{1}{2}\left[\frac{(\boldsymbol{x}_t - \sqrt{\alpha_t}\boldsymbol{x}_{t-1})^2}{\beta_t} + \frac{(\boldsymbol{x}_{t-1} - \sqrt{\bar{\alpha}_{t-1}}\boldsymbol{x}_0)^2}{1-\bar{\alpha}_{t-1}} - \frac{(\boldsymbol{x}_t - \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0)^2}{1-\bar{\alpha}_t}\right]\right) \\
&\propto \exp\left(-\frac{1}{2}\left[\left(\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}}\right)\boldsymbol{x}_{t-1}^2 - 2\left(\frac{\sqrt{\alpha_t}}{\beta_t}\boldsymbol{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}}\boldsymbol{x}_0\right)\boldsymbol{x}_{t-1} + C(\boldsymbol{x}_t, \boldsymbol{x}_0)\right]\right)
\end{aligned}$$

If we expand $\exp\left(-\frac{1}{2}(\frac{x_{t-1}-\mu}{\sigma})^2\right) = \exp\left(-\frac{1}{2}(\frac{1}{\sigma^2}\boldsymbol{x}_{t-1}^2 - 2\frac{\mu}{\sigma^2}\boldsymbol{x}_{t-1} + \frac{\mu^2}{\sigma^2})\right)$, we can see that

$$\tilde{\beta}_t = 1/\left(\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}}\right) \tag{2.27}$$

$$\tilde{\mu}(\boldsymbol{x}_t, \boldsymbol{x}_0) = \left(\frac{\sqrt{\alpha_t}}{\beta_t}\boldsymbol{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}}\boldsymbol{x}_0\right) / \left(\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}}\right) \tag{2.28}$$

$$= \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\boldsymbol{x}_t + \frac{\sqrt{\bar{\alpha}_t}\beta_t}{1-\bar{\alpha}_t}\boldsymbol{x}_0 \tag{2.29}$$

$$= \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_t\right) \tag{2.30}$$

Since $\tilde{\beta}_t$ only depends on variables which we know from the forward diffusion process, we can calculate $\tilde{\beta}_t$ and thus we do not need to train β_θ. This leaves $\tilde{\mu}(\boldsymbol{x}_t, \boldsymbol{x}_0)$ as the only unknown. Hence, in order to find the estimate $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ we need to train $\mu_\theta(\boldsymbol{x}_t, t)$ to predict $\tilde{\mu}(\boldsymbol{x}_t, \boldsymbol{x}_0)$ and set $\beta_\theta = \tilde{\beta}_t$. Since $\boldsymbol{x}_t$ is an input in training we essentially only need to train the noise term $\boldsymbol{\epsilon}_\theta$.

$$\mu_\theta(\boldsymbol{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\boldsymbol{x}_t, t)\right) \tag{2.31}$$

During training the evidence lower bound (see Sec. 2.3.1 for derivation) on the negative log likelihood would be optimised

$$\mathbb{E}[-\log p_\theta(\boldsymbol{x}_0)] \leq \mathbb{E}_q\left[-\log \frac{p_\theta(\boldsymbol{x}_{0:T})}{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\right] = \mathbb{E}_q\left[-\log p_\theta(\boldsymbol{x}_T)\sum_{t\geq 1}\log\frac{p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})}\right] := L_{\text{ELBO}} \tag{2.32}$$

which can be rewritten as

$$\begin{aligned} L_{\text{ELBO}} &= \mathbb{E}_q[D_{\text{KL}}\left(q\left(\boldsymbol{x}_T \mid \boldsymbol{x}_0\right) \| p_\theta\left(\boldsymbol{x}_T\right)\right) \\ &+ \sum_{t=2}^{T} D_{\text{KL}}\left(q\left(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0\right) \| p_\theta\left(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t\right)\right) - \log p_\theta\left(\boldsymbol{x}_0 \mid \boldsymbol{x}_1\right)] \end{aligned} \tag{2.33}$$

Ho, Jain, and Abbeel 2020 found that using a simplified loss function improved sample quality

$$L_{\text{simple}}(\theta) \coloneqq \mathbb{E}_{t,\mathbf{x}_0,\epsilon}\left[||\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)||^2\right] \tag{2.34}$$

If $T \to \infty$, then $\boldsymbol{x}_T$ will follow a Gaussian distribution, which means that we can sample new images from a Gaussian distribution once we have found $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ by training the NN.

2.3.4 Sampling

Once the neural network has been trained there are a multitude of ways one can approach the sampling of new datapoints. Here we focus on the sampling methods for denoising diffusion probabilistic models specifically. Ho, Jain, and Abbeel 2020 made use of ancestral sampling where one starts with a Gaussian field and then iterates backwards, removing noise at each time-step. Y. Song et al. 2021 showed that the forward as well as reverse diffusion process can be written as stochastic differential equations (SDE). The reverse time SDE only depends on the score, which allows for the SDE to be numerically solved. They also proposed an equivalent ODE which can be solved directly from noise, skipping many iteration steps and thus speeding up the sampling process.

Ancestral sampling

Given that ancestral sampling is an iterative process, one starts by drawing a sample from a standard Normal distribution $\boldsymbol{x}_T \sim \mathcal{N}(0, 1)$, with a mean of zero and a variance of one. In a second step, one removes one layer of noise to find $\boldsymbol{x}_{T-1}$

$$\boldsymbol{x}_{T-1} = \frac{1}{\sqrt{\bar{\alpha}_T}}\left(\boldsymbol{x}_T - \frac{1-\alpha_T}{\sqrt{1-\bar{\alpha}_T}}\epsilon_\theta(\boldsymbol{x}_T, T)\right) + \sigma_T \boldsymbol{z} \tag{2.35}$$

where $\epsilon_\theta(\boldsymbol{x}_T, T)$ is the output of the neural network, $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$ and $\sigma_T = \sqrt{\tilde{\beta}_T}$. This process of removing noise at each time-step is then repeated (iterating over $t = T, \ldots, 1$) until the de-noised image $\boldsymbol{x}_0$ is generated. The full algorithm is shown in Table 2.1.

Sampling Algorithm
1: $\boldsymbol{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$
2: **for** $t = T, \ldots, 1$ **do**
3: $\quad \boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$ if $t > 1$ else $\boldsymbol{z} = 0$
4: $\quad \boldsymbol{x}_{t-1} = \frac{1}{\sqrt{\bar{\alpha}_t}}(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\boldsymbol{x}_t, t)) + \sigma_t \boldsymbol{z}$
5: **return** $\boldsymbol{x}_0$

Table 2.1: Summary of the ancestral sampling algorithm

Reverse time SDE

A stochastic differential equation (SDE)

$$\mathrm{d}\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{x}, t)\mathrm{d}t + \boldsymbol{G}(t)\mathrm{d}\boldsymbol{w} \tag{2.36}$$

with $\boldsymbol{f}(\boldsymbol{x}, t)$ being the drift coefficient and $\boldsymbol{G}(t)$ the scalar coefficient, can be discretised in the following way

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \boldsymbol{f}(\boldsymbol{x}_i, i) + \boldsymbol{G}(i)\boldsymbol{z}_i \tag{2.37}$$

If we compare this to Eq. 2.19 we find $\boldsymbol{f}(\boldsymbol{x}_i, i) = (\sqrt{1-\beta_i} - 1)\boldsymbol{x}_i$ and $\boldsymbol{G}(i) = \sqrt{\beta_i}$. In order to find $\boldsymbol{x}_0$ from $\boldsymbol{x}_T$ the SDE needs to be reversed. Y. Song et al. 2021 proposed the following reverse time SDE and its discretisation

$$\mathrm{d}\boldsymbol{x} = \left[\boldsymbol{f}(\boldsymbol{x}, t) - \boldsymbol{G}(t)\boldsymbol{G}(t)^\top \nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})\right] \mathrm{d}t + \boldsymbol{G}(t)\mathrm{d}\boldsymbol{w} \tag{2.38}$$

$$\boldsymbol{x}_i = \boldsymbol{x}_{i+1} - \boldsymbol{f}(\boldsymbol{x}_{i+1}, i+1) + \boldsymbol{G}_{i+1}\boldsymbol{G}_{i+1}^\top \boldsymbol{s}_\theta^*(\boldsymbol{x}_{i+1}, i+1) + \boldsymbol{G}_{i+1}\boldsymbol{z}_{i+1} \tag{2.39}$$

They also showed that the ancestral sampling in Ho, Jain, and Abbeel 2020 is equivalent to the discretised SDE of the following form

$$\boldsymbol{x}_i = (2 - \sqrt{1-\beta_{i+1}})\boldsymbol{x}_{i+1} + \beta_{i+1}\boldsymbol{s}_\theta^*(\boldsymbol{x}_{i+1}, i+1) + \sqrt{\beta_{i+1}}\boldsymbol{z}_{i+1} \tag{2.40}$$

Probability flow ODE

The forward SDE can also be written as an ODE with the following form

$$\mathrm{d}\boldsymbol{x} = \left\{\boldsymbol{f}(\boldsymbol{x}, t) - \frac{1}{2}\boldsymbol{G}(t)\boldsymbol{G}(t)^\top \nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})\right\} \mathrm{d}t \tag{2.41}$$

Y. Song et al. 2021 proposed to integrate this probability flow ODE backwards in time by discretising it

$$\boldsymbol{x}_i = \boldsymbol{x}_{i+1} - \boldsymbol{f}_{i+1}(\boldsymbol{x}_{i+1}) + \frac{1}{2}\boldsymbol{G}_{i+1}\boldsymbol{G}_{i+1}^\top \boldsymbol{s}_\theta^*(\boldsymbol{x}_{i+1}, i+1) \tag{2.42}$$

which they showed in the specific case of a DDPM to be

$$\boldsymbol{x}_i = (2 - \sqrt{1-\beta_{i+1}})\boldsymbol{x}_{i+1} + \frac{1}{2}\beta_{i+1}\boldsymbol{s}_\theta^*(\boldsymbol{x}_{i+1}, i+1) \tag{2.43}$$

This ODE can then either be solved iteratively or by using any ODE solver method (e.g. Runge-Kutta).

Chapter 3

Experiment

3.1 Training data

The data used for training was generated by Fluri et al. 2019 using N-body simulations run with PkdGrav3 (J. G. Stadel 2001). The simulations assumed a flat ΛCDM universe and generated particle positions in 3D boxes for 57 different cosmologies which only differ in the choice of Ω_m and σ_8. The remaining parameters were chosen as $\Omega_b = 0.0493$, $H_0 = 67.36$ and $n_s = 0.9649$ which correspond to the results obtained by Planck 2018 (Collaboration et al. 2020). The convergence maps were then produced with UFalcon (Sgier et al. 2021) by projecting the particle densities along the line of sight, according to a lensing kernel to model the effects of gravitational lensing. For training, we then split the data into training and test cosmologies, the choice of which was identical to Perraudin et al. 2020 and can be seen in Fig. 3.1. Due to each cosmology having 12 N-body runs and each run producing 1000 cut-outs from the full sky, the training data consisted of 550k images.

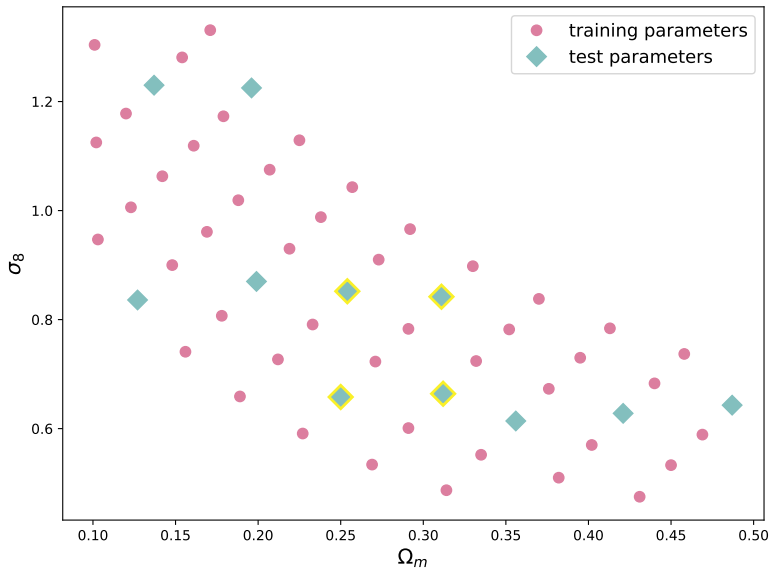

Figure 3.1: Above one can see the parameter space of the cosmologies generated by the N-body simulation. The blue diamonds are the cosmologies which were not used for training and thus can be used as test data. The diamonds which have a yellow glow are the cosmologies which we focus on in our analysis and are labelled in Tab. 3.1.

Before training, the data first underwent some preprocessing. Since the convergence maps

	Ω_m	σ_8
A	0.250	0.658
B	0.254	0.862
C	0.312	0.664
D	0.311	0.842

Table 3.1: Acronyms for the four test cosmologies.

range from regions of low density to extremely high density a key idea was to first normalise the data by taking the logarithm. This reduces the dynamic range of the data and thus makes it easier to learn. After this, we tried two different approaches. Once we scaled the data to lie exactly in the range $[-1, 1]$ and once we standardised the data to have a mean of 0 and a standard deviation of 1.

3.2 Neural networks

3.2.1 Baseline: CWGAN

We used the publicly available[1] Conditional Wasserstein GAN (CWGAN) from Perraudin et al. 2020 as the baseline for our DDPM. The architecture of the GAN consists of a generator with three linear and five deconvolution modules and a discriminator with five convolution and four linear modules. They make use of a Wasserstein loss functions (which helps with the issue of mode collapse), RMSProp as an optimiser, a batch size of 64 and trained the network for around 170 hours on a GeForce GTX 1080 GPU. The conditionality was implemented by rescaling the latent vector with the following function

$$\hat{z} = f(z, y) = \left(l_0 + \frac{l_1 - l_0}{b - a}(y - a) \right) \frac{z}{\|z\|_2} \tag{3.1}$$

Here $y \in [a, b]$ is the parameter vector, in our case $\Omega_M \in [0.1, 0.5]$ and $\sigma_8 \in [0.4, 1.4]$, and $\ell_0 = 0.1\sqrt{n}$ and $\ell_1 = \sqrt{n}$ with n being the size of the latent vector. This function is applied to the latent vector before it is fed into the generator and was created with the idea to make the conditioning continuous. The generator of the CWGAN then learns to spot the differences in input (scaling of the latent vector) and extrapolates that to differences in output (different cosmologies). The discriminator in the training process is also always fed both the fake/real image and concatenates that with the fake/real parameter vector, giving the generator more incentive to produce mass maps conditional on the cosmological parameters assigned.

3.2.2 Conditional DDPM

We built the diffusion model based on a tutorial[2] with a UNet (Ronneberger, Fischer, and Brox 2015) with 5 downsampling and 5 upsampling layer, skip connections and sinusoidal position embedding (Vaswani et al. 2017). The skip connections allow for the full information to propagate unaltered and the position embedding allows the network to distinguish between the different time-steps or noise levels. This is done by using a sine and cosine

[1] `https://renkulab.io/gitlab/nathanael.perraudin/darkmattergan`

[2] `https://colab.research.google.com/drive/1sjy9odlSSy0RBVgMTgP7s99NXsqglsUL?usp=sharing#scrollTo=2fUPyJghdoUA`

positional embedding

$$PE(\mathrm{t}, 2i) = \sin\left(\frac{\mathrm{t}}{n^{2i/d}}\right) \tag{3.2}$$

$$PE(\mathrm{t}, 2i+1) = \cos\left(\frac{\mathrm{t}}{n^{2i/d}}\right) \tag{3.3}$$

where d is the dimension of the input embedding space which we chose to be $d = 32$ and t is the current time-step. Furthermore, n is a scalar which Vaswani et al. 2017 chose to be $n = 10000$ and i is the index $0 \leq i < d/2$ used to map to a column within the embedding matrix. The sine and cosine positional embeddings, together spanning the t-th row of the embedding matrix, are then concatenated and added to every block. This allows each time-step to be encoded onto a unique sinusoidal wave while simultaneously allowing the network to learn the relative position of one time-step to another. A visualisation of the position embedding of the first 100 time-steps in our DDPM can be seen in 3.2. Much more simply, we implemented the conditionality by embedding the labels into the network in every block. More specifically, we conditioned the neural network to generate images based on the choice of Ω_m and σ_8. A depiction of the full architecture of the neural network can

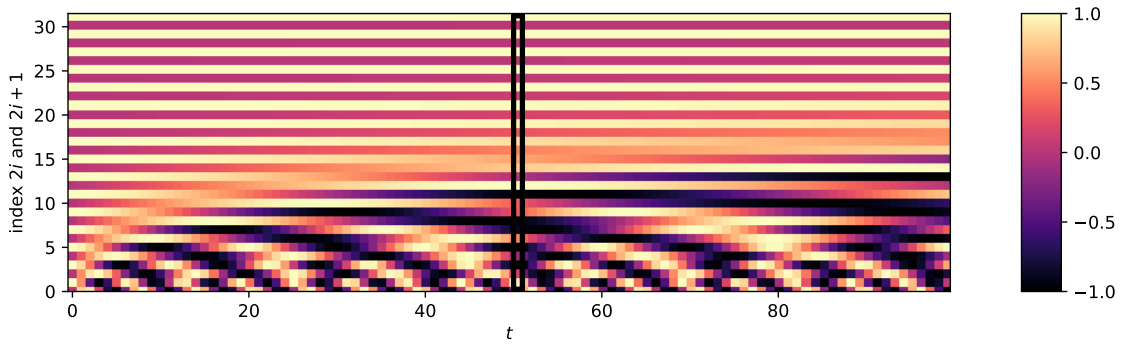

Figure 3.2: PE depicted for the first 100 time-steps of our DDPM. On the x-axis the time-steps are visible and on the y-axis one can see all indices up to the dimension of the position embedding, which we chose to be $d = 32$. The black box outlines the positional embedding of a single time-step, in this case $t = 50$. Extending the plot to $t = T$ would then show the entire PE-matrix.

be seen in Fig. 3.3. Each block consists of two convolutions with kernel size $k = 3$, stride $s = 1$ and padding $p = 1$ with ReLU activation functions, where in the first convolution the number of channels is doubled. In between, the labels as well as the time-steps are embedded (also with ReLU activation). Finally a down-/upsampling step is performed using a convolution with kernel size $k = 4$, stride $s = 2$ and padding $p = 1$ where the image size is quartered. The upsampling process makes use of the skip connections where information can flow unhindered.

During training we used a batch size of 40 and an L2 loss (see Eq. 2.34) as well as an Adam optimiser (Kingma and Ba 2017) with a learning rate of 10^{-3}. The models were trained with an NVIDIA Tesla P100 from the Swiss National Supercomputing Centre (CSCS) for 60k - 120k steps (18-36 hours), depending on the model, which is significantly shorter than the training for the CWGAN. Since the loss function converged fairly quickly and is not a direct measure of how well the DDPM is generating mass maps, we had to hand-pick the best checkpoints, a process also done by Ho, Jain, and Abbeel 2020. Every 500 steps five samples were drawn and both the pixel value distribution as well as the angular power-spectrum were calculated, which informed our decision on which checkpoints to further investigate.

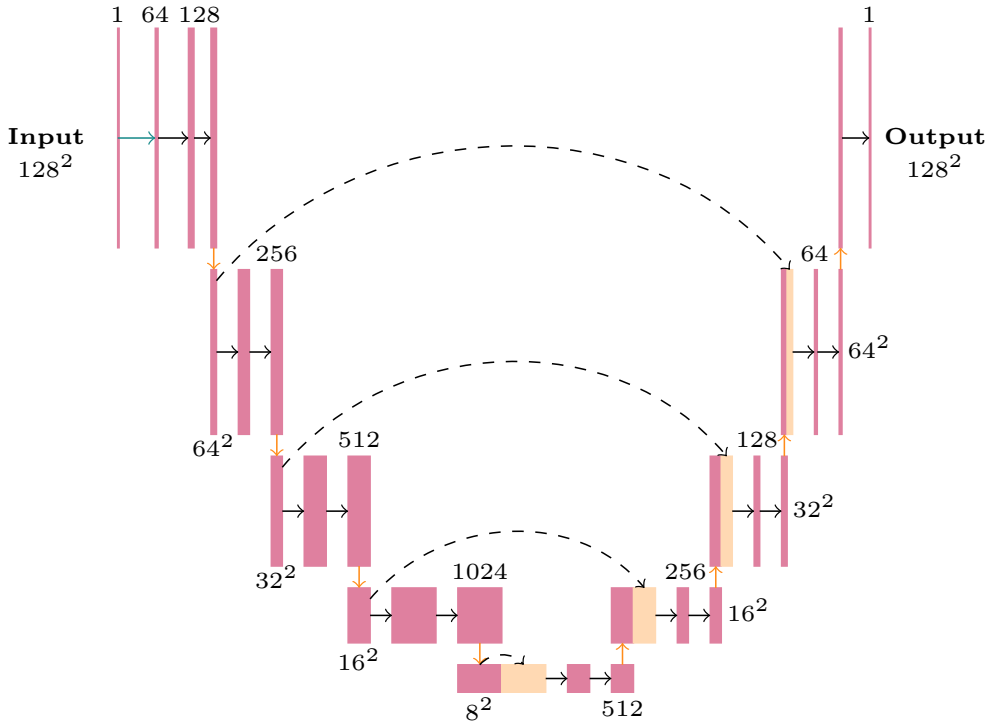

Figure 3.3: A depiction of the UNet used in the DDPM. The black arrows portray the 3×3 convolutions whereas the orange arrows refer to the down and upsampling process. In between the two black arrows both label and time-step embeddings are performed. The dashed black arrows visualise the skip connections and the orange blocks symbolise the data that has been added/brought forward by the skip connection. The squared numbers give insight to the size of the image at each step and the number above the blocks shows the number of channels at that stage.

Hyper-parameters

We discuss six different models which trained sufficiently well, an overview of which can be seen in Table 3.2. One of the three key parameters adjusted is the beta scheduler. Once we chose a linear beta scheduler with $T = 1000$ time-steps, $\beta_0 = 0.02$ and $\beta_T = 0.0001$ mirroring the choice of Ho, Jain, and Abbeel 2020. This choice of beta values is equivalent to doubling the amount of time-steps and halving the initial and final beta values $\beta_0 = 0.01$ and $\beta_T = 0.00005$. We would expect more time-steps to increase the accuracy of the DDPM and thus chose this as our second beta scheduler. Finally we also took a look at a novel beta scheduler which relies on linearly increasing the logged signal-to-noise ratio (Kingma, Salimans, et al. 2022). The SNR scheduler was adjusted such that the initial and final beta values would match up with the linear beta scheduler. A comparison between the two noise schedulers can be seen in Fig. 3.4. Another key aspect which we wanted to test is the impact of the depth of the UNet. For this reason, we compare three slightly different architectures of the UNet. Once we train the NN with a 5 layer UNet (shown in Fig. 2.7) and once with 4 and 3 layers only. Finally, as mentioned previously, we also compared a model trained on standardised versus scaled training data.

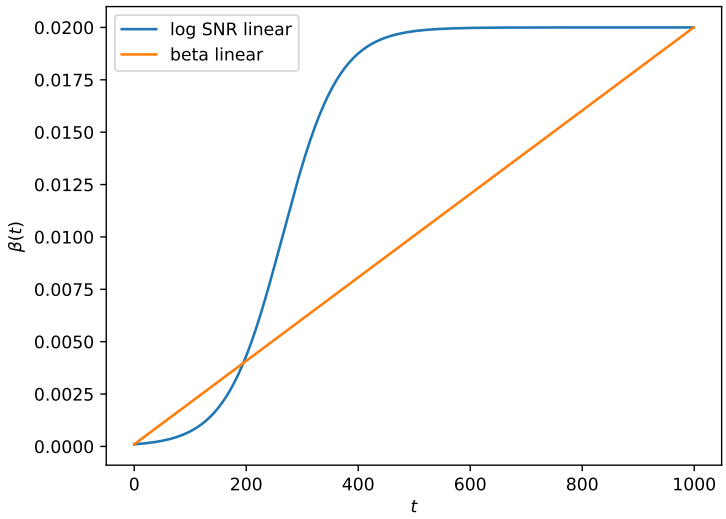

Figure 3.4: Both the linear beta scheduler as well as the SNR scheduler are shown in this plot. $T = 1000$ time-steps were chosen and the initial and final beta values were kept fixed at $\beta_0 = 0.02$ and $\beta_t = 0.0001$

Name	Preprocessing	Beta Scheduler	Architecture	# Trainable Parameters
N1000-5L	scaled	1000 linear	5-layer UNet	62'431'873
S1000-5L	standardised	1000 linear	5-layer UNet	62'431'873
S2000-5L	standardised	2000 linear	5-layer UNet	62'431'873
SSNR-5L	standardised	SNR	5-layer UNet	62'431'873
S2000-4L	standardised	2000 linear	4-layer UNet	11'442'305
S2000-3L	standardised	2000 linear	3-layer UNet	3'689'601

Table 3.2: Six different models with their key differences are shown in the table above.

3.3 Diagnostics

When judging the quality of natural images such as face one usually can do so quite well by eye. In our case we go a step further and use a multitude of summary statistics to effectively judge the quality of the generated images, a process which is not always possible with natural images. One of the first tests performed was comparing the pixel value distribution $N_{\text{pixels}}(\kappa)$ of the generated images to the test data not used in training. This already gives us insight into the NNs capabilities to learn the distribution of the data $P_{\text{data}}(x)$. We are not only interested in the values of the pixels but also how the pixel values are distributed throughout the image. Thus we also calculated the angular power-spectrum C_ℓ and the bispectrum B_ℓ. Consider and observable $\mathcal{O}$, which we can expand in terms of spherical harmonics $\mathcal{O} = \sum_{\ell=0}^{\infty}\sum_{m=-\ell}^{\ell}\mathcal{O}_{\ell m}Y_{\ell m}$ with the amplitude $\mathcal{O}_{\ell m}$. In cosmology, the angular power-spectrum is then defined as the variance of the amplitude $\mathcal{O}_{\ell m}$ of that observable $\langle\mathcal{O}_{\ell m}\mathcal{O}^*_{\ell' m'}\rangle = \delta_{\ell\ell'}\delta_{mm'}C_\ell$. The bispectrum would then be the 3-pt correlation function $\langle\mathcal{O}_{\ell m}\mathcal{O}_{\ell' m'}\mathcal{O}_{\ell'' m''}\rangle = \mathcal{G}^{\ell\ell'\ell''}_{mm'm''}B_{\ell\ell'\ell''}$, where $\mathcal{G}^{\ell\ell'\ell''}_{mm'm''}$ is the Gaunt integral.

Another tool to evaluate the morphology of large-scale cosmological distributions are the Minkowski functionals (Schmalzing, Kerscher, and Buchert 1995). More specifically, we are interested in the three morphological descriptors V_0, V_1 and V_2. All three describe a certain aspect of the pixels of an image above a certain threshold. V_0 describes the area of the

emerging islands, V_1 the circumference and V_2 gives the value for the Euler characteristic χ (number of emerging islands minus the number of isolated holes):

$$V_0(\kappa) = N_{\text{pixels}}(> \kappa) \tag{3.4}$$
$$V_1(\kappa) = -4N_{\text{pixels}}(> \kappa) + 2E(> \kappa) \tag{3.5}$$
$$V_2(\kappa) \equiv \chi(\kappa) = N_{\text{pixels}}(> \kappa) - E(> \kappa) + V(> \kappa) \tag{3.6}$$

where $E(> \kappa)$ are the edges of the emerging islands and $V(> \kappa)$ the vertices. To compare all these statistics to the test data, we used the fractional difference

$$f = \frac{x_{\text{data}} - x_{\text{sample}}}{x_{\text{data}}} \tag{3.7}$$

with a preferred value below 5%. Furthermore, the Wasserstein-1 distance (Kantorovich 1960) allows us to compare the distribution of the pixels within the generated images to the data and was thus also used as a diagnostic tool.

$$W_1(\mu_{\text{data}}, \mu_{\text{sample}}) = \int_{\mathbb{R}} |F_{\text{data}}(x) - F_{\text{sample}}(x)| dx \tag{3.8}$$

Chapter 4

Results

After completing training on all models mentioned in Table 3.2 we found that the model with standardised data, a linear beta scheduler with 2000 time-steps and a three layer UNet produced the most favourable checkpoint. Later in this chapter (Sec. 4.2) we examine the effectiveness of the other models. In Fig. 4.1 one can see four samples generated with the S2000-3L DDPM, alongside images drawn from the test data for comparative purposes, for four different test cosmologies. By just looking at the images, it is impossible to tell which one was generated by the DDPM and which stems from the test data. Furthermore, qualitatively the DDPM is able to predict the visual structures of the different cosmologies, signifying that it has successfully conditioned itself on the parameters Ω_m and σ_8. In order to compute the summary statistics we generated 100 samples for each of the four test cosmologies illustrated in Fig. 3.1. The resulting angular power-spectrum, bispectrum and the pixel value distribution, along with the fractional difference of the DDPM samples and the N-Body test data, are depicted in Figure 4.2. The power-spectrum as well as the bispectrum were calculated using LensTools (Petri 2016). By looking at the top four plots we can see that the power-spectrum is consistent with the test data within $<5\%$ for small and medium scales, however, there is a visible deviation at large scales. In contrast,

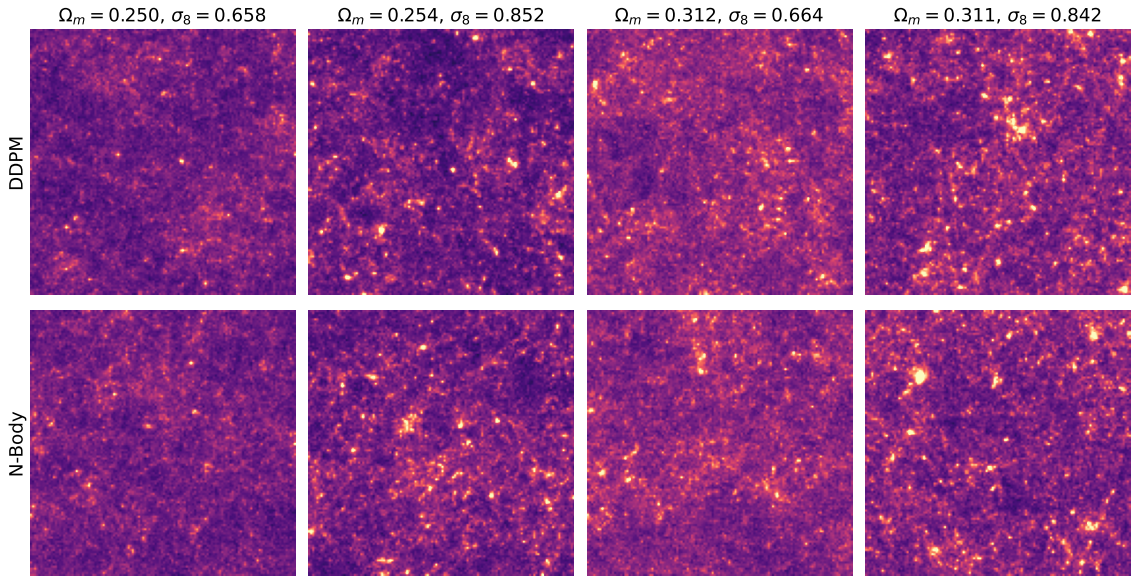

Figure 4.1: Images generated by the **S2000-3L** DDPM for four different cosmologies and the test data (N-Body) for comparison.

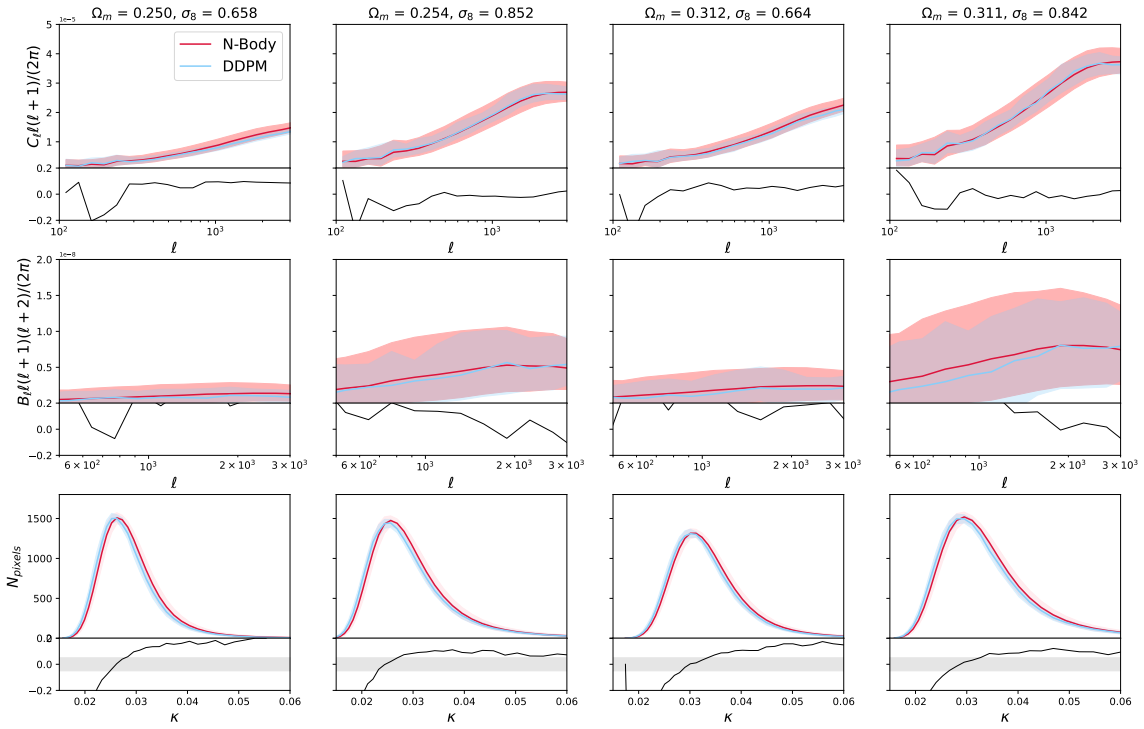

Figure 4.2: Power-spectrum, bispectrum and pixel value distribution of both the N-Body data and samples from the **S2000-3L** DDPM. The blue and red bands correspond to the 68% confidence limit. Below the plots the fractional difference between the N-Body data and the samples is shown in black where the grey bar corresponds to a <5% deviation.

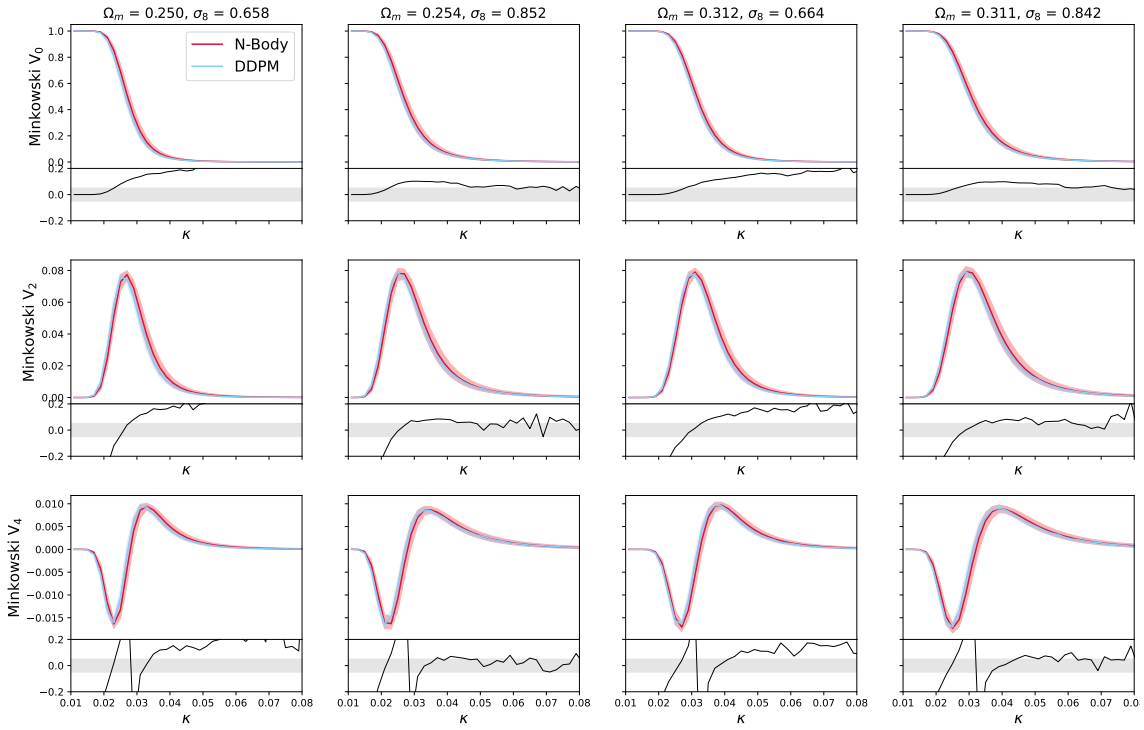

Figure 4.3: All three Minkowski functionals of both the N-Body data and samples from the **S2000-3L** DDPM. The configurations are the same as in Fig. 4.2

the median of bispectrum does not follow the test data well with a fractional difference that is often above 20%. Nevertheless the median still lies within the 68% confidence interval.

The pixel value distribution leaves room for improvement as it does not follow the test data perfectly, but instead displays a fractional difference of approximately 10%. Here it is important to note that when aligning the mean of the convergence κ of the samples with the dataset we observe agreements within $<5\%$, as we can see in Fig. 4.4. While it would be preferable for the DDPM to accurately estimate the pixel value distribution, from an observational point of view the mean of the convergence is irrelevant since one observes shear, from which one can only find a degenerate convergence with an added constant.

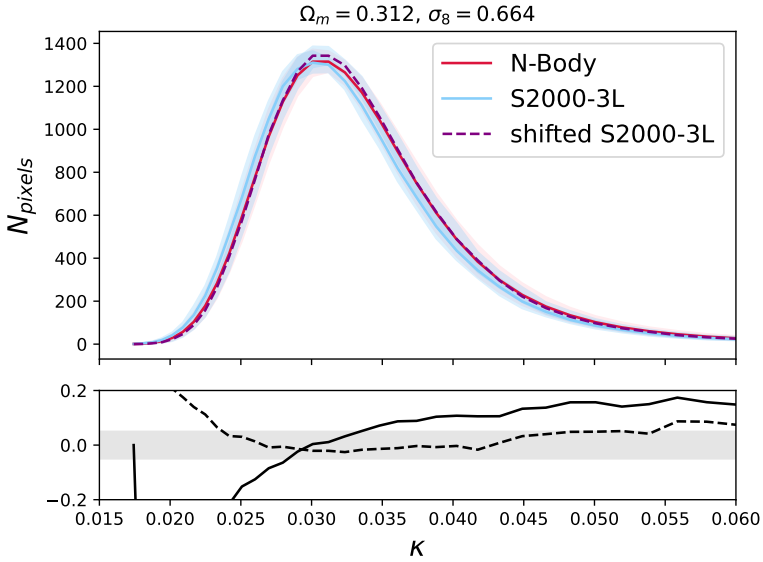

Figure 4.4: Pixel value distribution of the N-Body data and samples from the **S2000-3L** DDPM once unchanged and once aligned with the mean of the N-Body data. The blue, red and purple bands correspond to the 68% confidence limit. Below the plot the fractional difference between the N-Body data and the samples is shown in black where the grey bar corresponds to a $<5\%$ deviation.

In Fig. 4.3 one can see the three Minkowski functionals for the same four test cosmologies. To calculate these functionals, we utilised the LensTools package once again. Upon examining the first functional V_0 we can observe that the model predicts the area of emerging islands with reasonable accuracy, where in two cosmologies we see a $<5\%$ fractional difference. However, the cosmologies A and C are not being fit very well. We can see a similar trend for the circumference V_1 and Euler characteristics V_2, where once again the cosmologies B and D follow the test data well, whereas the other two cosmologies exhibit a deviation. Upon revisiting Fig. 4.1 we can see that the generated cosmologies that more accurately match the test data seem to be more clustered. This could be a reason why the neural network has an easier time learning the structure of those images.

To further analyse the results, we evaluated the Wasserstein-1 distance for 20 samples of each set of cosmological parameters. Prior to calculating it we standardised the dataset and the samples (to have a mean of $\mu = 0$ and variance of $\sigma = 1$) with respect to the training data for each set of cosmological parameters, equivalently to Perraudin et al. 2020. This was done due to the Wasserstein-1 distance being a scale-dependent measure. In this case a 1σ shift of the mean would lead to a Wasserstein-1 distance of $W_1 = 1$ and a doubling of the variance would correspond to $W_1 \approx 0.8$ (Perraudin et al. 2020). In Fig. 4.5 one can see the W_1 values as well as absolute fractional difference of the power-spectrum of all parameters. At first glance it is apparent that the DDPM has difficulty extending to low matter density Ω_m. Nevertheless, it is able to explore previously unseen cosmologies

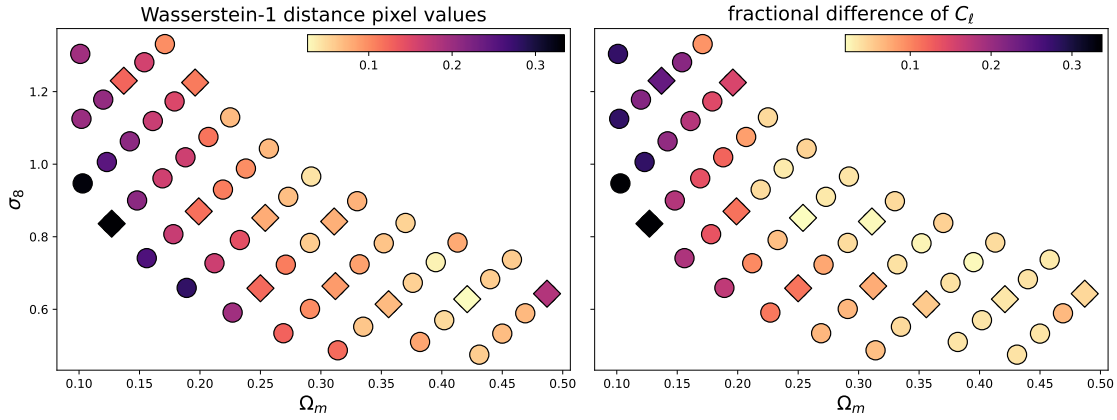

Figure 4.5: (*left*) Wasserstein-1 distance and (*right*) average absolute fractional difference between the median of the power-spectrum (linearly binned) of 20 generated samples and of the test/training N-Body data for all available sets of cosmological parameters.

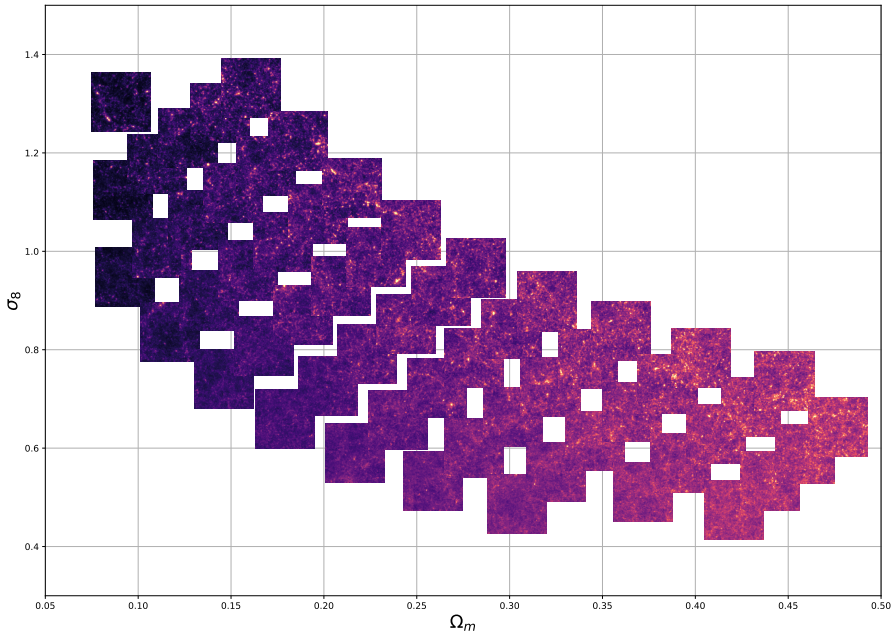

Figure 4.6: N-Body images of different cosmologies, with their position on the plot providing information on their cosmological parameters Ω_m and σ_8. An equal plot using samples from the DDPM can be seen in Fig. A.4 in Appendix B.

with ease, provided they do not lie at the edge. A visual representation of the different cosmologies can be seen in Fig. 4.6 from which we can tell that the cosmologies which the DDPM has difficulties with are ones which are very dim and sparse in structure.

4.1 Sampling methods

Sampling one image from the S2000-3L DDPM takes roughly 90 seconds, which is substantially shorter than running an N-Body simulation. Nevertheless it still takes longer compared to a GAN, which can generate hundreds of images in seconds. In this section we examine a few different sampling approaches applied to our trained DDPM: ancestral sampling, reverse-time SDE sampling, discretised ODE sampling and solving the ODE

with a blackbox solver by Y. Song et al. 2021[1]. We note, that the method of ancestral sampling is the standard sampling method which was also used in all other instances in this thesis and acts as a benchmark in this section. To provide a visual comparison, Fig. 4.7 displays the power-spectrum and pixel value distribution for 20 samples for each of the four sampling methods. At the top of the figure one can see four images, each generated with the same initial conditions. Both the images generated with ancestral and SDE sampling look different to the others which is due to the stochastic nature of the sampling process. Conversely, the ODE and blackbox ODE sampler (Runge-Kutta of order 5) produce images which resemble each other, as the sampling process is deterministic in nature. While the specific cosmology investigated here has the parameters of D, similar behaviours were also observed for the other test cosmologies. Upon examining the summary statistics we see that the reverse-time SDE sampling performs similarly to ancestral sampling, as predicted by Eq. 2.40. However, the discretised as well as blackbox ODE solver have difficulties generating outputs which follow the test data. It is worth highlighting that the blackbox ODE solver produces one image within 13 seconds which is is significantly shorter than all other three sampling methods (for which it takes 90 seconds). Hence to prioritise accuracy, one would choose the ancestral sampling method, while the blackbox ODE sampler is preferable for those prioritising speed.

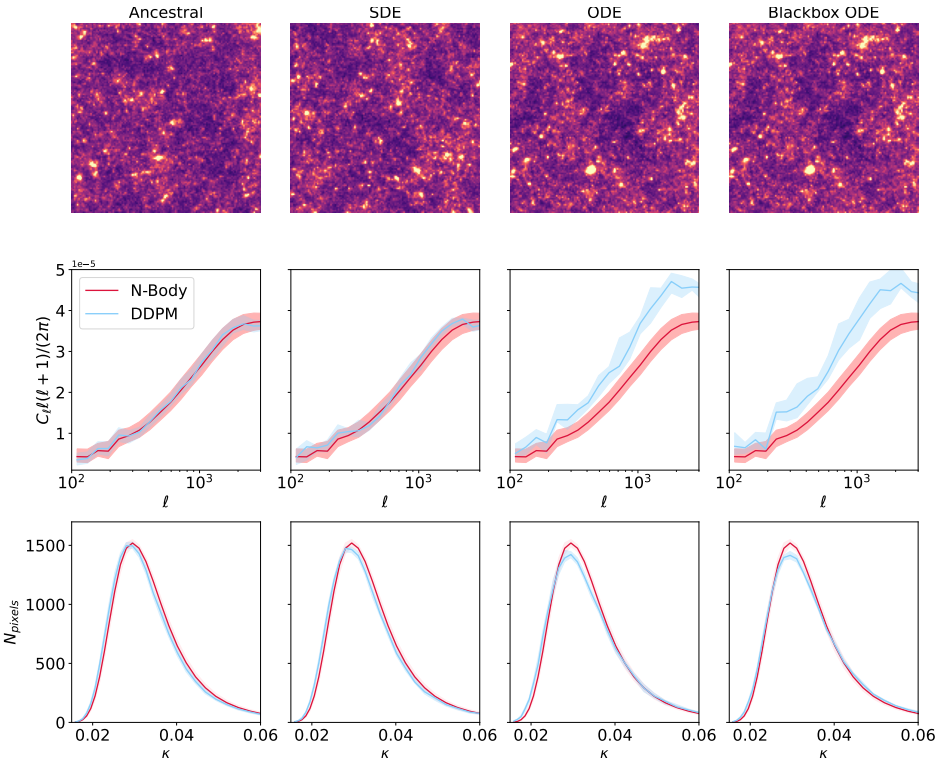

Figure 4.7: Different sampling approaches applied to the S2000-3L DDPM with cosmological C. At the top four images generated with the same initial conditions and below the power-spectrum as well as the pixel value distribution are depicted.

4.2 Impact of hyper-parameters

4.2.1 Standardised vs scaled

In this section, we present a comparison between two preprocessing techniques: standardising the training data to have a mean of 0 and a standard deviation of 1, and scaling

[1] `https://github.com/yang-song/score_sde`

the training data to lie within the range of $[-1, 1]$. In Ho, Jain, and Abbeel 2020 they also use a scaling approach where they scale images with integers in $[0, 255]$ to lie between $[-1, 1]$. Nevertheless, here we are not dealing with traditional images but rather density maps and hence this is not necessarily the optimal approach. In Fig. 4.8 one can see some summary statistics for both the N1000-5L DDPM and S1000-5L DDPM respectively. The model trained on the scaled data fails to match both the power-spectrum as well as the bispectrum. Conversely, training on standardised data shows better results on the power-spectrum, especially on small scales. On large scales there are some deviations, nonetheless the median of the samples still lie within the error-bars of the test data, with the exception of the cosmology with parameters A. The median of the bispectrum does not follow the test data very well, however still remains largely within the error-bars. This could be a result of the large scatter of the bispectrum as well as the limited amount of samples drawn. Furthermore, the pixel value distribution is matched significantly better by the model trained on standardised data versus scaled data, especially samples drawn for the cosmology with parameters D which exhibit a fractional difference below 5%. Similarly, the Minkowski functionals for the S1000-5L DDPM outperform the N1000-5L DDPM, as seen in Fig. A.1 in Appendix A.

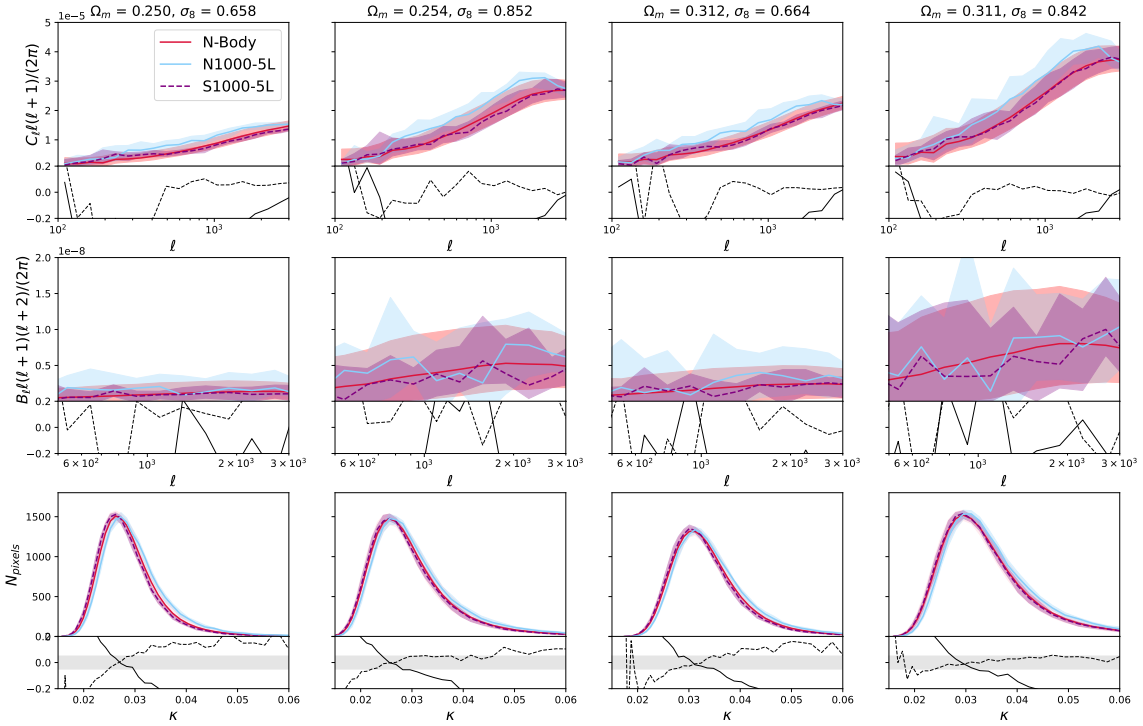

Figure 4.8: Power-spectrum, bispectrum and pixel value distribution of both the N-Body data and 20 samples from the **N1000-5L** DDPM and **S1000-5L** DDPM. The blue, red and purple bands correspond to the 68% confidence limit. Below the plots the fractional difference between the N-Body data and the samples is shown in black where the grey bar corresponds to a <5% deviation.

4.2.2 Beta scheduler

In addition to preprocessing, another important hyper-parameter of a DDPM is the beta- or noise-scheduler. Ho, Jain, and Abbeel 2020 use a beta scheduler with $T = 1000$ time-steps, $\beta_0 = 0.02$ and $\beta_T = 0.0001$. However, as mentioned in the Sec. 2.3.3 increasing the number of time-steps $T \to \infty$ results in $\boldsymbol{x}_T$ matching a Gaussian distribution more closely. Therefore, it could be preferable to use a larger number of time-steps. In our

case we chose to double the amount of time-steps to $T' = 2000$ and in order to keep the amount of noise added constant we halved the initial and final noise level to $\beta'_0 = 0.01$ and $\beta'_T = 0.00005$. In addition we also tested the SNR-noise scheduler as described in Sec. 4.2. The outcomes obtained from the S2000-5L DDPM and SSNR-5L DDPM are illustrated in Fig. 4.9. Concerning the power- and bispectrum it is unclear if the increased amount of time-steps or the SNR noise scheduler had an impact. Nevertheless, it can be argued that the SNR noise scheduler is able to match the power-spectrum to an equal degree on large as on small scales, unlike both of the linear beta scheduler. On the other hand, the pixel value distribution seems to be matched more consistently throughout all four test cosmologies by a model with more time-steps compared to the SNR noise scheduler. Similarly, the Minkowski functionals, shown in Fig. A.2, for the model with $T = 2000$ also match the test data more reliably compared to a model with a non-linear noise scheduler.

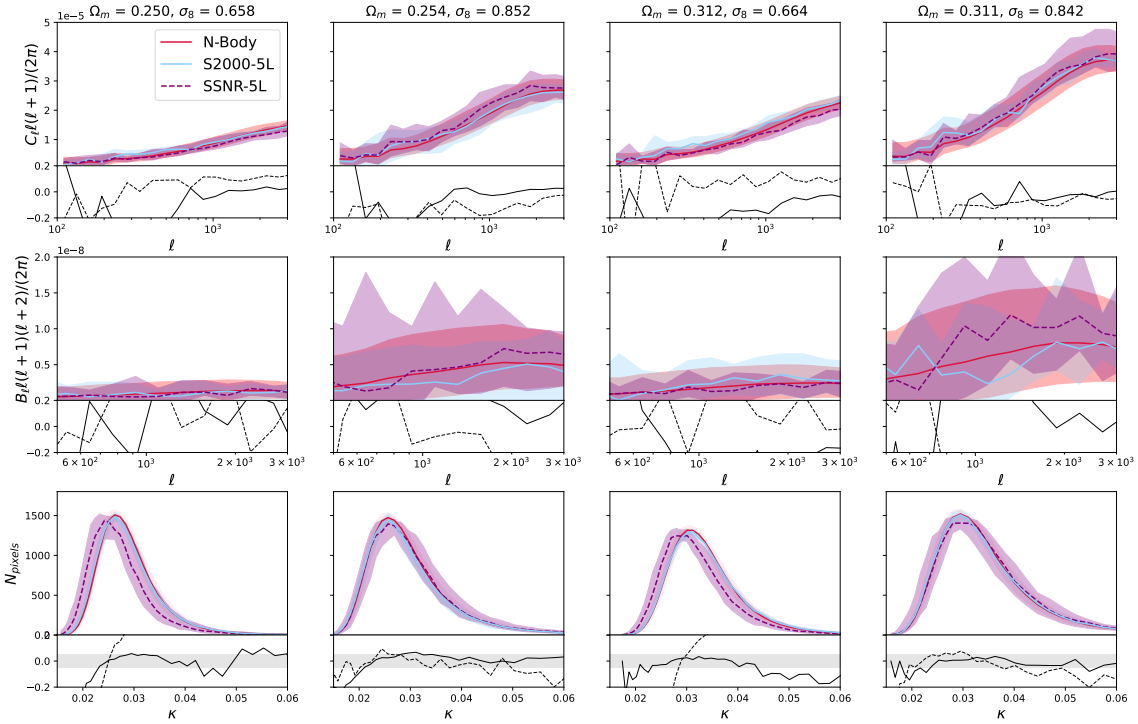

Figure 4.9: Power-spectrum, bispectrum and pixel value distribution of both the N-Body data and 20 samples from the **S2000-5L** DDPM and **SSNR-5L** DDPM. The blue, red and purple bands correspond to the 68% confidence limit. Below the plots the fractional difference between the N-Body data and the samples is shown in black where the grey bar corresponds to a <5% deviation.

4.2.3 Architecture

A final hyper-parameter of interest is the number of UNet layers, and thus the number of trainable parameters. Fig. 4.10 shows the output of a trained DDPM with four layers compared to five layers. For both, the power-spectrum of the generated images has difficulties conforming to the test data, particularly on large scales. This seems to be an issue of the DDPM throughout all choices of hyper-parameters and it is unclear what is causing this excess in power-spectrum on large scales. Nevertheless, four layers seems to expand the accuracy to slightly larger scales than for five layers, while three layers extend to even larger scales (as seen previously in Fig. 4.2). Conversely, more layers enhance the accuracy of the pixel value distribution as well as the Minkowski functionals (see Fig. A.3). Finally, the bispectrum performs equally poorly for all three models but still remains largely within the error-bars.

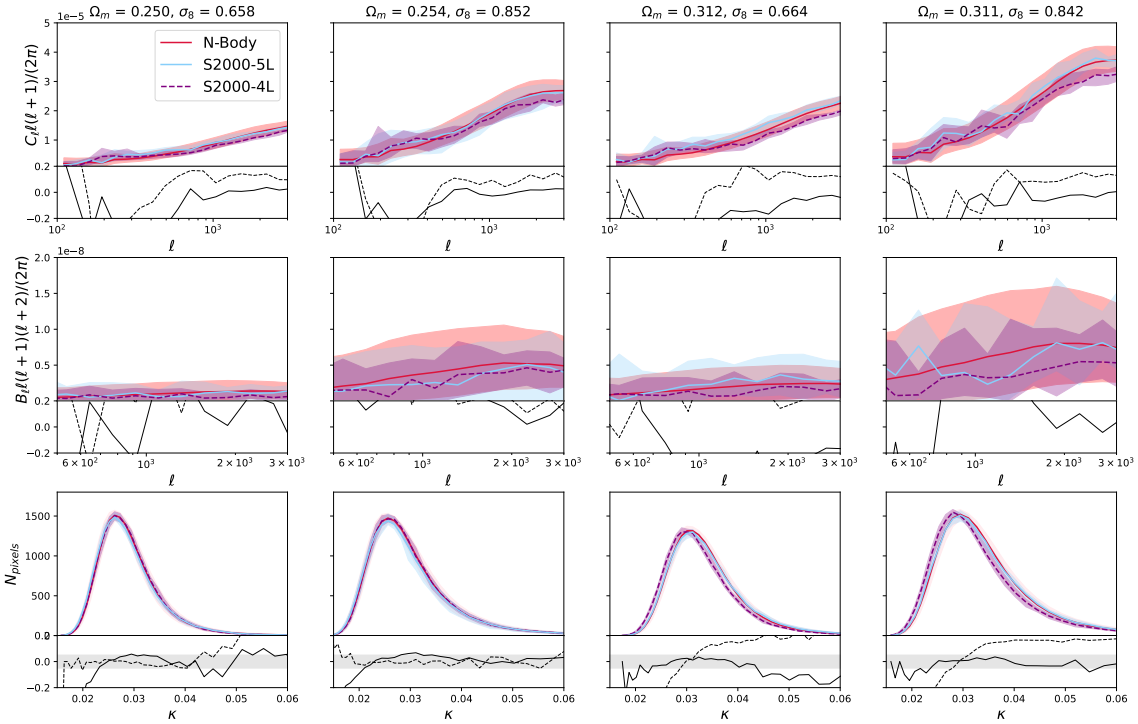

Figure 4.10: Power-spectrum, bispectrum and pixel value distribution of both the N-Body data and 20 samples from the **S2000-4L** DDPM and **S2000-5L** DDPM. The blue, red and purple bands correspond to the 68% confidence limit. Below the plots the fractional difference between the N-Body data and the samples is shown in black where the grey bar corresponds to a <5% deviation.

4.3 DDPM vs CWGAN

Finally, we compare the trained S2000-3L DDPM to the results from the CWGAN (Perraudin et al. 2020)[2]. In order to compare the two generative models, we sampled 100 mass maps from the DDPM and 100 mass maps from the CWGAN for each of the four test cosmologies. Upon examination of the power-spectrum, bispectrum and pixel value distribution, as shown in Fig. 4.11, it seems that the CWGAN generates samples of slightly higher accuracy. For the power-spectrum both models follow the test data with equal proficiency for small scales, whereas on large scales the CWGAN outperforms the DDPM by remaining largely within the <5% confidence limit. Once again we observe this excess in the power-spectrum at large scales for the DDPM. The bispectrum seems to also be better fit by the CWGAN, albeit not to the same extent as the power-spectrum. Here it is important to note, that we are not extending to as large scales as for the power-spectrum. Moving on to the pixel value distribution, we notice that the DDPM and CWGAN follow the training data comparably well. We can see a similar pattern in the Minkowski functionals, depicted in Fig. 4.12 where the CWGAN generally deviates less from the test data compared to the DDPM. However, this trend is primarily observed for the cosmology A. Finally, in Fig. 4.13 one can observe the spread of the power-spectrum for the N-Body data, CWGAN and DDPM samples of the four test cosmologies A, B, C and D. As we can see both the DDPM and CWGAN estimate the spread of the data well on large scales. Surprisingly, on small scaler the DDPM then under-predicts the spread whereas the CW-

[2]Unfortunately, we initially worked with data which had slight deviations from the data used to train the CWGAN. It is still not clear to us how these two datasets differ and hence we needed to retrain our model on the new data which is the reason why the following plots do not resemble Fig. 4.2 and Fig. 4.3 exactly.

GAN tends to over-predict it. Improving on this measure is of importance if one wishes to use the generated convergence maps for covariance estimation.

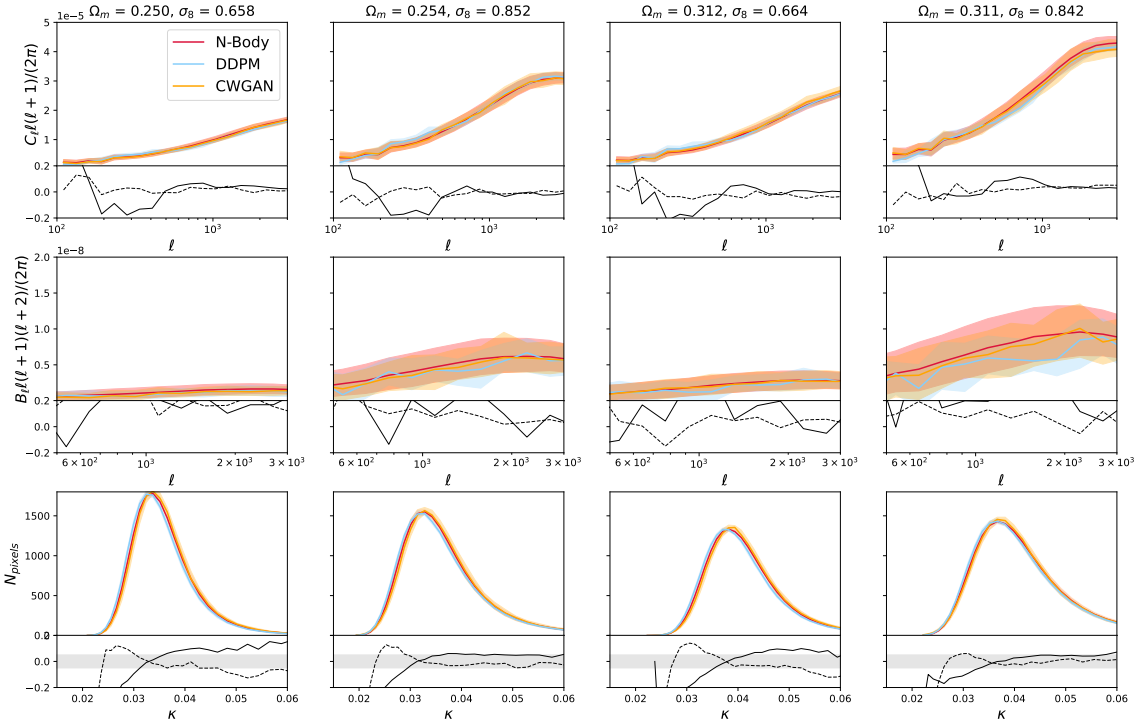

Figure 4.11: Power-spectrum, bispectrum and pixel value distribution of both the N-Body data, CWGAN and **S2000-3L** DDPM samples. The blue, red and orange bands correspond to the 32% and 68% percentiles. Below the plots the fractional difference between the N-Body data and the DDPM/CWGAN samples is shown in the continuous/dashed black line where the grey bar corresponds to a <5% deviation from the test data.

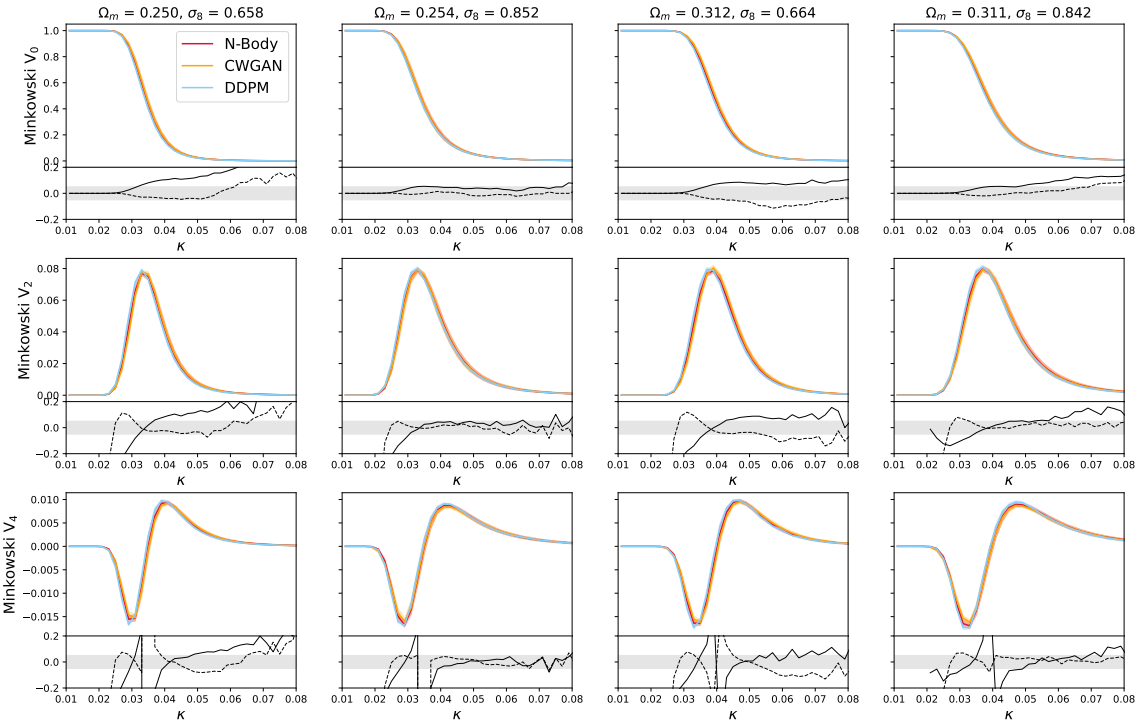

Figure 4.12: All three Minkowski functionals of both the N-Body data, CWGAN samples and samples from the **S2000-3L** DDPM. The configuration of the plot is the same as in Fig. 4.11

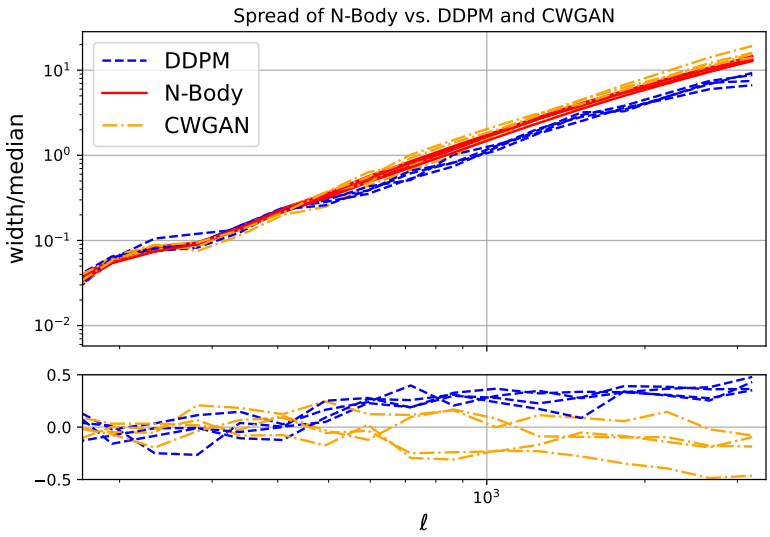

Figure 4.13: The spread (width over median) of power-spectrum of the N-Body data as well as CWGAN and DDPM samples. The bottom plot shows the fractional difference between the N-body spread and DDPM/CWGAN.

Chapter 5

Conclusion

In this thesis we explored the use of a DDPM in generating convergence maps conditioned on the cosmological parameters σ_8 and Ω_m. To establish the CWGAN as a baseline, we chose the dataset and parameter distribution identically to Perraudin et al. 2020. Additionally, we investigate the impact different hyper-parameters and sampling methods have on the quality of the generated images.

We found the model which produced the most favourable checkpoint used standardised training data, a linear beta scheduler with $T = 2000$ time-steps as well as a three layer UNet. Visually, the DDPM is able to successfully produce convergence maps conditioned on σ_8 and Ω_m. Concerning the summary statistics, the power-spectrum of the DDPM matches the N-Body data within the >5% confidence limit for matter densities of $\Omega_m > 0.25$. This trend can also be seen when investigating the Wasserstein-1 distance, where the DDPM is able to approximate cosmologies with larger Ω_m with higher accuracy. Higher order statistics, such as the Minkowski functionals and the bispectrum, have a harder time agreeing with the test data but generally still lie within the 68% confidence limit. When investigating different sampling methods, we found that ancestral sampling approximates the test data most accurately. Nevertheless, the reverse time SDE approach still shows promising results and sampling the data by solving the blackbox ODE increases the sampling speed seven-fold. When investigating the impact of the hyper-parameters we found that for all models the DDPM has difficulties approximating the power-spectrum at large scales. Nevertheless, by decreasing the number of layers in the UNet (and thus decreasing the number of trainable parameters) one can extend the accuracy of the power-spectrum to larger scales. We also found that a linear beta scheduler works best for our setup and that more intermediate time-steps increase the accuracy on all summary statistics. Similarly, training on standardised data outperforms training on scaled data. Finally, when comparing the best checkpoint with the baseline, we observed only marginally better results for the CWGAN.

In conclusion, we showed that the denoising diffusion probabilistic model is a promising new generative tool in cosmology. Due to its stable and easy training as well as high image output quality it has already been shown to outperform GANs (Dhariwal and Nichol 2021) in natural image generation, but in this thesis we hope to have shown that it also demonstrates optimistic results for more complex data such as convergence maps. Additionally, the DDPM we trained was a standard implementation of the model and still allows for further modification. There are many projects (e.g. Hong et al. 2023, J. Song, Meng, and Ermon 2022, Nichol and Dhariwal 2021, Ho, Saharia, et al. 2021) expanding on the architecture of a DDPM and are showing promising results for natural image generation, paving the way for future applications in the field of physics.

Acknowledgement

First and foremost, I would like to thank Prof. Dr. Alexandre Refregier for the opportunity to work on this project. I am also very thankful for the time he dedicated to our weekly meetings which sparked new ideas when stuck and were helpful in providing structure in the six months of this project. I would also like to thank Dr. Tilman Troester and Arne Thomsen for their guidance and help, as well as their patience with my many questions. Their expertise and insights were invaluable in shaping the direction of my thesis. I am also thankful to Dr. Tomasz Kacprzak for providing the data and helping me navigate the CWGAN. A special thank you goes to my fellow master students (the cosmo sprouts), Benjamin, Luca, Luis, Hao and Simon for the many wonderful lunches and for their camaraderie. Finally, I would also like to thank Fabian for his support and understanding during my ups and downs and for being my rock throughout these last six months.

Appendices

A Minkowski functionals for hyper-parameters

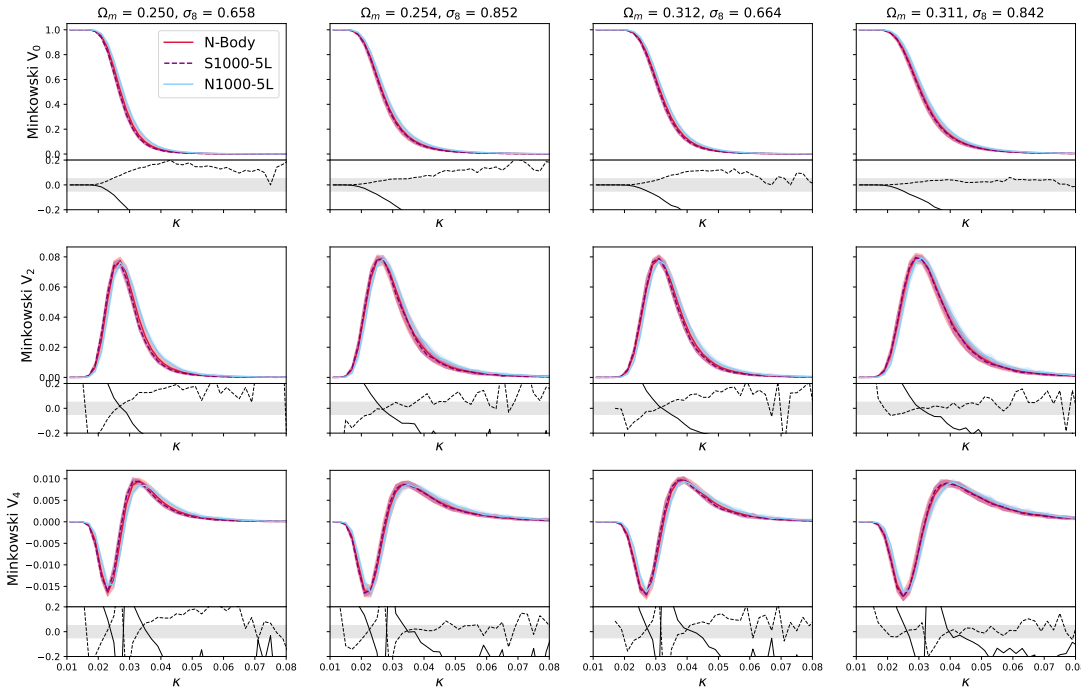

Figure A.1: All three Minkowski functionals of both the N-Body data and 20 samples from the **N1000-5L** DDPM and **S1000-5L** DDPM. The blue, red and purple bands correspond to the 68% confidence limit. Below the plots the fractional difference between the N-Body data and the samples is shown in black where the grey bar corresponds to a <5% deviation.

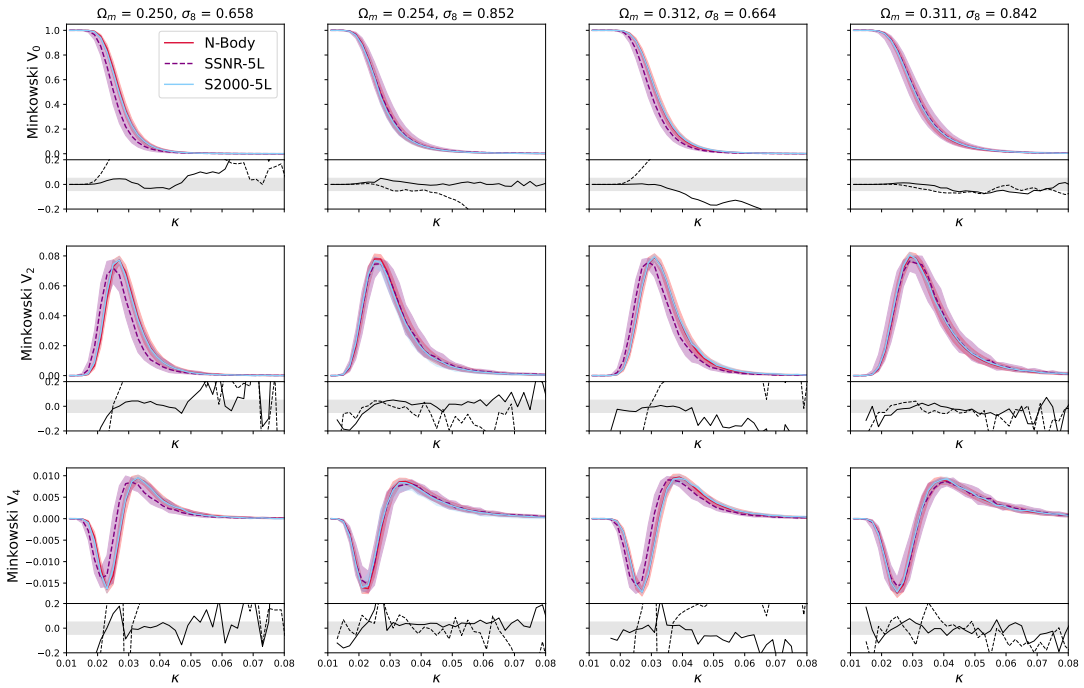

Figure A.2: The same plot structure as in Fig. A.1 but for the **S2000-5L** and **SSNR-5L** DDPM.

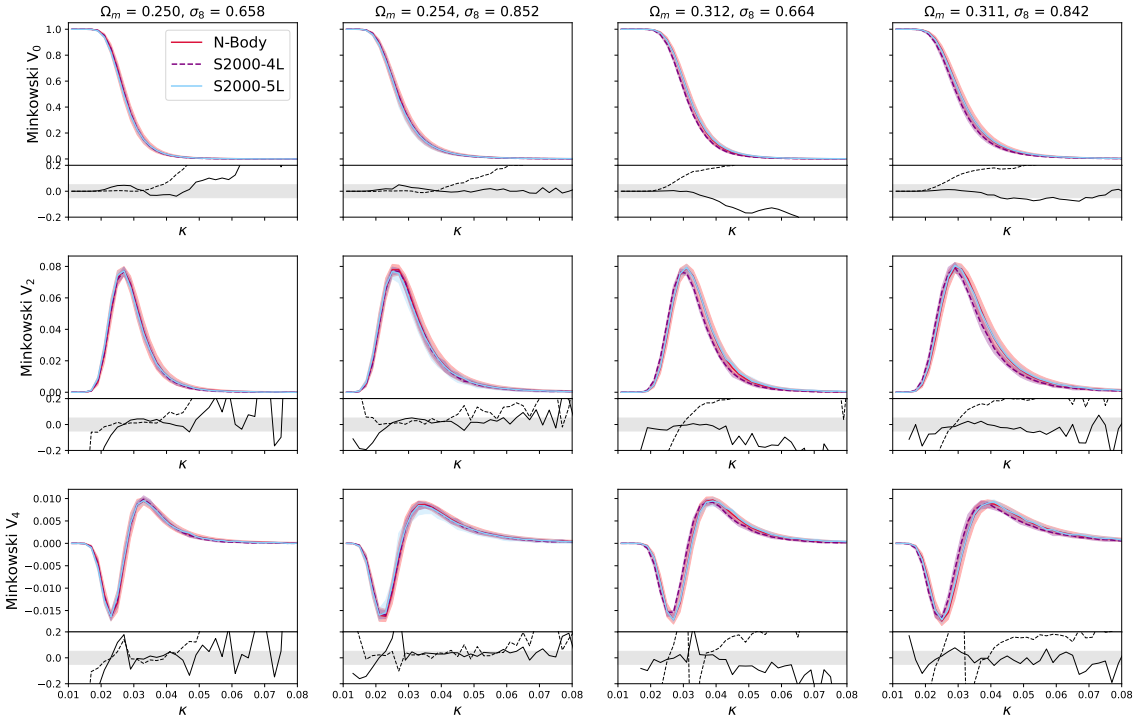

Figure A.3: All three Minkowski functionals of both the N-Body data and 20 samples from the **S2000-4L** DDPM and **S2000-5L** DDPM. The blue, red and purple bands correspond to the 68% confidence limit. Below the plots the fractional difference between the N-Body data and the samples is shown in black where the grey bar corresponds to a <5% deviation.

B Gallery

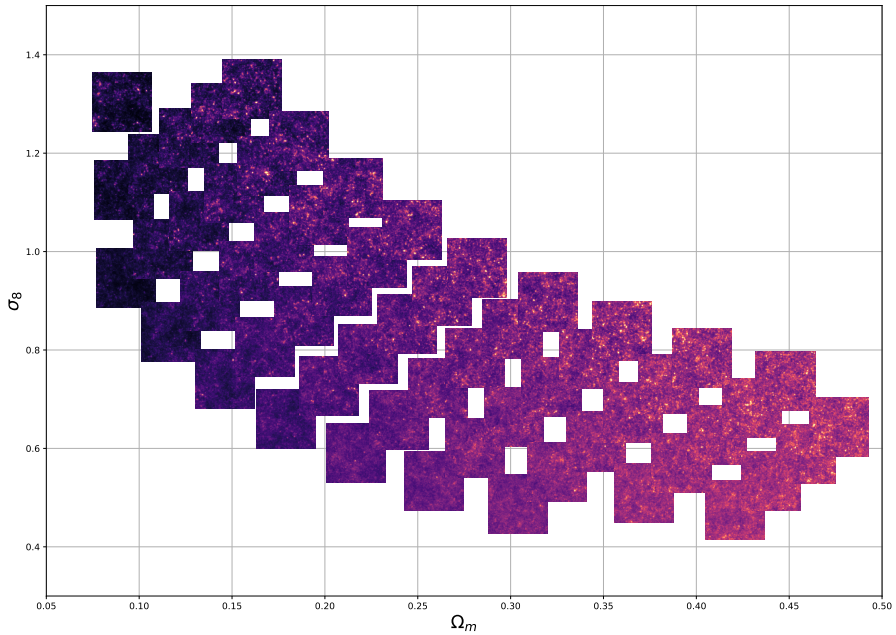

Figure A.4: Images generated by the S2000-3L DDPM for different cosmologies, with their position on the plot providing information on their cosmological parameters Ω_m and σ_8

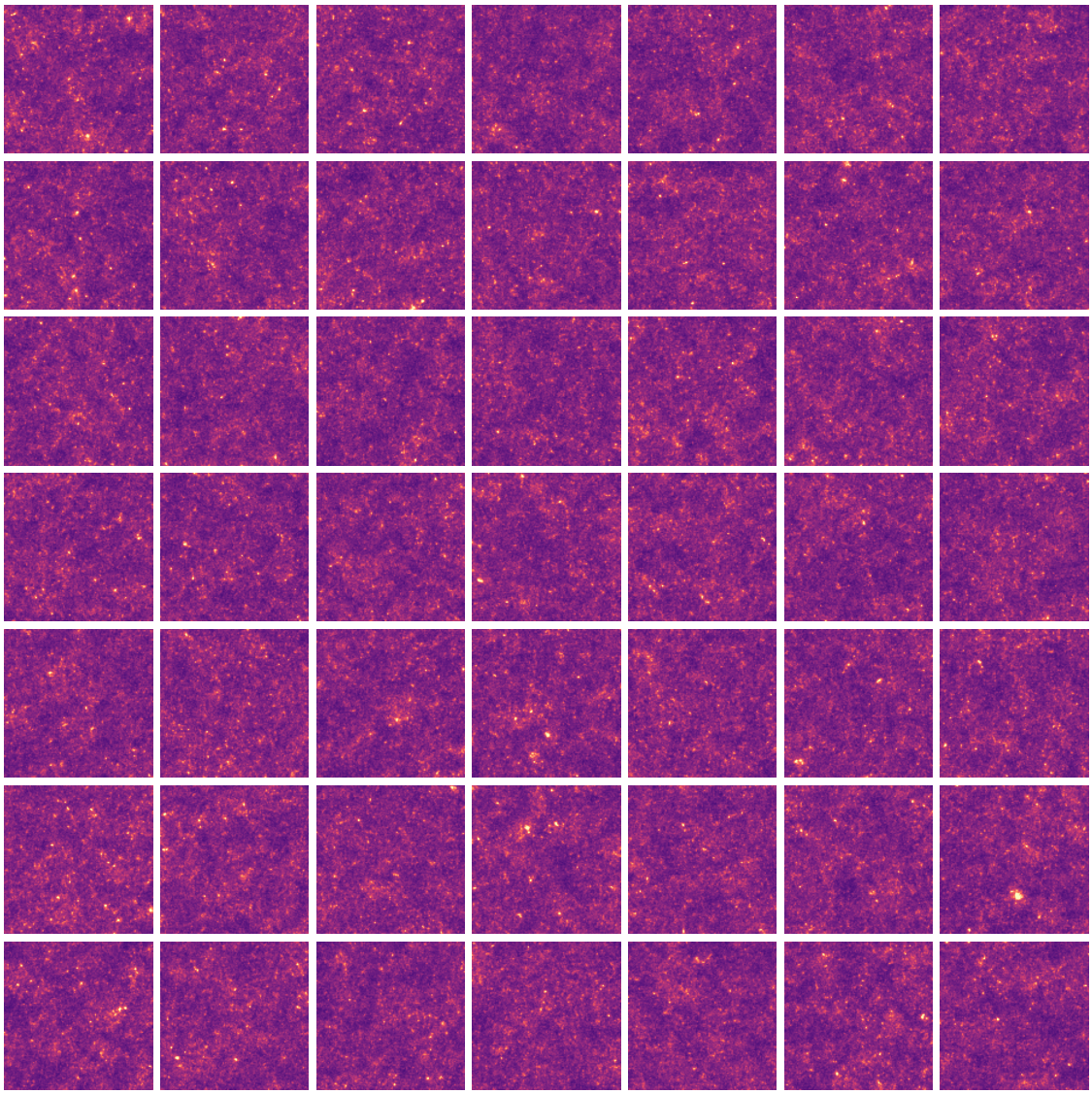

Figure A.5: A grid of images generated by the S2000-3L for test cosmology A.

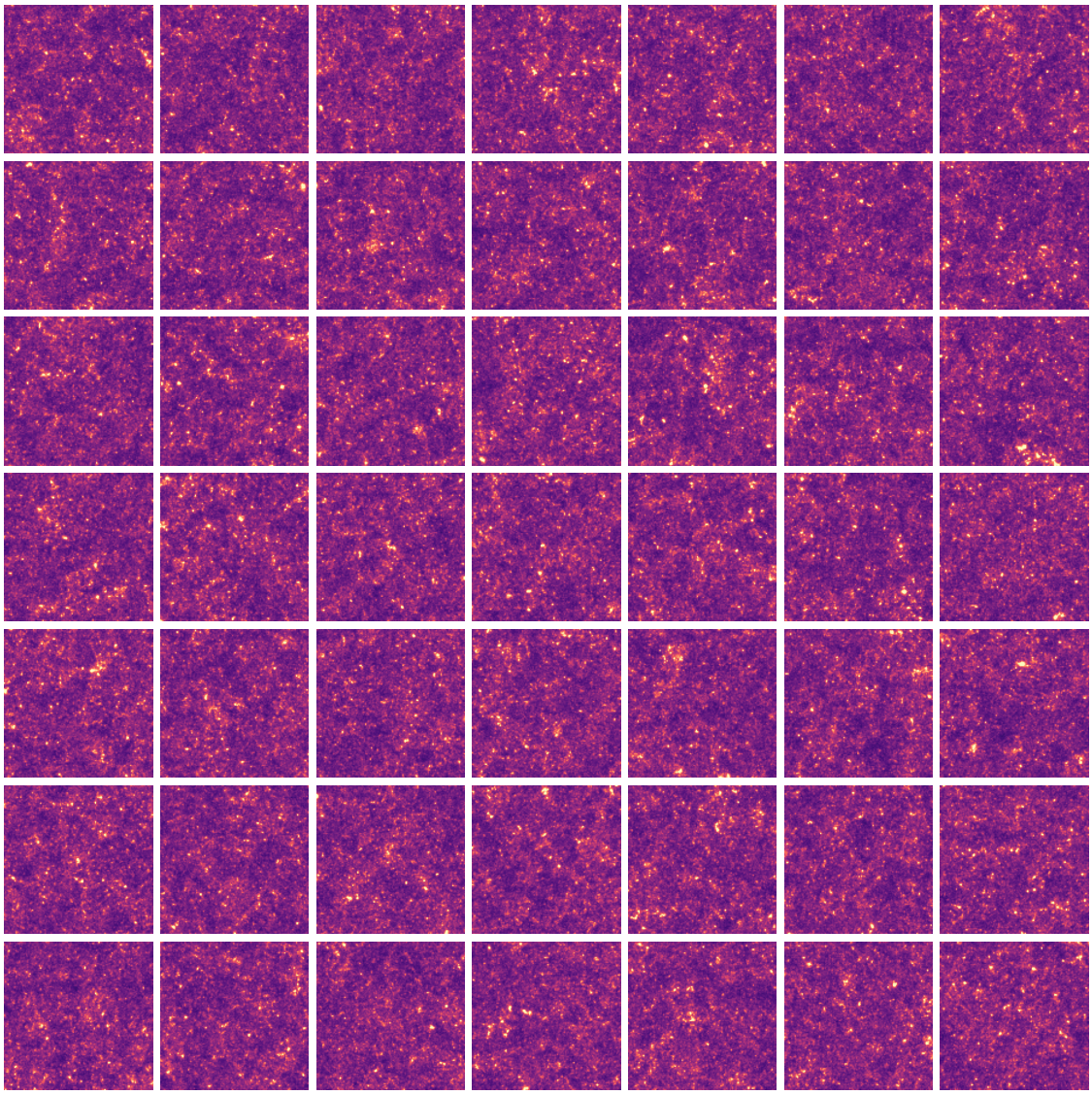

Figure A.6: A grid of images generated by the S2000-3L for test cosmology B.

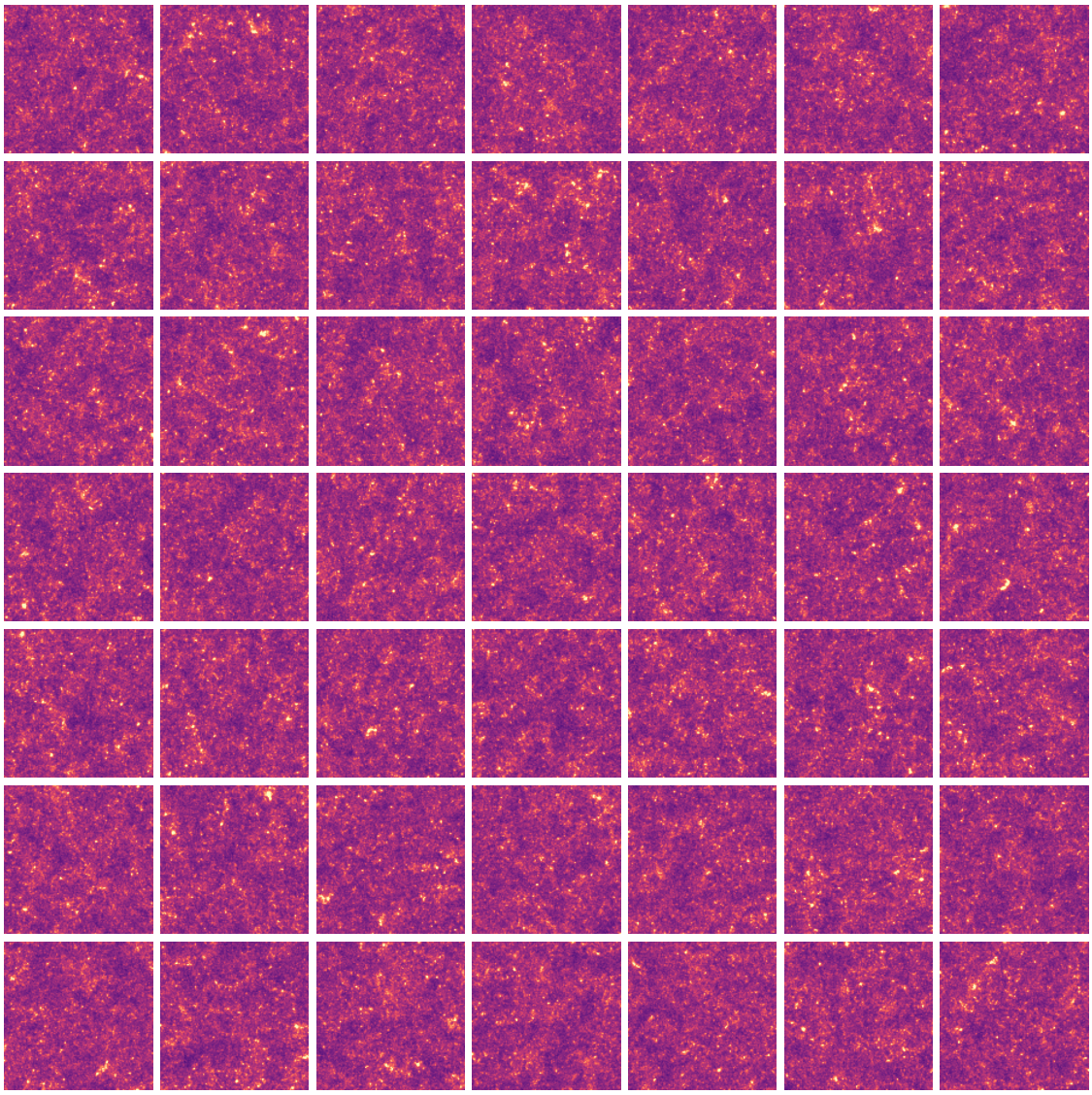

Figure A.7: A grid of images generated by the S2000-3L for test cosmology C.

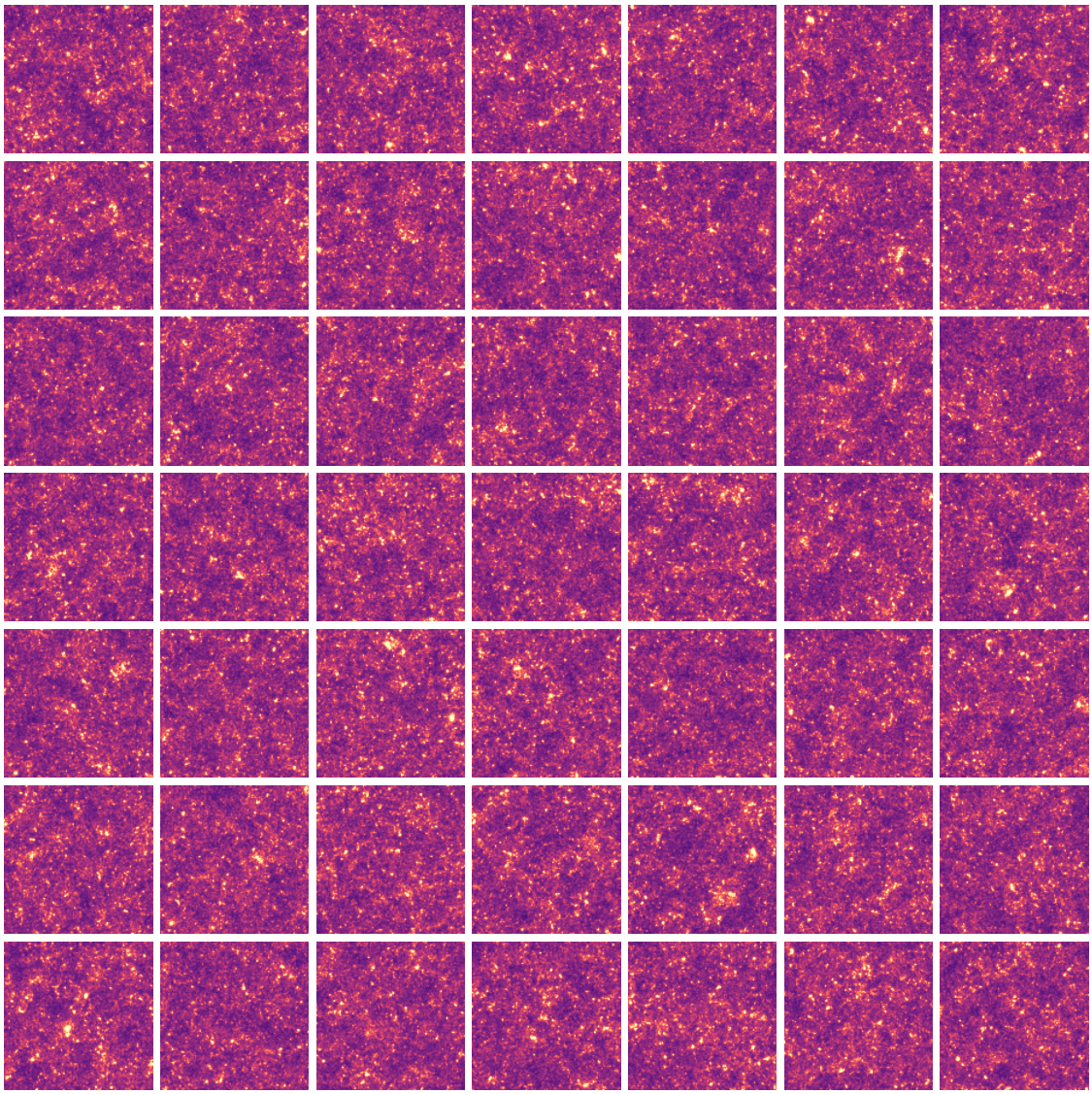

Figure A.8: A grid of images generated by the S2000-3L for test cosmology D.

Bibliography

Arjovsky, Martin, Soumith Chintala, and Léon Bottou (Dec. 2017). *Wasserstein GAN*. DOI: 10.48550/arXiv.1701.07875. arXiv: arXiv:1701.07875. (Visited on 12/20/2022).

Böhm, Vanessa and Uroš Seljak (Sept. 2022). *Probabilistic Autoencoder*. arXiv: arXiv:2006.05479. (Visited on 03/16/2023).

Bozza, V. and L. Mancini (Aug. 2004). "Gravitational Lensing by Black Holes: A Comprehensive Treatment and the Case of the Star S2". In: *The Astrophysical Journal* 611.2, pp. 1045–1053. ISSN: 0004-637X, 1538-4357. DOI: 10.1086/422309. (Visited on 03/20/2023).

Bozza, V., S. Calchi Novati, and L. Mancini (Feb. 2008). "Gravitational Lensing by the Supermassive Black Hole in the Center of M31". In: *Il Nuovo Cimento B* 122.5, pp. 579–583. ISSN: 03693554, 03693554. DOI: 10.1393/ncb/i2007-10386-6. arXiv: 0711.0750 [astro-ph, physics:gr-qc]. (Visited on 03/20/2023).

Brock, Andrew, Jeff Donahue, and Karen Simonyan (Feb. 2019). *Large Scale GAN Training for High Fidelity Natural Image Synthesis*. DOI: 10.48550/arXiv.1809.11096. arXiv: arXiv:1809.11096. (Visited on 12/19/2022).

Collaboration, Planck et al. (Sept. 2020). "Planck 2018 Results. VI. Cosmological Parameters". In: *Astronomy & Astrophysics* 641, A6. ISSN: 0004-6361, 1432-0746. DOI: 10.1051/0004-6361/201833910. arXiv: 1807.06209 [astro-ph]. (Visited on 01/31/2023).

Dhariwal, Prafulla and Alex Nichol (June 2021). *Diffusion Models Beat GANs on Image Synthesis*. DOI: 10.48550/arXiv.2105.05233. arXiv: arXiv:2105.05233. (Visited on 03/22/2023).

Einstein, Albert (Nov. 1914). *Volume 6: The Berlin Years: Writings, 1914-1917 (English Translation Supplement) Page 30 (42 of 462)*. https://einsteinpapers.press.princeton.edu/vol6-trans/42. (Visited on 03/20/2023).

Fluri, Janis et al. (Sept. 2019). "Cosmological Constraints with Deep Learning from KiDS-450 Weak Lensing Maps". In: *Physical Review D* 100.6, p. 063514. ISSN: 2470-0010, 2470-0029. DOI: 10.1103/PhysRevD.100.063514. arXiv: 1906.03156 [astro-ph]. (Visited on 01/31/2023).

Goodfellow, Ian et al. (2014). "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems*. Vol. 27. Curran Associates, Inc. (Visited on 12/19/2022).

Herrera-Martín, Antonio et al. (Feb. 2019). "Strong Gravitational Lensing by Wave Dark Matter Halos". In: *The Astrophysical Journal* 872.1, p. 11. ISSN: 0004-637X. DOI: 10.3847/1538-4357/aafaf0. (Visited on 03/20/2023).

Hinton, G. E. and R. R. Salakhutdinov (July 2006). "Reducing the Dimensionality of Data with Neural Networks". In: *Science* 313.5786, pp. 504–507. DOI: 10.1126/science.1127647. (Visited on 03/22/2023).

Ho, Jonathan, Ajay Jain, and Pieter Abbeel (Dec. 2020). *Denoising Diffusion Probabilistic Models*. DOI: 10.48550/arXiv.2006.11239. arXiv: 2006.11239 [cs, stat]. (Visited on 09/18/2022).

Ho, Jonathan, Chitwan Saharia, et al. (Dec. 2021). *Cascaded Diffusion Models for High Fidelity Image Generation*. DOI: 10.48550/arXiv.2106.15282. arXiv: arXiv:2106.15282. (Visited on 03/22/2023).

Hong, Susung et al. (Feb. 2023). *Improving Sample Quality of Diffusion Models Using Self-Attention Guidance*. arXiv: arXiv:2210.00939. (Visited on 03/20/2023).

Hopkins, Philip F. (June 2015). "A New Class of Accurate, Mesh-Free Hydrodynamic Simulation Methods". In: *Monthly Notices of the Royal Astronomical Society* 450.1, pp. 53–110. ISSN: 0035-8711, 1365-2966. DOI: 10.1093/mnras/stv195. (Visited on 03/18/2023).

Jarzynski, C. (Nov. 1997). "Equilibrium Free-Energy Differences from Nonequilibrium Measurements: A Master-Equation Approach". In: *Physical Review E* 56.5, pp. 5018–5035. ISSN: 1063-651X, 1095-3787. DOI: 10.1103/PhysRevE.56.5018. (Visited on 12/13/2022).

Kantorovich, L. V. (July 1960). "Mathematical Methods of Organizing and Planning Production". In: *Management Science*. DOI: 10.1287/mnsc.6.4.366. (Visited on 03/22/2023).

Karras, Tero, Miika Aittala, et al. (2021). "Alias-Free Generative Adversarial Networks". In: *Advances in Neural Information Processing Systems*. Vol. 34. Curran Associates, Inc., pp. 852–863. (Visited on 03/20/2023).

Karras, Tero, Samuli Laine, and Timo Aila (Mar. 2019). *A Style-Based Generator Architecture for Generative Adversarial Networks*. DOI: 10.48550/arXiv.1812.04948. arXiv: arXiv:1812.04948. (Visited on 12/19/2022).

Kingma, Diederik P. and Jimmy Ba (Jan. 2017). *Adam: A Method for Stochastic Optimization*. DOI: 10.48550/arXiv.1412.6980. arXiv: arXiv:1412.6980. (Visited on 02/14/2023).

Kingma, Diederik P., Tim Salimans, et al. (June 2022). *Variational Diffusion Models*. DOI: 10.48550/arXiv.2107.00630. arXiv: 2107.00630 [cs, stat]. (Visited on 09/18/2022).

Kingma, Diederik P. and Max Welling (May 2014). *Auto-Encoding Variational Bayes*. arXiv: arXiv:1312.6114. (Visited on 03/20/2023).

– (2019). "An Introduction to Variational Autoencoders". In: *Foundations and Trends® in Machine Learning* 12.4, pp. 307–392. ISSN: 1935-8237, 1935-8245. DOI: 10.1561/2200000056. arXiv: 1906.02691 [cs, stat]. (Visited on 03/16/2023).

Kullback, S. and R. A. Leibler (Mar. 1951). "On Information and Sufficiency". In: *The Annals of Mathematical Statistics* 22.1, pp. 79–86. ISSN: 0003-4851, 2168-8990. DOI: 10.1214/aoms/1177729694. (Visited on 03/20/2023).

Liu, A. et al. (Feb. 2023). "X-Ray Analysis of JWST's First Galaxy Cluster Lens SMACS J0723.3-7327". In: *Astronomy & Astrophysics* 670, A96. ISSN: 0004-6361, 1432-0746. DOI: 10.1051/0004-6361/202245118. arXiv: 2210.00633 [astro-ph]. (Visited on 03/20/2023).

Makhzani, Alireza et al. (May 2016). *Adversarial Autoencoders*. DOI: 10.48550/arXiv.1511.05644. arXiv: arXiv:1511.05644. (Visited on 03/16/2023).

Martinelli, Matteo and Isaac Tutusaus (Aug. 2019). "CMB Tensions with Low-Redshift H_0 and S_8 Measurements: Impact of a Redshift-Dependent Type-Ia Supernovae Intrinsic Luminosity". In: *Symmetry* 11.8, p. 986. ISSN: 2073-8994. DOI: 10.3390/sym11080986. arXiv: 1906.09189 [astro-ph]. (Visited on 03/20/2023).

Mudur, Nayantara and Douglas P. Finkbeiner (Nov. 2022). *Can Denoising Diffusion Probabilistic Models Generate Realistic Astrophysical Fields?* arXiv: arXiv:2211.12444. (Visited on 11/28/2022).

Neal, Radford M. (Sept. 1998). *Annealed Importance Sampling*. arXiv: arXiv:physics/9803008. (Visited on 12/13/2022).

Ng, Andrew and Michael Jordan (2001). "On Discriminative vs. Generative Classifiers: A Comparison of Logistic Regression and Naive Bayes". In: *Advances in Neural Information Processing Systems*. Vol. 14. MIT Press. (Visited on 03/20/2023).

Nichol, Alex and Prafulla Dhariwal (Feb. 2021). *Improved Denoising Diffusion Probabilistic Models*. arXiv: arXiv:2102.09672. (Visited on 12/14/2022).

Pandey, Kushagra et al. (Nov. 2022). *DiffuseVAE: Efficient, Controllable and High-Fidelity Generation from Low-Dimensional Latents*. DOI: 10.48550/arXiv.2201.00308. arXiv: arXiv:2201.00308. (Visited on 03/22/2023).

Perraudin, Nathanaël et al. (Apr. 2020). *Emulation of Cosmological Mass Maps with Conditional Generative Adversarial Networks*. (Visited on 09/18/2022).

Petri, Andrea (Oct. 2016). "Mocking the Weak Lensing Universe: The LensTools Python Computing Package". In: *Astronomy and Computing* 17, pp. 73–79. ISSN: 22131337. DOI: 10.1016/j.ascom.2016.06.001. arXiv: 1606.01903 [astro-ph]. (Visited on 03/12/2023).

Potter, Douglas, Joachim Stadel, and Romain Teyssier (Sept. 2016). *PKDGRAV3: Beyond Trillion Particle Cosmological Simulations for the Next Era of Galaxy Surveys*. arXiv: arXiv:1609.08621. (Visited on 03/18/2023).

Ronneberger, Olaf, Philipp Fischer, and Thomas Brox (2015). "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Ed. by Nassir Navab et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 234–241. ISBN: 978-3-319-24574-4. DOI: 10.1007/978-3-319-24574-4_28.

Saad, Muhammad Muneeb, Ruairi O'Reilly, and Mubashir Husain Rehmani (Dec. 2022). *A Survey on Training Challenges in Generative Adversarial Networks for Biomedical Image Analysis*. arXiv: arXiv:2201.07646. (Visited on 03/20/2023).

Schmalzing, J., M. Kerscher, and T. Buchert (Oct. 1995). *Minkowski Functionals in Cosmology*. arXiv: arXiv:astro-ph/9508154. (Visited on 03/12/2023).

Scott Dodelson and Fabian Schmidt (2020). *Modern Cosmology*. Second Edition. Academic Press.

Sgier, Raphael et al. (Feb. 2021). "Fast Lightcones for Combined Cosmological Probes". In: *Journal of Cosmology and Astroparticle Physics* 2021.02, pp. 047–047. ISSN: 1475-7516. DOI: 10.1088/1475-7516/2021/02/047. arXiv: 2007.05735 [astro-ph]. (Visited on 03/24/2023).

Smith, Michael J et al. (Feb. 2022). "Realistic Galaxy Image Simulation via Score-Based Generative Models". In: *Monthly Notices of the Royal Astronomical Society* 511.2, pp. 1808–1818. ISSN: 0035-8711, 1365-2966. DOI: 10.1093/mnras/stac130. (Visited on 03/18/2023).

Sohl-Dickstein, Jascha et al. (n.d.). "Deep Unsupervised Learning Using Nonequilibrium Thermodynamics". In: ().

Song, Jiaming, Chenlin Meng, and Stefano Ermon (Oct. 2022). *Denoising Diffusion Implicit Models*. arXiv: arXiv:2010.02502. (Visited on 12/14/2022).

Song, Yang et al. (Feb. 2021). *Score-Based Generative Modeling through Stochastic Differential Equations*. DOI: 10.48550/arXiv.2011.13456. arXiv: 2011.13456 [cs, stat]. (Visited on 09/18/2022).

Stadel, Joachim Gerhard (Jan. 2001). "Cosmological N-body Simulations and Their Analysis". PhD thesis, p. 3657. (Visited on 01/31/2023).

Ullmo, Marion, Aurélien Decelle, and Nabila Aghanim (July 2021). "Encoding Large Scale Cosmological Structure with Generative Adversarial Networks". In: *Astronomy & Astrophysics* 651, A46. ISSN: 0004-6361, 1432-0746. DOI: 10.1051/0004-6361/202039866. arXiv: 2011.05244 [astro-ph]. (Visited on 09/22/2022).

Vaswani, Ashish et al. (Dec. 2017). *Attention Is All You Need*. arXiv: arXiv:1706.03762. (Visited on 10/13/2022).

Vincent, Pascal et al. (2010). "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion". In: *Journal of Machine Learning Research* 11.110, pp. 3371–3408. ISSN: 1533-7928. (Visited on 10/25/2022).

Wang, Zhendong et al. (Oct. 2022). *Diffusion-GAN: Training GANs with Diffusion*. arXiv: arXiv:2206.02262. (Visited on 03/22/2023).

Weng, Lilian (July 2021). *What Are Diffusion Models?* https://lilianweng.github.io/posts/2021-07-11-diffusion-models/. (Visited on 03/26/2023).

Wiatrak, Maciej, Stefano V. Albrecht, and Andrew Nystrom (Mar. 2020). *Stabilizing Generative Adversarial Networks: A Survey*. arXiv: arXiv:1910.00927. (Visited on 03/20/2023).

Wing Hei Yiu, Timothy, Janis Fluri, and Tomasz Kacprzak (Dec. 2021). *A Tomographic Spherical Mass Map Emulator of the KiDS-1000 Survey Using Conditional Generative Adversarial Networks*. (Visited on 09/18/2022).

Xu, Yilun et al. (Oct. 2022). *Poisson Flow Generative Models*. arXiv: arXiv:2209.11178. (Visited on 03/20/2023).

Yacoby, Yaniv, Weiwei Pan, and Finale Doshi-Velez (Mar. 2021). "Failure Modes of Variational Autoencoders and Their Effects on Downstream Tasks". In: (visited on 03/20/2023).

Zhu, Jun-Yan et al. (Aug. 2020). *Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks*. DOI: 10.48550/arXiv.1703.10593. arXiv: arXiv:1703.10593. (Visited on 12/19/2022).