Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Computer Vision Lab

Conditioning of DDPMs on Accelerated MRI

Semester Thesis

Lionel Peer
Department of Information Technology and Electrical Engineering

Advisors: Georg Brunner & Emiljo Mëhillaj
Supervisor: Prof. Dr. Ender Konukoglu
Computer Vision Laboratory, Group for Biomedical Image Computing
Department of Information Technology and Electrical Engineering

January 10, 2020

Abstract

The abstract gives a concise overview of the work you have done. The reader shall be able to decide whether the work which has been done is interesting for him by reading the abstract. Provide a brief account on the following questions:

- What is the problem you worked on? (Introduction)
- How did you tackle the problem? (Materials and Methods)
- What were your results and findings? (Results)
- Why are your findings significant? (Conclusion)

The abstract should approximately cover half of a page, and does generally not contain citations.

Acknowledgements

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Give an introduction to the topic you have worked on:

- *What is the rationale for your work?* Give a sufficient description of the problem, e.g. with a general description of the problem setting, narrowing down to the particular problem you have been working on in your thesis. Allow the reader to understand the problem setting.
- *What is the scope of your work?* Given the above background, state briefly the focus of the work, what and how you did.
- *How is your thesis organized?* It helps the reader to pick the interesting points by providing a small text or graph which outlines the organization of the thesis. The structure given in this document shows how the general structuring shall look like. However, you may fuse chapters or change their names according to the requirements of your thesis.

1.1 Focus of this Work

1.2 Thesis Organization

Chapter 2

Related Work

2.1 DDPMs for Image Synthesis

2.2 Image-Guided Diffusion and Reconstruction

Chapter 3

Materials and Methods

The objectives of the “Materials and Methods” section are the following:

- *What are tools and methods you used?* Introduce the environment, in which your work has taken place - this can be a software package, a device or a system description. Make sure sufficiently detailed descriptions of the algorithms and concepts (e.g. math) you used shall be placed here.

- *What is your work?* Describe (perhaps in a separate chapter) the key component of your work, e.g. an algorithm or software framework you have developed.

3.1 Latent Variable Models

Before getting started it is important to define the terms used in the next sections, since they all stem from Bayesian statistics. The Bayesian theorem can be written as

$$p(z|x) = \frac{p(x|z)p(z)}{p(x)} \tag{3.1}$$

where it is implicitly assumed that p is a probability density function over two continuous random variables x and z. The formula holds in general, but in generative modeling and machine learning it is usually assumed that the letter z represents a random variable in a latent (unobserved) space from which – after successful training – new data can be generated by sampling. This requires that $p(z)$ is a simple distribution from which sampling is easy and that the trained model is capable of mapping values from the latent distribution to the true data distribution.

Using above described ordering, the four terms in this formula use distinct names:

$p(x)$ is called the *evidence* or the *marginal likelihood*. It encompasses the actual observations of the data.

$p(z)$ is called the *prior*, since it exposes information on z before any conditioning.

$p(z|x)$ is called the *posterior*. It describes the distribution over z after (*post*) having seen the evidence x.

$p(x|z)$ is called the *likelihood*. It gives the literal likelihood of observing an example x when choosing the latent space to be a specific z.

3.2 Variational Autoencoders

One of the most straightforward examples of a generative model, where the goal is to find such a latent space representation of the training sample distribution, is the Variational Autoencoder (VAE) [6]. The name of the VAE stems from the Autoencoder, a network that tries to recreate its output through a bottleneck and thereby learns a compressed representation of the data. [8] Autoencoders bear similarity to other dimension reduction methods like Principal Component Analysis (PCA) and therefore were first published under the name *Nonlinear principal component analysis*. The *variational* part in the VAE stems from the fact that it does not only learn to recreate input samples through dimensionality reduction, but is also optimized to represent the distribution over the training samples as a combination of a parameterized latent distribution $p_{\theta_z}(z)$ and a neural network mapping $p_{\theta_{NN}}(x|z)$ between the latent space and the sample space, termed decoder. The latent distribution is chosen such that sampling from it is easy (e.g. a multivariate Gaussian, with the parameters being vectors of means and variances). With sufficient dimensionality reduction the encoding should not overfit and the latent space should be a good approximation of the true data manifold. This enables the creation of data, by sampling the latent space and mapping it to the output.

Marginalizing $p_\theta(x)$ requires another approximation of the posterior $p(z|x)$ with a neural network, termed the encoder $p_{\theta_{NN_{in}}}(z|x)$.

$$p_\theta(x) = \int p_{\theta_{NN}}(x|z)p_{\theta_z}(z)dz = \frac{p_{\theta_{NN_{out}}}(x|z)p_{\theta_z}(z)}{p_{\theta_{NN_{in}}}(z|x)} \tag{3.2}$$

A schematic of a VAE, separated into these 3 factors – encoder, latent distribution and decoder – is shown in Fig. 3.1.

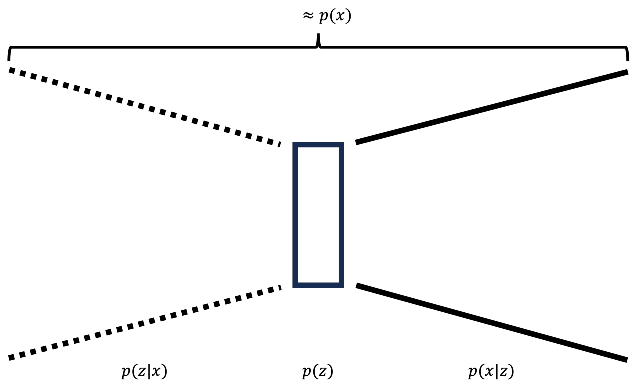

Figure 3.1: VAE schematic: $p(x)$ is approximated through a latent variable model where posterior and likelihood are modeled with neural networks and the prior on the latent variable is modeled through a simple parameterized distribution (often Gaussian). The hope is, that after training, sampling from $p(z)$ and passing it through the neural network $p_{\theta_{NN}}(x|z)$, is the same as sampling from $p(x)$.

3.3 KL Divergence and Variational Lower Bound

In VAEs, the encoder $p_\theta(z|x)$ needs to approximate the posterior $p(z|x)$, therefore a differentiable loss function is needed that compares two probability distributions. One such heavily used measure is the KL (Kullback-Leibler) divergence

$$KL\left[p_\theta(z|x)||p(z|x)\right] = \int \log \frac{p_\theta(z|x)}{p(z|x)} p_\theta(z|x)dz \tag{3.3}$$

which has the properties of being strictly non-negative and is only 0 if the two distributions are equal.

At the same time, the output of the VAE should fit the true data distribution well, e.g. should maximize the log-likelihood of the evidence $p_\theta(x|z)$. This term is mainly responsible for the reconstruction of truthful samples from the latent space. Combining the two terms and inverting the signs (for minimization rather than maximization) gives a loss function known as ELBO (evidence lower bound) or VLB (variational lower bound).

$$\mathcal{L}_{VLB} = -\log p_\theta(x|z) + KL\left[p_\theta(z|x)||p(z|x)\right] \tag{3.4}$$

3.4 Diffusion Denoising Probabilistic Models

Diffusion Denoising Probabilistic Models (DDPMs or Diffusion Models) are a generative model that learn the distribution of images in a training set. During training, sample images are gradually destroyed by adding noise over many iterations and a neural network is trained, such that these steps can be inverted.

As the name suggests, image content is diffused in timesteps, therefore we use the random variable $\boldsymbol{x}_0$ to represent our original training images, $\boldsymbol{x}_t$ for (partially noisy) images at an intermediate timestep and $\boldsymbol{x}_T$ for images at the end of the process where all information has been destroyed and the distribution $q(\boldsymbol{x}_T)$ largely follows an isotropic Gaussian distribution.

The goal is to train a network that creates a less noisy image $\boldsymbol{x}_{t-1}$ from $\boldsymbol{x}_t$. If this is achieved we should be able to sample some new $\boldsymbol{x}_T$ and generate new samples from the training distribution $q(\boldsymbol{x}_0)$ by passing this noisy image many times through the network until the noise is fully removed.

3.4.1 Forward Diffusion Process

Mathematical Description

In order to derive a training objective it is important to understand the workings of the *forward diffusion process*. During this process, i.i.d (independent and identically distributed) Gaussian noise is applied to the image over many discrete timesteps. A *variance schedule* defines the means and variances ($\sqrt{1-\beta}$ and β) of the added noise at every timestep. [5] The whole process can be expressed as a Markov chain (depicted in Fig. 3.2), with the factorization

$$q(\boldsymbol{x}_T|\boldsymbol{x}_0) = q(\boldsymbol{x}_0)\prod_{t=1}^{T} q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) \tag{3.5}$$

where the transition distributions $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}(\sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t I)$. An example of iterative destruction of an image by this process is shown in Fig. 3.3.

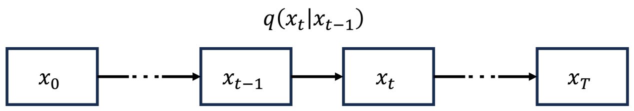

Figure 3.2: Forward Diffusion Process: An image is iteratively destroyed by adding normally distributed noise, according to a schedule. This represents a Markov process with the transition probability $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$.

Gladly it is not necessary to sample noise again and again in order to arrive at $\boldsymbol{x}_t$, since Ho et al. derived a closed-form solution to the sampling procedure. [5] For this, the variance schedule is first reparameterized as $1-\beta = \alpha$

$$q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}(\sqrt{\alpha_t}\boldsymbol{x}_{t-1}, (1-\alpha_t)\boldsymbol{I}) \tag{3.6}$$

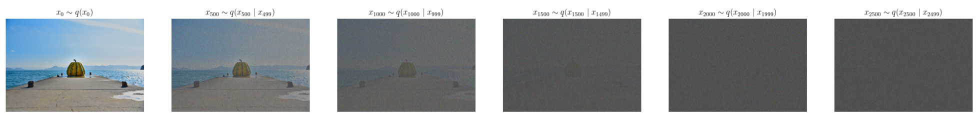
$x_0 \sim q(x_0)$
$x_{500} \sim q(x_{500} \mid x_{499})$
$x_{1000} \sim q(x_{1000} \mid x_{999})$
$x_{1500} \sim q(x_{1500} \mid x_{1499})$
$x_{2000} \sim q(x_{2000} \mid x_{1999})$
$x_{2500} \sim q(x_{2500} \mid x_{2499})$

Figure 3.3: Example of Iterative Image Destruction through Forward Diffusion Process: The indices give the time step in the iterative destruction process, where β was created according to a linear noise variance schedule (5000 steps from in the 0.001 to 0.02 range and picture resolution of 4016 by 6016 pixels).

and the closed-form solution for $q(\boldsymbol{x}_t|\boldsymbol{x}_0)$ is derived by introducing the cumulative product $\bar{\alpha_t} = \prod_{s=1}^{t} \alpha_s$ as

$$q(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\sqrt{\bar{\alpha_t}}\boldsymbol{x}_0, (1 - \bar{\alpha_t})\boldsymbol{I}) \tag{3.7}$$

A choice of $\bar{\alpha_t} \in [0, 1]$ in above parameterizaiton ensures that the variance does not explode in the process, but that the SNR (signal-to-noise-ratio) still goes to 0 by gradually attenuating the means, corresponding to the original image. Thanks to the reparameterization with $\bar{\alpha_t}$, the forward process is also not restricted anymore to discrete timesteps, but a continuous schedule can be used. [7, 13]

The derivation that leads from Eq. 3.6 to Eq. 3.7 is left to appendix B.1.

Influence of Scheduling Functions

The process of information destruction is dependent on the chosen variance schedule, the number of steps and the image size. Beyond the most simple case – a constant variance over time – Ho et al. opted for the second most simple option, a linear schedule, where the variance β_t grows linearly in t. [5] Nichol et al. later found that a cosine-based schedule gives better results on lower resolution images, since it does not destruct information quite as quickly, making it more informative in the last few timesteps. They also mention that their cosine schedule is purely based on intuition and they similar functions to perform equally well. [10] Own experiments exploring above mentioned parameters are explained in 4.1 and plots of the two different variance schedules are visible in Fig. 4.1.

3.4.2 Reverse Diffusion Process

DDPMs can be viewed as latent space models in a similar way that Generative Adversarial Nets or Variational Autoencoders can. [4, 6]

In DDPMs the reverse process is again a Markov chain and can therefore again be factorized as

$$q(\boldsymbol{x}_0|\boldsymbol{x}_T) = q(\boldsymbol{x}_T) \prod_{t=T}^{1} q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) \tag{3.8}$$

which means that our network does not learn to approximate the full inversion, but rather just the transition probabilities $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ in the chain, which are transitions between several intermediate latent distributions. Sohl-Dickstein et al. further showed that the reverse transitions are also Gaussian in the limit of $t \to 0$, e.g. as long as the diffusion steps are small enough. We therefore approximate

$$q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) \approx p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t), \boldsymbol{\Sigma}_\theta(\boldsymbol{x}_t, t)). \tag{3.9}$$

3.4.3 Loss Functions

The combination of forward $q(\boldsymbol{x}_T|\boldsymbol{x}_0)$ and reverse process $q(\boldsymbol{x}_0|\boldsymbol{x}_T)$ can be viewed as a chain of many VAEs and we can again formulate a variational lower bound objective (Eq. 3.4) that maximizes log-likelihood of the output and matches transition probabilities. [10]

$$\mathcal{L}_0 = -\log p_\theta(x_0|x_1) \tag{3.10}$$
$$\mathcal{L}_{1:T-1} = KL\left[q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t)\right] \tag{3.11}$$
$$\mathcal{L}_T = KL\left[q(x_T|x_0)||p(x_T)\right] \tag{3.12}$$

The exact posterior $q(x_{t-1}|x_t, x_0)$ is tractable for specific samples of the training distribution, therefore $\mathcal{L}_{1:T-1}$ could be calculated, since KL divergence has a closed form solution for two Gaussian distributions. $\mathcal{L}_T$ is independent of θ and therefore not used for optimization. It should anyway be very close to zero if the parameterization of the forward process is correct and forward diffused samples get close to $\mathcal{N}(0, \boldsymbol{I})$. The first term is only used for performance evaluation in terms of log-likelihood, but not in optimization.

Another simplification is usually taken and $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ only approximates the means $\boldsymbol{\mu}_\theta$ and not the variances. For small enough timesteps, the means determine the transitional distributions much stronger than the variances. The network is furhter usually trained to not predict the means directly, but the noise and the means are then determined through a reparameterization. [5, 10]

3.5 Image Guided Diffusion

Both, Choi et al. and Lugmayr et al. make use of unconditional DDPMs for image-guided diffusion for the tasks of image translation in the former and in-painting in the latter. [1, 9] Similarly, classifier guidance or CLIP-guidance can be used on unconditional and conditional DDPMs to produce samples of a specific class or matching a prompt in the unconditional case or to further trade off sample variability for sample fidelity. [3] We show here that both approaches can be interpreted as special cases of MAP (maximum a posteriori) estimation and that they can be generalized to other means of guidance during the reverse process.

3.5.1 MAP Estimation for Inverse Problems

MAP estimation is a statistical concept that is often used for optimizing the parameters θ of a parameterized distribution to observed data $z \sim p(z)$ following Bayes rule

$$\hat{\theta}_{MAP} = \underset{\theta}{\operatorname{argmax}}\, p(\theta|z) = \underset{\theta}{\operatorname{argmax}}\, \frac{p(z|\theta)p(\theta)}{p(z)} \tag{3.13}$$

or for inverse problems

$$\hat{x}_{MAP} = \underset{x}{\operatorname{argmax}}\, p(x|s) = \underset{x}{\operatorname{argmax}}\, \frac{p(s|x)p(x)}{p(s)} \tag{3.14}$$

where $s \sim p(s)$ is the evidence that is provided by the measured signal, $p(x)$ is a prior on the desired reconstruction and the likelihood term $p(s|x)$ enforces a data consistency between the measured signal and the true distribution, usually a forward model of the data-corrupting process. Since maximizing $p(x|s)$ is the same as maximizing $\log(x|s)$ and $p(s)$ is independent of θ, we can separate the product into a sum.

$$\hat{x}_{MAP} = \underset{x}{\operatorname{argmax}}\, logp(x|s) = \underset{x}{\operatorname{argmax}} \log \frac{p(s|x)p(x)}{p(s)} \tag{3.15}$$

$$= \underset{x}{\operatorname{argmax}} \log p(s|x)p(x) \tag{3.16}$$

$$= \underset{x}{\operatorname{argmax}} \log f(x,s) \tag{3.17}$$

Such problems are usually optimized using iterative optimization schemes such as gradient ascent $x_{t+1} = x_t + \lambda \nabla_x f(x,s)$, with λ being the step length. It is usually helpful to separate the data consistency term (likelihood) and regularizer (prior) into individual terms for joint optimization with their respective gradients and weights.

$$x_{i+1} = x_i + \lambda_1 \nabla_{x_i} \log p(s|x_i) + \lambda_2 \nabla_{x_i} \log p(x_i) \tag{3.18}$$

3.5.2 DDPMs as Priors

DDPMs approximate a data distribution over training images $p(x)$ and according to Song et al., they do so by learning to approximate gradients of the distribution and taking a gradient ascent step with every iteration of the reverse diffusion process. [12] With this interpretation the DDPM has the exact same form as Eq. 3.18 without the data consistency term $p(s|x)$.

$$x_{t-1} = x_t + \nabla_{x_t} \log p(x_t) \tag{3.19}$$

3.5.3 Classifier Guidance

Classifier guidance as termed by Nichol et al. introduces a data consistency term $p(s|x_t)$ in the form of a classifier trained on noisy images, where s is the random variable expressing if an image belongs to a certain class. [3, 11] Conditioning on a classifier is sucessfully used by taking gradient ascent steps not only in the direction that maximizes the prior $p(x)$ in a DDPM $\nabla_{x_t} \log p(x_t)$, but also the direction of this conditioning term $\nabla_{x_t} \log p(s|x_t)$. In total, this is equal to Eq. 3.18

$$x_{t+1} = \underbrace{x_t + \nabla_{x_t} \log p(x_t)}_{x'_{t+1}} + \lambda \nabla_{x_t} \log p(s|x_t) \tag{3.20}$$

with x'_{t+1} being the prediction of the reverse diffusion steps before any conditioning and λ an arbitrary factor determining the strength of the guidance.

3.5.4 Gradients and Closed-Forms of Data Consistency Terms

Starting from Eq. 3.19, the data consistency term can be reintroduced into the DDPM model.

Choi et al. guide the diffusion process by trying to match the low frequency content of a target image to the low frequency content of the prediction $\operatorname{argmin}_x \phi(s) - phi(x)$. They do this by using a very simple linear data consistency function, corresponding to a difference between linear low-pass filtered representations, which they call ILVR (iterative latent variable refinement)

$$x_{t-1} = \phi(s_t) + (I - \phi)(x_t) \tag{3.21}$$

$$x_{t-1} = x_t + \phi(s_t) - \phi(x_t) \tag{3.22}$$

$$x_{t-1} = x_t + \phi(s_t - x_t) \tag{3.23}$$

where ϕ is a linear filter operation and s_t is obtained by using the forward process on the target image. [1]

Similarly, Lugmayr et al. use a conditioning on known parts of the image for the inpainting operation, which effectively boils down to applying a linear mask that zeroes out known parts in the prediction and replaces them with outputs from the forward process.

$$x_{t-1} = x_t - \mathcal{M}(x_t) + \mathcal{M}(s_t) \tag{3.24}$$

$$x_{t-1} = x_t + \mathcal{M}(s_t - x_t) \tag{3.25}$$

As is easily seen, Eq. 3.21 and Eq. 3.24 take the same form and can be interpretated as simultaneously taking gradient ascent steps for optimizing $\operatorname{argmax}_x \log p(x)$ and $\operatorname{argmax}_x \log p(s|x)$.

Assuming distribution over latent predictions and targets at diffusion timestep t, $p(s|t)$ and $p(x|t)$, and $\mathcal{L}$ corresponding to an arbitrary linear operation

$$\mathcal{L}(p(s|t) - p(x|t)) = \nabla_{x_t} \log p(s_t|x_t) \tag{3.26}$$

$$\int \mathcal{L}(p(s|t) - p(x|t))dt = \int \nabla_{x_t} \log p(s_t|x_t)dt \tag{3.27}$$

$$\int \mathcal{L}(p(s|t))dt - \int \mathcal{L}(p(x|t))dt = \log p(s|x) \tag{3.28}$$

$$\mathcal{L}\left(\int p(s|t)dt\right) - \mathcal{L}\left(\int p(x|t)dt\right) = \log p(s|x) \tag{3.29}$$

$$\mathcal{L}(p(s)) - \mathcal{L}(p(x)) = \log p(s|x) \tag{3.30}$$

where the left side of the equation is the original

Chapter 4

Experiments and Results

Describe the evaluation you did in a way, such that an independent researcher can repeat it. Cover the following questions:

- *What is the experimental setup and methodology?* Describe the setting of the experiments and give all the parameters in detail which you have used. Give a detailed account of how the experiment was conducted.
- *What are your results?* In this section, a *clear description* of the results is given. If you produced lots of data, include only representative data here and put all results into the appendix.

4.1 Influence of Schedules and Image Size on the Forward Diffusion

Ho et al. had derived a closed form solution to the forward process of DDPMs and Nichol et al. investigated alternative options for the noise scheduling. [5, 10] They concluded that the important parameters to model are not the variances β of the transitions, but the variances $1 - \bar{\alpha}$ of the closed-form forward process, since they are the ones responsible for the destruction of information.

They decided to go with a squared cosine function, since this would be close to linear smooth out towards the critical beginning and end points of the process. In Fig.4.1 you can see how $1 - \bar{\alpha}$ and β behave for both approaches. It is immediately visible that the variances reach the maximum too early and flatten out for the linear schedule. This leads to the intuition that the last few steps are not very useful.

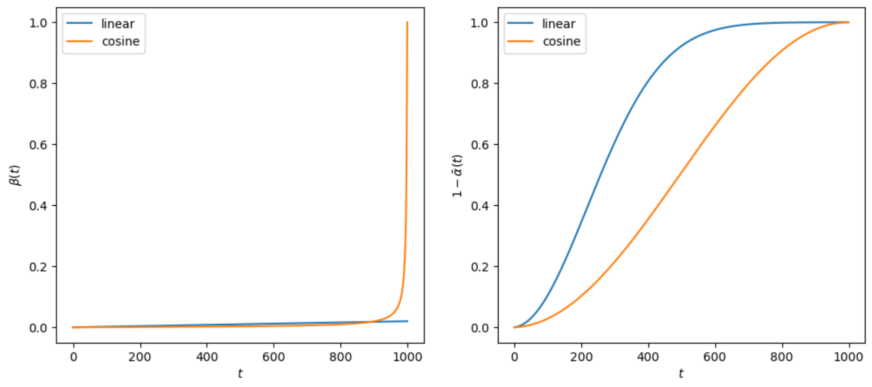

Figure 4.1: Variance Schedule Approaches: Modeling the $1 - \bar{\alpha}$ as an approximate linear function (right cosine) and deriving β (left cosine), or modeling β as a linear function (left linear) and deriving $1 - \bar{\alpha}$.

The intution can experimentally confirmed by measuring how closely we get to isotropic noise when passing samples through the forward process. For this a batch of 50 times the same image was passed through the different steps of the process and the covariance matrix was calculated. As a metric for how close the covariance matrix was to the identity covariance matrix of pure i.i.d Gaussian noise, the identity matrix was subtracted and the mean of the absolute value of the matrix calculated. The results can be seen in Fig. 4.2 and confirm the intuition: When using linear scheduling we reach the closest point to pure noise already after around 600 steps for small images, and after around 700 for larger images. Cosine scheduling also performs worse on smaller images than on larger ones, but is still capable providing value for at least 850 timesteps.

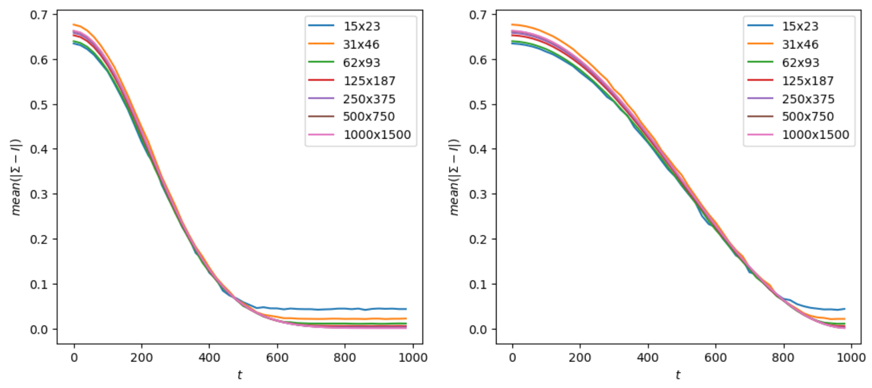

Figure 4.2: Closeness to noise for linear scheduling (left) and cosine scheduling (right).

Chapter 5

Discussion

The discussion section gives an interpretation of what you have done [2]:

- *What do your results mean?* Here you discuss, but you do not recapitulate results. Describe principles, relationships and generalizations shown. Also, mention inconsistencies or exceptions you found.
- *How do your results relate to other's work?* Show how your work agrees or disagrees with other's work. Here you can rely on the information you presented in the "related work" section.
- *What are implications and applications of your work?* State how your methods may be applied and what implications might be.

Make sure that introduction/related work and the discussion section act as a pair, i.e. "be sure the discussion section answers what the introduction section asked" [2].

Chapter 6

Conclusion

List the conclusions of your work and give evidence for these. Often, the discussion and the conclusion sections are fused.

References

[1] Jooyoung Choi et al. *ILVR: Conditioning Method for Denoising Diffusion Probabilistic Models*. 2021. arXiv: `2108.02938 [cs.CV]`.

[2] R.A. Day and B. Gastel. *How to Write and Publish a Scientific Paper*. Cambridge University Press, 2006.

[3] Prafulla Dhariwal and Alex Nichol. *Diffusion Models Beat GANs on Image Synthesis*. 2021. arXiv: `2105.05233 [cs.LG]`.

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

[5] Jonathan Ho, Ajay Jain, and Pieter Abbeel. *Denoising Diffusion Probabilistic Models*. 2020. arXiv: `2006.11239 [cs.LG]`.

[6] Diederik P Kingma and Max Welling. *Auto-Encoding Variational Bayes*. 2022. arXiv: `1312.6114 [stat.ML]`.

[7] Diederik P. Kingma et al. *Variational Diffusion Models*. 2023. arXiv: `2107.00630 [cs.LG]`.

[8] Mark A. Kramer. "Nonlinear principal component analysis using autoassociative neural networks". In: *AIChE Journal* 37.2 (1991), pp. 233–243. DOI: `https://doi.org/10.1002/aic.690370209`. eprint: `https://aiche.onlinelibrary.wiley.com/doi/pdf/10.1002/aic.690370209`. URL: `https://aiche.onlinelibrary.wiley.com/doi/abs/10.1002/aic.690370209`.

[9] Andreas Lugmayr et al. *RePaint: Inpainting using Denoising Diffusion Probabilistic Models*. 2022. arXiv: `2201.09865 [cs.CV]`.

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

[11] Jascha Sohl-Dickstein et al. *Deep Unsupervised Learning using Nonequilibrium Thermodynamics*. 2015. arXiv: `1503.03585 [cs.LG]`.

[12] Yang Song and Stefano Ermon. *Generative Modeling by Estimating Gradients of the Data Distribution*. 2020. arXiv: `1907.05600 [cs.LG]`.

[13] Yang Song et al. *Score-Based Generative Modeling through Stochastic Differential Equations*. 2021. arXiv: `2011.13456 [cs.LG]`.

Appendix A

The First Appendix

In the appendix, list the following material:

- Data (evaluation tables, graphs etc.)
- Program code
- Further material

Appendix B

Extended Derivations

B.1 Forward Process Closed-Form

Starting with transition distributions

$$q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}(\sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t I) \tag{B.1}$$

the reparameterization $\alpha = 1 - \beta$ is introduced

$$q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}(\sqrt{\alpha_t}\boldsymbol{x}_{t-1}, (1-\alpha)I) \tag{B.2}$$

which can also be formulated as

$$q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \sqrt{\alpha_t}\boldsymbol{x}_{t-1} + \sqrt{1-\alpha_t}\mathcal{N}(\mathbf{0}, \boldsymbol{I}). \tag{B.3}$$

For coherent indexing it is beneficial to switch to notation using random variables

$$\boldsymbol{x}_t = \sqrt{\alpha_t}\boldsymbol{x}_{t-1} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon_{t-1}} \tag{B.4}$$

where $\boldsymbol{\epsilon_{t-1}} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and the earlier $\boldsymbol{x}_t$ can be recursively inserted into the formula. Recalling that the sum $Z = X + Y$ of two normally distributed random variables $X \sim \mathcal{N}(\mu_X, \sigma_Y^2)$ and $Y \sim \mathcal{N}(\mu_Y, \sigma_Y^2)$ is again normally distributed according to $Z \sim \mathcal{N}(\mu_X + \mu_Y, \sigma_X^2 + \sigma_Y^2)$

$$\begin{aligned}
x_t &= \sqrt{\alpha_t}\left(\sqrt{\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{1-\alpha_{t-1}}\boldsymbol{\epsilon}_{t-2}\right) + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_{t-1} && \text{(B.5)}\\
&= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{\alpha_t(1-\alpha_{t-1})}\boldsymbol{\epsilon}_{t-2} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_{t-1} && \text{(B.6)}\\
&= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{\alpha_t(1-\alpha_{t-1}) + (1-\alpha_t)}\bar{\boldsymbol{\epsilon}}_{t-2} && \text{(B.7)}
\end{aligned}$$

where $\bar{\boldsymbol{\epsilon}}_{t-2}$ is the sum of the random variables up to $t-2$ (again Gaussian). The second term can of course be simplified to

$$\boldsymbol{x}_t = \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\bar{\boldsymbol{\epsilon}}_{t-2} \tag{B.8}$$

which is exactly the same form as in Eq. B.4. Therefore the final form is

$$\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_{t-2} + \sqrt{1-\bar{\alpha}_t}\bar{\boldsymbol{\epsilon}}_{t-2} \tag{B.9}$$

with $\bar{\alpha}_t = \prod_{s=1}^{t}\alpha_s$ as before.