ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

CVL Computer Vision Lab

My first and last thesis
Subtitle Subtitle Subtitle

Master's Thesis

Eckhart Immerheiser
Department of ...

Advisors: Egon Hasenfratz-Schreier
Supervisor: Prof. Dr. Luc van Gool

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

Describe the other's work in the field, with the following purposes in mind:

- *Is the overview concise?* Give an overview of the most relevant work to the needed extent. Make sure the reader can understand your work without referring to other literature.
- *Does the compilation of work help to define the "niche" you are working in?* Another purpose of this section is to lay the groundwork for showing that you did significant work. The selection and presentation of the related work should enable you to name the implications, differences and similarities sufficiently in the "discussion" section.

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.

Chapter 4

Generative Machine Learning

4.1 Bayesian Formulation of 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

$$p(z|x) = \frac{p(x|z)p(z)}{p(x)} \tag{4.1}$$

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 latent space (unobserved) from which – after training – can be sampled to generate new data, whereas x is the random variable that represents the training samples (observed space).

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.

4.2 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) [5]. 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. [6] 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 learn to recreate input samples, but is rather 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. The latent distribution is chosen such that

sampling from it is easy, which allows the neural network to create new data samples (e.g. a multivariate Gaussian).

Marginalizing $p_\theta(x) = \int p_{\theta_{NN}}(x|z)p_{\theta_z}(z)dz = \frac{p_{\theta_{NN_{out}}}(x|z)p_{\theta_z}(z)}{p_{\theta_{NN_{in}}}(z|x)}$ requires another approximation of the intractable posterior $p_{\theta_{NN_{in}}}(z|x)$ during training. A schematic of a VAE is shown in Fig. 4.1.

The neural networks usually do not contain stochastic layers, but are deterministic mappings between latent space and sample space $x \sim p_{\theta_{NN_{out}}}(x|z) \Rightarrow x = f_{\theta_{NN_{out}}}(z)$. The hope is, that after training the encoder $p_{\theta_{NN_{in}}}$ can be removed and sampling from z is the same as sampling from x.

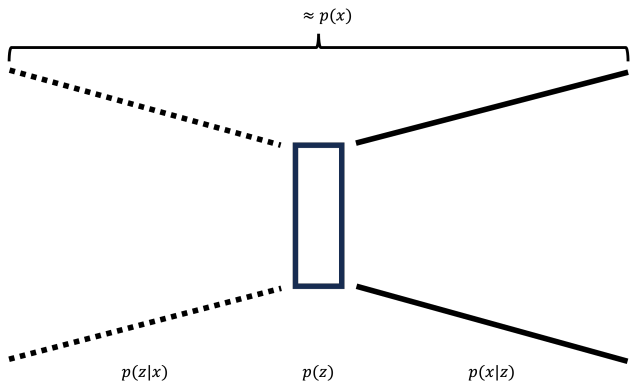

Figure 4.1: VAE schematic: $p(x)$ is approximated through a latent variable model where posterior and likelihood are modeled with neural networks and the prior on the latent variable is modeled through a simple parameterized distribution (often Gaussian). The hope is, that after training, sampling from $p(z)$ and passing it through the neural network $p_{\theta_{NN}}(x|z)$, is the same as sampling from $p(x)$.

4.3 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 almost all information has been destroyed and the distribution $q(\boldsymbol{x}_T)$ largely follows an isotropic Gaussian distribution – a Gaussian distribution with the identity matrix as covariance matrix, but a non-zero means vector.

Once the network is trained to create a less noisy image $\boldsymbol{x}_{t-1}$ from $\boldsymbol{x}_t$, 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 it many times through the network until all noise is removed.

4.3.1 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 at variances ($\sqrt{1-\beta}$ and β) of the added noise at every timestep. [4] The whole process can be expressed as a Markov chain (depicted in

Fig. 4.2)

$$q(\boldsymbol{x}_T|\boldsymbol{x}_0) = q(\boldsymbol{x}_0)\prod_{t=1}^{T} q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) \tag{4.2}$$

with 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. 4.3.

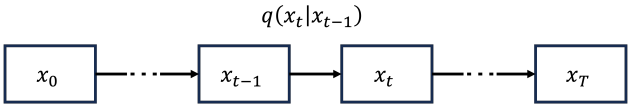

Figure 4.2: Forward Diffusion Process: An image is iteratively destroyed by adding normally distributed noise, according to a schedule. This represents a Markov process with the transition probability $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$.

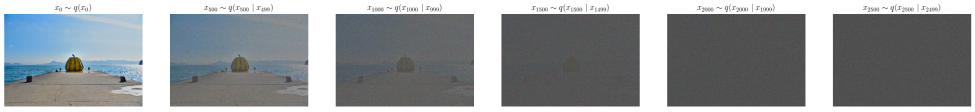

Figure 4.3: Example of Iterative Image Destruction through Forward Diffusion Process: The indices give the time step in the iterative destruction process, where β was created according to a linear noise variance schedule (5000 steps from in the 0.001 to 0.02 range and picture resolution of 4016 by 6016 pixels).

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. [4] For this, the variance schedule is first reparameterized as $1 - \beta = \alpha$

$$q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}(\sqrt{\alpha_t}\boldsymbol{x}_{t-1}, (1-\alpha_t)\boldsymbol{I}) \tag{4.3}$$

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

$$q(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0, (1-\bar{\alpha}_t)\boldsymbol{I}) \tag{4.4}$$

The derivation that leads from Eq. 4.3 to Eq. 4.4 will be 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. [4] Nichol et al. later found that a cosine-based schedule gives better results, since it does not destruct information quite as quickly, making it more informative in the last few timesteps. [7] Own experiments exploring above mentioned parameters are explained in 5.1.

4.3.2 Reverse Diffusion Process

DDPMs can be viewed as latent space models in a similar way that Generative Adversarial Nets or Variational Autoencoders can. [3, 5]

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

$$q(\boldsymbol{x}_0|\boldsymbol{x}_T) = q(\boldsymbol{x}_T)\prod_{t=T}^{1} q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) \tag{4.5}$$

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.

4.4 Latent Variable Models Compared

	VAE	GAN	DDPM
prior	$p(z)$, parameterized of any shape	$p(z)$, parameterized of any shape	$p(\boldsymbol{x}_t)$, same shape as samples
posterior	$p(z\|x)$, modeled with neural network	$p(z\|x)$, modeled with loss function	$q(\boldsymbol{x}_t\|\boldsymbol{x}_{t-1})$, modeled as step in forward diffusion process
likelihood	$p(x\|z) = f_{NN}(z)$, modeled with neural network	$p(x\|z) = f_{NN}(z)$, modeled with neural network	$q(\boldsymbol{x}_{t-1}\|\boldsymbol{x}_t) = \mathcal{N}(k_1 f_{NN}, k_2 f_{NN})$, modeled with Gaussian sampling with parameters estimated by neural network

4.5 Loss Functions

In generative machine learning we would want our model to learn the distribution that generated out training examples. Often this distribution is conditioned on some description (e.g. text) or on the corruption process in our case, where we use generative models to solve inverse problems. Assuming our original data distribution (of images) is $p(x)$, then we try to find a parameterized variational machine learning model ($q_\theta(x)$) that will closely match the data distribution.

In order for this $q_\theta(x)$ to be trained we need a differentiable loss function that expresses "closeness" in a distributional sense. The usual approach to this is to use the Kullback-Leibler (KL) divergence.

4.5.1 Kullback-Leibler Divergence

4.5.2 Wasserstein Distance

A different approach to comparing the similarity of distributions is the Wasserstein metric, successfully used in the Wasserstein GAN. [1].

4.5.3 Variational Lower Bound

As mentioned previously, the goal is to approximate a true data distribution $p^*(x)$ with a parameterized distribution $p_\theta(x) = \int p_\theta(x|z)p(z)dz$, from which sampling is easy, since the prior $p(z)$ is very simple.

Chapter 5

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.

5.1 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. [4, 7] 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.5.1 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.

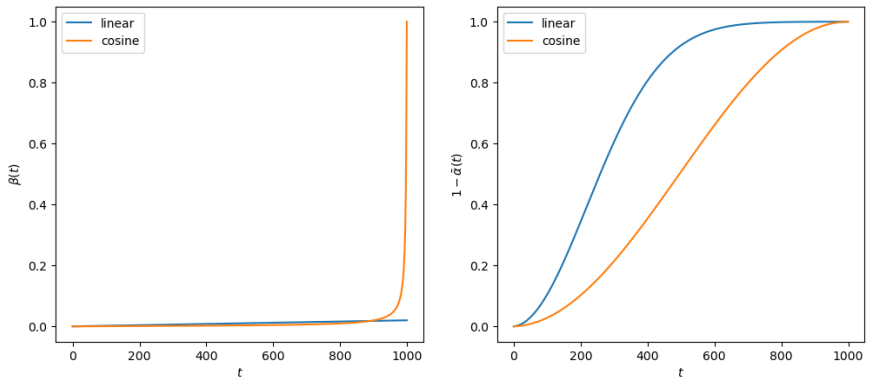

Figure 5.1: Variance Schedule Approaches: Modeling the $1 - \bar{\alpha}$ as an approximate linear function (right cosine) and deriving β (left cosine), or modeling β as a linear function (left linear) and deriving $1 - \bar{\alpha}$.

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. 5.2 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.

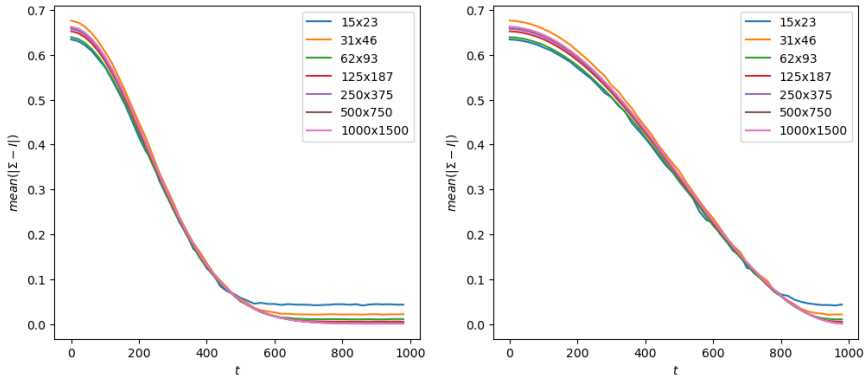

Figure 5.2: Closeness to noise for linear scheduling (left) and cosine scheduling (right).

Chapter 6

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 7

Conclusion

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

References

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. *Wasserstein GAN*. 2017. arXiv: 1701.07875 [stat.ML].

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

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

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

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

[6] 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.

[7] Alex Nichol and Prafulla Dhariwal. *Improved Denoising Diffusion Probabilistic Models*. 2021. arXiv: 2102.09672 [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 $Z \sim \mathcal{N}(\mu_X + \mu_Y, \sigma_X^2 + \sigma_Y^2)$

$$x_t = \sqrt{\alpha_t}\left(\sqrt{\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{1-\alpha_{t-1}}\boldsymbol{\epsilon}_{t-2}\right) + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_{t-1} \tag{B.5}$$

$$= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{\alpha_t(1-\alpha_{t-1})}\boldsymbol{\epsilon}_{t-2} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_{t-1} \tag{B.6}$$

$$= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{\alpha_t(1-\alpha_{t-1}) + (1-\alpha_t)}\bar{\boldsymbol{\epsilon}}_{t-2} \tag{B.7}$$

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.