DTU Compute
Department of Applied Mathematics and Computer Science

Denoising Diffusion Probabilistic Models for Super Resolution

Master Thesis

Denoising Diffusion Probabilistic Models for Super Resolution

Master Thesis
July, 2023

By
Olgeir Ingi Árnason

Cover photo: Vibeke Hempler, 2012
Published by: DTU, Department of Applied Mathematics and Computer Science, Richard Petersens Plads , Building 324, 2800 Kgs. Lyngby Denmark
www.compute.dtu.dk
ISSN: [0000-0000] (electronic version)
ISBN: [000-00-0000-000-0] (electronic version)

ISSN: [0000-0000] (printed version)
ISBN: [000-00-0000-000-0] (printed version)

Approval

This thesis has been prepared over five months at the Section for Visual Computing, Department of Applied Mathematics Computer Science, at the Technical University of Denmark, DTU, in partial fulfilment for the degree Master of Science in Engineering, MSc Eng.

It is assumed that the reader has a basic knowledge in the areas of deep learning and probability theory.

Olgeir Ingi Árnason - s212564

.....

Signature

.....

Date

Abstract

Single image super resolution techniques are used to generate high resolution images from a single low resolution image. Deep learning methods such as GANs and flow-based models have outperformed traditional methods, achieving state of the art results. These methods do however have known shortcomings. Namely, GANs have been shown to be prone to mode collapse and are difficult to train whereas flow based methods are subject to strong architectural constraints which enforces the use of models with large footprints. These shortfalls motivate the use of different models that are able to mitigate or eliminate these flaws.

Diffusion models have recently gained traction in the field of generative modeling due to their impressive performance on such tasks, outperforming state of the art methods in some cases. This motivates the use of diffusion models for tasks such as SR.

In this thesis we test multiple diffusion models with different parameters on three SR ratios (2x, 4x, 8x) and explore the effects of said diffusion model parameters on different super resolution ratios.

We report that the 500 step cosine schedule models achieve the best results in each SR ratio category:

- 2x SR ratio: 36.356 HR-PSNR, 37.852 LR-PSNR and 0.950 SSIM.
- 4x SR ratio. 35.113 HR-PSNR, 35.744 LR-PSNR and 0.880 SSIM.
- 8x SR ratio: 33.708 HR-PSNR, 34.040 LR-PSNR and 0.751 SSIM.

Acknowledgements

[Vedrana Andersen Dahl], [Associate Professor], [Supervisor]
[Department of Applied Mathematics and Computer Science]

[Anders Bjorholm Dahl], [Professor, Head of section], [Supervisor]
[Department of Applied Mathematics and Computer Science]

[Patrick Møller Jensen], [Postdoc], [Co-supervisor]
[Department of Applied Mathematics and Computer Science]

Contents

1 Introduction

In recent years, the field of computer vision has witnessed remarkable advancements, particularly in the domain of image super resolution (SR). SR techniques aim to enhance the resolution and detail of low-resolution images, enabling applications in various domains, including medical imaging, satellite imaging, surveillance, and consumer electronics. Among the diverse approaches for SR, diffusion models have gained significant attention due to their ability to generate high-quality images while preserving fine details.

Diffusion models have demonstrated impressive performance in various computer vision tasks, including image generation, denoising, and inpainting. These models utilize iterative processes to transform low-resolution images into high-resolution counterparts by iteratively refining the pixel values. Unlike traditional SR methods, diffusion models are data-driven and do not rely on explicit image priors or handcrafted features. Instead, they learn the distribution of high-resolution images from a given dataset and generate realistic results while also mitigating some of the shortcomings of previous learning based methods.

While diffusion models have shown promising results, there is still a need for a comprehensive understanding of their underlying mechanisms and the impact of various factors on their performance. To bridge this gap, this thesis presents an ablation study on diffusion models for SR on different SR ratios. The primary objective of this study is to dissect and analyze the key components and design choices of diffusion models to identify their strengths, weaknesses, and limitations.

1.1 Research Objectives

The main objectives of this thesis are as follows:

1.1.1 Investigation of Diffusion Model Components

- Explore different diffusion model components and describe their impact on SR performance.
- Analyze the effectiveness of various diffusion model components, such as number of forward process steps and noise schedules on different SR ratios.

1.1.2 Examination of Ill-Posed Problem Solving of Diffusion Models

- Implement various model performance metrics that capture the models ability to solve ill-posed problems such as SR.
- Explore the effect of different SR ratios on aforementioned performance metrics and gauge the capabilities and limitations of diffusion models to solve such problems.

1.2 Significance and Expected Contributions

The findings of this ablation study on diffusion models will provide a deeper understanding of diffusion models' capabilities and limitations, allowing researchers and practitioners to make informed decisions when designing and utilizing these models.

The key contributions of this study are as follows:

- Identification of crucial architectural components for diffusion models in the context of SR, providing insights into their impact on performance.

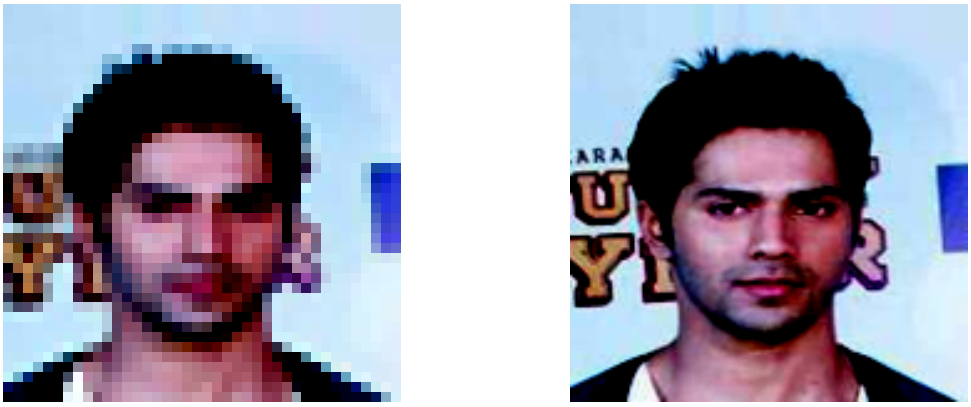
- Evaluation and comparison of model performance on various SR ratios to gauge diffusion model abilities to solve ill-posed problems.

By addressing these objectives and providing insights into diffusion models for SR, this thesis aims to pave the way for further advancements in the field, facilitating the development of more effective diffusion model methods to solve SR problems.

2 Background

2.1 Super Resolution

Super resolution (SR) refers to the task of enhancing the resolution of an image or video beyond its original low-resolution (LR) form. In this thesis, we refer to increased resolution when the pixel count is increased. The SR problem has attracted significant attention in various fields, including computer vision, image processing, and multimedia applications. The ability to generate high-quality, high-resolution images from low-resolution inputs has wide-ranging practical implications, such as in surveillance, medical imaging, satellite imaging, and digital photography. Figure 2.1 illustrates the goal of SR where a HR image is created from its corresponding LR image.

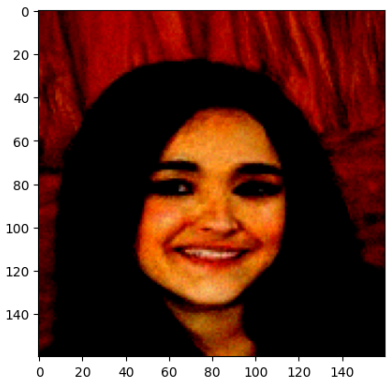


(a) LR image. (b) HR image.

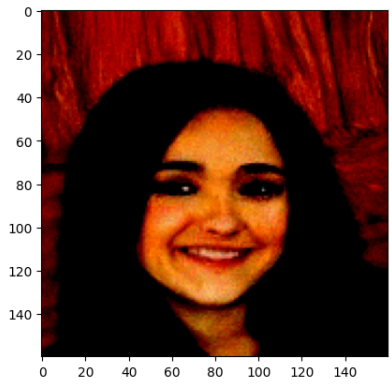
Figure 2.1: Super resolution aims to reconstruct the HR image from the LR image.

Super resolution problems are often considered ill-posed, referring to a situation where a problem lacks unique, stable solutions. In the context of super resolution, this means that there are typically multiple possible HR images that could correspond to a given LR image, and the problem of selecting the ”correct” high-resolution image becomes challenging or impossible. In other words, the ill-posed nature of super resolution problems implies that there is an inherent ambiguity in the mapping from low-resolution to high-resolution images. Without additional constraints or prior knowledge, it becomes challenging to uniquely determine the exact details and structures that were present in the original high-resolution image.

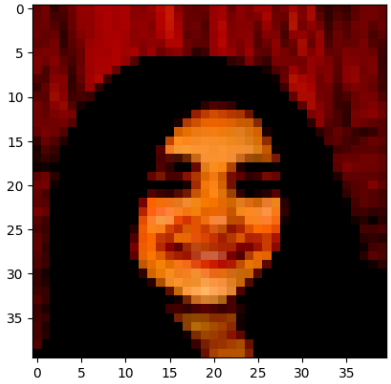
The ill-posed nature of super resolution problems arises due to the inherent loss of information during the process of downsampling or capturing low-resolution images. When an image is captured or downsampled, high-frequency details and fine-grained information are lost, resulting in a lossy representation. Consequently, the missing high-frequency information cannot be uniquely determined from the low-resolution data alone, making it difficult to obtain a single ground truth solution. This property is reflected visually in figure 2.2.



(a) HR image 1.



(b) HR image 2.



(c) Corresponding LR image.

Figure 2.2: Two different HR images that get downscaled to the same LR image.

In the past, several methods have been proposed to address the super resolution problem. Early approaches were based on interpolation techniques, which aimed to estimate the high-resolution (HR) image by upsampling the available LR data. These methods typically employed simple interpolation algorithms, such as bilinear or bicubic interpolation. While these techniques provided a quick and straightforward solution, they often resulted in blurred or unrealistic HR images due to the lack of accurate information about the missing high-frequency details.

To overcome the limitations of interpolation-based methods, researchers turned to more sophisticated approaches, including example-based methods, regularization-based methods, and learning-based methods. Example-based methods relied on external databases of HR images to learn the relationship between LR and HR image pairs, allowing them to reconstruct the missing high-frequency information. These methods used techniques like sparse coding, patch matching, and non-local self-similarity to find similar image patches and transfer the details to the LR image.

Regularization-based methods exploited image priors and assumptions to regularize the SR problem. Techniques like total variation regularization and sparse representation modeling were used to promote sparsity and encourage the reconstruction of high-frequency details. These methods leveraged the inherent properties of natural images to generate more visually pleasing HR results.

However, the recent breakthroughs in deep learning have revolutionized the field of su-

per resolution. Deep learning-based methods, particularly convolutional neural networks (CNNs), have shown exceptional performance in SR tasks. These methods learn an end-to-end mapping function from LR to HR images using large-scale datasets. The deep networks are trained to capture complex relationships between LR and HR images, enabling them to generate highly realistic and visually appealing HR images. Notable architectures in deep learning-based SR include PSNR oriented methods, flow based methods and GANs. However, PSNR methods have been known to cause over smoothing of the HR image, flow based methods suffer from very large model footprints due to strong architectural contraints and lastly, GANs have been shown to be prone to mode collapse and tend to be difficult to train.

Current research in super resolution continues to explore new directions and improvements. Some recent advancements include attention mechanisms, which enable the network to focus on relevant image regions, and GANs, which enhance perceptual quality and texture details. Additionally, more recent methods such as diffusion models have gained traction due to their performance and ease of training.

2.2 Convolutional Neural Networks

Convolutional Neural Networks have been shown to be very effective for generative tasks such as Super-Resolution (SR) problems due to their ability to automatically learn and extract relevant features from low-resolution images. The convolutional layers capture local patterns, while pooling layers aid in reducing spatial dimensions, making them effective at learning hierarchical representations in images. By leveraging this, CNNs have been effective at solving SR problems and significantly enhance the resolution of images, providing sharper and more detailed results than traditional interpolation techniques.

The main idea behind convolutional CNNs as opposed to a traditional artificial neural network (ANN) is that in CNNs each unit receives input only from a small subset of neurons in the previous layer whereas neurons in the current layer of fully connected networks are connected to every neuron in the previous layer. The convolution procedure is done using kernels where each kernel has learnable weights and is slid over the input performing element wise multiplication and summation. Put mathematically a convolution operation is described as follows

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n-m]$$

Figure 2.3 shows a visual representation of what a convolution really is computing. Moreover, in a convolutional neural network many of these convolution operations are stacked one after the other creating multiple kernels each with learnable weights.

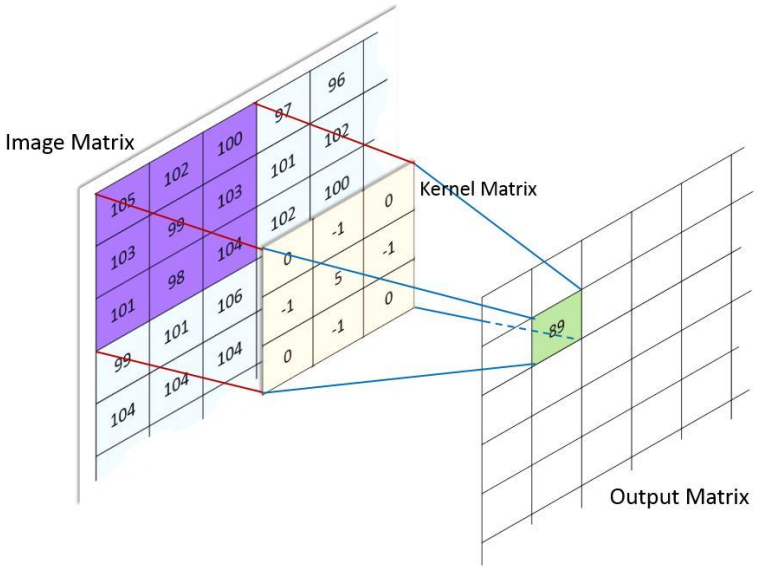


Figure 2.3: Convolution operation visualized.
Image source: https://brilliant.org/wiki/convolutional-neural-network/

2.2.1 Pooling

The pooling operation is used to compress the size of the convolution layers. The factor by which the convolution layers are compressed is called the downsampling factor which is determined by the stride of the pooling operation. Several pooling methods exist such as average pooling and median pooling but by far the most popular pooling method is the max pooling which essentially reduces the dimensions of the input by picking the maximum value of a neighbourhood of the current input. The max pooling method has also been shown to make CNNs more robust to small perturbations of the input making it a good choice in most situations.

2.2.2 Activation functions

The activation functions purpose is to add non-linearity to the model since a series of linear operations can be reduced to a single linear operation and limits the models ability to learn complex functions. A plethora of different activation functions for machine learning exist but the most popular one is the ReLU function also known as the Rectified Linear Unit defined mathematically as $\text{ReLU}(x) = \max(0, x)$.

2.2.3 U-net architecture

The U-Net architecture is a popular CNN architecture commonly used for image segmentation and generative tasks such as SR. It is also the architecture most commonly utilized by diffusion models. The U-Net architecture gets its name from its U-shaped design, which consists of an encoder path and a corresponding decoder path. The encoder path performs a pooling operation such as max pooling (downsampling), while the decoder path performs upsampling and reconstructs the image. The resulting network architecture has the same input and output dimensions as illustrated by figure 2.4.

The encoder path usually consists of convolutional layers followed by activation functions and pooling layers while the decoder path consists of upsampling operations followed by convolutional layers and activation functions.

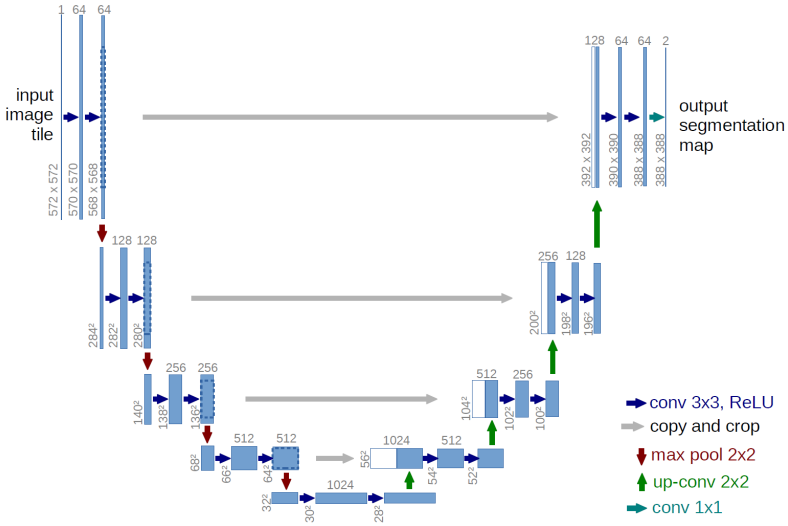


Figure 2.4: General U-net architecture (Ronneberger et al. 2015).

2.2.4 Embeddings

Embeddings in neural networks are most often used to incorporate additional information in the model that cannot be used directly as input for reasons such as architectural constraints for example. In the context of SR, the LR image is often embedded into the model such that it guides the convolution process from beginning to end. This technique will be used extensively in this thesis as certain architectural constraints of diffusion models make it difficult to take the LR image as direct input of the model.

Additionally, embeddings are often learned before being incorporated in the model. For instance, in the case of SR, the LR image is often passed through an encoder to either adjust the dimensionality of the LR image to the model dimensions such that it can be embedded into the model while also making it possible for the LR encoder to learn to better seperate different LR images and leaving less work for the U-net to do.

Two methods of embedding information into neural nets are most commonly used. The first one consisting of adding or multiplying the values from the embedding directly into the network at hand thus altering the output values of a certain layer of the network. The second method involves concatenating the embedding into the model instead of adding it directly to the output of the neurons. In this thesis we solely utilize the second method for all models.

3 Diffusion Models

Diffusion models have gained significant attention and popularity in the field of machine learning recently. These models provide a powerful framework for modeling complex datasets, particularly in the context of sequential/structured data. They have shown promising results in generative modeling such as image and sound synthesis. At their core, diffusion models are probabilistic models that capture the evolution of data over time. They aim to describe the dynamics of data generation processes and provide a mechanism for generating realistic samples from the underlying data distribution. Unlike traditional generative models, such as autoregressive models or GANs, diffusion models focus on modeling the transformation process from a simple distribution, often a Gaussian distribution, to the target data distribution. This transformation of the data is visualized by a stochastic differential equation for a simple 1D case in figure 3.1.

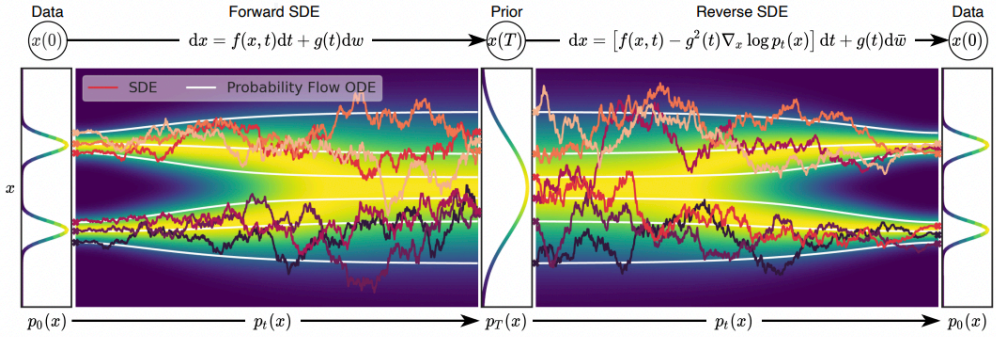


Figure 3.1: Transformation process from an underlying data distribution to a Gaussian distribution and back to the underlying data distribution illustrated by an SDE (Song et al. 2021).

Diffusion models are latent variable models parameterized by a Markov chain trained using the variational lower bound of the negative log likelihood. Diffusion models can typically be split into two parts, the first being the forward process and the latter being the reverse process which is learned by the model. The Markov chain $q(x_{1:T}|x_0)$ (also called the forward process or diffusion process) is a chain of length T that incrementally adds Gaussian noise to the data while the reverse process $p_\theta(x_{t-1}|x_t)$ is also a Markov chain of the same length as the forward process with learned transitions starting at $p(x_T)$. The graphical representation of diffusion models is illustrated in figure 3.2 where x_0 is the observed image and the x_t are the latents.

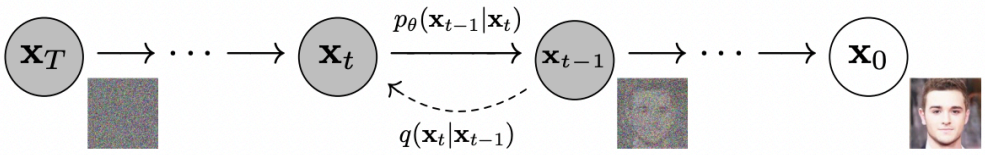


Figure 3.2: Graphical model of diffusion models. (Ho et al. 2020)

3.1 Forward Process

As mentioned already, the forward process $q(x_{1:T}|x_0)$ is a Markov chain of length T that incrementally adds gaussian noise to the data until it has been completely destroyed. The noise added at each timestep t is determined by the noise schedule β_t which are held as constant hyperparameters. The forward process is parameterized and visualized by the following:

$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1}) \tag{3.1}$$

Where

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t \mathbf{I}) \tag{3.2}$$

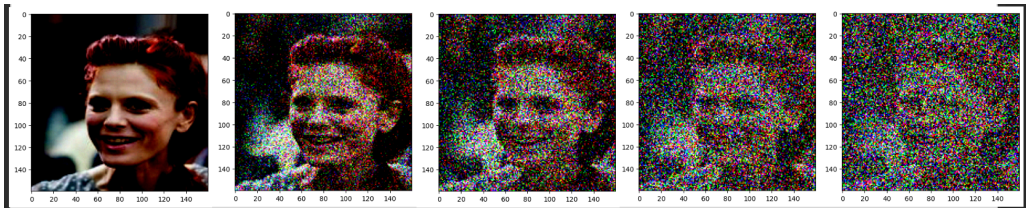


Figure 3.3: Truncated linear diffusion process in 40 step increments.

The visualized forward process clearly illustrates the importance of the noise schedule and choice of β_t since too many steps in the forward process will lead to redundant steps that do nothing except for adding noise to an already completely gaussian image. Two methods of defining a noise schedule are generally used which is either a linear noise schedule as done in (Ho et al. 2020) or a cosine schedule as done in (Nichol & Dhariwal 2021).

Consequently, the choice of the amount of steps in the forward process is also of great importance for diffusion models as each β_t corresponds to a certain timestep and therefore the amount of steps in the forward process and the noise schedule itself are closely tied together and have been shown to greatly affect the performance of diffusion models. Both the forward process steps and the type of noise schedule were tested for their effect on diffusion model performance on different SR ratios in this thesis.

While the noise schedule and forward process steps have been shown to contribute to the overall performance of diffusion models, there is a property of the forward process that makes the training of diffusion models very efficient and in general, tractable for practical applications. This property consists of the fact that since the forward process incrementally adds Gaussian noise to the data, it therefore admits a closed form solution for sampling an arbitrary step in the Markov chain. By reparameterizing such that $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod^t \alpha_t$ we have

$$\begin{aligned} x_t &= \sqrt{\alpha_t}x_{t-1} + \sqrt{1-\alpha_t}\epsilon_{t-1} \\ &= \sqrt{\alpha_t\alpha_{t-1}}x_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\hat{\epsilon}_{t-2} \\ &= \dots \\ &= \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon_t \end{aligned}$$

Where $\epsilon \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and $\hat{\epsilon}$ merges two Gaussian distributions. The arbitrary timestep sample then becomes:

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)\mathbf{I}) \tag{3.3}$$

This property breaks up the forward process Markov chain and will make the training of diffusion models much more efficient as will be discussed in the training section of this chapter.

3.2 Reverse Process

The goal of diffusion models is to learn how to reverse the forward process $q(x_{t-1}|x_t)$ which will in turn allow us to sample the underlying distribution of the data from Gaussian noise input $\boldsymbol{x}_T \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$. However, estimating $q(x_{t-1}|x_t)$ is in general intractable since it is a function of the entire dataset. Instead, neural networks are utilized in order to learn how to approximate the conditional probabilities denoted by

$$p_\theta(\boldsymbol{x}_{0:T}) = p_\theta(\boldsymbol{x}_T) \prod_{t=1}^{T} p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) \tag{3.4}$$

Where

$$p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_{t-1}; \boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t), \boldsymbol{\Sigma}_\theta(\boldsymbol{x}_t, t)) \tag{3.5}$$

It is worth noting that the reverse conditional probabilites are only tractable when conditioned on x_0:

$$q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_{t-1}; \tilde{\mu}(\boldsymbol{x}_t, \boldsymbol{x}_0), \tilde{\beta}_t \boldsymbol{I}) \tag{3.6}$$

This means that a model can not be trained to reverse the forward process on one dataset for example and then do it correctly on images from another dataset. Note also that in the case of SR, since the LR image is typically embedded into the model, the conditional probabilities in equation (3.5) change such that they include the LR image and become

$$p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \text{LR}) = \mathcal{N}(\boldsymbol{x}_{t-1}; \boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t, \text{LR}), \boldsymbol{\Sigma}_\theta(\boldsymbol{x}_t, t, \text{LR})) \tag{3.7}$$

Figure 3.4 illustrates reverse process done by a fully trained model.

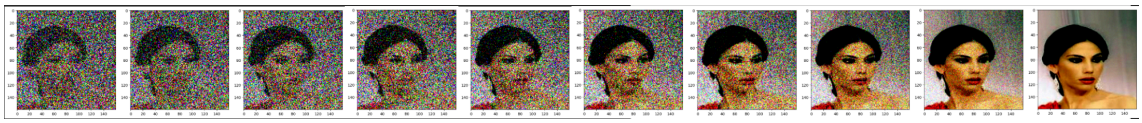


Figure 3.4: Truncated reverse process in 20 step increments.

3.3 Noise schedules

As already mentioned, the type of noise schedule has been shown to affect model performance of diffusion models since noisier samples in the forward process contribute less to the optimization of the negative log likelihood (NLL) (Nichol Dhariwal 2021).

Note that in diffusion models, there is a trade-off between the number of steps in the forward process and computational resources during inference since the model will have to do one forward pass for every step in the forward process. This means that if a model has 1000 forward process steps, a 1000 forward passes will have to be done in order to create a single sample.

This trade-off and the fact that noisier samples of the forward process contribute less to the optimization of the NLL motivates the use of noise schedules beyond the constant noise schedule where all β_t are set to a constant. Ideally, the forward process has fewer steps while the noise schedule ensures that the samples in the forward process contain as many steps as possible that contain little noise while still completely contaminating the data with noise by the end of the forward process. This would minimize the computational resources during inference while also allowing for faster optimization of the NLL.

Two types of noise schedules are typically used for diffusion models, the linear noise schedule and the cosine noise schedule. Figure 3.5 reflects the difference between the two schedules in a 200 step forward process. Note that the difference between to two schedules becomes more apparent in longer forward processes.

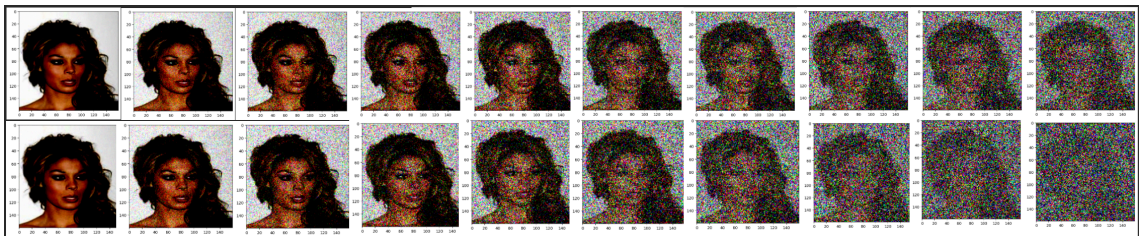


Figure 3.5: Difference between the linear noise schedule (above) and the cosine noise schedule (below).

Recall that the forward process is parameterized by

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

The linear noise schedule uses linearly increasing β_t on a pre-determined interval. The interval $[\beta_1 = 10^{-4}, \beta_T = 0.02]$ proposed in (Ho et al. (2020)) is commonly used in the diffusion model literature and will also be used as the linear noise schedule interval in this thesis.

The cosine noise schedule proposed in (Nichol & Dhariwal 2021) was designed to achieve a linear drop-off of $\bar{\alpha}_t$ near the middle of the process while making small changes near the ends of the diffusion process where $t = 0$ or $t = T$. The β_t parameters of the cosine schedule are parameterized as follows:

$$\beta_t = \text{clip}(1 - \frac{\bar{\alpha}_t}{\bar{\alpha}_{t-1}}, 0.99) \tag{3.8}$$

Where

$$\bar{\alpha}_t = \frac{f(t)}{f(0)} \tag{3.9}$$

And

$$f(t) = \cos(\frac{t/T + s}{1 + s} \cdot \frac{\pi}{2})^2 \tag{3.10}$$

Where s is a small offset set to $s = 0.08$ to prevent β_t from being too small when $t = 0$.

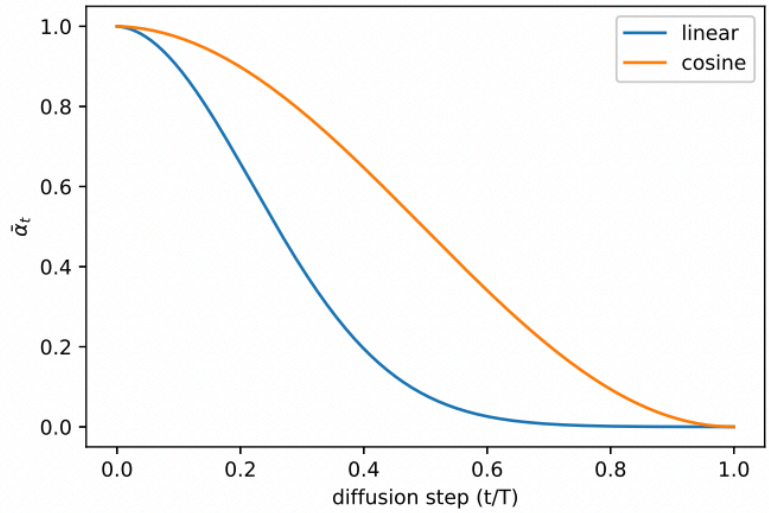


Figure 3.6: Difference of linear and cosine noise schedules in terms of $\bar{\alpha}_t$ (Nichol & Dhariwal 2021).

Figure 3.6 illustrates the difference between the two schedules in terms of $\bar{\alpha}_t$ which ultimately reflects the total noise accumulated at each timestep.

3.4 Training

Diffusion models are trained by optimizing the variational lower bound of the NLL

$$\begin{aligned}
-\log p_\theta(\boldsymbol{x}_0) &\leq -\log p_\theta(\boldsymbol{x}_0) + D_{KL}(q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)||p_\theta(\boldsymbol{x}_{1:T}|x_0)) \\
&= -\log p_\theta(\boldsymbol{x}_0) + E_{\boldsymbol{x}_{1:T}\sim q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}[\log \frac{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}{p_\theta(\boldsymbol{x}_{0:T})/p_\theta(\boldsymbol{x}_0)}] \\
&= -\log p_\theta(\boldsymbol{x}_0) + E_q[\log \frac{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}{p_\theta(\boldsymbol{x}_{0:T})} + \log p_\theta(\boldsymbol{x}_0)] \\
&= E_q[\log \frac{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}{p_\theta(\boldsymbol{x}_{0:T})}] \\
&:= L_{VLB}
\end{aligned}$$

Where p_θ is the approximated reverse process learned by the network. Additionally, the objective still needs to be converted such that it can be computed in an efficient and fast manner. The following derivations are a key step in making the training of diffusion models efficient enough for practical applications.

$$
\begin{aligned}
L_{VLB} &= E_q[\log \frac{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}{p_\theta(\boldsymbol{x}_{0:T})}] \\
&= E_q[\log \frac{\prod_{t=1}^{T} q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})}{p_\theta(\boldsymbol{x}_T)\prod_{t=1}^{T} p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}] \\
&= E_q[-\log p_\theta(\boldsymbol{x}_T) + \sum_{t=1}^{T} \log \frac{q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})}{p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}] \\
&= E_q[-\log p_\theta(\boldsymbol{x}_T) + \sum_{t=2}^{T} \log \frac{q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})}{p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)} + \log \frac{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}{p_\theta(\boldsymbol{x}_0|\boldsymbol{x}_1)}] \\
&= E_q[-\log p_\theta(\boldsymbol{x}_T) + \sum_{t=2}^{T} \log (\frac{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)}{p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)} \cdot \frac{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)}) + \log \frac{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}{p_\theta(\boldsymbol{x}_0|\boldsymbol{x}_1)}] \\
&= E_q[-\log p_\theta(\boldsymbol{x}_T) + \sum_{t=2}^{T} \log \frac{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)}{p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)} + \sum_{t=2}^{T} \log \frac{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)} + \log \frac{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}{p_\theta(\boldsymbol{x}_0|\boldsymbol{x}_1)}] \\
&= E_q[-\log p_\theta(\boldsymbol{x}_T) + \sum_{t=2}^{T} \log \frac{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)}{p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)} + \log \frac{q(\boldsymbol{x}_T|\boldsymbol{x}_0)}{q(\boldsymbol{x}_1|\boldsymbol{x}_0)} + \log \frac{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}{p_\theta(\boldsymbol{x}_0|\boldsymbol{x}_1)}] \\
&= E_q[\log \frac{q(\boldsymbol{x}_T|\boldsymbol{x}_0)}{p_\theta(\boldsymbol{x}_T)} + \sum_{t=2}^{T} \log \frac{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)}{p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)} - \log p_\theta(\boldsymbol{x}_0|\boldsymbol{x}_1)]
\end{aligned}
$$

The objective can then be further rewritten in terms of a series of KL-divergences as follows (Sohl-Dickenstein et al. (2015)):

$$
\begin{aligned}
L_{VLB} &= E_q[D_{KL}(q(\boldsymbol{x}_T|\boldsymbol{x}_0)||p_\theta(\boldsymbol{x}_T)) + \sum_{t=2}^{T} D_{KL}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)||p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)) - \log p_\theta(\boldsymbol{x}_0|\boldsymbol{x}_1)] \\
&:= L_T + L_{T-1} + \cdots + L_0
\end{aligned}
\tag{3.11}
$$

Note that both the forward process and the reverse process have the same functional form when β_t are small enough (Sohl-Dickenstein et al. (2015)). Moreover, L_{VLB} is a series of KL divergences between $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ and forward process posteriors $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$ which consequently are both Gaussian. This allows for closed form computation of the individual terms of L_{VLB} , allowing for more efficient training by removing any numerical approximations needed for computing the KL-divergence.

Furthermore, the forward process property presented in equation (3.3) breaks up the forward process, removing the dependence on each timestep sample from the previous timestep sample. This property allows for further optimization of the training efficiency of diffusion models and instead of optimizing every term of L_{VLB} at the same time, we can instead optimize random terms of L_{VLB}. By reparameterizing $\boldsymbol{\mu}_\theta$ in the learned reverse process $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_{t-1}; \boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t), \boldsymbol{\Sigma}_\theta(\boldsymbol{x}_t, t))$ the individual terms of L_{VLB} become

$$
L_t = E_{\boldsymbol{x},\boldsymbol{\epsilon}}[\frac{(1-\alpha_t)^2}{2\alpha_t(1-\bar{\alpha}_t)||\boldsymbol{\Sigma}_\theta(\boldsymbol{x}_t, t)||_2^2}||\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t, t)||^2] \tag{3.12}
$$

Additionally by using a simplified training objective proposed in (Ho et al. (2020)) they found that it resulted in better visually looking samples while also making the training more stable. Consequently, the variance term $\mathbf{\Sigma}_\theta$ can also be ignored during training, simplifying the training even more. The final simplified training objective is

$$L_t = E_{\boldsymbol{x},\boldsymbol{\epsilon}}[||\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t, t)||^2] \tag{3.13}$$

Note that the final training objective is then just the mean squared error of the noise added at timestep t and the noise predicted by the model at timestep t. While other training objectives have been proposed, they will be considered out of the scope of this thesis and all models trained in this thesis project use the simplified objective.The training sequence of diffusion models can then be summarized by algorithm 1:

Algorithm 1 Diffusion model training

1: **do**
2: $\boldsymbol{x}_0 \sim q(\boldsymbol{x}_0)$
3: $t \sim$ Uniform(1,2,...,T)
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
5: Take optimization step on the simplified training objective
6: $\nabla_\theta||\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t, t)||^2$
7: **until** done

3.5 Inference

The goal of doing inference in diffusion models is to iteratively remove noise from a completely noisy input and generate a clean and realistic sample. As mentioned already, there is a trade-off between the number of steps in the forward process and computational resources during inference. This is due to the fact that diffusion models need to do as many forward passes of the network as there are steps in the forward process. The reason for this is that during training, the model learns the individual transitions of the reverse process given the randomly drawn timestep.

Inference in diffusion models for image generation then starts from a completely Gaussian input image and iteratively removes the noise from the image until the forward process has been completely reversed. The inference sequence of diffusion models is then summarized by algorithm 2

Algorithm 2 Diffusion model inference

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

By now it should be clear that since diffusion models start their sampling process from a randomized Gaussian input, the inference is inherently stochastic. This means that SR diffusion models can create different HR images from the same LR image as illustrated by figure 3.7.

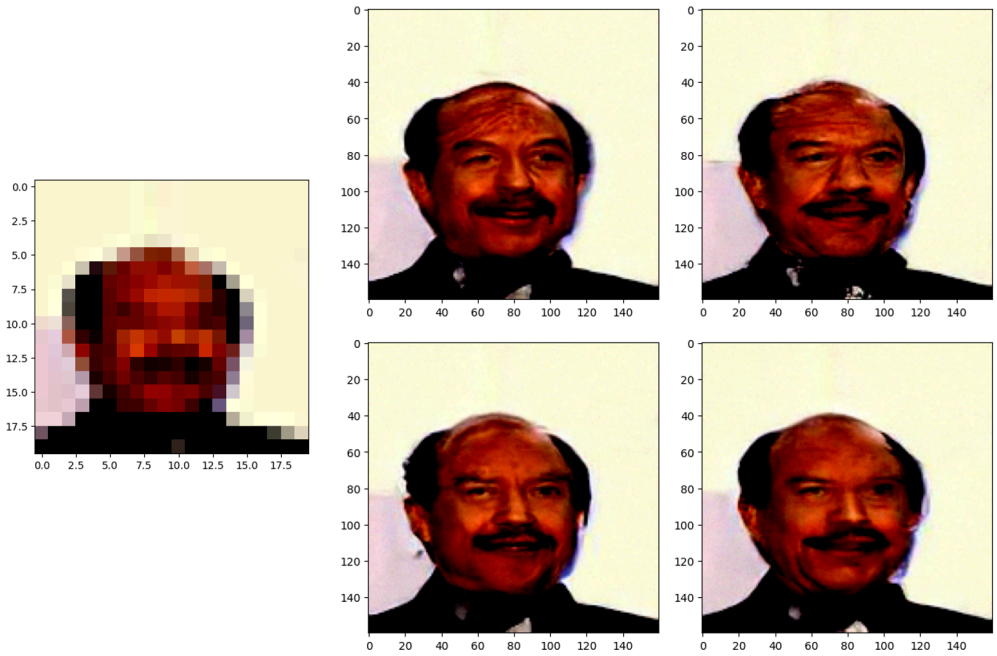


Figure 3.7: Multiple HR images (right) created from the same LR image (left).

This property is interesting by itself but it also makes performance evaluation of diffusion models an interesting topic since the images created by the model are different each time.

3.6 Model Embeddings

Due to the training objective of diffusion models being the mean squared error of the noise added by the forward process at timestep t and the noise removed by the model at the same timestep, the input of the model needs to be a sample from the forward process and the output is then the predicted previous sample in the forward process. This imposes certain architectural constraints on the model and other inputs such as LR images must be embedded into the model.

While not all diffusion models are SR models and therefore do not need to embed conditionals such as LR images into the model, they do all need to embed the current timestep that is being predicted. This is due to the fact that noise schedules typically do not have constant noise parameters β_t and therefore the noise added at each timestep varies. There are many ways to do this but in this thesis project we utilize a single hidden layer fully connected network that takes as input the current timestep and outputs a vector that gets rearranged and concatenated into the convolutional layers of the U-net.

In the case of SR, the LR variant of the desired HR image must bu embedded into the model as well as the timestep. In this thesis project we utilize a LR convolutional encoder that takes as input the LR image and ouputs an image that has the same dimensions as the desired HR image and is embedded into the first convolutional layer of the U-net such that it guides the diffusion process from beginning to end. Note that the architecture of the LR encoder varies slightly for different SR ratios such that it outputs and image that always has the same dimensions as the HR image.

4 Experimental Settings

4.1 Set-up

The hardware used in this thesis was the DTU HPC which is a computing cluster owned by DTU. More precisely, all models in this thesis were trained using the NVIDIA a100. Additionally, all models developed in this thesis were implemented in PyTorch.

4.2 Dataset

The dataset used in this thesis project was the CelebA dataset which is a large scale face attributes dataset with more than 200k images of celebrity faces, each with 40 binary attribute annotations and is to be used for non-commercial use only. Each image is an RGB colored image with dimensions 178x218. Images from the CelebA dataset also cover a large variation of poses and background clutter of over 10k different celebrities. The dataset is split into 3 parts, the first one being the training part which consists of approximately 162k images, the second one being the validation part which consists of roughly 20k images and finally the testing part which consists of the remaining 20k images.

Note that the choice of dataset is somewhat arbitrary for this thesis but we wanted a dataset that was large and complex enough for non-trivial image synthesis and the CelebA fit those criteria.

Figure 4.1: Images from the CelebA dataset.

Although the dataset contains 40 binary attributes corresponding to each image, those were not used in this thesis project simply to keep the problem at hand as SR focused as

possible.

4.3 Data Preprocessing

Before the data can be used for training the models it needs to be processed such that architectural contraints of convolutional neural networks can be accommodated. The data preprocessing is split into two parts, one for the HR images and the other for LR images. Many methods for preprocessing image data exist but in our case, we only need to downscale images to fit model restrictions and to create the corresponding LR images to guide the denoising process done by the model. Image downscaling is commonly done using simple interpolation methods such as bilinear interpolation or bicubic interpolation. In this thesis project we use bicubic interpolation for every downscaling operation of the data.

4.3.1 Bicubic Interpolation

Bicubic interpolation is a commonly used method in image processing and as a simple method for solving SR problems. The main advantages of such interpolation methods is that they are fast, easy to implement and have shown to be better than bilinear interpolation for most tasks. Bicubic interpolation uses a 4x4 patch of pixels or 16 pixel values when up or downscaling whereas bilinear interpolation uses only 4 pixels thus resulting in less accurate samples compared to bicubic interpolation. Bicubic interpolation calculates the interpolated surface as follows:

$$p(x,y) = \sum_{i=0}^{3}\sum_{j=0}^{3} a_{ij}x^i y^j$$

And the problem revolves around determining the 16 different a_{ij} coefficients.

4.3.2 HR Data Preprocessing

As already mentioned, the CelebA dataset consists of RGB images of size 178x218. Because of this consistency, HR data preprocessing is technically not needed as the pixel values of the images already have a favorable range for CNNs ($[-1, 1]$). However, in the name of dimensionality reduction and model simplification, we decided to downscale every image from 178x218 to 160x160 using bicubic interpolation. This downscaling to 160x160 is an arbitrary choice and could have been chosen differently as long as the downscaling is not of such factor that is completely distorts the image. The 160x160 images are then considered as HR ground truth images and are diffused and denoised in the training process. An important note here is that this downscaling of the HR data maintains the ill-posed and ambiguous nature of the SR problem and should not affect the comparisons of different models.

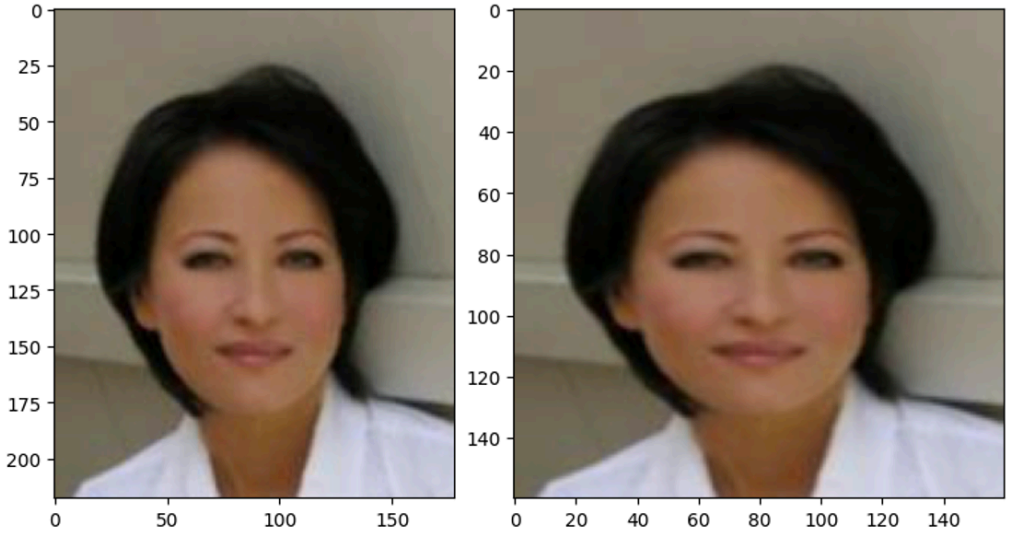


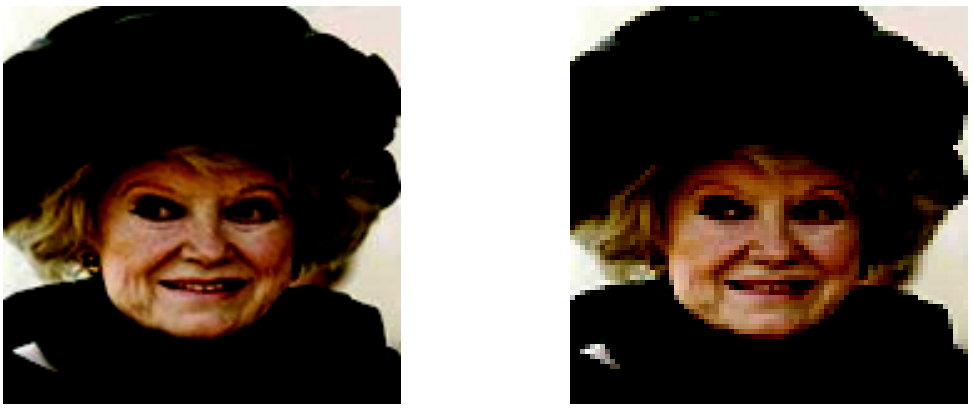
Figure 4.2: 178x218 CelebA image (left) and the same image downscaled to 160x160 (right).

As illustrated by figure 4.2 the HR data preprocessing of downscaling from 178x218 to 160x160 does not significantly distort the image and it is still recognizable as the same image by the human eye.

4.3.3 LR Data Preprocessing

For the LR data, three different preprocessing methods were used, one for each resolution ratio (2x, 4x, 8x). All of which consist of downscaling the HR images to the appropriate dimensions in each resolution ratio case:

- For the 2x resolution ratio models, the 160x160 HR images are downscaled to 80x80
- For the 4x resolution ratio models, the 160x160 HR images are downscaled to 40x40
- For the 8x resolution ratio models, the 160x160 HR images are downscaled to 20x20



(a) 160x160 image. (b) 80x80 image.

Figure 4.3: 160x160 CelebA image downscaled to 80x80.

(a) 160x160 image.

(b) 40x40 image.

Figure 4.4: 160x160 CelebA image downscaled to 40x40.

(a) 160x160 image.

(b) 20x20 image.

Figure 4.5: 160x160 CelebA image downscaled to 20x20.

As illustrated by figures 4.3, 4.4 and 4.5 we can see that the SR problem becomes more difficult with each factor the ratio is increased by and more HR images correspond to the same LR image.

4.4 Models

In this thesis we want to investigate the impact of different diffusion model parameters on model performance for increasingly harder SR problems with higher resolution ratios. As already mentioned, the parameters we are interested in are the amount of steps in the forward process and the corresponding noise schedule parameters β_t. As a result, we train three models with different amounts of steps (50, 200, 500) per noise schedule per resolution ratio with two noise schedules (linear and cosine) and three resolution ratios (2x, 4x, 8x) resulting in 6 models per resolution ratio and 18 models in total. More concretely, all models trained and evaluated in this thesis are summarized in table 4.1:

Steps	Noise Schedule	SR ratio
50	Linear	2x
200	Linear	2x
500	Linear	2x
50	Cosine	2x
200	Cosine	2x
500	Cosine	2x
50	Linear	4x
200	Linear	4x
500	Linear	4x
50	Cosine	4x
200	Cosine	4x
500	Cosine	4x
50	Linear	8x
200	Linear	8x
500	Linear	8x
50	Cosine	8x
200	Cosine	8x
500	Cosine	8x

Table 4.1: Summary of every model trained and evaluated in this thesis.

All models use the same baseline U-net architecture illustrated in figure 4.6 but the LR encoder varies slightly between resolution ratios to accommodate the different dimensions of the LR embeddings in each case such that they fit into the first convolutional block of the U-net which is of dimensions 160x160.

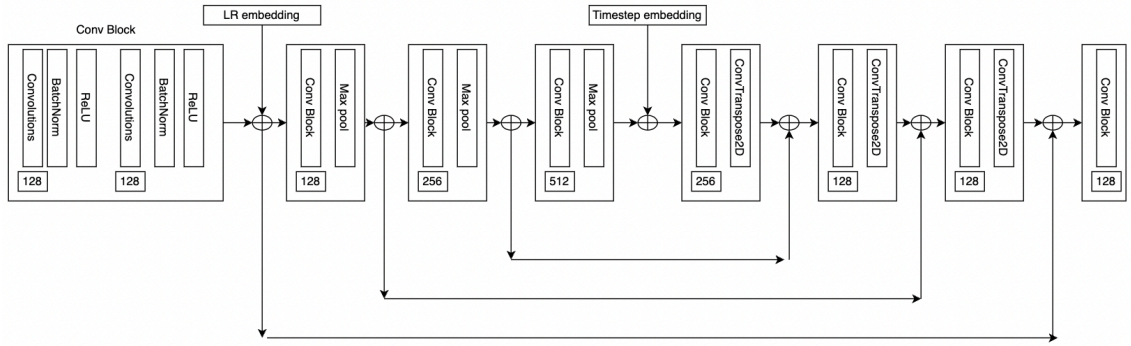


Figure 4.6: Model architecture.

As illustrated by figure 4.6 the U-net is of depth 3 with the LR embedding at the first convolutional block while the timestep embedding is concatenated at the bottom of the U-net. Each convolutional block of the U-net consists of convolutional operations followed by a batch normalization which is then followed by an activation function (ReLU) and finally an up or downscaling operation such as ConvTranspose2D or max pooling.

As for the LR encoders, they consist of convolutional blocks followed by varying amounts of ConvTranspose2D (upscaling) operations followed by the ReLU activation function such that the dimensions of the output are 160x160 with 16 channels in total.

Note that all models trained and evaluated in this thesis have roughly 15.5 million learnable

parameters varying slightly based on the type of LR encoder used.

4.5 Training

An important part of deep learning models such as diffusion models is how they are trained. Training these models typically involves many hyperparameters such as the learning rate and the type of optimizer the training process uses. These hyperparameters have been shown to impact model performance greatly and have been researched extensively. While these hyperparameters are interesting in the context of diffusion models and SR we choose to keep them constant across all models to ensure fair model comparison between all parameters tested in this thesis.

The training related parameters that were chosen and used for every model in this thesis are as follows:

- The optimizer used for training is the Adam optimizer.
- The training time is set to 10 epochs which translates to roughly 100k optimization steps and takes roughly 8 hours per model.
- Batch size is set to 16 images at a time.
- The learning rate is set to $1e^{-4}$ and is linearly decreased by a factor of $1/10$ each epoch.

4.6 Model Assessment

Assessing SR models has proven to be a difficult problem as mathematical metrics typically are not able to capture what makes an images visually appealing to our eyes. Due to this phenomenon, the field of model assessment in the context of SR is an active research field to this day. In this thesis we aim to assess two properties of diffusion models for SR. The first is that we want to assess the impact of different diffusion model parameters on general model performance for SR across different SR ratios. The second is to gauge whether or not the HR diffusion model samples get downscaled to the same LR image that guided the denoising process and how the different diffusion model parameters affect that ability of the model.

By increasing the resolution ratio of the SR problem we inherently create more ambiguity between the HR and LR images i.e. by increasing the resolution ratio, there are more HR images that correspond to and get downscaled to the same LR image. By experimenting with different resolution ratios we wish to gauge the capabilities of diffusion models to handle this increased ambiguity by altering the amount of steps in the forward process and the noise schedule parameters. In other words, we want assess the models in this thesis using metrics that can ideally capture the models ability to create HR images that are either very close to the original HR image and if that is too difficult or impossible for the model, does it create HR images that correspond to the LR image that was embedded into the model during inference.

Put mathematically, if we let $h(\boldsymbol{x}|\text{LR})$ be the conditional probability distribution of HR images that get downscaled to the same LR image and $l_\theta(\boldsymbol{x}|\text{LR})$ be the conditional distribution learned by the model, we want to assess if these distributions are the same or at least similar. Note that the stochastic nature of diffusion model inference allows us sample the learned distributions and get different results each time as opposed to completely deterministic methods which generate the same HR image for a given LR image every time. The conditional probabilites $h(\boldsymbol{x}|\text{LR})$ and $l_\theta(\boldsymbol{x}|\text{LR})$ are high dimensional and in general intractable and we therefore have to assess this property in an empirical manner.

The way we chose to assess this is by evaluating the models using both the peak signal-to-noise ratio between ground truth HR images and generated HR images by the model (HR-PSNR) and also the peak signal-to-noise ratio between the ground truth LR images and HR images generated by the model that are downscaled to the appropriate LR dimensions in each SR ratio case (LR-PSNR). In this way we use the HR-PSNR to assess the models ability to replicate HR images given the corresponding LR image which is ultimately the goal of SR while also using the LR-PSNR to calculate how close the downscaled generated image is to the LR ground truth image. For instance, if the HR-PSNR of the model is low but the LR-PSNR of the same model is high, we can infer that the model does not do a good job of replicating the HR images as indicated by the low HR-PSNR but the conditional distributions $h(\boldsymbol{x}|\text{LR})$ and $l_\theta(\boldsymbol{x}|\text{LR})$ seem to be similar as indicated by the high LR-PSNR.

It is important to note that while the ratio between the LR-PSNR and HR-PSNR does in some way reflect the similarity of the conditional distributions $h(\boldsymbol{x}|\text{LR})$ and $l_\theta(\boldsymbol{x}|\text{LR})$ it is by no means a perfect method to do this nor the standard in general. It is simply our way of evaluating the difference between the two distributions in an empirical manner.

4.6.1 PSNR

Peak Signal-to-Noise Ratio (PSNR) is a metric used to quantify the quality of an image by comparing it to the original image. It measures the ratio between the maximum possible power of a signal and the power of the noise that distorts it.

The PSNR is calculated using the following equivalent formulas:

$$\text{PSNR} = 10 \cdot log_{10}(\frac{\text{max}_I^2}{\text{MSE}}) = 20 \cdot log_{10}(\frac{\text{max}_I}{\sqrt{\text{MSE}}})$$

Where max_I represents the maximum pixel value of the image which is typically 255 for 8-bit images and MSE is the mean squared error between the images.

The PSNR value is expressed in decibels and provides an approximation of image quality in terms of how much noise the image contains compared to the ground truth image. Since diffusion models create their samples by denoising a completely noisy image, the PSNR is used to indicate how well the model achieves this. Note that a higher PSNR indicates higher image quality whereas a lower PSNR indicates worse image quality. A rule of thumb is that a PSNR above 30 dB is considered acceptable while values above 40 dB are considered very high.

Note that the PSNR is a pixel wise difference metric and does therefore in many cases not accurately reflect the perceptual quality of images.

4.6.2 Structural Similarity

The structural similarity (SSIM) is another image quality metric that measures the similarity between two images while taking into account both pixel wise differences and structural information of images. Unlike the PSNR, the SSIM aims to evaluate perceptual aspects of images. The SSIM measures similarity between images by evaluating luminance, contrast and structure on a scale from -1 to 1 where -1 denotes complete dissimilarity and 1 represents complete similarity (identical images).

- Luminance represents the overall brightness of an image which is the same as pixel value intensity. The SSIM compares the mean pixel values of both images which is then used to calculate the luminance similarity of the images.

- Contrast represents the differences in texture and sharpness between the two images. The SSIM evaluates the standard deviation of pixel intensities which is then used to calculate the contrast similarity.
- Structure represents the correlations between neighboring pixels which captures the image's texture and organization. The SSIM calculates the structural similarity by comparing the covariance of pixel intensities of the two images.

Additionally, the SSIM can only be calculated on a single channel of an image and since the images of the CelebA dataset are RGB images, they need to be transformed before calculating the SSIM. Two methods are typically used to solve this problem and the first one is converting the RGB image to a grayscale image and calculate the SSIM between the grayscale images and the second is converting the RGB image to the YCbCr format and calculate the SSIM using the Y channel. In this thesis we explicitly use the first method i.e. we convert the images to grayscale and calculate the SSIM using the following formula:

$$\text{SSIM} = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2) + c_2}$$

5 Results

In this chapter we present all models trained in this thesis and their corresponding performance metrics. As already mentioned, we train 6 models for each SR ratio with 3 different amounts of steps (50, 200, 500) for each noise schedule (linear, cosine). The models are then evaluated using the HR-PSNR, LR-PSNR and SSIM performance metrics to be compared against one another. In this thesis we are mainly interested in how well the models are able to replicate the HR images which is captured by the HR-PSNR and the SSIM. Additionally we look at whether or not the samples created by the models have a strong consistency with the LR images i.e. whether or not the model samples get downscaled to the same LR image even though they are not identical to the HR ground truth image. This property is reflected in the ratio between the HR-PSNR and LR-PSNR where a low HR-PSNR paired with a high LR-PSNR indicates a model possesses this property for instance.

All performance metrics across all SR ratios tested in this thesis were evaluated using the average of 1000 model samples.

5.1 2x Models

In this section we present the results for the 2x models and show samples from these models.

Steps	Noise Schedule	HR-PSNR	LR-PSNR	LR-PSNR/HR-PSNR	SSIM
50	Linear	32.366	32.115	0.992	0.658
200	Linear	36.011	37.482	1.041	0.944
500	Linear	36.070	37.621	1.043	0.943
50	Cosine	33.418	33.723	1.009	0.827
200	Cosine	35.346	36.473	1.032	0.933
500	Cosine	36.356	37.852	1.041	0.950

Table 5.1: Performance metrics of all 2x models.

As expressed by table 5.1, model performance increases with increased amounts of steps in the forward process and the best models across all performance metrics are the 500 step models. This is consistant with findings of other diffusion model papers such as (Ho et al. 2020) which investigates diffusion models in general and is not specified to SR. Moreover, the noise schedule significantly affects the performance of the models and the cosine schedule positively impacts the metrics in all cases models except for the 200 step models. The positive impact of the cosine schedule is also consistent with findings of other research papers such as (Nichol & Dhariwal 2021) which investigates the impact of the noise schedule in diffusion models not specified to SR.

As for the models ability to create samples that get downscaled to the correct LR image we direct our attention to the differences in the HR-PSNR and LR-PSNR. Interestingly, the noise schedule seems to have little impact on the models ability in this regard whereas the amount of steps in the forward process seems to greatly affect this property for the 2x models and especially so when going from 50 to 200 steps.

Figure 5.1: LR image (left), HR image (middle) and predicted image (right) from the 2x 50 step linear noise schedule model.

Figure 5.2: LR image (left), HR image (middle) and predicted image (right) from the 2x 200 step linear noise schedule model.

Figure 5.3: LR image (left), HR image (middle) and predicted image (right) from the 2x 500 step linear noise schedule model.

Figure 5.4: LR image (left), HR image (middle) and predicted image (right) from the 2x 50 step cosine noise schedule model.

Figure 5.5: LR image (left), HR image (middle) and predicted image (right) from the 2x 200 step cosine noise schedule model.

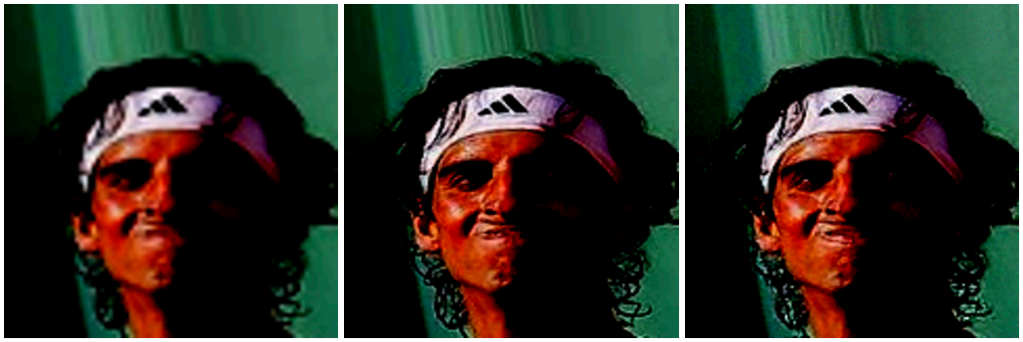


Figure 5.6: LR image (left), HR image (middle) and predicted image (right) from the 2x 500 step cosine noise schedule model.

5.2 4x Models

In this section we present the results for the 4x models and samples from selected models.

Steps	Noise Schedule	HR-PSNR	LR-PSNR	LR-PSNR/HR-PSNR	SSIM
50	Linear	31.988	31.578	0.987	0.512
200	Linear	33.894	34.453	1.017	0.835
500	Linear	34.913	35.626	1.020	0.873
50	Cosine	33.163	33.324	1.005	0.746
200	Cosine	34.222	34.841	1.018	0.841
500	Cosine	35.113	35.744	1.018	0.880

Table 5.2: Performance metrics of all 4x models.

Table 5.2 contains all the performance metrics and models for the 4x resolution ratio and the general performance trend of these models in terms of replicating the HR image seems to increase with the increased amount of steps in the forward process which again is consistent with results of other diffusion model papers and the 2x models. The 4x models also show an increase in performance when going from the linear noise schedule to the cosine schedule. The impact of the noise schedule is consistent across all the different step models as opposed to the 2x models.

The 4x models also possess similar characteristics when it comes to creating images that get downscaled to the correct LR image. The 50 step models are especially bad at this and the linear 50 step models in both 2x and 4x resolution ratios have worse LR-PSNR metrics compared to the HR-PSNR. In contrast, the 200 and 500 step models have higher LR-PSNR than HR-PSNR indicating that the learned conditional distributions of those models more closely resemble the conditional distribution of the HR images that correspond to a certain LR image. The ratio between the HR-PSNR and LR-PSNR does however indicate that the 4x models are worse at this than the 2x models.

Figure 5.7: LR image (left), HR image (middle) and predicted image (right) from the 4x 50 step linear noise schedule model.

Figure 5.8: LR image (left), HR image (middle) and predicted image (right) from the 4x 200 step linear noise schedule model.

Figure 5.9: LR image (left), HR image (middle) and predicted image (right) from the 4x 500 step linear noise schedule model.

Figure 5.10: LR image (left), HR image (middle) and predicted image (right) from the 4x 50 step cosine noise schedule model.

Figure 5.11: LR image (left), HR image (middle) and predicted image (right) from the 4x 200 step cosine noise schedule model.

Figure 5.12: LR image (left), HR image (middle) and predicted image (right) from the 4x 500 step cosine noise schedule model.

5.3 8x Models

In this section we present the results for the 8x models and samples from selected models.

Steps	Noise Schedule	HR-PSNR	LR-PSNR	LR-PSNR/HR-PSNR	SSIM
50	Linear	31.877	31.600	0.991	0.463
200	Linear	33.351	33.606	1.008	0.738
500	Linear	33.381	33.466	1.003	0.750
50	Cosine	32.100	31.870	0.993	0.526
200	Cosine	33.559	33.811	1.008	0.750
500	Cosine	33.708	34.040	1.010	0.751

Table 5.3: Performance metrics of all 8x models.

Table 5.3 summarizes the performance of the 8x models and these models follow a similar trend to the 4x and 2x models when it comes to the impact of the amount of steps in the forward process on replicating HR images. In general, more steps lead to increased performance across all metrics but the diminishing returns on model performance with more steps is more apparent in the 8x models and especially so when going from 200 to 500 steps. Interestingly, the noise schedule does impact model performance as it does for the 4x and 2x models where in general the cosine schedule performs better than the linear noise schedule. However, while the cosine noise schedule does impact the PSNR metrics of the 8x models it has less impact on the SSIM in the 200 and 500 step models.

The 8x models are the worst out of the bunch when it comes to creating images that get downscaled to the correct image as indicated by the LR-PSNR/HR-PSNR ratio. They do however have similar characteristics to the 4x and 2x models where the 50 step models are the worst at this and are the only models that have a lower LR-PSNR than HR-PSNR. Furthermore, the 200 and 500 step models have a higher LR-PSNR when compared to the HR-PSNR but very marginally so, especially when compared to the 2x models.

Figure 5.13: LR image (left), HR image (middle) and predicted image (right) from the 8x 50 step linear noise schedule model.

Figure 5.14: LR image (left), HR image (middle) and predicted image (right) from the 8x 200 step linear noise schedule model.

Figure 5.15: LR image (left), HR image (middle) and predicted image (right) from the 8x 500 step linear noise schedule model.

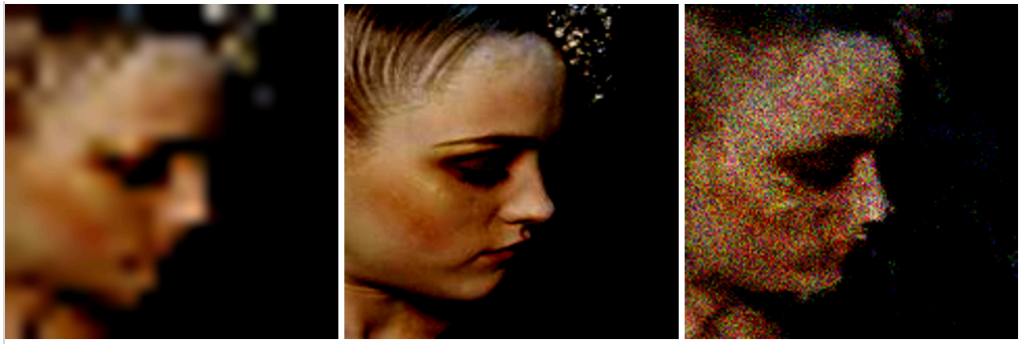


Figure 5.16: LR image (left), HR image (middle) and predicted image (right) from the 8x 50 step cosine noise schedule model.

Figure 5.17: LR image (left), HR image (middle) and predicted image (right) from the 8x 200 step cosine noise schedule model.

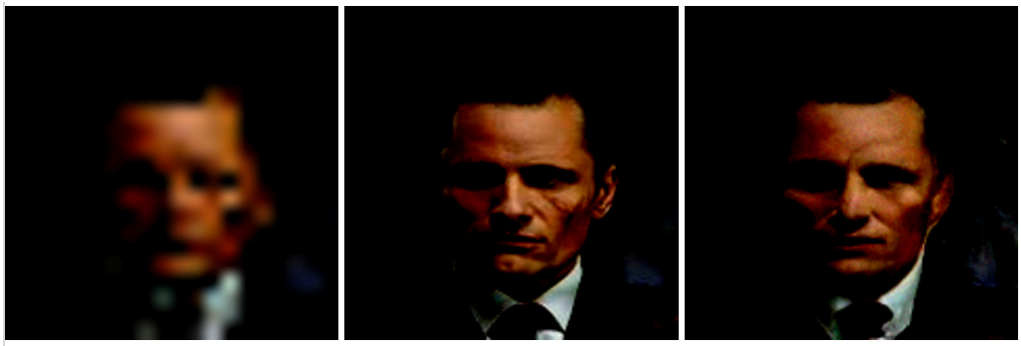


Figure 5.18: LR image (left), HR image (middle) and predicted image (right) from the 8x 500 step cosine noise schedule model.

6 Final Remarks

In this chapter we discuss our findings in this thesis and conclude the report. Finally, we discuss where there is room for improvement and further research.

6.1 Discussion

6.1.1 Results

Overall, the results are consistant with findings of other diffusion model research papers where the amount of forward steps positively affects model performance across all performance metrics tested in this thesis. Moreover, the noise schedule also affects the models ability to replicate HR images which results in improved metrics with the exception of the LR-PSNR/HR-PSNR ratio. This means that the cosine noise schedule does increase both the HR-PSNR and the LR-PSNR but it does so such that it has little effect on the ratio between the two.

A surprising finding in this thesis is that there seems to be a trend where the LR-PSNR/HR-PSNR ratio gets worse with increased resolution ratios. Increasing the resolution ratio inherently creates more ambiguity in the SR problem and intuitively one might think that this would make it easier to create HR images that get downscaled to the same LR image since there are more HR images that get downscaled to the same LR image. This is however not the case according to the findings of this thesis. It is important to note however that (Li et al. 2021) report strong LR consistency on the 8x SR problem but they only report their findings on 8x models so it is difficult to say whether or not their 2x and 4x models would have even stronger LR consistencies. The biggest difference between the models in this thesis and their models is the training scheme. They train their models for longer by a large margin. This does suggest that the training scheme used affects the models ability to create HR images that get downscaled to the same LR images.

6.1.2 Stochastic Inference

The stochastic inference of diffusion models is a very interesting property especially so in the case of SR, that most other deep learning models do not possess. This property of diffusion models is the largest motivation factor behind comparing the conditional distributions $h(\boldsymbol{x}|\text{LR})$ and $l_\theta(\boldsymbol{x}|\text{LR})$ mentioned in chapter 4. The main idea is that if these conditional distributions are the same or very similar and we have a model that can sample this distribution, then it is effectively possible to sample multiple HR images that all correspond to the same LR image and compare them against one another and even get point estimates of certain probabilities. For instance, if there is a medical image that is upscaled by a diffusion model multiple times, 20% of the different HR images created by the model could reveal a certain disease while the other 80% do not reveal the same disease. In that case a probability point estimate for a certain disease is acquired.

While the stochastic property of diffusion model inference does have interesting potential applications it also has a drawback. Namely, the performance evaluation of diffusion models has an additional stochastic component. Since deep learning models are typically evaluated empirically using the samples they create, this would mean that the estimated performance metrics could change drastically each time they are calculated. To combat this in this thesis project, we aim to have the sample size large enough to create a reliable estimate.

6.1.3 LR-PSNR/HR-PSNR ratio

Even though the idea of having diffusion models learn the correct conditional distributions is tempting, evaluating this property is difficult to do since the distributions are in general intractable. This intractability means that the evaluation of this property has to be done empirically. In this thesis we chose to do this by using the LR-PSNR/HR-PSNR ratio as our empirical estimate of this property.

As already mentioned, it is important to note that the LR-PSNR/HR-PSNR ratio is by no means the perfect metric that reflects the true similarity of the conditional distributions. We leave it to future work to figure out a better way of evaluating the similarity of the two conditional distributions and how deviations in that similarity affect the model performance.

6.1.4 Practical applications

Diffusion models have shown great promise in generative tasks such as image synthesis and other reconstruction problems such as SR. Even though they show great promise and have outperformed other state-of-the-art deep learning models in various tasks, it remains to be seen how they can or will be used for practical applications where reliability is of major importance. Practical applications of this sort are for instance medical image analysis/segmentation. Until now, diffusion models have mostly been used for research and applications such as avatar creation. Thus they have yet to break into the high impact practical applications.

In our opinion, diffusion models show great potential and could even be used for tasks such as medical image analysis in their current state. It is however important to note that they should not be blindly relied on for these high impact applications but rather used as a tool to help professionals better diagnose their patients for instance.

In order for diffusion models to be blindly relied on we feel that further research is required and even then, they might not be the right tool for all tasks.

6.2 Conclusion

Throughout this thesis we have investigated diffusion models and their various architectural components for the purpose of solving SR for images. Based on the results presented in chapter 5, we conclude that the effect of the forward process steps and noise schedule for SR stay consistent with findings of other diffusion model research papers that are not SR specific. That is, the amount of forward process steps positively affects model performance and the choice of noise schedule also affects model performance where the cosine schedule pulls ahead of the linear schedule. This stays true across all SR ratios tested in this thesis. As for the models ability to create images that are downscaled to the correct LR image, we conclude that the forward process steps do affect this property significantly while the choice of noise schedule seems to have less impact in this regard. This property is affected by the SR ratio at hand and we report that higher resolution ratios benefit less from the amount of forward process steps than lower resolution ratios.

6.3 Future Work

6.3.1 Conditional distribution comparison

As already mentioned in the discussion section of this chapter, we leave it to future work to better evaluate the true similarity of the conditional distributions since the LR-PSNR/HR-PSNR ratio is unlikely to accurately reflect this property.

6.3.2 Training

According to (Li et al. 2021) they utilize diffusion models and report metrics on 8x models using the CelebA dataset showing strong consistency between the LR and HR images

and reporting LR-PSNR over 50 dB while their HR-PSNR is lower reporting in around 25 dB. The main differences between the models used in this thesis and their models is that they utilize a pretrained LR encoder and they also train their models for 35 hours with a slightly different learning rate schedule whereas in this thesis they are only trained for roughly 8 hours. This suggests that the training scheme majorly affects the models ability to create HR images that get downscaled to the same LR image. Therefore we leave it to future work to further investigate the effects of different training schemes on model performance.

6.3.3 Model comparison

In this thesis we are mainly focused on the effects of different diffusion model parameters on model performance, namely the forward process steps and the noise schedule across different resolution ratios. We have therefore not put our attention in creating the best possible diffusion models in terms of model performance but rather created many models to cover as many bases as possible within the given time. Because of the limited time, we have chosen to not compare our diffusion models to other state-of-the-art models for a couple of reasons. First, there do not seem to be many papers that utilize the same dataset as us and comparison between models is therefore difficult to gauge. Second, other state-of-the-art models usually are trained for a very long time to achieve the best possible results and comparing our models which were trained for 8 hours to models that were trained for weeks doesn't really say much about the differences between the models but it rather reflects the difference in training. State-of-the-art diffusion models have frequently been compared to other state-of-the-art deep learning models and have been reported outperforming other models such as GANs on various performance metrics on various tasks. Thus we leave it to future work to continue this comparison between different models.

6.3.4 Model size

Model size is another interesting parameter to investigate in terms of model performance as deep learning models typically get better as their model size gets bigger. Our models come in at 15.5 million learnable parameters which in the context of deep learning models is considered very small. Additionally, the CelebA dataset is a large scale dataset and would likely benefit from a larger model since a small model like ours used on such a large dataset and trained for a short amount of time is likely to be underfitted to the data. Unfortunately, due to time constraints, we were not able to experiment with different model sizes and how they affect the model performance. Thus we leave it to future work to research how model size affects diffusion models.

6.3.5 Datasets

As already mentioned, the CelebA dataset is a large scale dataset with over 200k images. Since this is the only dataset tested in this thesis, it would be interesting to see how different datasets with different properties affect model performance. Moreover, the model size, dataset and training time are all linked in this way and their relationship would also be interesting to investigate not only for diffusion models.

7 Bibliography

Ho et al. (2021). ”Cascaded Diffusion Models for High Fidelity Image Generation”. In ”https://arxiv.org/pdf/2106.15282.pdf”.

Li et al. (2021). ”SRDiff: Single Image Super-Resolution with Diffusion Probabilistic Models”. In ”https://arxiv.org/pdf/2104.14951.pdf”.

Nichol & Dhariwal (2021). ”Improved Denoising Diffusion Probabilistic Models”. In ”https://arxiv.org/pdf/2102.09672.pdf”.

Rombach et al. (2022). ”High-Resolution Image Synthesis with Latent Diffusion Models”. In ”https://arxiv.org/pdf/2112.10752.pdf”.

Sohl-Dickenstein et al. (2015). ”Deep Unsupervised Learning using Nonequilibrium Thermodynamics”. In ”https://arxiv.org/pdf/1503.03585.pdf”.

Ho et al. (2020). ”Denoising Diffusion Probabilistic Models”. In ”https://arxiv.org/pdf/2006.11239.pdf”.

Ronneberger et al. (2015). ”U-Net: Convolutional Networks for Biomedical Image Segmentation”. In ”https://arxiv.org/pdf/1505.04597.pdf”.

Song et al. (2021). ”Score-based Generative Modeling Through Stochastic Differential Equations”. In ”https://arxiv.org/pdf/2011.13456.pdf”.

Kong et al. (2021). ”DiffWave: A Versatile Diffusion Model for Audio Synthesis”. In ”https://arxiv.org/pdf/2009.09761.pdf”.

Vaswani et al. (2017). ”Attention Is All You Need”. In ”https://arxiv.org/pdf/1706.03762.pdf”.

Technical
University of
Denmark

Richard Petersens Plads , Building 324
2800 Kgs. Lyngby
Tlf. 4525 1700

www.compute.dtu.dk