Introduction to Diffusion Models

Andreas Bagge & Gustav Rørhauge

January 5, 2024

Contents

1 Introduction

Diffusion models are a class of generative models that aim to learn the latent structure of complex data, such as images. These latent structures are underlying structures that are used in the generative process behind the data. Generative models are models that can generate new data similar to the data on which they are trained. They have many potential applications, such as data augmentation, image synthesis, video generation, molecule design, and text-to-image generation. However, generative modeling is also a very challenging task, as it requires capturing the high-dimensional and multimodal distribution of natural data.

Diffusion models are based on the idea of reversing a diffusion process, which is a stochastic process that gradually adds noise to the data until it reaches a predefined noise level. The diffusion process can be seen as a way of destroying the information in the data while preserving some of its statistical properties. By learning to reverse this process, diffusion models can recover the original data from the noisy data, and thus generate new data from pure noise.

2 The forward process

The diffusion model can be interpreted as a two-part system: A forward and a backward process. The forward process steadily noisifies the images in a Markovian chain, where a series of $t = 0 \dots T$ timesteps following a noising schedule transforms the image into new images that more and more closely resemble pure Gaussian noise. The forward process can be described as:

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

Where β is a fixed schedule parameter (although it can be learned). Simply put, β_t is just a scalar value pertaining to the timestep t.
Imagine that you have some image that is then flattened into an n-dimensional vector:

$$\boldsymbol{x}_{\mathrm{t}} = [\boldsymbol{x}_{t,0}, \boldsymbol{x}_{t,1}, \cdots \boldsymbol{x}_{t,n}]$$

such that n is the number of pixels in the image and such that $\boldsymbol{x}_{t,l}$ is the l'th pixel in the image. Then, a slightly more noisy version of $\boldsymbol{x}_t$ can be calculated by sampling a new pixel-value using $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ and the parameterization trick for each pixel in $\boldsymbol{x}_t$:

$$\boldsymbol{x}_{t+1} = \sqrt{1-\beta_t}\boldsymbol{x}_t + \beta_t\mathbf{I}$$

Such that each new pixel-value, $\boldsymbol{x}_{t+1,l}$ is drawn from a normal distribution with mean $\sqrt{1-\beta_t}\boldsymbol{x}_{t,l}$ and standard deviation β_t.
Obviously, there are many ways to make data more and more noisy and this is just one of them. However, picking the forward process to follow a Gaussian distribution will, as usual, give some very nice properties later on. Each image in the Markov chain depends only on the prior image. So, given x_0, our ground truth image, the following images can be written in a sequence as:

$$\begin{gathered}
\boldsymbol{x}_0 \sim q(\boldsymbol{x}_0) \\
\boldsymbol{x}_1 \sim \mathcal{N}(\sqrt{1-\beta_1}\boldsymbol{x}_0, \beta_1\mathbf{I}) \\
\boldsymbol{x}_2 \sim \mathcal{N}(\sqrt{1-\beta_2}\boldsymbol{x}_1, \beta_2\mathbf{I}) \\
\vdots \\
\boldsymbol{x}_{T-1} \sim \mathcal{N}(\sqrt{1-\beta_{T-1}}\boldsymbol{x}_{T-2}, \beta_{T-1}\mathbf{I}) \\
\boldsymbol{x}_T \sim \mathcal{N}(\sqrt{1-\beta_T}\boldsymbol{x}_{T-1}, \beta_T\mathbf{I})
\end{gathered}$$

Where $\boldsymbol{x}_0 \ \sim q(\boldsymbol{x}_0)$ is drawn from the true distribution of the data, i.e. it is an image from our dataset. Even though the forward process is totally fixed, it is important to understand that it is not deterministic; we don't know the exact values of $\boldsymbol{x}_1, \boldsymbol{x}_1, \dots \boldsymbol{x}_T$, only their distributions.
So, to obtain $\boldsymbol{x}_t$, we have to fist obtain $\boldsymbol{x}_1, \boldsymbol{x}_2, \dots \boldsymbol{x}_{t-1}$. This can be a costly affair, especially for large T and for large data. However, due to our choice of noise scheduling, this can be circumvented by the

”repeated reparameterization trick”. Instead of sampling t times to obtain $\boldsymbol{x}_t$, it is possible to go from an input image $\boldsymbol{x}_0$ directly to any timestep in the forward process: First, we'll introduce α_t:

$$\alpha_t = 1 - \beta_t$$

Using α_t, the forward process can be rewritten as:

$$\boldsymbol{x}_t \sim \mathcal{N}(\sqrt{\alpha_t}\boldsymbol{x}_{t-1}, (1-\alpha_t)\mathbf{I})$$

Sampling from a normal distribution (reparameterization trick)

Given:

$$p(x) = \mathcal{N}(\mu, \sigma^2)$$

A sample $\boldsymbol{x}$ from p can be drawn by calculating:

$$x = \mu + \sigma\epsilon, \qquad \epsilon \sim \mathcal{N}(0,1)$$

Using the reparameterization trick, the next step in the forward process can also be expressed as:

$$\boldsymbol{x}_t = \sqrt{\alpha_t}\boldsymbol{x}_{t-1} + \sqrt{1-\alpha_t}\epsilon_t$$

x_{t-1} can be rewritten in the same way:

$$\boldsymbol{x}_{t-1} = \sqrt{\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{1-\alpha_{t-1}}\epsilon_{t-1}$$

Where ϵ_t and ϵ_{t-1} are independent and identically distributed. Combining the two expressions above we obtain:

$$\begin{aligned}
\boldsymbol{x}_t &= \sqrt{\alpha_t}(\sqrt{\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{1-\alpha_{t-1}}\epsilon_{t-1}) + \sqrt{1-\alpha_t}\epsilon_t \\
&= \sqrt{\alpha_t}\sqrt{\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{\alpha_t}\sqrt{1-\alpha_{t-1}}\epsilon_{t-1} + \sqrt{1-\alpha_t}\epsilon_t \\
&= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \underbrace{\sqrt{\alpha_t - \alpha_{t-1}\alpha_t}\epsilon_{t-1} + \sqrt{1-\alpha_t}\epsilon_t}_{\text{Sum of Gaussians}} \\
&= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{\sqrt{\alpha_t - \alpha_{t-1}\alpha_t}^2\epsilon_{t-1} + \sqrt{1-\alpha_t}^2\epsilon_t} \\
&= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{1-\alpha\alpha_{t-1}}\epsilon^*
\end{aligned}$$

Sum of centered Gaussians (i.i.d)

We'll now use the definition of a sum of centered Gaussians. The mean and standard deviation of the resulting Gaussian distribution is defined as follows:

$$\begin{gathered}
X \sim \mathcal{N}(0, \sigma_x^2) \\
Y \sim \mathcal{N}(0, \sigma_y^2) \\
X + Y \sim \mathcal{N}(0, \sigma_x^2 + \sigma_y^2)
\end{gathered}$$

Given that X and Y are not correlated.

Now, the above can be repeated by writing the expression for $\boldsymbol{x}_{t-2}$ using the reparameterization trick. This can be done all the way down to $\boldsymbol{x}_0$ by when we stop. The expression for $\boldsymbol{x}_t$ will then be:

$$\begin{aligned}
\boldsymbol{x}_t &= \sqrt{\alpha_t\alpha_{t-1}\dots\alpha_1}\boldsymbol{x}_0 + \sqrt{1-\alpha\alpha_{t-1}\dots\alpha_1}\epsilon^* \\
&= \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon^* \\
q(\boldsymbol{x}_t|\boldsymbol{x}_0) &= \mathcal{N}(\boldsymbol{x}_t, \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0, (1-\bar{\alpha}_t)\mathbf{I})
\end{aligned}$$

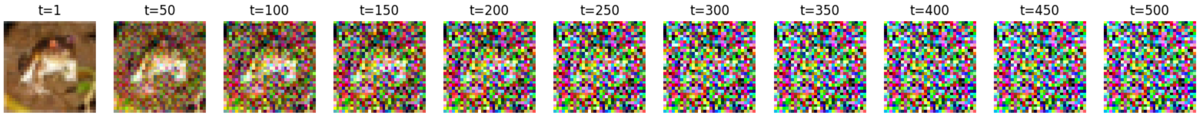

Figure 1: Gradually more noisy versions of the original $\boldsymbol{x}_0$. Here, each image have been sampled from the distribution pertaining to the timestep t

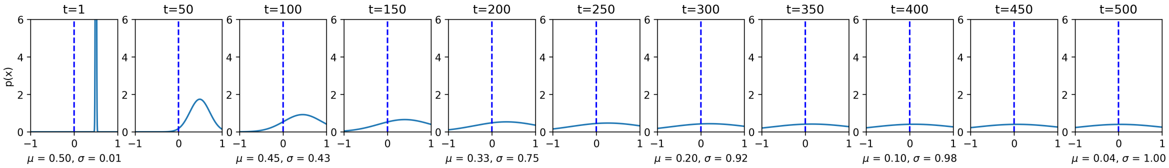

Figure 2: The distribution for the pixel value for each timestep given $\boldsymbol{x}_0 = 0.5$. Clearly, the distribution approaches a standard normal distribution.

Where $\bar{\alpha}_t = \prod_{t=1}^{T} \alpha_t$. We can visualize what happens with an image when running it through the encoder by sampling from the distribution pertaining to each timestep. For example, if we let $\boldsymbol{x}_0 = \text{frog}$, we can obtain the noisy images visualized in figure 1. Another way of interpreting what exactly happens in the forward process is to look at the resulting distribution of some pixel value after each timestep. This change in distribution can be seen in figure 2. Evidently, the distribution of the pixel gets shifted closer and closer towards a standard normal distribution. This also explains why the images become more and more noisy; they carry less and less information from the original image and come closer and closer to Gaussian noise.

2.1 Other types of noise schedules

The purpose of the noise scheduler is to gradually add more and more noise to the original picture such that the final image resembles Gaussian noise. Remember how we parameterized each image in the chain of noisy images:

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

So, given that we want $\boldsymbol{x}_t$ to resemble $\boldsymbol{x}_0$ for small $t \approx 1$, and we want $\boldsymbol{x}_t$ to resemble $\mathcal{N}(0,1)$ for big $t \approx T$, it should be obvious that we want to model $\bar{\alpha}_t$ such that $\bar{\alpha}_1 \approx 0$ and $\bar{\alpha}_T \approx 1$, and such that $\bar{\alpha}_t$ is monotonously decreasing on the range from $t = 1 \ldots T$. This is also exactly the case using the linear noise schedule used in the chapter above. But obviously, there are other ways of defining the noise schedule such that $\bar{\alpha}_t$ obtains the sought-after behavior. For example, in [ND21] they define $\bar{\alpha}_t$ to follow a cosine wave defined as:

$$\bar{\alpha}_t = \frac{f(t)}{f(0)}, \qquad f(t) = \cos\left(\frac{t/T + s}{1+s} \cdot \frac{\pi}{2}\right)^2$$

The authors discover that this noise schedule performs better than the ”traditional” linear noise schedule. A comparison between the behavior of linear and cosine noise schedules can be seen in figure 3.

2.2 What is our goal?

Until now, we have formulated the forward process of our model. That is, we have formulated a way of gradually producing more and more noisy images originating from some ground truth image called $\boldsymbol{x}_0$ such that the final image resembles simple Gaussian noise. Our goal is now to go backward and try to reproduce the original image. In other words, in the forward process, we modeled the next image based on the prior, such that we had $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$. Now, we want to model the prior image based on the next one; we want to obtain the distribution of $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$. Just like in the forward process, we

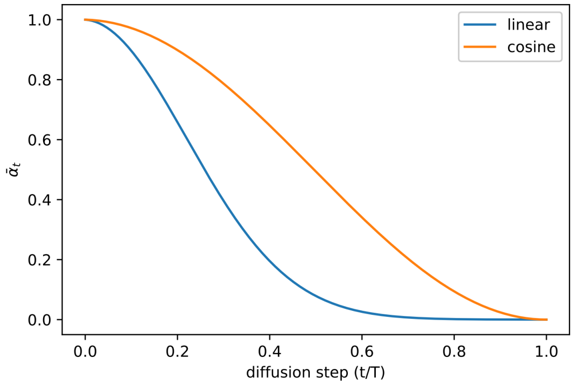

Figure 3: Comparison of the behavior of $\bar{\alpha}_t$ depending on the type of noise schedule. Obtained from: [ND21]

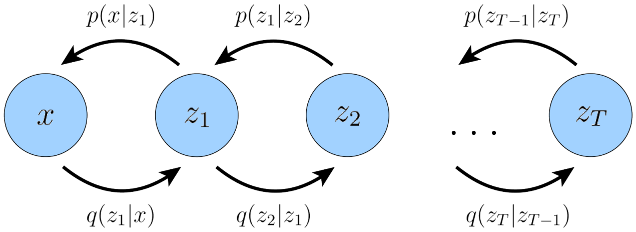

Figure 4: Given some forward diffusion process, $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$, that produces more and more noisy images, we also want the backward process, $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$, that produces less and less noisy images. Obtained from [Luo22].

would also this backward process to produce new images in a discrete manner. The difference is now that we want the images to become less and less noisy. Also, like in the forward process, we actually do not want to learn the image pertaining to each timestep in the backward process but rather the distribution of the image. One question is how to describe this backward process, another is how to optimize it. To do this, we need to use maximum likelihood estimation. The idea behind the forward and backward diffusion process is visualized in figure 4 as well as in figure 1.

3 The ELBO

Imagine that we have some data denoted $\boldsymbol{x}$. These pictures are our "ground truth" meaning we interpret them as having no noise. In the following, we will specifically denote the data $\boldsymbol{x}$ as $\boldsymbol{x}_0$ when the characteristic of "no noise" is important. In other words, they are the 0'th element in our Markov chain before any noise is added. As usual, we wish to model the distribution of these data,

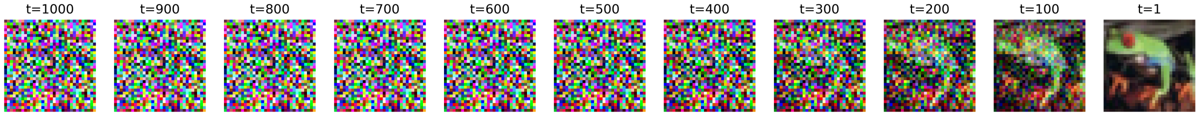

Figure 5: Visualizing the reverse process. We start with $\boldsymbol{x}_T$ and want to go backward and obtain the true underlying image (remember, we don't want $\boldsymbol{x}_1$ but instead $\boldsymbol{x}_0$, but this simply requires one more call of the reverse process). Obviously, since $\boldsymbol{x}_T$ almost resembles Gaussian noise, it is not possible to reobtain all information in the original picture, but we can get close.

$p(\boldsymbol{x}) = p(\boldsymbol{x}_0)$. One way of doing this is by marginalization:

$$p(\boldsymbol{x}_0) = \int \int \cdots \int p(\boldsymbol{x}_0, \boldsymbol{x}_1, \cdots \boldsymbol{x}_t)\, d\boldsymbol{x}_1\, d\boldsymbol{x}_2 \cdots\, d\boldsymbol{x}_t$$

To make this notation prettier, we will introduce $x_{0:T} = (x_0, x_1, \cdots x_T)$. Also, we will simply write one integral which implicitly contains the others. Then, we get:

$$p(\boldsymbol{x}_0) = \int p(\boldsymbol{x}_{0:T})\, d\boldsymbol{x}_{1:T}$$

We now want to manipulate the above expression until we get to some expression that can be evaluated on a computer and that somehow involves the mentioned backward process that we are interested in. To do this, we do a trick: we multiply by $\frac{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)} = 1$:

$$p(\boldsymbol{x}_0) = \int p(\boldsymbol{x}_{0:T}) \frac{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\, d\boldsymbol{x}_{1:T}$$

Expectation value

The expectation value is defined as:

$$\mathbb{E}_{p(x)}[f(x)] = \int p(x) f(x)\, dx$$

Where p is a probability density function over x and f is some function of x.

If we let $p = q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)$ and $f = \frac{p(\boldsymbol{x}_{0:T})}{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}$, then:

$$p(\boldsymbol{x}_0) = \int p(\boldsymbol{x}_{0:T}) \frac{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\, d\boldsymbol{x}_{1:T} = \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\frac{p(\boldsymbol{x}_{0:T})}{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\right]$$

In a second, we'll take the logarithm on both sides of the above equation. Logarithms usually make expressions easier to deal with since products can be split into sums. Also, in our case, it opens the possibility of using Jensen's Inequality to simplify things further:

Jensens Inequality

Jensen's inequality says that:

$$\log\left(\mathbb{E}[\boldsymbol{x}]\right) \geq \mathbb{E}[\log(\boldsymbol{x})]$$

Jensen's equality sometimes helps get a lower bound on expressions that are otherwise intractable.

We take the logarithm on both sides and apply Jensen's inequality:

$$\log(p(\boldsymbol{x}_0)) = \log\left(\mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\frac{p(\boldsymbol{x}_{0:T})}{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\right]\right) \geq \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log\left(\frac{p(\boldsymbol{x}_{0:T})}{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\right)\right]$$

Great! Now, let's expand the expressions in the fraction:

$$\begin{aligned}
p(\boldsymbol{x}_{0:T}) &= p(\boldsymbol{x}_0, \boldsymbol{x}_1 \cdots \boldsymbol{x}_T) \\
&= p(\boldsymbol{x}_0|\boldsymbol{x}_{1:T}) p(\boldsymbol{x}_1|\boldsymbol{x}_{2:T}) \cdots p(\boldsymbol{x}_{T-1}|\boldsymbol{x}_T) p(\boldsymbol{x}_T) \\
&= p(\boldsymbol{x}_0|\boldsymbol{x}_1) p(\boldsymbol{x}_1|\boldsymbol{x}_2) \cdots p(\boldsymbol{x}_{T-1}|\boldsymbol{x}_T) p(\boldsymbol{x}_T) \\
&= p(\boldsymbol{x}_T) \prod_{t=1}^{T} p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)
\end{aligned}$$

We can rewrite the probabilities on the form $p(\boldsymbol{x}_t|\boldsymbol{x}_{(t+1):T})$ as $p(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})$ due to our assumptions, that is the that the distribution of some noisy image $\boldsymbol{x}_t$ ONLY depends on the prior image $\boldsymbol{x}_{t+1}$.

Now, let us look at the denominator:

$$\begin{aligned}
q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0) &= \frac{q(\boldsymbol{x}_{1:T}, \boldsymbol{x}_0)}{q(\boldsymbol{x}_0)} \\
&= \frac{q(\boldsymbol{x}_0, \boldsymbol{x}_1 \cdots \boldsymbol{x}_T)}{q(\boldsymbol{x}_0)} \\
&= \frac{q(\boldsymbol{x}_T|\boldsymbol{x}_{(T-1):0})q(\boldsymbol{x}_{T-1}|\boldsymbol{x}_{(T-2):0}) \cdots q(\boldsymbol{x}_1|\boldsymbol{x}_0)q(\boldsymbol{x}_0)}{q(\boldsymbol{x}_0)} \\
&= \frac{q(\boldsymbol{x}_T|\boldsymbol{x}_{T-1})q(\boldsymbol{x}_{T-1}|\boldsymbol{x}_{T-2}) \cdots q(\boldsymbol{x}_1|\boldsymbol{x}_0)q(\boldsymbol{x}_0)}{q(\boldsymbol{x}_0)} \\
&= q(\boldsymbol{x}_T|\boldsymbol{x}_{T-1})q(\boldsymbol{x}_{T-1}|\boldsymbol{x}_{T-2}) \cdots q(\boldsymbol{x}_1|\boldsymbol{x}_0) \\
&= \prod_{t=1}^{T} q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})
\end{aligned}$$

Once again, we used the properties of the Markov chain to simplify the above.
We'll insert these new expressions in the fraction:

$$\begin{aligned}
\log(p(\boldsymbol{x}_0)) &\geq \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log\left(\frac{p(\boldsymbol{x}_T)\prod_{t=1}^{T} p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{\prod_{t=1}^{T} q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})}\right)\right] \\
&= \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log\left(\frac{p(\boldsymbol{x}_T)p(\boldsymbol{x}_0|\boldsymbol{x}_1)\prod_{t=2}^{T} p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\boldsymbol{x}_T|\boldsymbol{x}_{T-1})\prod_{t=1}^{T-1} q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})}\right)\right] \\
&= \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log\left(\frac{p(\boldsymbol{x}_T)p(\boldsymbol{x}_0|\boldsymbol{x}_1)\prod_{t=1}^{T-1} p(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})}{q(\boldsymbol{x}_T|\boldsymbol{x}_{T-1})\prod_{t=1}^{T-1} q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})}\right)\right] \\
&= \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log\frac{p(\boldsymbol{x}_T)p(\boldsymbol{x}_0|\boldsymbol{x}_1)}{p(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})}\right] + \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log\prod_{t=1}^{T-1}\frac{p(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})}{q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})}\right] \\
&= \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log p(\boldsymbol{x}_0|\boldsymbol{x}_1)\right] + \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log\frac{p(\boldsymbol{x}_T)}{q(\boldsymbol{x}_T|\boldsymbol{x}_{T-1}}\right] + \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\sum_{t=1}^{T-1}\log\frac{p(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})}{q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})}\right] \\
&= \mathbb{E}_{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}\left[\log p(\boldsymbol{x}_0|\boldsymbol{x}_1)\right] + \mathbb{E}_{q(\boldsymbol{x}_{T-1},\boldsymbol{x}_T|\boldsymbol{x}_0)}\left[\log\frac{p(\boldsymbol{x}_T)}{q(\boldsymbol{x}_T|\boldsymbol{x}_{T-1})}\right] + \mathbb{E}_{q(\boldsymbol{x}_{t-1},\boldsymbol{x}_t\boldsymbol{x}_{t+1}|\boldsymbol{x}_0)}\sum_{t=1}^{T-1}\left[\log\frac{p(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})}{q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})}\right]
\end{aligned}$$

We arrive at the following expression:

$$\begin{aligned}
\log(p(\boldsymbol{x}_0)) \geq \underbrace{\mathbb{E}_{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}\left[\log p(\boldsymbol{x}_0|\boldsymbol{x}_1)\right]}_{\text{reconstruction term}} - \underbrace{\mathbb{E}_{q(\boldsymbol{x}_{T-1}|\boldsymbol{x}_0)}\left[D_{KL}(q(\boldsymbol{x}_T|\boldsymbol{x}_{T-1})||p(\boldsymbol{x}_T))\right]}_{\text{prior matching term}} \\
- \sum_{t=1}^{T-1}\underbrace{\mathbb{E}_{q(\boldsymbol{x}_{t-1},\boldsymbol{x}_{t+1}|\boldsymbol{x}_0)}\left[D_{KL}(q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})||p(\boldsymbol{x}_t|\boldsymbol{x}_{t+1}))\right]}_{\text{consistency term}}
\end{aligned}$$

We will call this above expression the "naive lower bound" for reasons that will become apparent in a second. It is worth looking at this naive lower bound and considering what the different terms mean:

- *The reconstruction term* models the probability of the original data given the first-step latent layer. If you have studied VAEs before, you'll recognize this term. It is also the term that "connects" the model with the real-world data. This term is high if we can predict the real image given the first-step latent layer.

- *The prior matching term* makes sure that our last-step latent variable follows some prior $p(\boldsymbol{x}_T)$. It is minimized if our distribution matches the prior. Normally, we put $p(\boldsymbol{x}_T) = \mathcal{N}(0, 1)$. We'll discuss this much more later.

- The consistency term makes sure that the distribution of $\boldsymbol{x}_t$ is consistent, such that the distribution is the same when we're going forward and backward in the diffusion process. Since the KL-divergence is a kind of similarity measure between probability distributions, this term is minimized if $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ is the same as or close to $p(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})$.

Now, we are ready to train a model. "Simply" define $p_\theta(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})$ with learnable parameters, and use equation (1) for $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ to optimize the naive lower bound. We could continue explaining how exactly one would set up such an algorithm, but as it turns out, this will only give sup-optimal results. There is a better way of reformulating the lower bound and making it more "aware" by conditioning on available information.

3.1 Conditioning on $\boldsymbol{x}_0$

We'll now try to improve the before-mentioned lower bound. We'll do this by conditioning on available information. Realize that the following must hold due to the Markov chain:

$$q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{x}_0)$$

Properties of Markov chain

Due to the Markov chain, the following is true:

$$p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, a) = p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) \qquad \text{and} \qquad q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, a) = q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$$

For any a.

That is, that every $\boldsymbol{x}_t$ only depends on $\boldsymbol{x}_{t-1}$ (as described above). Therefore, we can condition on whatever variables we want. Let us insert this and get:

$$\begin{aligned}
\log(p(\boldsymbol{x}_0)) &\geq \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log\left(\frac{p(\boldsymbol{x}_T)p(\boldsymbol{x}_0|\boldsymbol{x}_1)\prod_{t=2}^{T}p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\boldsymbol{x}_1|\boldsymbol{x}_0)\prod_{t=2}^{T}q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{x}_0)}\right)\right] \\
&= \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log\left(\frac{p(\boldsymbol{x}_T)p(\boldsymbol{x}_0|\boldsymbol{x}_1)}{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}\prod_{t=2}^{T}\frac{p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{x}_0)}\right)\right] \\
&= \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log\left(\frac{p(\boldsymbol{x}_T)p(\boldsymbol{x}_0|\boldsymbol{x}_1)}{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}\right) + \log\left(\prod_{t=2}^{T}\frac{p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{x}_0)}\right)\right]
\end{aligned}$$

We can rewrite $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{x}_0)$ using Bayes:

$$q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{x}_0) = \frac{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)q(\boldsymbol{x}_t|\boldsymbol{x}_0)}{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)}$$

This is also inserted into the expression. Thereafter, we can go on algebra-autopilot:

$$\begin{aligned}
\log(p(\boldsymbol{x}_0) \geq \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)} &\left[\log\left(\frac{p(\boldsymbol{x}_T)p(\boldsymbol{x}_0|\boldsymbol{x}_1)}{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}\right) + \log\left(\prod_{t=2}^{T}\frac{p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{\frac{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)q(\boldsymbol{x}_t|\boldsymbol{x}_0)}{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)}}\right)\right] \\
= \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)} &\left[\log\left(\frac{p(\boldsymbol{x}_T)p(\boldsymbol{x}_0|\boldsymbol{x}_1)}{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}\right) + \log\left(\prod_{t=2}^{T}\frac{p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)}{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\right)\right] \\
= \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)} &\left[\log\left(\frac{p(\boldsymbol{x}_T)p(\boldsymbol{x}_0|\boldsymbol{x}_1)}{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}\right) + \log\left(\prod_{t=2}^{T}\frac{p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)}\right) + \log\left(\prod_{t=2}^{T}\frac{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)}{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\right)\right] \\
= \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)} &\left[\log\left(\frac{p(\boldsymbol{x}_T)p(\boldsymbol{x}_0|\boldsymbol{x}_1)}{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}\right) + \log\left(\prod_{t=2}^{T}\frac{p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)}\right) + \log\left(\frac{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}{q(\boldsymbol{x}_T|\boldsymbol{x}_0)}\right)\right] \\
= \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)} &\left[\log\left(\frac{p(\boldsymbol{x}_T)p(\boldsymbol{x}_0|\boldsymbol{x}_1)}{q(\boldsymbol{x}_T|\boldsymbol{x}_0)}\right) + \log\left(\prod_{t=2}^{T}\frac{p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)}\right)\right] \\
= \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)} &\left[\log\left(p(\boldsymbol{x}_0|\boldsymbol{x}_1)\right) + \log\left(\frac{q(\boldsymbol{x}_T)}{p(\boldsymbol{x}_T|\boldsymbol{x}_0)}\right) + \log\left(\prod_{t=2}^{T}\frac{p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)}\right)\right] \\
= \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)} &\left[\log(p(\boldsymbol{x}_0|\boldsymbol{x}_1)\right] + \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log\left(\frac{p(\boldsymbol{x}_T)}{q(\boldsymbol{x}_T|\boldsymbol{x}_0)}\right)\right] + \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log\left(\prod_{t=2}^{T}\frac{p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)}\right)\right] \\
= \mathbb{E}_{q(\boldsymbol{x}_1|\boldsymbol{x}_0)} &\left[\log(p(\boldsymbol{x}_0|\boldsymbol{x}_1)\right] + \mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[\log\left(\frac{p(\boldsymbol{x}_T)}{q(\boldsymbol{x}_T|\boldsymbol{x}_0)}\right)\right] + \sum_{t=2}^{T}\mathbb{E}_{q(\boldsymbol{x}_t,\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)}\left[\log\left(\frac{p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)}\right)\right]
\end{aligned}$$

Please note how the expectation in the first term changes from $\mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}(\cdots)$ to $\mathbb{E}_{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}(\cdots)$, and similar for the other expectations. This can be derived from the fact that the function inside the expectation value only contains a subset of the conditional events.

Expectation value over a subset of conditional events

Conditional events can be ignored in the distribution involved in an expectation value if the said conditional events aren't included in the expression inside the expectation value. This can formally be described as:

$$\begin{aligned}
\mathbb{E}_{q(a,b|c)}(f(a)) &= \int\int q(a,b|c)f(a)\,da\,db \\
&= \int f(a)\int q(a,b|c)\,db\,da \\
&= \int f(a)q(a|c)\,da \\
&= \mathbb{E}_{q(a|c)}(f(a))
\end{aligned}$$

Kullback-Leibler divergence

The Kullback-Leibler divergence (often shortened as KL-divergence) is a kind of similarity measure between distributions. A low Kullback-Leibler divergence means that the distributions are similar to each other. The KL-divergence is defined as:

$$D_{KL}(p(x)||q(x)) = -\int p(x)\log\left(\frac{q(x)}{p(x)}\right)\,dx = -\mathbb{E}\left[\log\left(\frac{q(x)}{p(x)}\right)\right]$$

We'll apply the KL-divergence to the expression above:

$$\log(p(\boldsymbol{x}_0) \geq \mathbb{E}_{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}[\log(p(\boldsymbol{x}_0|\boldsymbol{x}_1)] - \mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[D_{KL}(q(\boldsymbol{x}_T|\boldsymbol{x}_0||p(\boldsymbol{x}_t))\right] + \sum_{t=2}^{T}\mathbb{E}_{q(\boldsymbol{x}_t,\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)}\left[\log\left(\frac{p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)}\right)\right]$$

The very last term in the above expression can also be rewritten using the KL-divergence. This is not totally trivial though, and it requires that we rewrite the expectation value as the initial integral it arose from:

$$\begin{aligned}
\mathbb{E}_{q(\boldsymbol{x}_t\boldsymbol{x}_{t-1}|x0)}\left[\log\left(\frac{p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)}\right)\right] &= \int\int q(\boldsymbol{x}_t,\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)\log\left(\frac{p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)}\right)dx_t\,dx_{t-1}\\
&= \int\int q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)q(\boldsymbol{x}_t|\boldsymbol{x}_0)\log\left(\frac{p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)}\right)d\boldsymbol{x}_t\,dx_{t-1}\\
&= \int q(\boldsymbol{x}_t|\boldsymbol{x}_0)\int q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)\log\left(\frac{p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)}\right)dx_{t-1}\,dx_t\\
&= -\int q(\boldsymbol{x}_t|\boldsymbol{x}_0)D_{KL}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)||p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))\,dx_t\\
&= -\mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[D_{KL}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)||p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))\right]
\end{aligned}$$

Now, we can write our final lower bound for the logarithm of $p(\boldsymbol{x}_0)$. This expression is also called the evidence lower bound, shortened ELBO:

$$\begin{aligned}
\log(p(\boldsymbol{x}_0)) \geq \mathbb{E}_{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}[\log(p_\theta(\boldsymbol{x}_0|\boldsymbol{x}_1)] - \mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[D_{KL}(q(\boldsymbol{x}_T|\boldsymbol{x}_0||p(\boldsymbol{x}_T))\right]&\\
-\sum_{t=2}^{T}\mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[D_{KL}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)||p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))\right]&
\end{aligned}$$

Now, this is the expression for the ELBO, which we want to maximize. Oftentimes, we put a negative sign in front and the problem therefore becomes a minimization problem.

4 Interpreting the ELBO

Let us look at the different terms in the ELBO. First, let us name the terms appropriately:

$$\begin{aligned}
\log(p(\boldsymbol{x}_0)) \geq \underbrace{\mathbb{E}_{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}[\log(p_\theta(\boldsymbol{x}_0|\boldsymbol{x}_1)]}_{L_0} - \underbrace{\mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[D_{KL}(q(\boldsymbol{x}_T|\boldsymbol{x}_0||p(\boldsymbol{x}_T))\right]}_{L_T}&\\
-\underbrace{\sum_{t=2}^{T}\mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[D_{KL}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)||p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))\right]}_{L_{t-1}}& \quad (2)
\end{aligned}$$

- L_0 is the reconstruction error. It examines whether the ground truth image $\boldsymbol{x}_0$ scores high in the distribution resulting from the second to last layer in the backward diffusion process, $\boldsymbol{x}_1$.
- L_T is the "prior matching term". It examines whether the last layer in the forward diffusion process matches the prior. We haven't chosen a prior yet, but considering that we have chosen the diffusion process to go towards a standard normal, it would be natural to choose $p(\boldsymbol{x}_T) \sim \mathcal{N}(0,1)$. This term has no trainable parameters and can therefore be ignored when training the model since it won't affect the gradients.
- L_{t-1} is the denoising matching term. It examines whether our learnable denoising transition step $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ matches the ground truth denoising step $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)$. To minimize this term, the two distributions should be as close to each other as possible therefore minimizing the KL divergence between them. The question is: how do we parameterize these distributions?

4.1 Why can we ignore L_T

Let's discuss the term L_T. We will not allocate any learnable parameters to this term and in practice the importance of this will be negligible, assuming we intelligently choose our prior. It will not impact our model's choice of the optimal parameter values, θ^*. To be precise, we could choose to define

$p(\boldsymbol{x}_T)$ as a standard, normal Gaussian. And since we have chosen our diffusion schedule such that $\boldsymbol{x}_T$ approximately ends up looking like a standard Gaussian, this term will virtually be equal to zero anyways.

In order to convince you, and ourselves, that this term is virtually equal to zero during training as seen in [HJA20], we have done some experimentation with the noising schedule and prior. It is important that the chosen prior distribution $p(\boldsymbol{x}_T)$ and the final distribution in the noising process are very similar, since our model is trying to work backwards from this prior. If these are too dissimilar, there is a dissonance between the stopping point of the forward process and the starting point of the backward process, thus making the learning problem infinitely harder. In order to investigate this we took an image(here from MNIST) and ran it through the forward process with $\beta = (0.0001, 0.02)$ and $T = 500$ and compared the resulting distribution to $p(\boldsymbol{x}_T)$ throughout, see figure 6. We use this to argue that satisfactory asymptotic behaviour is present and we can safely disregard this term in the ELBO. A visual representation of the noisified input image can be seen in 1.

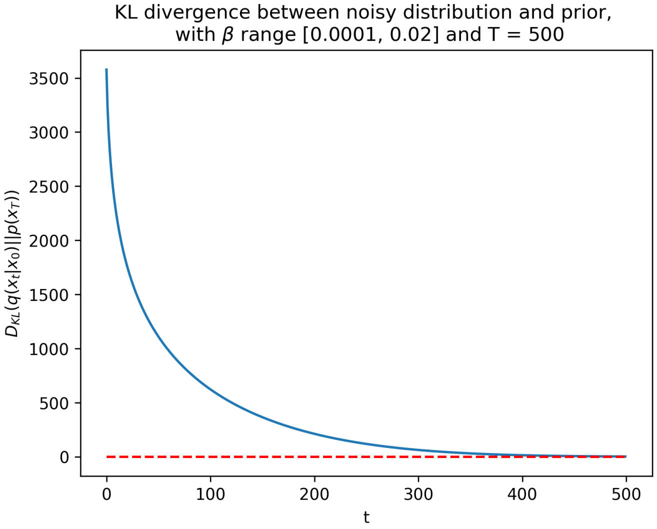

Figure 6: Comparison of KL divergences for each timestep.

4.2 Deriving the distributions in L_{t-1}

As mentioned earlier, we can minimize L_{t-1} if we let $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) = q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$. We are going to call $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$ the ”ground-truth denoising transition step”; this makes sense since it has access to the original image $\boldsymbol{x}_0$. However, obviously, it is not possible to completely match our learnable denoising step with the ground-truth denoising step, simply because the ground-truth step has access to the ground-truth picture $\boldsymbol{x}_0$ and our learnable step doesn't. To overcome this problem, we will derive the actual distribution of the ground-truth step using Bayes:

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

In the second to last line, we implicitly include $C(\boldsymbol{x}_t, \boldsymbol{x}_0)$ and can therefore complete the square (the square is on the form $(x_{t-1} - \mu_q)^2$, so the curious reader will just have to confirm that $\frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}C(\boldsymbol{x}_t, \boldsymbol{x}_0)$ is actually equal to μ_q^2).

Therefore, given some noisy picture $\boldsymbol{x}_t$, we can derive the distribution of the slightly less noisy picture $\boldsymbol{x}_{t-1}$ and sample from this distribution. To make things shorter, we will define the following mean and variance:

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

$$\Sigma_q(t) = \frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{I}$$

$$\Downarrow$$

$$q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) = \mathcal{N}(\mu_q(\boldsymbol{x}_t, t), \Sigma_q(t))$$

To make the following derivations more readable, we'll denote $\boldsymbol{\Sigma}_q(t) = \sigma_q^2(t)\mathbf{I}$, such that $\sigma_q^2(t) = \frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}$. Our goal is to match our learnable denoising step with the ground-truth denoising step. Since the ground-truth denoising step have been shown to follow a Gaussian distribution, we can likewise parameterize the learnable step as a Gaussian in almost the same way, and we can name the mean and variance of the this distributions appropriately. That is:

$$p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_{t-1}; \mu_\theta(\boldsymbol{x}_t, t), \sigma_\theta^2(t))$$

Here, we can realize that the "true" variance, $\Sigma_q(t) = \sigma_q^2(t)$, is a function of only t and therefore we reuse this variance in the learnable transition step!
Likewise, we would also like to use the true mean, $\mu_q(\boldsymbol{x}_t, t)$. However, since this mean depends on $\boldsymbol{x}_0$, we can not use it directly (remember, we will not have access to $\boldsymbol{x}_0$ in our learnable denoising step, only to $\boldsymbol{x}_t$!). However, perhaps we can somehow approximate $\boldsymbol{x}_0$ on the basis of the available information. In particular, we could approximate $\boldsymbol{x}_0$ via a neural network, using $\boldsymbol{x}_t$ and t as the inputs. In other words, we can write:

$$\mu_\theta(\boldsymbol{x}_t, t) = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\boldsymbol{x}_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\hat{\boldsymbol{x}}_0(\boldsymbol{x}_t, t)}{1-\bar{\alpha}_t}$$

Therefore, the learnable denoising step has been parameterized to follow a Gaussian distribution with parameters $\mu_\theta(\boldsymbol{x}_t, t)$ and $\Sigma_q(t)$:

$$p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) = \mathcal{N}\left(\boldsymbol{x}_{t-1}; \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\boldsymbol{x}_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\hat{\boldsymbol{x}}_0(\boldsymbol{x}_t, t)}{1-\bar{\alpha}_t}, \frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{I}\right)$$

Now that we have parameterized the two distributions $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ and $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$, we can insert these distributions into the formula for the KL-divergence and calculate the term L_{t-1}. And since both distributions are normal distributions, the KL-divergence turns out to be extremely easy to calculate.

KL-divergence for multivariate Gaussians

Given two multivariate normal distributions such that:

$$p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}; \mu_{\boldsymbol{x}}, \boldsymbol{\Sigma}_{\boldsymbol{x}}) \qquad q(\boldsymbol{x}) = \mathcal{N}(\mathbf{y}; \mu_y, \boldsymbol{\Sigma}_y)$$

Then the KL-divergence can be calculated as:

$$D_{KL}(p(\boldsymbol{x})||q(\boldsymbol{x})) = \frac{1}{2}\left[\log\frac{|\boldsymbol{\Sigma}_y|}{|\boldsymbol{\Sigma}_x|} - d + \mathrm{tr}(\boldsymbol{\Sigma}_y^{-1}\boldsymbol{\Sigma}_x) + (\mu_y - \mu_x)^T\boldsymbol{\Sigma}_y^{-1}(\mu_y - \mu_x)\right]$$

Let us substitute in our distributions (to make things more readable we'll write $\mu_q = \mu_q(\boldsymbol{x}_t, t)$ and $\mu_\theta = \mu_\theta(\boldsymbol{x}_t, t)$):

$$\begin{aligned}
D_{KL}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)||p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)) &= D_{KL}(\mathcal{N}(\boldsymbol{x}_{t-1}; \mu_q, \boldsymbol{\Sigma}_q(t))||\mathcal{N}(\boldsymbol{x}_{t-1}; \mu_\theta, \boldsymbol{\Sigma}_q(t)) \\
&= \frac{1}{2}\left[\log\frac{|\boldsymbol{\Sigma}_q(t)|}{|\boldsymbol{\Sigma}_q(t)|} - d + \mathrm{tr}(\boldsymbol{\Sigma}_q(t)^{-1}\boldsymbol{\Sigma}_q(t)) + (\mu_\theta - \mu_q)^T\boldsymbol{\Sigma}_q(t)^{-1}(\mu_\theta - \mu_q)\right] \\
&= \frac{1}{2}\left[\log(1) - d + d + (\mu_\theta - \mu_q)^T\boldsymbol{\Sigma}_q(t)^{-1}(\mu_\theta - \mu_q)\right] \\
&= \frac{1}{2}\left[(\mu_\theta - \mu_q)^T\sigma_q^2(t)^{-1}\mathbf{I}(\mu_\theta - \mu_q)\right] \\
&= \frac{1}{2\sigma_q^2(t)}\left[(\mu_\theta - \mu_q)^T\mathbf{I}(\mu_\theta - \mu_q)\right] \\
&= \frac{1}{2\sigma_q^2(t)}||\mu_\theta - \mu_q||_2^2
\end{aligned}$$

So, the KL-divergence boils down to being some factor, which depends on t, times the squared distance between the means of the two distributions. This should make somewhat good sense considering that

the covariance matrices were identical.
Therefore, the term L_{t-1} can be rewritten as:

$$L_{t-1} = \sum_{t=2}^{T} \mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)} \left[\frac{1}{2\sigma_q^2(t)} ||\mu_\theta(\boldsymbol{x}_t, t) - \mu_q(\boldsymbol{x}_t, t)||_2^2 \right]$$

4.3 Deriving an expression for L_0

Now let's look at L_0:

$$L_0 = \mathbb{E}_{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}[\log(p_\theta(\boldsymbol{x}_0|\boldsymbol{x}_1)]$$

Remembering how we chose to parameterize the distribution p_θ, we can rewrite the above as:

$$\begin{aligned} L_0 &= \mathbb{E}_{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}[\log(\mathcal{N}(\boldsymbol{x}_0|\mu_\theta(\boldsymbol{x}_1, 1), \Sigma_q(1))] \\ &= \mathbb{E}_{q(\boldsymbol{x}_1|\boldsymbol{x}_0)} \left[-\frac{k}{2}\log(2\pi) - \frac{k}{2}\log(\sigma_q^2(1)) - \frac{1}{2\sigma_q^2(1)} ||x_0 - \mu_\theta||_2^2 \right] \\ &= \mathbb{E}_{q(\boldsymbol{x}_1|\boldsymbol{x}_0)} \left[C - \frac{1}{2\sigma_q^2(1)} ||\mu_\theta(\boldsymbol{x}_1, 1) - \boldsymbol{x}_0||_2^2 \right] = C - \mathbb{E}_{q(\boldsymbol{x}_1|\boldsymbol{x}_0)} \left[\frac{1}{2\sigma_q^2(1)} ||\mu_\theta(\boldsymbol{x}_1, 1) - \boldsymbol{x}_0||_2^2 \right] \end{aligned}$$

Where $C = -\frac{k}{2}\log(2\pi) - \frac{k}{2}\log(\sigma_q(1))$ is some constant that doesn't have any parameters and therefore doesn't affect the gradients (remember that the variance is fixed). Technically, we could also choose to learn the variance to potentially optimize this term further.

Logarithm of multivariate Gaussian distribution with diagonal covariance matrix with constant terms

$$\begin{aligned} \log(\mathcal{N}(x|\mu, \Sigma)) &= \log\left((2\pi)^{-k/2)} \det(\Sigma)^{-1/2} \exp\left\{ -\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu) \right\} \right) \\ &= -\frac{k}{2}\log(2\pi) - \frac{1}{2}\log(\det(\Sigma)) - \frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu) \end{aligned}$$

Since Σ is a diagonal matrix with constant terms, it can be written as:

$$\Sigma = \sigma^2(t)\mathbf{I}$$

Where $\mathbf{I}$ is the identity matrix. In that case, the determinant simply becomes the product of the diagonal elements. Therefore, we get:

$$\begin{aligned} \log(\mathcal{N}(x|\mu, \Sigma)) &= -\frac{k}{2}\log(2\pi) - \frac{1}{2}\log(\sigma^{2k}(t)) - \frac{1}{2}(x-\mu)^T \left(\frac{1}{\sigma^2(t)}\mathbf{I} \right) (x-\mu) \\ &= -\frac{k}{2}\log(2\pi) - \frac{k}{2}\log(\sigma^2(t)) - \frac{1}{2\sigma^2}(x-\mu)^T(x-\mu) \\ &= -\frac{k}{2}\log(2\pi) - \frac{k}{2}\log(\sigma^2(t)) - \frac{1}{2\sigma^2}||x-\mu||_2^2 \end{aligned}$$

4.4 Combining the expressions

Let's combine what we have so far: we have expressions for L_0 and L_{t-1} and we can therefore rewrite the ELBO:

$$\begin{aligned} &\log(p(\boldsymbol{x}_0)) \geq L_0 - L_T - L_{t-1} \\ &= C - L_T - \mathbb{E}_{q(\boldsymbol{x}_1|\boldsymbol{x}_0)} \left[\frac{1}{2\sigma_q^2(1)} ||\mu_\theta(\boldsymbol{x}_1, 1) - \boldsymbol{x}_0||_2^2 \right] - \sum_{t=2}^{T} \mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)} \left[\frac{1}{2\sigma_q^2(t)} ||\mu_\theta(\boldsymbol{x}_t, t) - \mu_q(\boldsymbol{x}_t, t)||_2^2 \right] \\ &= C - L_T - \sum_{t=1}^{T} \mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)} \left[\frac{1}{2\sigma_q^2(t)} ||\mu_\theta(\boldsymbol{x}_t, t) - \mu_q(\boldsymbol{x}_t, t)||_2^2 \right] \end{aligned}$$

We can extend the summation sign by realizing that $\mu_q(\boldsymbol{x}_1, 1) = \boldsymbol{x}_0$ (this can be shown algebraically or simply by remembering the purpose of the ground truth denoising step).
Since C and L_T contain no learnable parameters, we are only interested in minimizing the last term. Also, we can move the constant $\frac{1}{2\sigma_q^2(t)}$ outside the expectation value since it is only a function of t. Therefore, in the end, we end up with the following optimization problem:

$$\theta^* = \arg\min_{\theta} \sum_{t=1}^{T} \frac{1}{2\sigma_q^2(t)} \mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)} \left[||\mu_\theta(\boldsymbol{x}_t, t) - \mu_q(\boldsymbol{x}_t, t)||_2^2 \right]$$

5 The simple loss

In the prior section, we deduced a very simple loss function from the initial ELBO by realizing that we could ignore some constants since they don't influence the gradients of our network. The loss function looked as follows:

$$\mathcal{L} = \sum_{t=1}^{T} \frac{1}{2\sigma_q^2(t)} \mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)} \left[||\mu_\theta(\boldsymbol{x}_t, t) - \mu_q(\boldsymbol{x}_t, t)||_2^2 \right] \tag{3}$$

Let's insert our expressions for μ_θ and μ_q:

$$\begin{aligned}
\mathcal{L} &= \sum_{t=1}^{T} \frac{1}{2\sigma_q^2(t)} \mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)} \left[||\mu_\theta(\boldsymbol{x}_t, t) - \mu_q(\boldsymbol{x}_t, t)||_2^2 \right] \\
&= \sum_{t=1}^{T} \frac{1}{2\sigma_q^2(t)} \mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)} \left[\left|\left| \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\boldsymbol{x}_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\hat{\boldsymbol{x}}_0(\boldsymbol{x}_t, t)}{1-\bar{\alpha}_t} - \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\boldsymbol{x}_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\boldsymbol{x}_0}{1-\bar{\alpha}_t} \right|\right|_2^2 \right] \\
&= \sum_{t=1}^{T} \frac{1}{2\sigma_q^2(t)} \mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)} \left[\left|\left| \frac{\sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)(\hat{\boldsymbol{x}}_0(\boldsymbol{x}_t, t) - \boldsymbol{x}_0)}{1-\bar{\alpha}_t} \right|\right|_2^2 \right] \\
&= \sum_{t=1}^{T} \frac{1}{2\sigma_q^2(t)} \frac{\bar{\alpha}_{t-1}(1-\alpha_t)^2}{(1-\bar{\alpha}_t)^2} \mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)} \left[||\hat{\boldsymbol{x}}_0(\boldsymbol{x}_t, t) - \boldsymbol{x}_0||_2^2 \right]
\end{aligned}$$

So it turns out that the loss function actually amounts to minimizing the squared distance between the real ground truth image $\boldsymbol{x}_0$ and our obtained guess $\hat{\boldsymbol{x}}_0$.
And we can take this one step further. We can rewrite the expression for $\boldsymbol{x}_0$ (by remembering the extended reparameterization trick from earlier):

$$\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon^* \quad \Rightarrow \quad \boldsymbol{x}_0 = \frac{\boldsymbol{x}_t - \sqrt{1-\bar{\alpha}_t}\epsilon^*}{\sqrt{\bar{\alpha}_t}}$$

We can insert this into the expression for μ_q:

$$\begin{aligned}
\mu_q(\boldsymbol{x}_t, t) &= \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\boldsymbol{x}_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\boldsymbol{x}_0}{1-\bar{\alpha}_t} \\
&= \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\boldsymbol{x}_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\frac{\boldsymbol{x}_t - \sqrt{1-\bar{\alpha}_t}\epsilon^*}{\sqrt{\bar{\alpha}_t}}}{1-\bar{\alpha}_t} \\
&= \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\boldsymbol{x}_t + (1-\alpha_t)\frac{\boldsymbol{x}_t - \sqrt{1-\bar{\alpha}_t}\epsilon^*}{\sqrt{\alpha_t}}}{1-\bar{\alpha}_t} \\
&= \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\boldsymbol{x}_t}{(1-\bar{\alpha}_t)} + \frac{(1-\alpha_t)\boldsymbol{x}_t}{(1-\bar{\alpha}_t)\sqrt{\alpha_t}} - \frac{(1-\alpha_t)\sqrt{1-\bar{\alpha}_t}}{(1-\bar{\alpha}_t)\sqrt{\alpha_t}}\epsilon^* \\
&= \left(\frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{(1-\bar{\alpha}_t)} + \frac{(1-\alpha_t)}{(1-\bar{\alpha}_t)\sqrt{\alpha_t}}\right)\boldsymbol{x}_t - \frac{(1-\alpha_t)\sqrt{1-\bar{\alpha}_t}}{(1-\bar{\alpha}_t)\sqrt{\alpha_t}}\epsilon^* \\
&= \left(\frac{\alpha_t(1-\bar{\alpha}_{t-1})}{(1-\bar{\alpha}_t)\sqrt{\alpha_t}} + \frac{(1-\alpha_t)}{(1-\bar{\alpha}_t)\sqrt{\alpha_t}}\right)\boldsymbol{x}_t - \frac{(1-\alpha_t)\sqrt{1-\bar{\alpha}_t}}{(1-\bar{\alpha}_t)\sqrt{\alpha_t}}\epsilon^* \\
&= \frac{\alpha_t(1-\bar{\alpha}_{t-1}) + 1 - \alpha_t}{(1-\bar{\alpha}_t)\sqrt{\alpha_t}}\boldsymbol{x}_t - \frac{(1-\alpha_t)\sqrt{1-\bar{\alpha}_t}}{(1-\bar{\alpha}_t)\sqrt{\alpha_t}}\epsilon^* \\
&= \frac{1-\bar{\alpha}_t}{(1-\bar{\alpha}_t)\sqrt{\alpha_t}}\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}\epsilon^* \\
&= \frac{1}{\sqrt{\alpha_t}}\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}\epsilon^*
\end{aligned}$$

Likewise, since we want to parameterize our forward and backward process in the same way, we can rewrite our expression for μ_θ in the same way. However, since the expression no longer contains $\hat{\boldsymbol{x}}_0$ we instead choose to predict the noise ϵ^*. We will call this expression $\hat{\epsilon}_\theta(\boldsymbol{x}_t, t)$

$$\mu_\theta = \frac{1}{\sqrt{\alpha_t}}\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}\hat{\epsilon}_\theta(\boldsymbol{x}_t, t)$$

We can insert this new expression for μ_q and μ_θ into the expression for $\mathcal{L}$ exactly in the same way as we did for our prior definition of μ_q and μ_θ:

$$\begin{aligned}
\mathcal{L} &= \sum_{t=1}^{T} \frac{1}{2\sigma_q^2(t)} \mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[||\mu_\theta(\boldsymbol{x}_t, t) - \mu_q(\boldsymbol{x}_t, t)||_2^2\right] \\
&= \sum_{t=1}^{T} \frac{1}{2\sigma_q^2(t)} \mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[\left|\left|\left(\frac{1}{\sqrt{\alpha_t}}\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}\hat{\epsilon}_\theta(\boldsymbol{x}_t, t)\right) - \left(\frac{1}{\sqrt{\alpha_t}}\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}\epsilon^*\right)\right|\right|_2^2\right] \\
&= \sum_{t=1}^{T} \frac{1}{2\sigma_q^2(t)} \mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[\left|\left|\frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}\epsilon^* - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}\hat{\epsilon}_\theta(\boldsymbol{x}_t, t)\right|\right|_2^2\right] \\
&= \sum_{t=1}^{T} \frac{1}{2\sigma_q^2(t)} \mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[\left|\left|\frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}(\epsilon^* - \hat{\epsilon}_\theta(\boldsymbol{x}_t, t))\right|\right|_2^2\right] \\
&= \sum_{t=1}^{T} \frac{1}{2\sigma_q^2(t)} \frac{(1-\alpha_t)^2}{(1-\bar{\alpha}_t)\alpha_t} \mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[||\epsilon^* - \hat{\epsilon}_\theta(\boldsymbol{x}_t, t)||_2^2\right]
\end{aligned}$$

Therefore, if we instead choose to guess the noise $\hat{\epsilon}_\theta$ (instead of finding $\hat{\boldsymbol{x}}_{0\theta}$) then the objective function amounts to minimizing the distance between the real noise and our estimated noise. It turns out that this kind of parameterization empirically gives better results. Some of these results can be seen in [HJA20]. In this paper, they also argue that the loss function can be simplified even further by ignoring the factor in front of the KL-divergence, therefore obtaining the ”simple loss function”:

$$\mathcal{L}_{simple} = ||\epsilon^* - \hat{\epsilon}_\theta(\boldsymbol{x}_t, t)||_2^2, \qquad \boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon^*, \qquad t \sim \text{Uniform}(\{1, \ldots T\}), \qquad \epsilon^* \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$$

How can we arrive at this simple loss function? First of all, we can choose to "ignore" the expectation value by estimating it by drawing just a single sample from the distribution. More formally, we can write:

$$\begin{aligned}\mathcal{L} &= \sum_{t=1}^{T} \frac{1}{2\sigma_q^2(t)} \frac{(1-\alpha_t)^2}{(1-\bar{\alpha}_t)\alpha_t} \mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)} \left[||\epsilon^* - \hat{\epsilon}_\theta(\boldsymbol{x}_t, t)||_2^2 \right] \\ &\approx \sum_{t=1}^{T} \frac{1}{2\sigma_q^2(t)} \frac{(1-\alpha_t)^2}{(1-\bar{\alpha}_t)\alpha_t} ||\epsilon^* - \hat{\epsilon}_\theta(\boldsymbol{x}_t, t)||_2^2 \,, \qquad \boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon^*\end{aligned}$$

Next, we can realize that the sum is minimized if each term inside the sum is minimized. Here, we are making a big assumption; that the different network evaluations, i.e. $\hat{\epsilon}_\theta(\boldsymbol{x}_i, i)$ and $\hat{\epsilon}_\theta(\boldsymbol{x}_j, j)$ for all $i \neq j$, are decoupled such that the optimal parameters for the term belonging to i have no influence on the optimal parameters for the term belonging to j. To be correct, this assumption is not true for our model since we're using the same parameters for each t but it turns out that it is good enough to properly train the model.

In that case, we can optimize each term inside the sum:

$$\arg\min_\theta \frac{1}{2\sigma_q^2(t)} \frac{(1-\alpha_t)^2}{1-\bar{\alpha}_t\alpha_t} ||\epsilon^* - \hat{\epsilon}_\theta(\boldsymbol{x}_t, t)||_2^2 = \arg\min_\theta ||\epsilon^* - \hat{\epsilon}_\theta(\boldsymbol{x}_t, t)||_2^2$$

And if we want to optimize the term for all t it would be a good idea to sample t such that $t \sim \text{Uniform}(\{1, \ldots, T\})$. If we instead chose to first optimize the term for $t = 1$, then $t = 2$, and so on, we could very quickly run into problems with our wrong assumption about the model being decoupled since the model essentially would have "forgotten" the optimal parameters for $t = 1$ by the time it got to $t = T$. By randomly sampling t uniformly, we're constantly paying attention to all timesteps every once in a while (one could argue that it could be an idea to sample t such that some timesteps appear more often than others if those particular timesteps are important; actually, the constant $\frac{1}{2\sigma_q^2(t)} \frac{(1-\alpha_t)^2}{1-\bar{\alpha}_t\alpha_t}$ that was removed from the expression hints towards the lower t being more important. This can be confirmed by plotting the constant and observing that it is higher for low values of t).

5.1 Training and sampling

Here comes probably the most exciting part. How do we set up and train our diffusion model and how do we sample new pictures from it? Here, we can once again draw inspiration from [HJA20] and their algorithms. However, it would also be a good idea to go through some of the expressions that we have derived so far to get an intuitive understanding of what the sampling process actually does.

Remember, that we can generate arbitrarily noisy images from $\boldsymbol{x}_0$ using our forward process $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$. But when sampling we do not have access to $\boldsymbol{x}_0$; instead we want to generate some $\boldsymbol{x}_0$. We can do this using our backward process $p_\theta(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})$. Still, we need to start somewhere in order to begin sampling, that is, we need some $\boldsymbol{x}_t$ for $t \in [1, T]$. The problem is still that we don't know any $\boldsymbol{x}_t$ or their distribution. However, earlier we showed that the distribution of $p(\boldsymbol{x}_T)$ will approximately follow a normal Gaussian simply due to how the forward process was defined. We can use this fact to sample $\boldsymbol{x}_T \sim p(\boldsymbol{x}_T) \approx \mathcal{N}(0, \mathbf{I})$. Now that we have $\boldsymbol{x}_T$ we can sample $\boldsymbol{x}_{t-1}$. From $\boldsymbol{x}_{t-1}$ we can sample $\boldsymbol{x}_{t-2}$ and so on all the way down to $\boldsymbol{x}_0$ using $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) = \mathcal{N}(\mu_\theta, \sigma_q^2(t))$ (here, we can both use $\mu_\theta(\hat{\epsilon}_\theta(\boldsymbol{x}_t, t))$ and $\mu_\theta(\bar{\mathbf{x}}_0(\boldsymbol{x}_t, t))$. However, when we get to $p(\boldsymbol{x}_0|\boldsymbol{x}_1)$ we choose to let $\boldsymbol{x}_0 = \mu_\theta(\boldsymbol{x}_1)$ - we don't sample from the distribution but take the mean. This is done to avoid unnecessary noise added by the random process in the sampling procedure. In the end, we can sum up the sampling process as an algorithm:

Sampling algorithm

```
$\boldsymbol{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$
for $t$ from $T$ downto 1 do
    $\boldsymbol{x}_{t-1} \leftarrow \frac{1}{\sqrt{\alpha_t}}\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}\hat{\epsilon}_\theta(\boldsymbol{x}_t, t)$
    if $t > 0$ then
        $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$
        $\boldsymbol{x}_{t-1} \leftarrow \boldsymbol{x}_{t-1} + \mathbf{z}\sigma_q(t)$
    end if
end for
return $x_0$
```

Likewise, we can write an algorithm for training the model. Assuming that we are using the simple loss function, the algorithm for training the model can be written as:

Training algorithm

```
repeat
    $\boldsymbol{x}_0 \sim q(\boldsymbol{x}_0)$
    $t \sim \text{Uniform}(1, \ldots, T)$
    $\epsilon^* \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$
    $\boldsymbol{x}_t \leftarrow \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon^*$
    Take gradient descent step on $\nabla_\theta ||\epsilon^* - \hat{\epsilon}_\theta(\boldsymbol{x}_t, t)||_2^2$
until convergence
```

In practice, we also have to tell the model what timestep it is in. There are many ways to do this. For example, one could make T separate models, one for each $t = 1 \ldots T$ and then use the appropriate when calculating the loss. But this would require a lot of models. Another solution is to pass the timestep to the model as an integer. The problem here is that if the model input, the images, are very high dimensional and if we pass just a single integer, it'll get "washed out" and forgotten within the network. The most common solution, which we also used in our training, is to do sinusoidal encodings of the timesteps. The idea behind sinusoidal encodings is described in [VSP+23]. The good thing about sinusoidal encodings is that you can choose any time dimensionality and encode the timestep in this dimensionality. Therefore, the timestep input can be adjusted according to the size of the image input. More formally, the sinusoidal encodings are defined as:

$$\text{PosEnc}(t, i) = \begin{cases} \sin\left(\frac{t}{10000^{2i/d}}\right) & \text{if } i \text{ is even,} \\ \cos\left(\frac{t}{10000^{2i/d}}\right) & \text{if } i \text{ is odd,} \end{cases} \tag{4}$$

Where:

$\text{PosEnc}(pos, i)$ is the sinusoidal positional encoding at position pos and dimension i,
i is the dimension index within the encoding,
d is the total number of dimensions in the encoding.

Then, after obtaining $\text{PosEnc}(t) = [\text{PosEnc}(t, 0), \text{PosEnc}(t, 1), \ldots \text{PosEnc}(t, d-1)]$, we can pass this to the model as the timestep t. The sinusoidal encodings are also visualized in figure 7.

6 Results and comparing models

Using all of the above methods, we trained three different models: one small model using the simple loss (described in 5.1) one big model using the simple loss, and lastly one small model using the simple loss but for $\hat{\boldsymbol{x}}_0$ (i.e. same as in 5.1 but $\mathcal{L} = ||\hat{\boldsymbol{x}}_0(\boldsymbol{x}_t, t) - \boldsymbol{x}_0||_2^2$). Notice: the sampling algorithm for the model using the loss with $\hat{\boldsymbol{x}}_0$ is therefore also slightly different from the algorithm described in 5.1. To make things more readable, we'll call these three models respectively for $\epsilon_{\text{small}}, \epsilon_{\text{big}}$ and $\boldsymbol{x}_{0,\text{small}}$. Obviously, ϵ_{big} should perform better since it uses the better loss function (we haven't shown yet that

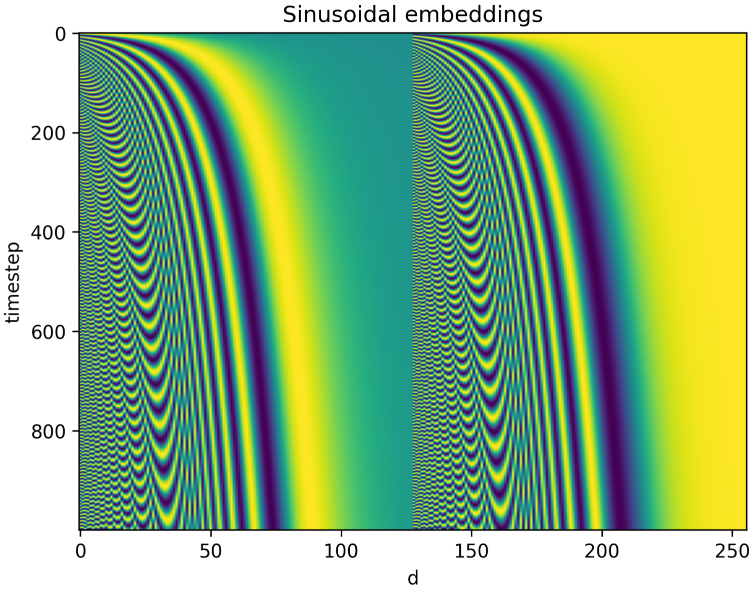

Figure 7: Sinusoidal encodings as described in [VSP+23] here for dimension $d = 256$ and $T = 1000$. The sinusoidal encoding for some timestep t is the horizontal cross-section pertaining to that t.

it is better, but in 5 we mention that empirical studies such as those conducted by [HJA20] shows that the simple loss performs better). We are training this big model to show what is possible with diffusion models. We are also training the two small models to show which loss is the best.

6.1 The dataset and the code

When training these models, we'll be using the CelebA dataset which is a dataset containing facial images of celebrities. We'll be using a subset of 50.000 images to train the small models, and we'll use a subset of 150.000 images to train the big model. All the images have been centered and downscaled to be 32 pixels high, and 32 pixels wide and they have three color channels. 9 example images from the dataset can be seen in figure 8.
All the code can be found at our github.

6.2 How to compare models

There are various ways of comparing how well the models perform. Here, we'll list a few of them:

- **Visual inspection**. The most important feature of generative models is the ability to generate real-looking images. Therefore, it can sometimes be sufficient to simply look at the images generated from different models and assess which model generates the best-looking images. This method is quick and requires almost no code. However, the method is also extremely subjective and it isn't possible to assign a single number to the performance of the model. Also, if the task is to generate new samples from a dataset containing images that are hard for humans to describe, it can be almost impossible to actually assess whether the model has captured important features (for example, it is very easy for us to assess the performance of a model trained on human faces, but what about a model trained on cell images? What does a cell look like? What characteristics are important to capture?).

- **The loss function**. The most obvious way to obtain a single number for the performance of different models is to compare the final test loss. However, often times the different models are trained using different loss functions (for example ϵ_{small} and $\hat{\boldsymbol{x}}_{0,\text{small}}$ uses different losses) and therefore the losses cannot be directly compared.

- **The ELBO**. Our original goal was to do maximum likelihood estimation and maximizing $p(\boldsymbol{x}_0)$ from where we derived the ELBO. Therefore, it seems obvious that we can use the ELBO

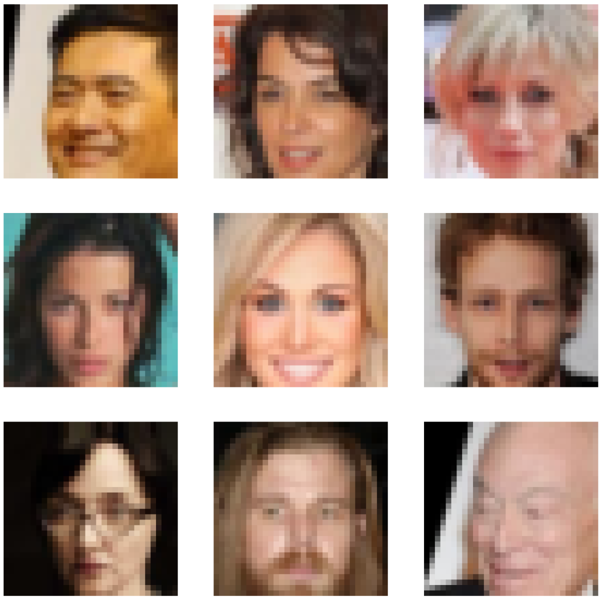

Figure 8: 9 random images from the downscaled CelebA dataset.

to measure model performance. Here, one can use the original definition of the ELBO from equation (2) or, if working on the assumptions described in this paper, simply use the simplified objective from equation (3) (which is technically not the correct ELBO since we have removed the constants, but it can still be used for direct comparisons of models that are made under the same assumptions). The problem here is that the ELBO, which is a lower bound of the likelihood, technically tells nothing about the quality of the images.

- **FID-score**. The Fréchet Inception Distance (FID) is a score metric used for evaluating generative models. It takes into account both the quality and diversity of generated images, offering a more comprehensive evaluation compared to some other metrics. The FID score is calculated by comparing a dataset of generated images to a dataset of real images. The FID score is calculated by obtaining statistics from the two different datasets using the InceptionV3 model [Wik23] and then using these statistics to calculate the distance between the distribution of the two datasets. The lower the FID score, the more similar the two datasets are, and the better the model.

Therefore, in order to fairly compare the results of our three models, we have calculated the FID-scores for the different models and visualized them in table 1, and we have included examples of generated pictures from the different models in figure 9 and 10 for visual comparison. We have also included a figure of the training, validation, and test loss of the simple model to examine whether the model converges nicely. Evidently, it does. We have omitted the same figure for the other models, but they behave similarly. See figure 12.

model	# parameters	FID Score (eq. 3)
eps_small	39902147	74.34
x0_small	39902147	102.52
eps_big	92690435	**54.20**

Table 1: FID scores for image distributions generated by the different models. The FID score is calculated between true images from the CelebA dataset and generated images from the models.

7 Why is the simple loss so much better?

As seen in table 1 and in the visual inspections, the $\boldsymbol{x}_0$-model performs much worse than the ϵ-model. This may seem surprising considering that they arise from the same formulas, such they mathemati-

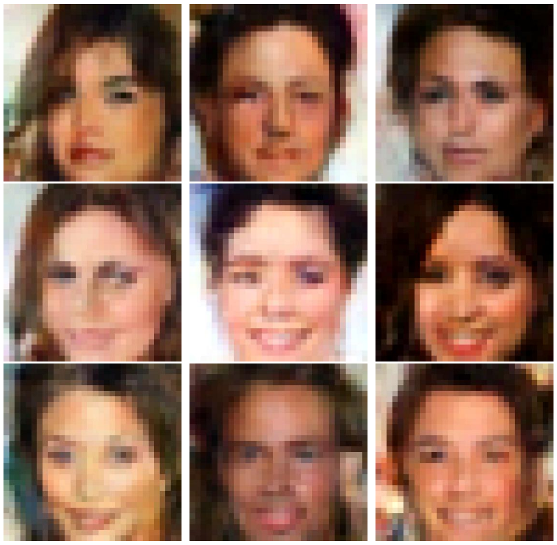

Figure 9: Randomly generated images from the ϵ_{small}-model

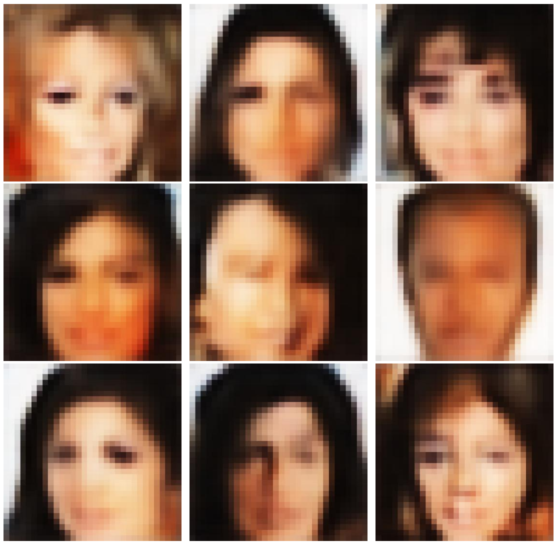

Figure 10: Randomly generated images from the $\boldsymbol{x}_{0\text{small}}$-model

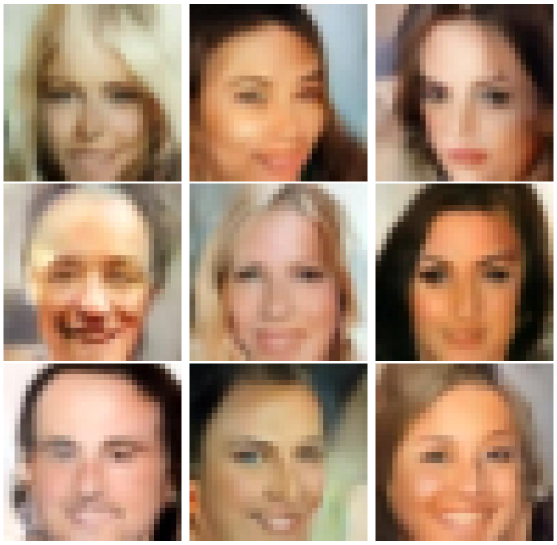

Figure 11: Randomly generated images from the ϵ_{big}-model

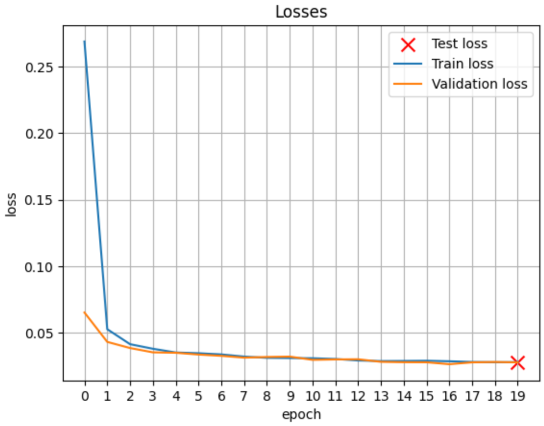

Figure 12: Train, validation, and test loss for each epoch for the $\boldsymbol{x}_{0\text{small}}$-model. The model converges nicely.

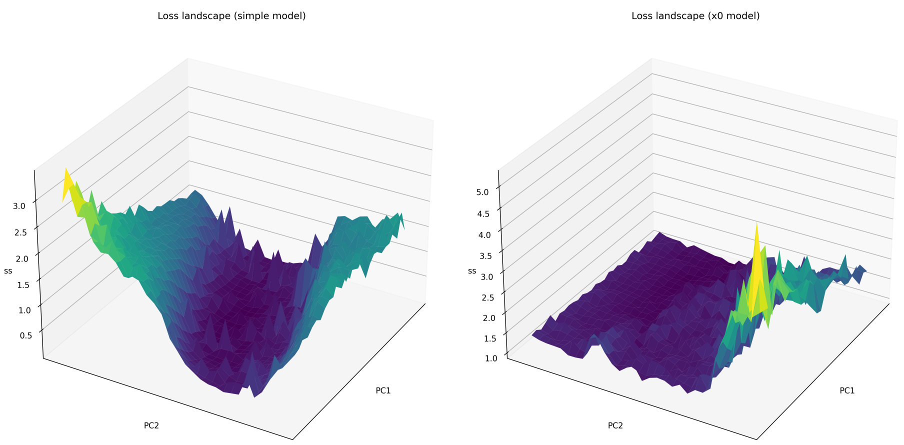

Figure 13: **Left:** loss landscape for the $\epsilon_{0\text{small}}$-model. **Right:** loss landscape for the $\boldsymbol{x}_{0\text{small}}$-model.

cally are identical. Still, for some reason, it seems easier for a neural network to predict the noise than it is to predict the ground truth. There is not a single good reason for why this is. But to get a sense of what happens in the parameter space when the models are searching for local minimum, we have plotted the loss landscapes for the two small models.

7.1 Loss landscape

The loss landscapes have been obtained by saving the model parameters for each epoch. Therefore, we get a matrix A of size $m \times n$ where m is the number of epochs and n is the number of parameters. Then, we perform PCA analysis on A and extract the first two principal components (given we want a loss landscape in 2D). We choose an appropriate region in 2D space, discretely iterate over the points in this 2D space, and project each point back into the parameter space using the inverse PCA transformation. Using these parameters, a test loss can be calculated and noted for this particular point in 2D space. In the end, we obtain a landscape in 2D space that we can visualize. Of course, this method doesn't give the full picture since there will be regions in the parameter space that we can still not see. Also, the method requires that the explained variance of the first two principal components is reasonable. We obtained around 60% explained variance using the first two principal components.
When plotting the loss landscapes we hope to find a nice, relatively smooth minimum, indicating that the model will have an easy time finding that minimum. We have visualized the loss landscapes for the two small models in figure 13. The loss landscapes show that the $\epsilon_{0\text{small}}$-model has a "dip", clearly indicating the minimum, while the $\boldsymbol{x}_{0\text{small}}$-model doesn't have a single, delimited, definite minimum. Actually, it is quite difficult to even see from the figure where the minimum is located. This also explains why the simple model is so much better at finding the optimal parameters, therefore producing the best results.

8 Final words & Future Work

Our goal with this paper was to create a self-contained document that can serve as a learning tool for other curious minds who wish to delve into this contemporary form of generative modelling. To achieve this we have gone into minute detail with the necessary mathematical concepts such as; Bayes' Theorem, Expectation, KL-divergence, Markov chains, etc. The aim with this approach is to allow the reader to go from tabula rasa to getting a fundamentally sound grasp on what diffusion models are and why and how they work. We show that the simple model $\epsilon_{0\text{small}}$-model outperforms the $\boldsymbol{x}_{0\text{small}}$-model, as others have previously experienced, yet it remains unclear why this learning problem is easier for neural networks, since it stems from the same mathematical formulation. We encourage further work on the explainability of these phenomena. Prospectively we also find the idea of a conditional diffusion process inspiring since this would allow for more precise and targeted generation, which could lead to advances in dataset generation for machine learning training. Both in artistic and scientific domains, the ability to guide synthesis by style or class has broad implications.

References

[HJA20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020.

[Luo22] Calvin Luo. Understanding diffusion models: A unified perspective, 2022.

[ND21] Alex Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models, 2021.

[VSP$^+$23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[Wik23] Wikipedia contributors. Inceptionv3 — Wikipedia, the free encyclopedia, 2023. [Online; accessed 2-November-2023].