Understanding Diffusion Models: A Unified Perspective

Calvin Luo
Google Research, Brain Team
calvinluo@google.com

August 26, 2022

Contents

Introduction: Generative Models

Given observed samples $\boldsymbol{x}$ from a distribution of interest, the goal of a **generative model** is to learn to *model* its true data distribution $p(\boldsymbol{x})$. Once learned, we can *generate* new samples from our approximate model at will. Furthermore, under some formulations, we are able to use the learned model to evaluate the likelihood of observed or sampled data as well.

There are several well-known directions in current literature, that we will only introduce briefly at a high level. Generative Adversarial Networks (GANs) model the sampling procedure of a complex distribution, which is learned in an adversarial manner. Another class of generative models, termed "likelihood-based", seeks to learn a model that assigns a high likelihood to the observed data samples. This includes autoregressive models, normalizing flows, and Variational Autoencoders (VAEs). Another similar approach is energy-based modeling, in which a distribution is learned as an arbitrarily flexible energy function that is then normalized.

Score-based generative models are highly related; instead of learning to model the energy function itself, they learn the *score* of the energy-based model as a neural network. In this work we explore and review diffusion models, which as we will demonstrate, have both likelihood-based and score-based interpretations. We showcase the math behind such models in excruciating detail, with the aim that anyone can follow along and understand what diffusion models are and how they work.

Background: ELBO, VAE, and Hierarchical VAE

For many modalities, we can think of the data we observe as represented or generated by an associated unseen *latent* variable, which we can denote by random variable $\boldsymbol{z}$. The best intuition for expressing this idea is through Plato's Allegory of the Cave. In the allegory, a group of people are chained inside a cave their entire life and can only see the two-dimensional shadows projected onto a wall in front of them, which are generated by unseen three-dimensional objects passed before a fire. To such people, everything they observe is actually determined by higher-dimensional abstract concepts that they can never behold.

Analogously, the objects that we encounter in the actual world may also be generated as a function of some higher-level representations; for example, such representations may encapsulate abstract properties such as color, size, shape, and more. Then, what we observe can be interpreted as a three-dimensional projection or instantiation of such abstract concepts, just as what the cave people observe is actually a two-dimensional projection of three-dimensional objects. Whereas the cave people can never see (or even fully comprehend) the hidden objects, they can still reason and draw inferences about them; in a similar way, we can approximate latent representations that describe the data we observe.

Whereas Plato's Allegory illustrates the idea behind latent variables as potentially unobservable representations that determine observations, a caveat of this analogy is that in generative modeling, we generally seek to learn lower-dimensional latent representations rather than higher-dimensional ones. This is because trying to learn a representation of higher dimension than the observation is a fruitless endeavor without strong priors. On the other hand, learning lower-dimensional latents can also be seen as a form of compression, and can potentially uncover semantically meaningful structure describing observations.

Evidence Lower Bound

Mathematically, we can imagine the latent variables and the data we observe as modeled by a joint distribution $p(\boldsymbol{x}, \boldsymbol{z})$. Recall one approach of generative modeling, termed "likelihood-based", is to learn a model to maximize the likelihood $p(\boldsymbol{x})$ of all observed $\boldsymbol{x}$. There are two ways we can manipulate this joint distribution to recover the likelihood of purely our observed data $p(\boldsymbol{x})$; we can explicitly marginalize out the latent variable $\boldsymbol{z}$:

$$p(\boldsymbol{x}) = \int p(\boldsymbol{x}, \boldsymbol{z}) d\boldsymbol{z} \tag{1}$$

or, we could also appeal to the chain rule of probability:

$$p(\boldsymbol{x}) = \frac{p(\boldsymbol{x}, \boldsymbol{z})}{p(\boldsymbol{z}|\boldsymbol{x})} \tag{2}$$

Directly computing and maximizing the likelihood $p(\boldsymbol{x})$ is difficult because it either involves integrating out all latent variables $\boldsymbol{z}$ in Equation 1, which is intractable for complex models, or it involves having access to a ground truth latent encoder $p(\boldsymbol{z}|\boldsymbol{x})$ in Equation 2. However, using these two equations, we can derive a term called the **E**vidence **L**ower **Bo**und (ELBO), which as its name suggests, is a lower bound of the evidence. The evidence is quantified in this case as the log likelihood of the observed data. Then, maximizing the ELBO becomes a proxy objective with which to optimize a latent variable model; in the best case, when the ELBO is powerfully parameterized and perfectly optimized, it becomes exactly equivalent to the evidence. Formally, the equation of the ELBO is:

$$\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{x}, \boldsymbol{z})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\right] \tag{3}$$

To make the relationship with the evidence explicit, we can mathematically write:

$$\log p(\boldsymbol{x}) \geq \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{x},\boldsymbol{z})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\right] \tag{4}$$

Here, $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ is a flexible approximate variational distribution with parameters $\boldsymbol{\phi}$ that we seek to optimize. Intuitively, it can be thought of as a parameterizable model that is learned to estimate the true distribution over latent variables for given observations $\boldsymbol{x}$; in other words, it seeks to approximate true posterior $p(\boldsymbol{z}|\boldsymbol{x})$. As we will see when exploring the Variational Autoencoder, as we increase the lower bound by tuning the parameters $\boldsymbol{\phi}$ to maximize the ELBO, we gain access to components that can be used to model the true data distribution and sample from it, thus learning a generative model. For now, let us try to dive deeper into why the ELBO is an objective we would like to maximize.

Let us begin by deriving the ELBO, using Equation 1:

$$\begin{aligned}
\log p(\boldsymbol{x}) &= \log \int p(\boldsymbol{x},\boldsymbol{z}) d\boldsymbol{z} && \text{(Apply Equation 1)} && (5)\\
&= \log \int \frac{p(\boldsymbol{x},\boldsymbol{z}) q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} d\boldsymbol{z} && \text{(Multiply by } 1 = \frac{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\text{)} && (6)\\
&= \log \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\frac{p(\boldsymbol{x},\boldsymbol{z})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\right] && \text{(Definition of Expectation)} && (7)\\
&\geq \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{x},\boldsymbol{z})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\right] && \text{(Apply Jensen's Inequality)} && (8)
\end{aligned}$$

In this derivation, we directly arrive at our lower bound by applying Jensen's Inequality. However, this does not supply us much useful information about what is actually going on underneath the hood; crucially, this proof gives no intuition on exactly why the ELBO is actually a lower bound of the evidence, as Jensen's Inequality handwaves it away. Furthermore, simply knowing that the ELBO is truly a lower bound of the data does not really tell us why we want to maximize it as an objective. To better understand the relationship between the evidence and the ELBO, let us perform another derivation, this time using Equation 2:

$$\begin{aligned}
\log p(\boldsymbol{x}) &= \log p(\boldsymbol{x}) \int q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) dz && \text{(Multiply by } 1 = \int q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) d\boldsymbol{z}\text{)} && (9)\\
&= \int q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})(\log p(\boldsymbol{x})) dz && \text{(Bring evidence into integral)} && (10)\\
&= \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log p(\boldsymbol{x})\right] && \text{(Definition of Expectation)} && (11)\\
&= \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{x},\boldsymbol{z})}{p(\boldsymbol{z}|\boldsymbol{x})}\right] && \text{(Apply Equation 2)} && (12)\\
&= \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{x},\boldsymbol{z}) q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}{p(\boldsymbol{z}|\boldsymbol{x}) q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\right] && \text{(Multiply by } 1 = \frac{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\text{)} && (13)\\
&= \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{x},\boldsymbol{z})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\right] + \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}{p(\boldsymbol{z}|\boldsymbol{x})}\right] && \text{(Split the Expectation)} && (14)\\
&= \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{x},\boldsymbol{z})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\right] + D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \parallel p(\boldsymbol{z}|\boldsymbol{x})) && \text{(Definition of KL Divergence)} && (15)\\
&\geq \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{x},\boldsymbol{z})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\right] && \text{(KL Divergence always } \geq 0\text{)} && (16)
\end{aligned}$$

From this derivation, we clearly observe from Equation 15 that the evidence is equal to the ELBO plus the KL Divergence between the approximate posterior $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ and the true posterior $p(\boldsymbol{z}|\boldsymbol{x})$. In fact, it was this KL Divergence term that was magically removed by Jensen's Inequality in Equation 8 of the first derivation. Understanding this term is the key to understanding not only the relationship between the ELBO and the evidence, but also the reason why optimizing the ELBO is an appropriate objective at all.

Firstly, we now know why the ELBO is indeed a lower bound: the difference between the evidence and the ELBO is a strictly non-negative KL term, thus the value of the ELBO can never exceed the evidence.

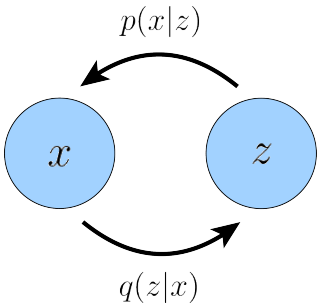

Figure 1: A Variational Autoencoder graphically represented. Here, encoder $q(\boldsymbol{z}|\boldsymbol{x})$ defines a distribution over latent variables $\boldsymbol{z}$ for observations $\boldsymbol{x}$, and $p(\boldsymbol{x}|\boldsymbol{z})$ decodes latent variables into observations.

Secondly, we explore why we seek to maximize the ELBO. Having introduced latent variables $\boldsymbol{z}$ that we would like to model, our goal is to learn this underlying latent structure that describes our observed data. In other words, we want to optimize the parameters of our variational posterior $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ to exactly match the true posterior distribution $p(\boldsymbol{z}|\boldsymbol{x})$, which is achieved by minimizing their KL Divergence (ideally to zero). Unfortunately, it is intractable to minimize this KL Divergence term directly, as we do not have access to the ground truth $p(\boldsymbol{z}|\boldsymbol{x})$ distribution. However, notice that on the left hand side of Equation 15, the likelihood of our data (and therefore our evidence term $\log p(\boldsymbol{x})$) is always a constant with respect to $\boldsymbol{\phi}$, as it is computed by marginalizing out all latents $\boldsymbol{z}$ from the joint distribution $p(\boldsymbol{x}, \boldsymbol{z})$ and does not depend on $\boldsymbol{\phi}$ whatsoever. Since the ELBO and KL Divergence terms sum up to a constant, any maximization of the ELBO term with respect to $\boldsymbol{\phi}$ necessarily invokes an equal minimization of the KL Divergence term. Thus, the ELBO can be maximized as a proxy for learning how to perfectly model the true latent posterior distribution; the more we optimize the ELBO, the closer our approximate posterior gets to the true posterior. Additionally, once trained, the ELBO can be used to estimate the likelihood of observed or generated data as well, since it is learned to approximate the model evidence $\log p(\boldsymbol{x})$.

Variational Autoencoders

In the default formulation of the Variational Autoencoder (VAE) [1], we directly maximize the ELBO. This approach is *variational*, because we optimize for the best $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ amongst a family of potential posterior distributions parameterized by $\boldsymbol{\phi}$. It is called an *autoencoder* because it is reminiscent of a traditional autoencoder model, where input data is trained to predict itself after undergoing an intermediate bottlenecking representation step. To make this connection explicit, let us dissect the ELBO term further:

$$\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{x}, \boldsymbol{z})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\right] = \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\right] \quad \text{(Chain Rule of Probability)} \tag{17}$$

$$= \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})\right] + \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{z})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\right] \quad \text{(Split the Expectation)} \tag{18}$$

$$= \underbrace{\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})\right]}_{\text{reconstruction term}} - \underbrace{D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \parallel p(\boldsymbol{z}))}_{\text{prior matching term}} \quad \text{(Definition of KL Divergence)} \tag{19}$$

In this case, we learn an intermediate bottlenecking distribution $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ that can be treated as an *encoder*; it transforms inputs into a distribution over possible latents. Simultaneously, we learn a deterministic function $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ to convert a given latent vector $\boldsymbol{z}$ into an observation $\boldsymbol{x}$, which can be interpreted as a *decoder*.

The two terms in Equation 19 each have intuitive descriptions: the first term measures the reconstruction likelihood of the decoder from our variational distribution; this ensures that the learned distribution is modeling effective latents that the original data can be regenerated from. The second term measures how similar the learned variational distribution is to a prior belief held over latent variables. Minimizing this term encourages the encoder to actually learn a distribution rather than collapse into a Dirac delta function. Maximizing the ELBO is thus equivalent to maximizing its first term and minimizing its second term.

A defining feature of the VAE is how the ELBO is optimized jointly over parameters $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$. The encoder of the VAE is commonly chosen to model a multivariate Gaussian with diagonal covariance, and the prior is often selected to be a standard multivariate Gaussian:

$$q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{z}; \boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x}), \boldsymbol{\sigma}^2_{\boldsymbol{\phi}}(\boldsymbol{x})\mathbf{I}) \tag{20}$$

$$p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}; \mathbf{0}, \mathbf{I}) \tag{21}$$

Then, the KL divergence term of the ELBO can be computed analytically, and the reconstruction term can be approximated using a Monte Carlo estimate. Our objective can then be rewritten as:

$$\arg\max_{\boldsymbol{\phi},\boldsymbol{\theta}} \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})\right] - D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \parallel p(\boldsymbol{z})) \approx \arg\max_{\boldsymbol{\phi},\boldsymbol{\theta}} \sum_{l=1}^{L} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}^{(l)}) - D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \parallel p(\boldsymbol{z})) \tag{22}$$

where latents $\{\boldsymbol{z}^{(l)}\}_{l=1}^{L}$ are sampled from $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$, for every observation $\boldsymbol{x}$ in the dataset. However, a problem arises in this default setup: each $\boldsymbol{z}^{(l)}$ that our loss is computed on is generated by a stochastic sampling procedure, which is generally non-differentiable. Fortunately, this can be addressed via the *reparameterization trick* when $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ is designed to model certain distributions, including the multivariate Gaussian.

The reparameterization trick rewrites a random variable as a deterministic function of a noise variable; this allows for the optimization of the non-stochastic terms through gradient descent. For example, samples from a normal distribution $x \sim \mathcal{N}(x; \mu, \sigma^2)$ with arbitrary mean μ and variance σ^2 can be rewritten as:

$$x = \mu + \sigma\epsilon \quad \text{with } \epsilon \sim \mathcal{N}(\epsilon; 0, \mathrm{I})$$

In other words, arbitrary Gaussian distributions can be interpreted as standard Gaussians (of which ϵ is a sample) that have their mean shifted from zero to the target mean μ by addition, and their variance stretched by the target variance σ^2. Therefore, by the reparameterization trick, sampling from an arbitrary Gaussian distribution can be performed by sampling from a standard Gaussian, scaling the result by the target standard deviation, and shifting it by the target mean.

In a VAE, each $\boldsymbol{z}$ is thus computed as a deterministic function of input $\boldsymbol{x}$ and auxiliary noise variable $\boldsymbol{\epsilon}$:

$$\boldsymbol{z} = \boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x}) + \boldsymbol{\sigma}_{\boldsymbol{\phi}}(\boldsymbol{x}) \odot \boldsymbol{\epsilon} \quad \text{with } \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\epsilon}; \mathbf{0}, \mathbf{I})$$

where $\odot$ represents an element-wise product. Under this reparameterized version of $\boldsymbol{z}$, gradients can then be computed with respect to $\boldsymbol{\phi}$ as desired, to optimize $\boldsymbol{\mu}_{\boldsymbol{\phi}}$ and $\boldsymbol{\sigma}_{\boldsymbol{\phi}}$. The VAE therefore utilizes the reparameterization trick and Monte Carlo estimates to optimize the ELBO jointly over $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$.

After training a VAE, generating new data can be performed by sampling directly from the latent space $p(\boldsymbol{z})$ and then running it through the decoder. Variational Autoencoders are particularly interesting when the dimensionality of $\boldsymbol{z}$ is less than that of input $\boldsymbol{x}$, as we might then be learning compact, useful representations. Furthermore, when a semantically meaningful latent space is learned, latent vectors can be edited before being passed to the decoder to more precisely control the data generated.

Hierarchical Variational Autoencoders

A Hierarchical Variational Autoencoder (HVAE) [2, 3] is a generalization of a VAE that extends to multiple hierarchies over latent variables. Under this formulation, latent variables themselves are interpreted as generated from other higher-level, more abstract latents. Intuitively, just as we treat our three-dimensional observed objects as generated from a higher-level abstract latent, the people in Plato's cave treat three-dimensional objects as latents that generate their two-dimensional observations. Therefore, from the perspective of Plato's cave dwellers, their observations can be treated as modeled by a latent hierarchy of depth two (or more).

Whereas in the general HVAE with T hierarchical levels, each latent is allowed to condition on all previous latents, in this work we focus on a special case which we call a Markovian HVAE (MHVAE). In a MHVAE, the generative process is a Markov chain; that is, each transition down the hierarchy is Markovian, where

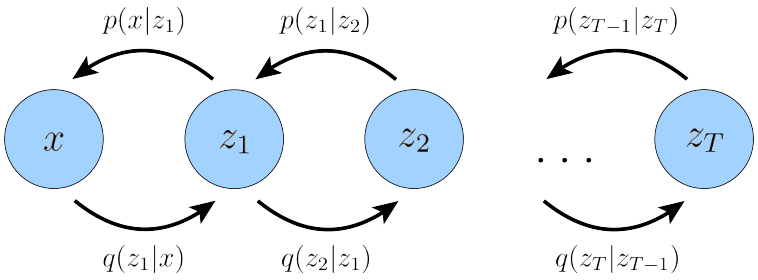

Figure 2: A Markovian Hierarchical Variational Autoencoder with T hierarchical latents. The generative process is modeled as a Markov chain, where each latent $\boldsymbol{z}_t$ is generated only from the previous latent $\boldsymbol{z}_{t+1}$.

decoding each latent $\boldsymbol{z}_t$ only conditions on previous latent $\boldsymbol{z}_{t+1}$. Intuitively, and visually, this can be seen as simply stacking VAEs on top of each other, as depicted in Figure 2; another appropriate term describing this model is a Recursive VAE. Mathematically, we represent the joint distribution and the posterior of a Markovian HVAE as:

$$p(\boldsymbol{x}, \boldsymbol{z}_{1:T}) = p(\boldsymbol{z}_T) p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}_1) \prod_{t=2}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t) \tag{23}$$

$$q_{\boldsymbol{\phi}}(\boldsymbol{z}_{1:T}|\boldsymbol{x}) = q_{\boldsymbol{\phi}}(\boldsymbol{z}_1|\boldsymbol{x}) \prod_{t=2}^{T} q_{\boldsymbol{\phi}}(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}) \tag{24}$$

Then, we can easily extend the ELBO to be:

$$\begin{aligned}
\log p(\boldsymbol{x}) &= \log \int p(\boldsymbol{x}, \boldsymbol{z}_{1:T}) d\boldsymbol{z}_{1:T} && \text{(Apply Equation 1)} && (25)\\
&= \log \int \frac{p(\boldsymbol{x}, \boldsymbol{z}_{1:T}) q_{\boldsymbol{\phi}}(\boldsymbol{z}_{1:T}|\boldsymbol{x})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}_{1:T}|\boldsymbol{x})} d\boldsymbol{z}_{1:T} && \text{(Multiply by } 1 = \frac{q_{\boldsymbol{\phi}}(\boldsymbol{z}_{1:T}|\boldsymbol{x})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}_{1:T}|\boldsymbol{x})}\text{)} && (26)\\
&= \log \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}_{1:T}|\boldsymbol{x})} \left[\frac{p(\boldsymbol{x}, \boldsymbol{z}_{1:T})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}_{1:T}|\boldsymbol{x})} \right] && \text{(Definition of Expectation)} && (27)\\
&\geq \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}_{1:T}|\boldsymbol{x})} \left[\log \frac{p(\boldsymbol{x}, \boldsymbol{z}_{1:T})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}_{1:T}|\boldsymbol{x})} \right] && \text{(Apply Jensen's Inequality)} && (28)
\end{aligned}$$

We can then plug our joint distribution (Equation 23) and posterior (Equation 24) into Equation 28 to produce an alternate form:

$$\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}_{1:T}|\boldsymbol{x})} \left[\log \frac{p(\boldsymbol{x}, \boldsymbol{z}_{1:T})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}_{1:T}|\boldsymbol{x})} \right] = \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}_{1:T}|\boldsymbol{x})} \left[\log \frac{p(\boldsymbol{z}_T) p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}_1) \prod_{t=2}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t)}{q_{\boldsymbol{\phi}}(\boldsymbol{z}_1|\boldsymbol{x}) \prod_{t=2}^{T} q_{\boldsymbol{\phi}}(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})} \right] \tag{29}$$

As we will show below, when we investigate Variational Diffusion Models, this objective can be further decomposed into interpretable components.

Variational Diffusion Models

The easiest way to think of a Variational Diffusion Model (VDM) [4, 5, 6] is simply as a Markovian Hierarchical Variational Autoencoder with three key restrictions:

- The latent dimension is exactly equal to the data dimension
- The structure of the latent encoder at each timestep is not learned; it is pre-defined as a linear Gaussian model. In other words, it is a Gaussian distribution centered around the output of the previous timestep
- The Gaussian parameters of the latent encoders vary over time in such a way that the distribution of the latent at final timestep T is a standard Gaussian

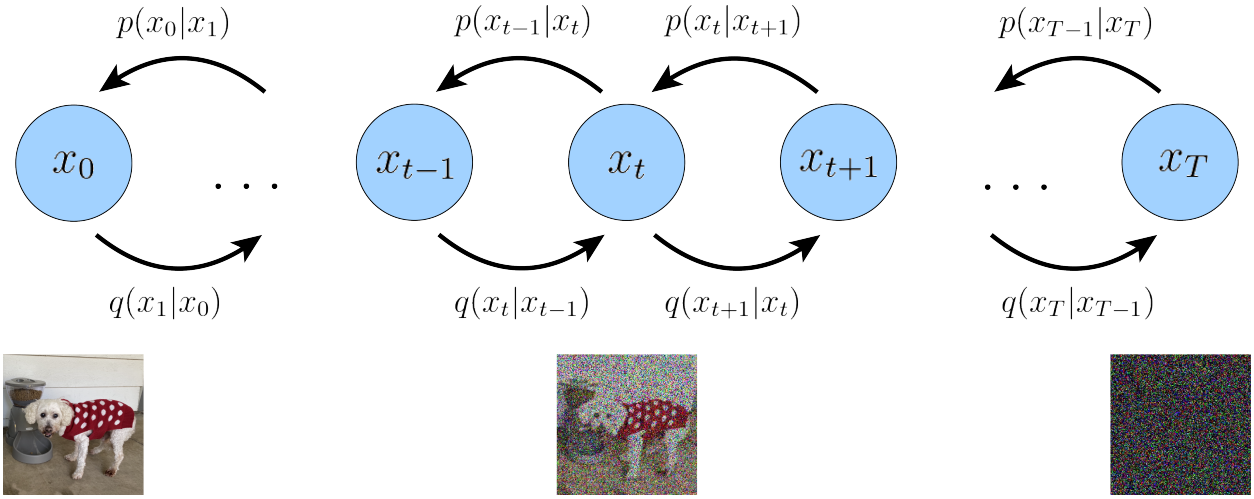

Figure 3: A visual representation of a Variational Diffusion Model; $\boldsymbol{x}_0$ represents true data observations such as natural images, $\boldsymbol{x}_T$ represents pure Gaussian noise, and $\boldsymbol{x}_t$ is an intermediate noisy version of $\boldsymbol{x}_0$. Each $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ is modeled as a Gaussian distribution that uses the output of the previous state as its mean.

Furthermore, we explicitly maintain the Markov property between hierarchical transitions from a standard Markovian Hierarchical Variational Autoencoder.

Let us expand on the implications of these assumptions. From the first restriction, with some abuse of notation, we can now represent both true data samples and latent variables as $\boldsymbol{x}_t$, where $t = 0$ represents true data samples and $t \in [1, T]$ represents a corresponding latent with hierarchy indexed by t. The VDM posterior is the same as the MHVAE posterior (Equation 24), but can now be rewritten as:

$$q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0) = \prod_{t=1}^{T} q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) \tag{30}$$

From the second assumption, we know that the distribution of each latent variable in the encoder is a Gaussian centered around its previous hierarchical latent. Unlike a Markovian HVAE, the structure of the encoder at each timestep t is not learned; it is fixed as a linear Gaussian model, where the mean and standard deviation can be set beforehand as hyperparameters [5], or learned as parameters [6]. We parameterize the Gaussian encoder with mean $\boldsymbol{\mu}_t(\boldsymbol{x}_t) = \sqrt{\alpha_t}\boldsymbol{x}_{t-1}$, and variance $\boldsymbol{\Sigma}_t(\boldsymbol{x}_t) = (1 - \alpha_t)\mathbf{I}$, where the form of the coefficients are chosen such that the variance of the latent variables stay at a similar scale; in other words, the encoding process is *variance-preserving*. Note that alternate Gaussian parameterizations are allowed, and lead to similar derivations. The main takeaway is that α_t is a (potentially learnable) coefficient that can vary with the hierarchical depth t, for flexibility. Mathematically, encoder transitions are denoted as:

$$q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{\alpha_t}\boldsymbol{x}_{t-1}, (1 - \alpha_t)\mathbf{I}) \tag{31}$$

From the third assumption, we know that α_t evolves over time according to a fixed or learnable schedule structured such that the distribution of the final latent $p(\boldsymbol{x}_T)$ is a standard Gaussian. We can then update the joint distribution of a Markovian HVAE (Equation 23) to write the joint distribution for a VDM as:

$$p(\boldsymbol{x}_{0:T}) = p(\boldsymbol{x}_T) \prod_{t=1}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) \tag{32}$$

where,

$$p(\boldsymbol{x}_T) = \mathcal{N}(\boldsymbol{x}_T; \mathbf{0}, \mathbf{I}) \tag{33}$$

Collectively, what this set of assumptions describes is a steady noisification of an image input over time; we progressively corrupt an image by adding Gaussian noise until eventually it becomes completely identical to pure Gaussian noise. Visually, this process is depicted in Figure 3.

Note that our encoder distributions $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ are no longer parameterized by $\boldsymbol{\phi}$, as they are completely modeled as Gaussians with defined mean and variance parameters at each timestep. Therefore, in a VDM, we are only interested in learning conditionals $p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$, so that we can simulate new data. After optimizing the VDM, the sampling procedure is as simple as sampling Gaussian noise from $p(\boldsymbol{x}_T)$ and iteratively running the denoising transitions $p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ for T steps to generate a novel $\boldsymbol{x}_0$.

Like any HVAE, the VDM can be optimized by maximizing the ELBO, which can be derived as:

$$\log p(\boldsymbol{x}) = \log \int p(\boldsymbol{x}_{0:T}) d\boldsymbol{x}_{1:T} \tag{34}$$

$$= \log \int \frac{p(\boldsymbol{x}_{0:T}) q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)} d\boldsymbol{x}_{1:T} \tag{35}$$

$$= \log \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)} \left[\frac{p(\boldsymbol{x}_{0:T})}{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)} \right] \tag{36}$$

$$\geq \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)} \left[\log \frac{p(\boldsymbol{x}_{0:T})}{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)} \right] \tag{37}$$

$$= \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)} \left[\log \frac{p(\boldsymbol{x}_T) \prod_{t=1}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{\prod_{t=1}^{T} q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})} \right] \tag{38}$$

$$= \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)} \left[\log \frac{p(\boldsymbol{x}_T) p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1) \prod_{t=2}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\boldsymbol{x}_T|\boldsymbol{x}_{T-1}) \prod_{t=1}^{T-1} q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})} \right] \tag{39}$$

$$= \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)} \left[\log \frac{p(\boldsymbol{x}_T) p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1) \prod_{t=1}^{T-1} p_{\boldsymbol{\theta}}(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})}{q(\boldsymbol{x}_T|\boldsymbol{x}_{T-1}) \prod_{t=1}^{T-1} q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})} \right] \tag{40}$$

$$= \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)} \left[\log \frac{p(\boldsymbol{x}_T) p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1)}{q(\boldsymbol{x}_T|\boldsymbol{x}_{T-1})} \right] + \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)} \left[\log \prod_{t=1}^{T-1} \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})}{q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})} \right] \tag{41}$$

$$= \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)} \left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1) \right] + \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)} \left[\log \frac{p(\boldsymbol{x}_T)}{q(\boldsymbol{x}_T|\boldsymbol{x}_{T-1})} \right] + \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)} \left[\sum_{t=1}^{T-1} \log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})}{q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})} \right] \tag{42}$$

$$= \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)} \left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1) \right] + \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)} \left[\log \frac{p(\boldsymbol{x}_T)}{q(\boldsymbol{x}_T|\boldsymbol{x}_{T-1})} \right] + \sum_{t=1}^{T-1} \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)} \left[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})}{q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})} \right] \tag{43}$$

$$= \mathbb{E}_{q(\boldsymbol{x}_1|\boldsymbol{x}_0)} \left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1) \right] + \mathbb{E}_{q(\boldsymbol{x}_{T-1},\boldsymbol{x}_T|\boldsymbol{x}_0)} \left[\log \frac{p(\boldsymbol{x}_T)}{q(\boldsymbol{x}_T|\boldsymbol{x}_{T-1})} \right] + \sum_{t=1}^{T-1} \mathbb{E}_{q(\boldsymbol{x}_{t-1},\boldsymbol{x}_t,\boldsymbol{x}_{t+1}|\boldsymbol{x}_0)} \left[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})}{q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})} \right] \tag{44}$$

$$= \underbrace{\mathbb{E}_{q(\boldsymbol{x}_1|\boldsymbol{x}_0)} \left[\log p_\theta(\boldsymbol{x}_0|\boldsymbol{x}_1) \right]}_{\text{reconstruction term}} - \underbrace{\mathbb{E}_{q(\boldsymbol{x}_{T-1}|\boldsymbol{x}_0)} \left[D_{\mathrm{KL}}(q(\boldsymbol{x}_T|\boldsymbol{x}_{T-1}) \parallel p(\boldsymbol{x}_T)) \right]}_{\text{prior matching term}} \tag{45}$$

$$- \sum_{t=1}^{T-1} \underbrace{\mathbb{E}_{q(\boldsymbol{x}_{t-1},\boldsymbol{x}_{t+1}|\boldsymbol{x}_0)} \left[D_{\mathrm{KL}}(q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) \parallel p_\theta(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})) \right]}_{\text{consistency term}}$$

The derived form of the ELBO can be interpreted in terms of its individual components:

1. $\mathbb{E}_{q(\boldsymbol{x}_1|\boldsymbol{x}_0)} \left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1) \right]$ can be interpreted as a *reconstruction term*, predicting the log probability of the original data sample given the first-step latent. This term also appears in a vanilla VAE, and can be trained similarly.

2. $\mathbb{E}_{q(\boldsymbol{x}_{T-1}|\boldsymbol{x}_0)} \left[D_{\mathrm{KL}}(q(\boldsymbol{x}_T|\boldsymbol{x}_{T-1}) \parallel p(\boldsymbol{x}_T)) \right]$ is a *prior matching term*; it is minimized when the final latent distribution matches the Gaussian prior. This term requires no optimization, as it has no trainable parameters; furthermore, as we have assumed a large enough T such that the final distribution is Gaussian, this term effectively becomes zero.

3. $\mathbb{E}_{q(\boldsymbol{x}_{t-1},\boldsymbol{x}_{t+1}|\boldsymbol{x}_0)} \left[D_{\mathrm{KL}}(q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})) \right]$ is a *consistency term*; it endeavors to make the distribution at $\boldsymbol{x}_t$ consistent, from both forward and backward processes. That is, a denoising step from a noisier image should match the corresponding noising step from a cleaner image, for every intermediate timestep; this is reflected mathematically by the KL Divergence. This term is minimized when we train $p_\theta(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})$ to match the Gaussian distribution $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$, which is defined in Equation 31.

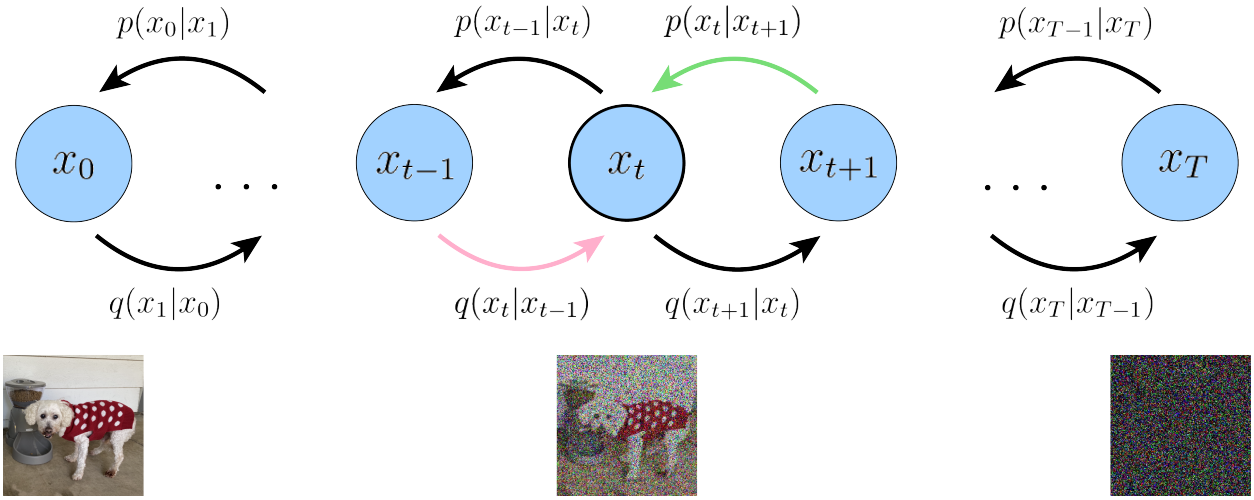

Figure 4: Under our first derivation, a VDM can be optimized by ensuring that for every intermediate $\boldsymbol{x}_t$, the posterior from the latent above it $p_{\boldsymbol{\theta}}(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})$ matches the Gaussian corruption of the latent before it $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$. In this figure, for each intermediate $\boldsymbol{x}_t$, we minimize the difference between the distributions represented by the pink and green arrows.

Visually, this interpretation of the ELBO is depicted in Figure 4. The cost of optimizing a VDM is primarily dominated by the third term, since we must optimize over all timesteps t.

Under this derivation, all terms of the ELBO are computed as expectations, and can therefore be approximated using Monte Carlo estimates. However, actually optimizing the ELBO using the terms we just derived might be suboptimal; because the consistency term is computed as an expectation over two random variables $\{\boldsymbol{x}_{t-1}, \boldsymbol{x}_{t+1}\}$ for every timestep, the variance of its Monte Carlo estimate could potentially be higher than a term that is estimated using only one random variable per timestep. As it is computed by summing up $T-1$ consistency terms, the final estimated value of the ELBO may have high variance for large T values.

Let us instead try to derive a form for our ELBO where each term is computed as an expectation over only one random variable at a time. The key insight is that we can rewrite encoder transitions as $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{x}_0)$, where the extra conditioning term is superfluous due to the Markov property. Then, according to Bayes rule, we can rewrite each transition as:

$$q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{x}_0) = \frac{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)q(\boldsymbol{x}_t|\boldsymbol{x}_0)}{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)} \tag{46}$$

Armed with this new equation, we can retry the derivation resuming from the ELBO in Equation 37:

$$\log p(\boldsymbol{x}) \geq \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log \frac{p(\boldsymbol{x}_{0:T})}{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\right] \tag{47}$$

$$= \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log \frac{p(\boldsymbol{x}_T)\prod_{t=1}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{\prod_{t=1}^{T} q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})}\right] \tag{48}$$

$$= \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log \frac{p(\boldsymbol{x}_T)p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1)\prod_{t=2}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\boldsymbol{x}_1|\boldsymbol{x}_0)\prod_{t=2}^{T} q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})}\right] \tag{49}$$

$$= \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log \frac{p(\boldsymbol{x}_T)p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1)\prod_{t=2}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\boldsymbol{x}_1|\boldsymbol{x}_0)\prod_{t=2}^{T} q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{x}_0)}\right] \tag{50}$$

$$= \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_T)p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1)}{q(\boldsymbol{x}_1|\boldsymbol{x}_0)} + \log \prod_{t=2}^{T} \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{x}_0)}\right] \tag{51}$$

$$= \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log \frac{p(\boldsymbol{x}_T)p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1)}{q(\boldsymbol{x}_1|\boldsymbol{x}_0)} + \log \prod_{t=2}^{T} \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{\frac{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)q(\boldsymbol{x}_t|\boldsymbol{x}_0)}{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)}}\right] \tag{52}$$

$$= \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log \frac{p(\boldsymbol{x}_T)p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1)}{q(\boldsymbol{x}_1|\boldsymbol{x}_0)} + \log \prod_{t=2}^{T} \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{\frac{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)q(\boldsymbol{x}_t|\boldsymbol{x}_0)}{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)}}\right] \tag{53}$$

$$= \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log \frac{p(\boldsymbol{x}_T)p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1)}{\cancel{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}} + \log \frac{\cancel{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}}{q(\boldsymbol{x}_T|\boldsymbol{x}_0)} + \log \prod_{t=2}^{T} \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)}\right] \tag{54}$$

$$= \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log \frac{p(\boldsymbol{x}_T)p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1)}{q(\boldsymbol{x}_T|\boldsymbol{x}_0)} + \sum_{t=2}^{T} \log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)}\right] \tag{55}$$

$$= \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1)\right] + \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log \frac{p(\boldsymbol{x}_T)}{q(\boldsymbol{x}_T|\boldsymbol{x}_0)}\right] + \sum_{t=2}^{T} \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)}\right] \tag{56}$$

$$= \mathbb{E}_{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1)\right] + \mathbb{E}_{q(\boldsymbol{x}_T|\boldsymbol{x}_0)}\left[\log \frac{p(\boldsymbol{x}_T)}{q(\boldsymbol{x}_T|\boldsymbol{x}_0)}\right] + \sum_{t=2}^{T} \mathbb{E}_{q(\boldsymbol{x}_t,\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)}\left[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)}\right] \tag{57}$$

$$= \underbrace{\mathbb{E}_{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1)\right]}_{\text{reconstruction term}} - \underbrace{D_{\mathrm{KL}}(q(\boldsymbol{x}_T|\boldsymbol{x}_0) \parallel p(\boldsymbol{x}_T))}_{\text{prior matching term}} - \sum_{t=2}^{T} \underbrace{\mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))\right]}_{\text{denoising matching term}} \tag{58}$$

We have therefore successfully derived an interpretation for the ELBO that can be estimated with lower variance, as each term is computed as an expectation of at most one random variable at a time. This formulation also has an elegant interpretation, which is revealed when inspecting each individual term:

1. $\mathbb{E}_{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1)\right]$ can be interpreted as a reconstruction term; like its analogue in the ELBO of a vanilla VAE, this term can be approximated and optimized using a Monte Carlo estimate.
2. $D_{\mathrm{KL}}(q(\boldsymbol{x}_T|\boldsymbol{x}_0) \parallel p(\boldsymbol{x}_T))$ represents how close the distribution of the final noisified input is to the standard Gaussian prior. It has no trainable parameters, and is also equal to zero under our assumptions.
3. $\mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))\right]$ is a *denoising matching term*. We learn desired denoising transition step $p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ as an approximation to tractable, ground-truth denoising transition step $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)$. The $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)$ transition step can act as a ground-truth signal, since it defines how to denoise a noisy image $\boldsymbol{x}_t$ with access to what the final, completely denoised image $\boldsymbol{x}_0$ should be. This term is therefore minimized when the two denoising steps match as closely as possible, as measured by their KL Divergence.

As a side note, one observes that in the process of both ELBO derivations (Equation 45 and Equation 58), only the Markov assumption is used; as a result these formulae will hold true for any arbitrary Markovian HVAE. Furthermore, when we set $T = 1$, both of the ELBO interpretations for a VDM exactly recreate the ELBO equation of a vanilla VAE, as written in Equation 19.

In this derivation of the ELBO, the bulk of the optimization cost once again lies in the summation term, which dominates the reconstruction term. Whereas each KL Divergence term $D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))$ is difficult to minimize for arbitrary posteriors in arbitrarily complex Markovian HVAEs due to the added complexity of simultaneously learning the encoder, in a VDM we can leverage the Gaussian transition assumption to make optimization tractable. By Bayes rule, we have:

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

As we already know that $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1},\boldsymbol{x}_0) = q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{\alpha_t}\boldsymbol{x}_{t-1}, (1-\alpha_t)\mathbf{I})$ from our assumption regarding encoder transitions (Equation 31), what remains is deriving for the forms of $q(\boldsymbol{x}_t|\boldsymbol{x}_0)$ and $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)$. Fortunately, these are also made tractable by utilizing the fact that the encoder transitions of a VDM are linear Gaussian models. Recall that under the reparameterization trick, samples $\boldsymbol{x}_t \sim q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ can be rewritten as:

$$\boldsymbol{x}_t = \sqrt{\alpha_t}\boldsymbol{x}_{t-1} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon} \quad \text{with } \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\epsilon}; \mathbf{0}, \mathbf{I}) \tag{59}$$

and that similarly, samples $\boldsymbol{x}_{t-1} \sim q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_{t-2})$ can be rewritten as:

$$\boldsymbol{x}_{t-1} = \sqrt{\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{1-\alpha_{t-1}}\boldsymbol{\epsilon} \quad \text{with } \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\epsilon}; \mathbf{0}, \mathbf{I}) \tag{60}$$

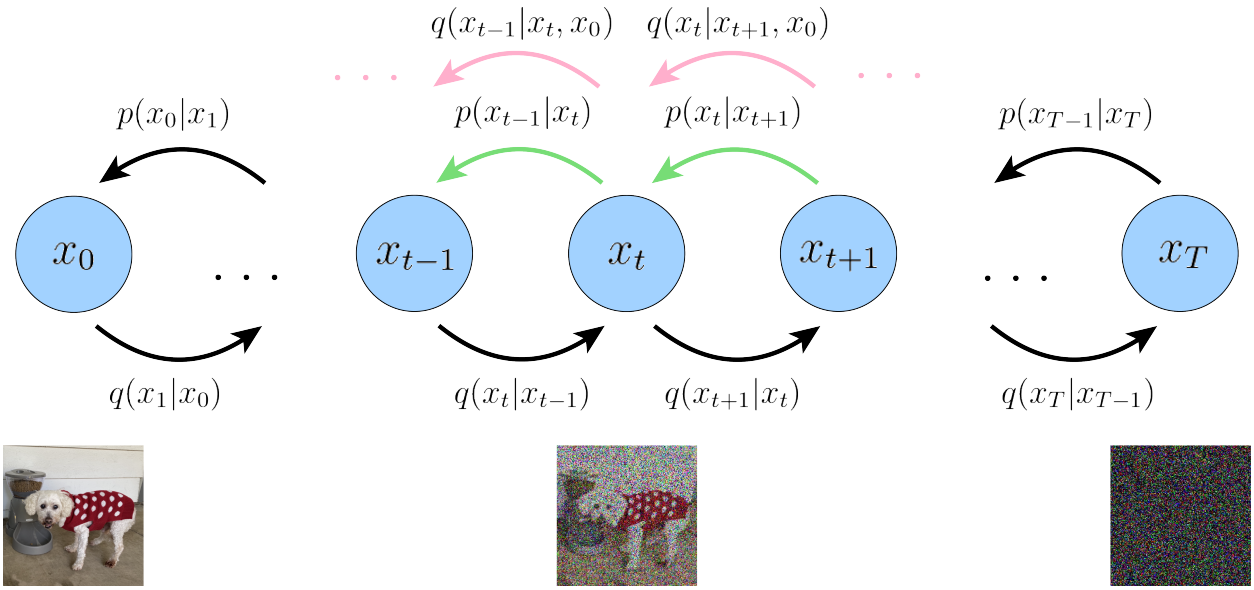

Figure 5: Depicted is an alternate, lower-variance method to optimize a VDM; we compute the form of ground-truth denoising step $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$ using Bayes rule, and minimize its KL Divergence with our approximate denoising step $p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$. This is once again denoted visually by matching the distributions represented by the green arrows with those of the pink arrows. Artistic liberty is at play here; in the full picture, each pink arrow must also stem from $\boldsymbol{x}_0$, as it is also a conditioning term.

Then, the form of $q(\boldsymbol{x}_t|\boldsymbol{x}_0)$ can be recursively derived through repeated applications of the reparameterization trick. Suppose that we have access to $2T$ random noise variables $\{\boldsymbol{\epsilon}_t^*, \boldsymbol{\epsilon}_t\}_{t=0}^T \overset{\text{iid}}{\sim} \mathcal{N}(\boldsymbol{\epsilon}; \mathbf{0}, \mathbf{I})$. Then, for an arbitrary sample $\boldsymbol{x}_t \sim q(\boldsymbol{x}_t|\boldsymbol{x}_0)$, we can rewrite it as:

$$\begin{aligned}
\boldsymbol{x}_t &= \sqrt{\alpha_t}\boldsymbol{x}_{t-1} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_{t-1}^* && (61)\\
&= \sqrt{\alpha_t}\left(\sqrt{\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{1-\alpha_{t-1}}\boldsymbol{\epsilon}_{t-2}^*\right) + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_{t-1}^* && (62)\\
&= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{\alpha_t - \alpha_t\alpha_{t-1}}\boldsymbol{\epsilon}_{t-2}^* + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_{t-1}^* && (63)\\
&= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{\sqrt{\alpha_t - \alpha_t\alpha_{t-1}}^2 + \sqrt{1-\alpha_t}^2}\boldsymbol{\epsilon}_{t-2} && (64)\\
&= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{\alpha_t - \alpha_t\alpha_{t-1} + 1 - \alpha_t}\boldsymbol{\epsilon}_{t-2} && (65)\\
&= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\boldsymbol{\epsilon}_{t-2} && (66)\\
&= \dots && (67)\\
&= \sqrt{\prod_{i=1}^{t}\alpha_i}\boldsymbol{x}_0 + \sqrt{1-\prod_{i=1}^{t}\alpha_i}\boldsymbol{\epsilon}_0 && (68)\\
&= \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_0 && (69)\\
&\sim \mathcal{N}(\boldsymbol{x}_t; \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0, (1-\bar{\alpha}_t)\,\mathbf{I}) && (70)
\end{aligned}$$

where in Equation 64 we have utilized the fact that the sum of two independent Gaussian random variables remains a Gaussian with mean being the sum of the two means, and variance being the sum of the two variances. Interpreting $\sqrt{1-\alpha_t}\boldsymbol{\epsilon}_{t-1}^*$ as a sample from Gaussian $\mathcal{N}(\mathbf{0}, (1-\alpha_t)\mathbf{I})$, and $\sqrt{\alpha_t - \alpha_t\alpha_{t-1}}\boldsymbol{\epsilon}_{t-2}^*$ as a sample from Gaussian $\mathcal{N}(\mathbf{0}, (\alpha_t - \alpha_t\alpha_{t-1})\mathbf{I})$, we can then treat their sum as a random variable sampled from Gaussian $\mathcal{N}(\mathbf{0}, (1-\alpha_t+\alpha_t-\alpha_t\alpha_{t-1})\mathbf{I}) = \mathcal{N}(\mathbf{0}, (1-\alpha_t\alpha_{t-1})\mathbf{I})$. A sample from this distribution can then be represented using the reparameterization trick as $\sqrt{1-\alpha_t\alpha_{t-1}}\boldsymbol{\epsilon}_{t-2}$, as in Equation 66.

We have therefore derived the Gaussian form of $q(\boldsymbol{x}_t|\boldsymbol{x}_0)$. This derivation can be modified to also yield the Gaussian parameterization describing $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)$. Now, knowing the forms of both $q(\boldsymbol{x}_t|\boldsymbol{x}_0)$ and $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)$, we can proceed to calculate the form of $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$ by substituting into the Bayes rule expansion:

$$q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) = \frac{q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{x}_0)q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)}{q(\boldsymbol{x}_t|\boldsymbol{x}_0)} \tag{71}$$

$$= \frac{\mathcal{N}(\boldsymbol{x}_t; \sqrt{\alpha_t}\boldsymbol{x}_{t-1}, (1-\alpha_t)\mathbf{I})\mathcal{N}(\boldsymbol{x}_{t-1}; \sqrt{\bar{\alpha}_{t-1}}\boldsymbol{x}_0, (1-\bar{\alpha}_{t-1})\mathbf{I})}{\mathcal{N}(\boldsymbol{x}_t; \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0, (1-\bar{\alpha}_t)\mathbf{I})} \tag{72}$$

$$\propto \exp\left\{-\left[\frac{(\boldsymbol{x}_t - \sqrt{\alpha_t}\boldsymbol{x}_{t-1})^2}{2(1-\alpha_t)} + \frac{(\boldsymbol{x}_{t-1} - \sqrt{\bar{\alpha}_{t-1}}\boldsymbol{x}_0)^2}{2(1-\bar{\alpha}_{t-1})} - \frac{(\boldsymbol{x}_t - \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0)^2}{2(1-\bar{\alpha}_t)}\right]\right\} \tag{73}$$

$$= \exp\left\{-\frac{1}{2}\left[\frac{(\boldsymbol{x}_t - \sqrt{\alpha_t}\boldsymbol{x}_{t-1})^2}{1-\alpha_t} + \frac{(\boldsymbol{x}_{t-1} - \sqrt{\bar{\alpha}_{t-1}}\boldsymbol{x}_0)^2}{1-\bar{\alpha}_{t-1}} - \frac{(\boldsymbol{x}_t - \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0)^2}{1-\bar{\alpha}_t}\right]\right\} \tag{74}$$

$$= \exp\left\{-\frac{1}{2}\left[\frac{(-2\sqrt{\alpha_t}\boldsymbol{x}_t\boldsymbol{x}_{t-1} + \alpha_t\boldsymbol{x}_{t-1}^2)}{1-\alpha_t} + \frac{(\boldsymbol{x}_{t-1}^2 - 2\sqrt{\bar{\alpha}_{t-1}}\boldsymbol{x}_{t-1}\boldsymbol{x}_0)}{1-\bar{\alpha}_{t-1}} + C(\boldsymbol{x}_t, \boldsymbol{x}_0)\right]\right\} \tag{75}$$

$$\propto \exp\left\{-\frac{1}{2}\left[-\frac{2\sqrt{\alpha_t}\boldsymbol{x}_t\boldsymbol{x}_{t-1}}{1-\alpha_t} + \frac{\alpha_t\boldsymbol{x}_{t-1}^2}{1-\alpha_t} + \frac{\boldsymbol{x}_{t-1}^2}{1-\bar{\alpha}_{t-1}} - \frac{2\sqrt{\bar{\alpha}_{t-1}}\boldsymbol{x}_{t-1}\boldsymbol{x}_0}{1-\bar{\alpha}_{t-1}}\right]\right\} \tag{76}$$

$$= \exp\left\{-\frac{1}{2}\left[(\frac{\alpha_t}{1-\alpha_t} + \frac{1}{1-\bar{\alpha}_{t-1}})\boldsymbol{x}_{t-1}^2 - 2\left(\frac{\sqrt{\alpha_t}\boldsymbol{x}_t}{1-\alpha_t} + \frac{\sqrt{\bar{\alpha}_{t-1}}\boldsymbol{x}_0}{1-\bar{\alpha}_{t-1}}\right)\boldsymbol{x}_{t-1}\right]\right\} \tag{77}$$

$$= \exp\left\{-\frac{1}{2}\left[\frac{\alpha_t(1-\bar{\alpha}_{t-1}) + 1 - \alpha_t}{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}\boldsymbol{x}_{t-1}^2 - 2\left(\frac{\sqrt{\alpha_t}\boldsymbol{x}_t}{1-\alpha_t} + \frac{\sqrt{\bar{\alpha}_{t-1}}\boldsymbol{x}_0}{1-\bar{\alpha}_{t-1}}\right)\boldsymbol{x}_{t-1}\right]\right\} \tag{78}$$

$$= \exp\left\{-\frac{1}{2}\left[\frac{\alpha_t - \bar{\alpha}_t + 1 - \alpha_t}{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}\boldsymbol{x}_{t-1}^2 - 2\left(\frac{\sqrt{\alpha_t}\boldsymbol{x}_t}{1-\alpha_t} + \frac{\sqrt{\bar{\alpha}_{t-1}}\boldsymbol{x}_0}{1-\bar{\alpha}_{t-1}}\right)\boldsymbol{x}_{t-1}\right]\right\} \tag{79}$$

$$= \exp\left\{-\frac{1}{2}\left[\frac{1-\bar{\alpha}_t}{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}\boldsymbol{x}_{t-1}^2 - 2\left(\frac{\sqrt{\alpha_t}\boldsymbol{x}_t}{1-\alpha_t} + \frac{\sqrt{\bar{\alpha}_{t-1}}\boldsymbol{x}_0}{1-\bar{\alpha}_{t-1}}\right)\boldsymbol{x}_{t-1}\right]\right\} \tag{80}$$

$$= \exp\left\{-\frac{1}{2}\left(\frac{1-\bar{\alpha}_t}{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}\right)\left[\boldsymbol{x}_{t-1}^2 - 2\frac{\left(\frac{\sqrt{\alpha_t}\boldsymbol{x}_t}{1-\alpha_t} + \frac{\sqrt{\bar{\alpha}_{t-1}}\boldsymbol{x}_0}{1-\bar{\alpha}_{t-1}}\right)}{\frac{1-\bar{\alpha}_t}{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}}\boldsymbol{x}_{t-1}\right]\right\} \tag{81}$$

$$= \exp\left\{-\frac{1}{2}\left(\frac{1-\bar{\alpha}_t}{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}\right)\left[\boldsymbol{x}_{t-1}^2 - 2\frac{\left(\frac{\sqrt{\alpha_t}\boldsymbol{x}_t}{1-\alpha_t} + \frac{\sqrt{\bar{\alpha}_{t-1}}\boldsymbol{x}_0}{1-\bar{\alpha}_{t-1}}\right)(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\boldsymbol{x}_{t-1}\right]\right\} \tag{82}$$

$$= \exp\left\{-\frac{1}{2}\left(\frac{1}{\frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}}\right)\left[\boldsymbol{x}_{t-1}^2 - 2\frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\boldsymbol{x}_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\boldsymbol{x}_0}{1-\bar{\alpha}_t}\boldsymbol{x}_{t-1}\right]\right\} \tag{83}$$

$$\propto \mathcal{N}(\boldsymbol{x}_{t-1}; \underbrace{\frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\boldsymbol{x}_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\boldsymbol{x}_0}{1-\bar{\alpha}_t}}_{\mu_q(\boldsymbol{x}_t, \boldsymbol{x}_0)}, \underbrace{\frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{I}}_{\boldsymbol{\Sigma}_q(t)}) \tag{84}$$

where in Equation 75, $C(\boldsymbol{x}_t, \boldsymbol{x}_0)$ is a constant term with respect to $\boldsymbol{x}_{t-1}$ computed as a combination of only $\boldsymbol{x}_t$, $\boldsymbol{x}_0$, and α values; this term is implicitly returned in Equation 84 to complete the square.

We have therefore shown that at each step, $\boldsymbol{x}_{t-1} \sim q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$ is normally distributed, with mean $\boldsymbol{\mu}_q(\boldsymbol{x}_t, \boldsymbol{x}_0)$ that is a function of $\boldsymbol{x}_t$ and $\boldsymbol{x}_0$, and variance $\boldsymbol{\Sigma}_q(t)$ as a function of α coefficients. These α coefficients are known and fixed at each timestep; they are either set permanently when modeled as hyperparameters, or treated as the current inference output of a network that seeks to model them. Following Equation 84, we can rewrite our variance equation as $\boldsymbol{\Sigma}_q(t) = \sigma_q^2(t)\mathbf{I}$, where:

$$\sigma_q^2(t) = \frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} \tag{85}$$

In order to match approximate denoising transition step $p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ to ground-truth denoising transition step $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$ as closely as possible, we can also model it as a Gaussian. Furthermore, as all α terms are known to be frozen at each timestep, we can immediately construct the variance of the approximate denoising transition step to also be $\boldsymbol{\Sigma}_q(t) = \sigma_q^2(t)\mathbf{I}$. We must parameterize its mean $\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$ as a function of $\boldsymbol{x}_t$, however, since $p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ does not condition on $\boldsymbol{x}_0$.

Recall that the KL Divergence between two Gaussian distributions is:

$$D_{\mathrm{KL}}(\mathcal{N}(\boldsymbol{x};\boldsymbol{\mu}_x,\boldsymbol{\Sigma}_x)\parallel\mathcal{N}(\boldsymbol{y};\boldsymbol{\mu}_y,\boldsymbol{\Sigma}_y))=\frac{1}{2}\left[\log\frac{|\boldsymbol{\Sigma}_y|}{|\boldsymbol{\Sigma}_x|}-d+\mathrm{tr}(\boldsymbol{\Sigma}_y^{-1}\boldsymbol{\Sigma}_x)+(\boldsymbol{\mu}_y-\boldsymbol{\mu}_x)^T\boldsymbol{\Sigma}_y^{-1}(\boldsymbol{\mu}_y-\boldsymbol{\mu}_x)\right] \tag{86}$$

In our case, where we can set the variances of the two Gaussians to match exactly, optimizing the KL Divergence term reduces to minimizing the difference between the means of the two distributions:

$$\begin{aligned}
&\arg\min_{\boldsymbol{\theta}} D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)\parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))\\
&=\arg\min_{\boldsymbol{\theta}} D_{\mathrm{KL}}(\mathcal{N}(\boldsymbol{x}_{t-1};\boldsymbol{\mu}_q,\boldsymbol{\Sigma}_q(t))\parallel\mathcal{N}(\boldsymbol{x}_{t-1};\boldsymbol{\mu}_{\boldsymbol{\theta}},\boldsymbol{\Sigma}_q(t))) &(87)\\
&=\arg\min_{\boldsymbol{\theta}}\frac{1}{2}\left[\log\frac{|\boldsymbol{\Sigma}_q(t)|}{|\boldsymbol{\Sigma}_q(t)|}-d+\mathrm{tr}(\boldsymbol{\Sigma}_q(t)^{-1}\boldsymbol{\Sigma}_q(t))+(\boldsymbol{\mu}_{\boldsymbol{\theta}}-\boldsymbol{\mu}_q)^T\boldsymbol{\Sigma}_q(t)^{-1}(\boldsymbol{\mu}_{\boldsymbol{\theta}}-\boldsymbol{\mu}_q)\right] &(88)\\
&=\arg\min_{\boldsymbol{\theta}}\frac{1}{2}\left[\log 1-d+d+(\boldsymbol{\mu}_{\boldsymbol{\theta}}-\boldsymbol{\mu}_q)^T\boldsymbol{\Sigma}_q(t)^{-1}(\boldsymbol{\mu}_{\boldsymbol{\theta}}-\boldsymbol{\mu}_q)\right] &(89)\\
&=\arg\min_{\boldsymbol{\theta}}\frac{1}{2}\left[(\boldsymbol{\mu}_{\boldsymbol{\theta}}-\boldsymbol{\mu}_q)^T\boldsymbol{\Sigma}_q(t)^{-1}(\boldsymbol{\mu}_{\boldsymbol{\theta}}-\boldsymbol{\mu}_q)\right] &(90)\\
&=\arg\min_{\boldsymbol{\theta}}\frac{1}{2}\left[(\boldsymbol{\mu}_{\boldsymbol{\theta}}-\boldsymbol{\mu}_q)^T\left(\sigma_q^2(t)\mathbf{I}\right)^{-1}(\boldsymbol{\mu}_{\boldsymbol{\theta}}-\boldsymbol{\mu}_q)\right] &(91)\\
&=\arg\min_{\boldsymbol{\theta}}\frac{1}{2\sigma_q^2(t)}\left[\|\boldsymbol{\mu}_{\boldsymbol{\theta}}-\boldsymbol{\mu}_q\|_2^2\right] &(92)
\end{aligned}$$

where we have written $\boldsymbol{\mu}_q$ as shorthand for $\boldsymbol{\mu}_q(\boldsymbol{x}_t,\boldsymbol{x}_0)$, and $\boldsymbol{\mu}_{\boldsymbol{\theta}}$ as shorthand for $\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_t,t)$ for brevity. In other words, we want to optimize a $\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_t,t)$ that matches $\boldsymbol{\mu}_q(\boldsymbol{x}_t,\boldsymbol{x}_0)$, which from our derived Equation 84, takes the form:

$$\boldsymbol{\mu}_q(\boldsymbol{x}_t,\boldsymbol{x}_0)=\frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\boldsymbol{x}_t+\sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\boldsymbol{x}_0}{1-\bar{\alpha}_t} \tag{93}$$

As $\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_t,t)$ also conditions on $\boldsymbol{x}_t$, we can match $\boldsymbol{\mu}_q(\boldsymbol{x}_t,\boldsymbol{x}_0)$ closely by setting it to the following form:

$$\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_t,t)=\frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\boldsymbol{x}_t+\sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\hat{\boldsymbol{x}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t,t)}{1-\bar{\alpha}_t} \tag{94}$$

where $\hat{\boldsymbol{x}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t,t)$ is parameterized by a neural network that seeks to predict $\boldsymbol{x}_0$ from noisy image $\boldsymbol{x}_t$ and time index t. Then, the optimization problem simplifies to:

$$\begin{aligned}
&\arg\min_{\boldsymbol{\theta}} D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)\parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))\\
&=\arg\min_{\boldsymbol{\theta}} D_{\mathrm{KL}}(\mathcal{N}\left(\boldsymbol{x}_{t-1};\boldsymbol{\mu}_q,\boldsymbol{\Sigma}_q\left(t\right)\right)\parallel\mathcal{N}\left(\boldsymbol{x}_{t-1};\boldsymbol{\mu}_{\boldsymbol{\theta}},\boldsymbol{\Sigma}_q\left(t\right)\right)) &(95)\\
&=\arg\min_{\boldsymbol{\theta}}\frac{1}{2\sigma_q^2(t)}\left[\left\|\frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\boldsymbol{x}_t+\sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\hat{\boldsymbol{x}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t,t)}{1-\bar{\alpha}_t}-\frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\boldsymbol{x}_t+\sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\boldsymbol{x}_0}{1-\bar{\alpha}_t}\right\|_2^2\right] &(96)\\
&=\arg\min_{\boldsymbol{\theta}}\frac{1}{2\sigma_q^2(t)}\left[\left\|\frac{\sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\hat{\boldsymbol{x}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t,t)}{1-\bar{\alpha}_t}-\frac{\sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\boldsymbol{x}_0}{1-\bar{\alpha}_t}\right\|_2^2\right] &(97)\\
&=\arg\min_{\boldsymbol{\theta}}\frac{1}{2\sigma_q^2(t)}\left[\left\|\frac{\sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)}{1-\bar{\alpha}_t}\left(\hat{\boldsymbol{x}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t,t)-\boldsymbol{x}_0\right)\right\|_2^2\right] &(98)\\
&=\arg\min_{\boldsymbol{\theta}}\frac{1}{2\sigma_q^2(t)}\frac{\bar{\alpha}_{t-1}(1-\alpha_t)^2}{(1-\bar{\alpha}_t)^2}\left[\|\hat{\boldsymbol{x}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t,t)-\boldsymbol{x}_0\|_2^2\right] &(99)
\end{aligned}$$

Therefore, optimizing a VDM boils down to learning a neural network to predict the original ground truth image from an arbitrarily noisified version of it [5]. Furthermore, minimizing the summation term of our derived ELBO objective (Equation 58) across all noise levels can be approximated by minimizing the expectation over all timesteps:

$$\arg\min_{\boldsymbol{\theta}}\mathbb{E}_{t\sim U\{2,T\}}\left[\mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)\parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))\right]\right] \tag{100}$$

which can then be optimized using stochastic samples over timesteps.

Learning Diffusion Noise Parameters

Let us investigate how the noise parameters of a VDM can be jointly learned. One potential approach is to model α_t using a neural network $\hat{\alpha}_{\boldsymbol{\eta}}(t)$ with parameters $\boldsymbol{\eta}$. However, this is inefficient as inference must be performed multiple times at each timestep t to compute $\bar{\alpha}_t$. Whereas caching can mitigate this computational cost, we can also derive an alternate way to learn the diffusion noise parameters. By substituting our variance equation from Equation 85 into our derived per-timestep objective in Equation 99, we can reduce:

$$\frac{1}{2\sigma_q^2(t)}\frac{\bar{\alpha}_{t-1}(1-\alpha_t)^2}{(1-\bar{\alpha}_t)^2}\left[\|\hat{\boldsymbol{x}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t,t)-\boldsymbol{x}_0\|_2^2\right] = \frac{1}{2\frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}}\frac{\bar{\alpha}_{t-1}(1-\alpha_t)^2}{(1-\bar{\alpha}_t)^2}\left[\|\hat{\boldsymbol{x}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t,t)-\boldsymbol{x}_0\|_2^2\right] \tag{101}$$

$$= \frac{1}{2}\frac{1-\bar{\alpha}_t}{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}\frac{\bar{\alpha}_{t-1}(1-\alpha_t)^2}{(1-\bar{\alpha}_t)^2}\left[\|\hat{\boldsymbol{x}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t,t)-\boldsymbol{x}_0\|_2^2\right] \tag{102}$$

$$= \frac{1}{2}\frac{\bar{\alpha}_{t-1}(1-\alpha_t)}{(1-\bar{\alpha}_{t-1})(1-\bar{\alpha}_t)}\left[\|\hat{\boldsymbol{x}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t,t)-\boldsymbol{x}_0\|_2^2\right] \tag{103}$$

$$= \frac{1}{2}\frac{\bar{\alpha}_{t-1}-\bar{\alpha}_t}{(1-\bar{\alpha}_{t-1})(1-\bar{\alpha}_t)}\left[\|\hat{\boldsymbol{x}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t,t)-\boldsymbol{x}_0\|_2^2\right] \tag{104}$$

$$= \frac{1}{2}\frac{\bar{\alpha}_{t-1}-\bar{\alpha}_{t-1}\bar{\alpha}_t+\bar{\alpha}_{t-1}\bar{\alpha}_t-\bar{\alpha}_t}{(1-\bar{\alpha}_{t-1})(1-\bar{\alpha}_t)}\left[\|\hat{\boldsymbol{x}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t,t)-\boldsymbol{x}_0\|_2^2\right] \tag{105}$$

$$= \frac{1}{2}\frac{\bar{\alpha}_{t-1}(1-\bar{\alpha}_t)-\bar{\alpha}_t(1-\bar{\alpha}_{t-1})}{(1-\bar{\alpha}_{t-1})(1-\bar{\alpha}_t)}\left[\|\hat{\boldsymbol{x}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t,t)-\boldsymbol{x}_0\|_2^2\right] \tag{106}$$

$$= \frac{1}{2}\left(\frac{\bar{\alpha}_{t-1}(1-\bar{\alpha}_t)}{(1-\bar{\alpha}_{t-1})(1-\bar{\alpha}_t)}-\frac{\bar{\alpha}_t(1-\bar{\alpha}_{t-1})}{(1-\bar{\alpha}_{t-1})(1-\bar{\alpha}_t)}\right)\left[\|\hat{\boldsymbol{x}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t,t)-\boldsymbol{x}_0\|_2^2\right] \tag{107}$$

$$= \frac{1}{2}\left(\frac{\bar{\alpha}_{t-1}}{1-\bar{\alpha}_{t-1}}-\frac{\bar{\alpha}_t}{1-\bar{\alpha}_t}\right)\left[\|\hat{\boldsymbol{x}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t,t)-\boldsymbol{x}_0\|_2^2\right] \tag{108}$$

Recall from Equation 70 that $q(\boldsymbol{x}_t|\boldsymbol{x}_0)$ is a Gaussian of form $\mathcal{N}(\boldsymbol{x}_t;\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0,(1-\bar{\alpha}_t)\mathbf{I})$. Then, following the definition of the signal-to-noise ratio (SNR) as $\text{SNR}=\frac{\mu^2}{\sigma^2}$, we can write the SNR at each timestep t as:

$$\text{SNR}(t) = \frac{\bar{\alpha}_t}{1-\bar{\alpha}_t} \tag{109}$$

Then, our derived Equation 108 (and Equation 99) can be simplified as:

$$\frac{1}{2\sigma_q^2(t)}\frac{\bar{\alpha}_{t-1}(1-\alpha_t)^2}{(1-\bar{\alpha}_t)^2}\left[\|\hat{\boldsymbol{x}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t,t)-\boldsymbol{x}_0\|_2^2\right] = \frac{1}{2}\left(\text{SNR}(t-1)-\text{SNR}(t)\right)\left[\|\hat{\boldsymbol{x}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t,t)-\boldsymbol{x}_0\|_2^2\right] \tag{110}$$

As the name implies, the SNR represents the ratio between the original signal and the amount of noise present; a higher SNR represents more signal and a lower SNR represents more noise. In a diffusion model, we require the SNR to monotonically decrease as timestep t increases; this formalizes the notion that perturbed input $\boldsymbol{x}_t$ becomes increasingly noisy over time, until it becomes identical to a standard Gaussian at $t=T$.

Following the simplification of the objective in Equation 110, we can directly parameterize the SNR at each timestep using a neural network, and learn it jointly along with the diffusion model. As the SNR must monotonically decrease over time, we can represent it as:

$$\text{SNR}(t) = \exp(-\omega_{\boldsymbol{\eta}}(t)) \tag{111}$$

where $\omega_{\boldsymbol{\eta}}(t)$ is modeled as a monotonically increasing neural network with parameters $\boldsymbol{\eta}$. Negating $\omega_{\boldsymbol{\eta}}(t)$ results in a monotonically decreasing function, whereas the exponential forces the resulting term to be positive. Note that the objective in Equation 100 must now optimize over $\boldsymbol{\eta}$ as well. By combining our parameterization of SNR in Equation 111 with our definition of SNR in Equation 109, we can also explicitly derive elegant forms for the value of $\bar{\alpha}_t$ as well as for the value of $1-\bar{\alpha}_t$:

$$\frac{\bar{\alpha}_t}{1-\bar{\alpha}_t} = \exp(-\omega_{\boldsymbol{\eta}}(t)) \tag{112}$$

$$\therefore \bar{\alpha}_t = \text{sigmoid}(-\omega_{\boldsymbol{\eta}}(t)) \tag{113}$$

$$\therefore 1-\bar{\alpha}_t = \text{sigmoid}(\omega_{\boldsymbol{\eta}}(t)) \tag{114}$$

These terms are necessary for a variety of computations; for example, during optimization, they are used to create arbitrarily noisy $\boldsymbol{x}_t$ from input $\boldsymbol{x}_0$ using the reparameterization trick, as derived in Equation 69.

Three Equivalent Interpretations

As we previously proved, a Variational Diffusion Model can be trained by simply learning a neural network to predict the original natural image $\boldsymbol{x}_0$ from an arbitrary noised version $\boldsymbol{x}_t$ and its time index t. However, $\boldsymbol{x}_0$ has two other equivalent parameterizations, which leads to two further interpretations for a VDM.

Firstly, we can utilize the reparameterization trick. In our derivation of the form of $q(\boldsymbol{x}_t|\boldsymbol{x}_0)$, we can rearrange Equation 69 to show that:

$$\boldsymbol{x}_0 = \frac{\boldsymbol{x}_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_0}{\sqrt{\bar{\alpha}_t}} \tag{115}$$

Plugging this into our previously derived true denoising transition mean $\boldsymbol{\mu}_q(\boldsymbol{x}_t, \boldsymbol{x}_0)$, we can rederive as:

$$\begin{aligned}
\boldsymbol{\mu}_q(\boldsymbol{x}_t, \boldsymbol{x}_0) &= \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\boldsymbol{x}_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\boldsymbol{x}_0}{1-\bar{\alpha}_t} && (116)\\
&= \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\boldsymbol{x}_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\frac{\boldsymbol{x}_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_0}{\sqrt{\bar{\alpha}_t}}}{1-\bar{\alpha}_t} && (117)\\
&= \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\boldsymbol{x}_t + (1-\alpha_t)\frac{\boldsymbol{x}_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_0}{\sqrt{\alpha_t}}}{1-\bar{\alpha}_t} && (118)\\
&= \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\boldsymbol{x}_t}{1-\bar{\alpha}_t} + \frac{(1-\alpha_t)\boldsymbol{x}_t}{(1-\bar{\alpha}_t)\sqrt{\alpha_t}} - \frac{(1-\alpha_t)\sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_0}{(1-\bar{\alpha}_t)\sqrt{\alpha_t}} && (119)\\
&= \left(\frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} + \frac{1-\alpha_t}{(1-\bar{\alpha}_t)\sqrt{\alpha_t}}\right)\boldsymbol{x}_t - \frac{(1-\alpha_t)\sqrt{1-\bar{\alpha}_t}}{(1-\bar{\alpha}_t)\sqrt{\alpha_t}}\boldsymbol{\epsilon}_0 && (120)\\
&= \left(\frac{\alpha_t(1-\bar{\alpha}_{t-1})}{(1-\bar{\alpha}_t)\sqrt{\alpha_t}} + \frac{1-\alpha_t}{(1-\bar{\alpha}_t)\sqrt{\alpha_t}}\right)\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}\boldsymbol{\epsilon}_0 && (121)\\
&= \frac{\alpha_t - \bar{\alpha}_t + 1 - \alpha_t}{(1-\bar{\alpha}_t)\sqrt{\alpha_t}}\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}\boldsymbol{\epsilon}_0 && (122)\\
&= \frac{1-\bar{\alpha}_t}{(1-\bar{\alpha}_t)\sqrt{\alpha_t}}\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}\boldsymbol{\epsilon}_0 && (123)\\
&= \frac{1}{\sqrt{\alpha_t}}\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}\boldsymbol{\epsilon}_0 && (124)
\end{aligned}$$

Therefore, we can set our approximate denoising transition mean $\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$ as:

$$\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) \tag{125}$$

and the corresponding optimization problem becomes:

$$\begin{aligned}
&\arg\min_{\boldsymbol{\theta}} D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))\\
&= \arg\min_{\boldsymbol{\theta}} D_{\mathrm{KL}}(\mathcal{N}\left(\boldsymbol{x}_{t-1}; \boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q\left(t\right)\right) \parallel \mathcal{N}\left(\boldsymbol{x}_{t-1}; \boldsymbol{\mu}_{\boldsymbol{\theta}}, \boldsymbol{\Sigma}_q\left(t\right)\right)) && (126)\\
&= \arg\min_{\boldsymbol{\theta}} \frac{1}{2\sigma_q^2(t)}\left[\left\|\frac{1}{\sqrt{\alpha_t}}\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) - \frac{1}{\sqrt{\alpha_t}}\boldsymbol{x}_t + \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}\boldsymbol{\epsilon}_0\right\|_2^2\right] && (127)\\
&= \arg\min_{\boldsymbol{\theta}} \frac{1}{2\sigma_q^2(t)}\left[\left\|\frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}\boldsymbol{\epsilon}_0 - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)\right\|_2^2\right] && (128)\\
&= \arg\min_{\boldsymbol{\theta}} \frac{1}{2\sigma_q^2(t)}\left[\left\|\frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}(\boldsymbol{\epsilon}_0 - \hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t))\right\|_2^2\right] && (129)\\
&= \arg\min_{\boldsymbol{\theta}} \frac{1}{2\sigma_q^2(t)}\frac{(1-\alpha_t)^2}{(1-\bar{\alpha}_t)\alpha_t}\left[\|\boldsymbol{\epsilon}_0 - \hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)\|_2^2\right] && (130)
\end{aligned}$$

Here, $\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$ is a neural network that learns to predict the source noise $\boldsymbol{\epsilon}_0 \sim \mathcal{N}(\boldsymbol{\epsilon}; \mathbf{0}, \mathbf{I})$ that determines $\boldsymbol{x}_t$ from $\boldsymbol{x}_0$. We have therefore shown that learning a VDM by predicting the original image $\boldsymbol{x}_0$ is equivalent to learning to predict the noise; empirically, however, some works have found that predicting the noise resulted in better performance [5, 7].

To derive the third common interpretation of Variational Diffusion Models, we appeal to Tweedie's Formula [8]. In English, Tweedie's Formula states that the true mean of an exponential family distribution, given samples drawn from it, can be estimated by the maximum likelihood estimate of the samples (aka empirical mean) plus some correction term involving the score of the estimate. In the case of just one observed sample, the empirical mean is just the sample itself. It is commonly used to mitigate sample bias; if observed samples all lie on one end of the underlying distribution, then the negative score becomes large and corrects the naive maximum likelihood estimate of the samples towards the true mean.

Mathematically, for a Gaussian variable $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{z}; \boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z)$, Tweedie's Formula states that:

$$\mathbb{E}\left[\boldsymbol{\mu}_z | \boldsymbol{z}\right] = \boldsymbol{z} + \boldsymbol{\Sigma}_z \nabla_{\boldsymbol{z}} \log p(\boldsymbol{z})$$

In this case, we apply it to predict the true posterior mean of $\boldsymbol{x}_t$ given its samples. From Equation 70, we know that:

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

Then, by Tweedie's Formula, we have:

$$\mathbb{E}\left[\boldsymbol{\mu}_{x_t} | \boldsymbol{x}_t\right] = \boldsymbol{x}_t + (1 - \bar{\alpha}_t)\nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t) \tag{131}$$

where we write $\nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t)$ as $\nabla \log p(\boldsymbol{x}_t)$ for notational simplicity. According to Tweedie's Formula, the best estimate for the true mean that $\boldsymbol{x}_t$ is generated from, $\boldsymbol{\mu}_{x_t} = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0$, is defined as:

$$\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 = \boldsymbol{x}_t + (1 - \bar{\alpha}_t)\nabla \log p(\boldsymbol{x}_t) \tag{132}$$

$$\therefore \boldsymbol{x}_0 = \frac{\boldsymbol{x}_t + (1 - \bar{\alpha}_t)\nabla \log p(\boldsymbol{x}_t)}{\sqrt{\bar{\alpha}_t}} \tag{133}$$

Then, we can plug Equation 133 into our ground-truth denoising transition mean $\boldsymbol{\mu}_q(\boldsymbol{x}_t, \boldsymbol{x}_0)$ once again and derive a new form:

$$\boldsymbol{\mu}_q(\boldsymbol{x}_t, \boldsymbol{x}_0) = \frac{\sqrt{\alpha_t}(1 - \bar{\alpha}_{t-1})\boldsymbol{x}_t + \sqrt{\bar{\alpha}_{t-1}}(1 - \alpha_t)\boldsymbol{x}_0}{1 - \bar{\alpha}_t} \tag{134}$$

$$= \frac{\sqrt{\alpha_t}(1 - \bar{\alpha}_{t-1})\boldsymbol{x}_t + \sqrt{\bar{\alpha}_{t-1}}(1 - \alpha_t)\frac{\boldsymbol{x}_t + (1-\bar{\alpha}_t)\nabla \log p(\boldsymbol{x}_t)}{\sqrt{\bar{\alpha}_t}}}{1 - \bar{\alpha}_t} \tag{135}$$

$$= \frac{\sqrt{\alpha_t}(1 - \bar{\alpha}_{t-1})\boldsymbol{x}_t + (1 - \alpha_t)\frac{\boldsymbol{x}_t + (1-\bar{\alpha}_t)\nabla \log p(\boldsymbol{x}_t)}{\sqrt{\alpha_t}}}{1 - \bar{\alpha}_t} \tag{136}$$

$$= \frac{\sqrt{\alpha_t}(1 - \bar{\alpha}_{t-1})\boldsymbol{x}_t}{1 - \bar{\alpha}_t} + \frac{(1 - \alpha_t)\boldsymbol{x}_t}{(1 - \bar{\alpha}_t)\sqrt{\alpha_t}} + \frac{(1 - \alpha_t)(1 - \bar{\alpha}_t)\nabla \log p(\boldsymbol{x}_t)}{(1 - \bar{\alpha}_t)\sqrt{\alpha_t}} \tag{137}$$

$$= \left(\frac{\sqrt{\alpha_t}(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t} + \frac{1 - \alpha_t}{(1 - \bar{\alpha}_t)\sqrt{\alpha_t}}\right)\boldsymbol{x}_t + \frac{1 - \alpha_t}{\sqrt{\alpha_t}}\nabla \log p(\boldsymbol{x}_t) \tag{138}$$

$$= \left(\frac{\alpha_t(1 - \bar{\alpha}_{t-1})}{(1 - \bar{\alpha}_t)\sqrt{\alpha_t}} + \frac{1 - \alpha_t}{(1 - \bar{\alpha}_t)\sqrt{\alpha_t}}\right)\boldsymbol{x}_t + \frac{1 - \alpha_t}{\sqrt{\alpha_t}}\nabla \log p(\boldsymbol{x}_t) \tag{139}$$

$$= \frac{\alpha_t - \bar{\alpha}_t + 1 - \alpha_t}{(1 - \bar{\alpha}_t)\sqrt{\alpha_t}}\boldsymbol{x}_t + \frac{1 - \alpha_t}{\sqrt{\alpha_t}}\nabla \log p(\boldsymbol{x}_t) \tag{140}$$

$$= \frac{1 - \bar{\alpha}_t}{(1 - \bar{\alpha}_t)\sqrt{\alpha_t}}\boldsymbol{x}_t + \frac{1 - \alpha_t}{\sqrt{\alpha_t}}\nabla \log p(\boldsymbol{x}_t) \tag{141}$$

$$= \frac{1}{\sqrt{\alpha_t}}\boldsymbol{x}_t + \frac{1 - \alpha_t}{\sqrt{\alpha_t}}\nabla \log p(\boldsymbol{x}_t) \tag{142}$$

Therefore, we can also set our approximate denoising transition mean $\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$ as:

$$\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\boldsymbol{x}_t + \frac{1-\alpha_t}{\sqrt{\alpha_t}}\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) \tag{143}$$

and the corresponding optimization problem becomes:

$$\begin{aligned}
&\arg\min_{\boldsymbol{\theta}} D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)) \\
&= \arg\min_{\boldsymbol{\theta}} D_{\mathrm{KL}}(\mathcal{N}\left(\boldsymbol{x}_{t-1}; \boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q\left(t\right)\right) \parallel \mathcal{N}\left(\boldsymbol{x}_{t-1}; \boldsymbol{\mu}_{\boldsymbol{\theta}}, \boldsymbol{\Sigma}_q\left(t\right)\right)) && (144) \\
&= \arg\min_{\boldsymbol{\theta}} \frac{1}{2\sigma_q^2(t)}\left[\left\|\frac{1}{\sqrt{\alpha_t}}\boldsymbol{x}_t + \frac{1-\alpha_t}{\sqrt{\alpha_t}}\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) - \frac{1}{\sqrt{\alpha_t}}\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{\alpha_t}}\nabla \log p(\boldsymbol{x}_t)\right\|_2^2\right] && (145) \\
&= \arg\min_{\boldsymbol{\theta}} \frac{1}{2\sigma_q^2(t)}\left[\left\|\frac{1-\alpha_t}{\sqrt{\alpha_t}}\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) - \frac{1-\alpha_t}{\sqrt{\alpha_t}}\nabla \log p(\boldsymbol{x}_t)\right\|_2^2\right] && (146) \\
&= \arg\min_{\boldsymbol{\theta}} \frac{1}{2\sigma_q^2(t)}\left[\left\|\frac{1-\alpha_t}{\sqrt{\alpha_t}}(\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) - \nabla \log p(\boldsymbol{x}_t))\right\|_2^2\right] && (147) \\
&= \arg\min_{\boldsymbol{\theta}} \frac{1}{2\sigma_q^2(t)}\frac{(1-\alpha_t)^2}{\alpha_t}\left[\|\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) - \nabla \log p(\boldsymbol{x}_t)\|_2^2\right] && (148)
\end{aligned}$$

Here, $\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$ is a neural network that learns to predict the score function $\nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t)$, which is the gradient of $\boldsymbol{x}_t$ in data space, for any arbitrary noise level t.

The astute reader will notice that the score function $\nabla \log p(\boldsymbol{x}_t)$ looks remarkably similar in form to the source noise $\boldsymbol{\epsilon}_0$. This can be shown explicitly by combining Tweedie's Formula (Equation 133) with the reparameterization trick (Equation 115):

$$\begin{aligned}
\boldsymbol{x}_0 = \frac{\boldsymbol{x}_t + (1-\bar{\alpha}_t)\nabla \log p(\boldsymbol{x}_t)}{\sqrt{\bar{\alpha}_t}} &= \frac{\boldsymbol{x}_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_0}{\sqrt{\bar{\alpha}_t}} && (149) \\
\therefore (1-\bar{\alpha}_t)\nabla \log p(\boldsymbol{x}_t) &= -\sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_0 && (150) \\
\nabla \log p(\boldsymbol{x}_t) &= -\frac{1}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_0 && (151)
\end{aligned}$$

As it turns out, the two terms are off by a constant factor that scales with time! The score function measures how to move in data space to maximize the log probability; intuitively, since the source noise is added to a natural image to corrupt it, moving in its opposite direction "denoises" the image and would be the best update to increase the subsequent log probability. Our mathematical proof justifies this intuition; we have explicitly shown that learning to model the score function is equivalent to modeling the negative of the source noise (up to a scaling factor).

We have therefore derived three equivalent objectives to optimize a VDM: learning a neural network to predict the original image $\boldsymbol{x}_0$, the source noise $\boldsymbol{\epsilon}_0$, or the score of the image at an arbitrary noise level $\nabla \log p(\boldsymbol{x}_t)$. The VDM can be scalably trained by stochastically sampling timesteps t and minimizing the norm of the prediction with the ground truth target.

Score-based Generative Models

We have shown that a Variational Diffusion Model can be learned simply by optimizing a neural network $\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$ to predict the score function $\nabla \log p(\boldsymbol{x}_t)$. However, in our derivation, the score term arrived from an application of Tweedie's Formula; this doesn't necessarily provide us with great intuition or insight into what exactly the score function is or why it is worth modeling. Fortunately, we can look to another class of generative models, Score-based Generative Models [9, 10, 11], for exactly this intuition. As it turns out, we can show that the VDM formulation we have previously derived has an equivalent Score-based Generative Modeling formulation, allowing us to flexibly switch between these two interpretations at will.

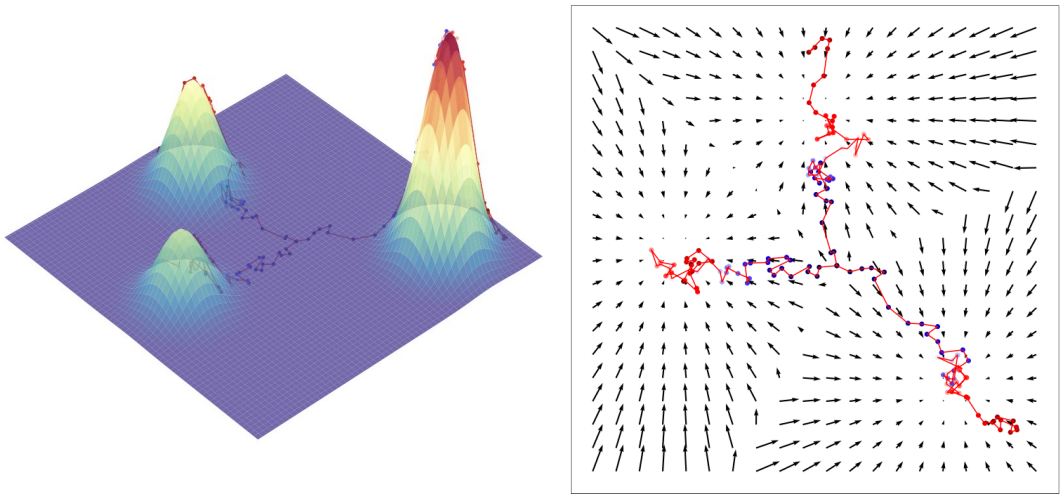

Figure 6: Visualization of three random sampling trajectories generated with Langevin dynamics, all starting from the same initialization point, for a Mixture of Gaussians. The left figure plots these sampling trajectories on a three-dimensional contour, while the right figure plots the sampling trajectories against the ground-truth score function. From the same initialization point, we are able to generate samples from different modes due to the stochastic noise term in the Langevin dynamics sampling procedure; without it, sampling from a fixed point would always deterministically follow the score to the same mode every trial.

To begin to understand why optimizing a score function makes sense, we take a detour and revisit energy-based models [12, 13]. Arbitrarily flexible probability distributions can be written in the form:

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \frac{1}{Z_{\boldsymbol{\theta}}} e^{-f_{\boldsymbol{\theta}}(\boldsymbol{x})} \tag{152}$$

where $f_{\boldsymbol{\theta}}(\boldsymbol{x})$ is an arbitrarily flexible, parameterizable function called the energy function, often modeled by a neural network, and $Z_{\boldsymbol{\theta}}$ is a normalizing constant to ensure that $\int p_{\boldsymbol{\theta}}(\boldsymbol{x}) d\boldsymbol{x} = 1$. One way to learn such a distribution is maximum likelihood; however, this requires tractably computing the normalizing constant $Z_{\boldsymbol{\theta}} = \int e^{-f_{\boldsymbol{\theta}}(\boldsymbol{x})} d\boldsymbol{x}$, which may not be possible for complex $f_{\boldsymbol{\theta}}(\boldsymbol{x})$ functions.

One way to avoid calculating or modeling the normalization constant is by using a neural network $\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x})$ to learn the score function $\nabla \log p(\boldsymbol{x})$ of distribution $p(\boldsymbol{x})$ instead. This is motivated by the observation that taking the derivative of the log of both sides of Equation 152 yields:

$$\begin{aligned}
\nabla_{\boldsymbol{x}} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}) &= \nabla_{\boldsymbol{x}} \log\left(\frac{1}{Z_{\boldsymbol{\theta}}} e^{-f_{\boldsymbol{\theta}}(\boldsymbol{x})}\right) && (153)\\
&= \nabla_{\boldsymbol{x}} \log \frac{1}{Z_{\boldsymbol{\theta}}} + \nabla_{\boldsymbol{x}} \log e^{-f_{\boldsymbol{\theta}}(\boldsymbol{x})} && (154)\\
&= -\nabla_{\boldsymbol{x}} f_{\boldsymbol{\theta}}(\boldsymbol{x}) && (155)\\
&\approx \boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}) && (156)
\end{aligned}$$

which can be freely represented as a neural network without involving any normalization constants. The score model can be optimized by minimizing the Fisher Divergence with the ground truth score function:

$$\mathbb{E}_{p(\boldsymbol{x})}\left[\|\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}) - \nabla \log p(\boldsymbol{x})\|_2^2\right] \tag{157}$$

What does the score function represent? For every $\boldsymbol{x}$, taking the gradient of its log likelihood with respect to $\boldsymbol{x}$ essentially describes what direction in data space to move in order to further increase its likelihood.

Intuitively, then, the score function defines a vector field over the entire space that data $\boldsymbol{x}$ inhabits, pointing towards the modes. Visually, this is depicted in the right plot of Figure 6. Then, by learning the score function of the true data distribution, we can generate samples by starting at any arbitrary point in the same space and iteratively following the score until a mode is reached. This sampling procedure is known as Langevin dynamics, and is mathematically described as:

$$\boldsymbol{x}_{i+1} \leftarrow \boldsymbol{x}_i + c\nabla \log p(\boldsymbol{x}_i) + \sqrt{2c}\boldsymbol{\epsilon}, \quad i = 0, 1, ..., K \tag{158}$$

where $\boldsymbol{x}_0$ is randomly sampled from a prior distribution (such as uniform), and $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\epsilon}; \mathbf{0}, \mathbf{I})$ is an extra noise term to ensure that the generated samples do not always collapse onto a mode, but hover around it for diversity. Furthermore, because the learned score function is deterministic, sampling with a noise term involved adds stochasticity to the generative process, allowing us to avoid deterministic trajectories. This is particularly useful when sampling is initialized from a position that lies between multiple modes. A visual depiction of Langevin dynamics sampling and the benefits of the noise term is shown in Figure 6.

Note that the objective in Equation 157 relies on having access to the ground truth score function, which is unavailable to us for complex distributions such as the one modeling natural images. Fortunately, alternative techniques known as score matching [14, 15, 16, 17] have been derived to minimize this Fisher divergence without knowing the ground truth score, and can be optimized with stochastic gradient descent.

Collectively, learning to represent a distribution as a score function and using it to generate samples through Markov Chain Monte Carlo techniques, such as Langevin dynamics, is known as Score-based Generative Modeling [9, 10, 11].

There are three main problems with vanilla score matching, as detailed by Song and Ermon [9]. Firstly, the score function is ill-defined when $\boldsymbol{x}$ lies on a low-dimensional manifold in a high-dimensional space. This can be seen mathematically; all points not on the low-dimensional manifold would have probability zero, the log of which is undefined. This is particularly inconvenient when trying to learn a generative model over natural images, which is known to lie on a low-dimensional manifold of the entire ambient space.

Secondly, the estimated score function trained via vanilla score matching will not be accurate in low density regions. This is evident from the objective we minimize in Equation 157. Because it is an expectation over $p(\boldsymbol{x})$, and explicitly trained on samples from it, the model will not receive an accurate learning signal for rarely seen or unseen examples. This is problematic, since our sampling strategy involves starting from a random location in the high-dimensional space, which is most likely random noise, and moving according to the learned score function. Since we are following a noisy or inaccurate score estimate, the final generated samples may be suboptimal as well, or require many more iterations to converge on an accurate output.

Lastly, Langevin dynamics sampling may not mix, even if it is performed using the ground truth scores. Suppose that the true data distribution is a mixture of two disjoint distributions:

$$p(\boldsymbol{x}) = c_1 p_1(\boldsymbol{x}) + c_2 p_2(\boldsymbol{x}) \tag{159}$$

Then, when the score is computed, these mixing coefficients are lost, since the log operation splits the coefficient from the distribution and the gradient operation zeros it out. To visualize this, note that the ground truth score function shown in the right Figure 6 is agnostic of the different weights between the three distributions; Langevin dynamics sampling from the depicted initialization point has a roughly equal chance of arriving at each mode, despite the bottom right mode having a higher weight in the actual Mixture of Gaussians.

It turns out that these three drawbacks can be simultaneously addressed by adding multiple levels of Gaussian noise to the data. Firstly, as the support of a Gaussian noise distribution is the entire space, a perturbed data sample will no longer be confined to a low-dimensional manifold. Secondly, adding large Gaussian noise will increase the area each mode covers in the data distribution, adding more training signal in low density regions. Lastly, adding multiple levels of Gaussian noise with increasing variance will result in intermediate distributions that respect the ground truth mixing coefficients.

Formally, we can choose a positive sequence of noise levels $\{\sigma_t\}_{t=1}^T$ and define a sequence of progressively perturbed data distributions:

$$p_{\sigma_t}(\boldsymbol{x}_t) = \int p(\boldsymbol{x})\mathcal{N}(\boldsymbol{x}_t; \boldsymbol{x}, \sigma_t^2\mathbf{I})d\boldsymbol{x} \tag{160}$$

Then, a neural network $\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}, t)$ is learned using score matching to learn the score function for all noise levels simultaneously:

$$\arg\min_{\boldsymbol{\theta}} \sum_{t=1}^{T} \lambda(t)\mathbb{E}_{p_{\sigma_t}(\boldsymbol{x}_t)}\left[\|\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}, t) - \nabla \log p_{\sigma_t}(\boldsymbol{x}_t)\|_2^2\right] \tag{161}$$

where $\lambda(t)$ is a positive weighting function that conditions on noise level t. Note that this objective almost exactly matches the objective derived in Equation 148 to train a Variational Diffusion Model. Furthermore, the authors propose annealed Langevin dynamics sampling as a generative procedure, in which samples are produced by running Langevin dynamics for each $t = T, T-1, ..., 2, 1$ in sequence. The initialization is chosen from some fixed prior (such as uniform), and each subsequent sampling step starts from the final samples of the previous simulation. Because the noise levels steadily decrease over timesteps t, and we reduce the step size over time, the samples eventually converge into a true mode. This is directly analogous to the sampling procedure performed in the Markovian HVAE interpretation of a Variational Diffusion Model, where a randomly initialized data vector is iteratively refined over decreasing noise levels.

Therefore, we have established an explicit connection between Variational Diffusion Models and Score-based Generative Models, both in their training objectives and sampling procedures.

One question is how to naturally generalize diffusion models to an infinite number of timesteps. Under the Markovian HVAE view, this can be interpreted as extending the number of hierarchies to infinity $T \to \infty$. It is clearer to represent this from the equivalent score-based generative model perspective; under an infinite number of noise scales, the perturbation of an image over continuous time can be represented as a stochastic process, and therefore described by a stochastic differential equation (SDE). Sampling is then performed by reversing the SDE, which naturally requires estimating the score function at each continuous-valued noise level [10]. Different parameterizations of the SDE essentially describe different perturbation schemes over time, enabling flexible modeling of the noising procedure [6].

Guidance

So far, we have focused on modeling just the data distribution $p(\boldsymbol{x})$. However, we are often also interested in learning conditional distribution $p(\boldsymbol{x}|y)$, which would enable us to explicitly control the data we generate through conditioning information y. This forms the backbone of image super-resolution models such as Cascaded Diffusion Models [18], as well as state-of-the-art image-text models such as DALL-E 2 [19] and Imagen [7].

A natural way to add conditioning information is simply alongside the timestep information, at each iteration. Recall our joint distribution from Equation 32:

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

Then, to turn this into a conditional diffusion model, we can simply add arbitrary conditioning information y at each transition step as:

$$p(\boldsymbol{x}_{0:T}|y) = p(\boldsymbol{x}_T)\prod_{t=1}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, y) \tag{162}$$

For example, y could be a text encoding in image-text generation, or a low-resolution image to perform super-resolution on. We are thus able to learn the core neural networks of a VDM as before, by predicting $\hat{\boldsymbol{x}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t, y) \approx \boldsymbol{x}_0$, $\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t, y) \approx \boldsymbol{\epsilon}_0$, or $\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t, y) \approx \nabla \log p(\boldsymbol{x}_t|y)$ for each desired interpretation and implementation.

A caveat of this vanilla formulation is that a conditional diffusion model trained in this way may potentially learn to ignore or downplay any given conditioning information. Guidance is therefore proposed as a way to more explicitly control the amount of weight the model gives to the conditioning information, at the cost of sample diversity. The two most popular forms of guidance are known as Classifier Guidance [10, 20] and Classifier-Free Guidance [21].

Classifier Guidance

Let us begin with the score-based formulation of a diffusion model, where our goal is to learn $\nabla \log p(\boldsymbol{x}_t|y)$, the score of the conditional model, at arbitrary noise levels t. Recall that ∇ is shorthand for $\nabla_{\boldsymbol{x}_t}$ in the interest of brevity. By Bayes rule, we can derive the following equivalent form:

$$\begin{aligned}
\nabla \log p(\boldsymbol{x}_t|y) &= \nabla \log \left(\frac{p(\boldsymbol{x}_t)p(y|\boldsymbol{x}_t)}{p(y)} \right) && (163)\\
&= \nabla \log p(\boldsymbol{x}_t) + \nabla \log p(y|\boldsymbol{x}_t) - \nabla \log p(y) && (164)\\
&= \underbrace{\nabla \log p(\boldsymbol{x}_t)}_{\text{unconditional score}} + \underbrace{\nabla \log p(y|\boldsymbol{x}_t)}_{\text{adversarial gradient}} && (165)
\end{aligned}$$

where we have leveraged the fact that the gradient of $\log p(y)$ with respect to $\boldsymbol{x}_t$ is zero.

Our final derived result can be interpreted as learning an unconditional score function combined with the adversarial gradient of a classifier $p(y|\boldsymbol{x}_t)$. Therefore, in Classifier Guidance [10, 20], the score of an unconditional diffusion model is learned as previously derived, alongside a classifier that takes in arbitrary noisy $\boldsymbol{x}_t$ and attempts to predict conditional information y. Then, during the sampling procedure, the overall conditional score function used for annealed Langevin dynamics is computed as the sum of the unconditional score function and the adversarial gradient of the noisy classifier.

In order to introduce fine-grained control to either encourage or discourage the model to consider the conditioning information, Classifier Guidance scales the adversarial gradient of the noisy classifier by a γ hyperparameter term. The score function learned under Classifier Guidance can then be summarized as:

$$\nabla \log p(\boldsymbol{x}_t|y) = \nabla \log p(\boldsymbol{x}_t) + \gamma \nabla \log p(y|\boldsymbol{x}_t) \tag{166}$$

Intuitively, when $\gamma = 0$ the conditional diffusion model learns to ignore the conditioning information entirely, and when γ is large the conditional diffusion model learns to produce samples that heavily adhere to the conditioning information. This would come at the cost of sample diversity, as it would only produce data that would be easy to regenerate the provided conditioning information from, even at noisy levels.

One noted drawback of Classifier Guidance is its reliance on a separately learned classifier. Because the classifier must handle arbitrarily noisy inputs, which most existing pretrained classification models are not optimized to do, it must be learned ad hoc alongside the diffusion model.

Classifier-Free Guidance

In Classifier-Free Guidance [21], the authors ditch the training of a separate classifier model in favor of an unconditional diffusion model and a conditional diffusion model. To derive the score function under Classifier-Free Guidance, we can first rearrange Equation 165 to show that:

$$\nabla \log p(y|\boldsymbol{x}_t) = \nabla \log p(\boldsymbol{x}_t|y) - \nabla \log p(\boldsymbol{x}_t) \tag{167}$$

Then, substituting this into Equation 166, we get:

$$\begin{aligned}
\nabla \log p(\boldsymbol{x}_t|y) &= \nabla \log p(\boldsymbol{x}_t) + \gamma \left(\nabla \log p(\boldsymbol{x}_t|y) - \nabla \log p(\boldsymbol{x}_t) \right) && (168)\\
&= \nabla \log p(\boldsymbol{x}_t) + \gamma \nabla \log p(\boldsymbol{x}_t|y) - \gamma \nabla \log p(\boldsymbol{x}_t) && (169)\\
&= \underbrace{\gamma \nabla \log p(\boldsymbol{x}_t|y)}_{\text{conditional score}} + \underbrace{(1-\gamma) \nabla \log p(\boldsymbol{x}_t)}_{\text{unconditional score}} && (170)
\end{aligned}$$

Once again, γ is a term that controls how much our learned conditional model cares about the conditioning information. When $\gamma = 0$, the learned conditional model completely ignores the conditioner and learns an unconditional diffusion model. When $\gamma = 1$, the model explicitly learns the vanilla conditional distribution without guidance. When $\gamma > 1$, the diffusion model not only prioritizes the conditional score function, but also moves in the direction away from the unconditional score function. In other words, it reduces the probability of generating samples that do not use conditioning information, in favor of the samples that explicitly do. This also has the effect of decreasing sample diversity at the cost of generating samples that accurately match the conditioning information.

Because learning two separate diffusion models is expensive, we can learn both the conditional and unconditional diffusion models together as a singular conditional model; the unconditional diffusion model can be queried by replacing the conditioning information with fixed constant values, such as zeros. This is essentially performing random dropout on the conditioning information. Classifier-Free Guidance is elegant because it enables us greater control over our conditional generation procedure while requiring nothing beyond the training of a singular diffusion model.

Closing

Allow us to recapitulate our findings over the course of our explorations. First, we derive Variational Diffusion Models as a special case of a Markovian Hierarchical Variational Autoencoder, where three key assumptions enable tractable computation and scalable optimization of the ELBO. We then prove that optimizing a VDM boils down to learning a neural network to predict one of three potential objectives: the original source image from any arbitrary noisification of it, the original source noise from any arbitrarily noisified image, or the score function of a noisified image at any arbitrary noise level. Then, we dive deeper into what it means to learn the score function, and connect it explicitly with the perspective of Score-based Generative Modeling. Lastly, we cover how to learn a conditional distribution using diffusion models.

In summary, diffusion models have shown incredible capabilities as generative models; indeed, they power the current state-of-the-art models on text-conditioned image generation such as Imagen and DALL-E 2. Furthermore, the mathematics that enable these models are exceedingly elegant. However, there still remain a few drawbacks to consider:

- It is unlikely that this is how we, as humans, naturally model and generate data; we do not generate samples as random noise that we iteratively denoise.
- The VDM does not produce interpretable latents. Whereas a VAE would hopefully learn a structured latent space through the optimization of its encoder, in a VDM the encoder at each timestep is already given as a linear Gaussian model and cannot be optimized flexibly. Therefore, the intermediate latents are restricted as just noisy versions of the original input.
- The latents are restricted to the same dimensionality as the original input, further frustrating efforts to learn meaningful, compressed latent structure.
- Sampling is an expensive procedure, as multiple denoising steps must be run under both formulations. Recall that one of the restrictions is that a large enough number of timesteps T is chosen to ensure the final latent is completely Gaussian noise; during sampling we must iterate over all these timesteps to generate a sample.

As a final note, the success of diffusion models highlights the power of Hierarchical VAEs as a generative model. We have shown that when we generalize to *infinite* latent hierarchies, even if the encoder is trivial and the latent dimension is fixed and Markovian transitions are assumed, we are still able to learn powerful models of data. This suggests that further performance gains can be achieved in the case of general, deep HVAEs, where complex encoders and semantically meaningful latent spaces can be potentially learned.

Acknowledgments: I would like to acknowledge Josh Dillon, Yang Song, Durk Kingma, Ben Poole, Jonathan Ho, Yiding Jiang, Ting Chen, Jeremy Cohen, and Chen Sun for reviewing drafts of this work and providing many helpful edits and comments. Thanks so much!

References

[1] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114, 2013.

[2] Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. Advances in neural information processing systems, 29, 2016.

[3] Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. Ladder variational autoencoders. Advances in neural information processing systems, 29, 2016.

[4] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In International Conference on Machine Learning, pages 2256–2265. PMLR, 2015.

[5] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. Advances in Neural Information Processing Systems, 33:6840–6851, 2020.

[6] Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. Advances in neural information processing systems, 34:21696–21707, 2021.

[7] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S Sara Mahdavi, Rapha Gontijo Lopes, et al. Photorealistic text-to-image diffusion models with deep language understanding. arXiv preprint arXiv:2205.11487, 2022.

[8] Bradley Efron. Tweedie's formula and selection bias. Journal of the American Statistical Association, 106(496): 1602–1614, 2011.

[9] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. Advances in Neural Information Processing Systems, 32, 2019.

[10] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. arXiv preprint arXiv:2011.13456, 2020.

[11] Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. Advances in neural information processing systems, 33:12438–12448, 2020.

[12] Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. A tutorial on energy-based learning. Predicting structured data, 1(0), 2006.

[13] Yang Song and Diederik P Kingma. How to train your energy-based models. arXiv preprint arXiv:2101.03288, 2021.

[14] Aapo Hyvärinen and Peter Dayan. Estimation of non-normalized statistical models by score matching. Journal of Machine Learning Research, 6(4), 2005.

[15] Saeed Saremi, Arash Mehrjou, Bernhard Schölkopf, and Aapo Hyvärinen. Deep energy estimator networks. arXiv preprint arXiv:1805.08306, 2018.

[16] Yang Song, Sahaj Garg, Jiaxin Shi, and Stefano Ermon. Sliced score matching: A scalable approach to density and score estimation. In Uncertainty in Artificial Intelligence, pages 574–584. PMLR, 2020.

[17] Pascal Vincent. A connection between score matching and denoising autoencoders. Neural computation, 23(7): 1661–1674, 2011.

[18] Jonathan Ho, Chitwan Saharia, William Chan, David J Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded diffusion models for high fidelity image generation. J. Mach. Learn. Res., 23:47–1, 2022.

[19] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. arXiv preprint arXiv:2204.06125, 2022.

[20] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. Advances in Neural Information Processing Systems, 34:8780–8794, 2021.

[21] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications, 2021.