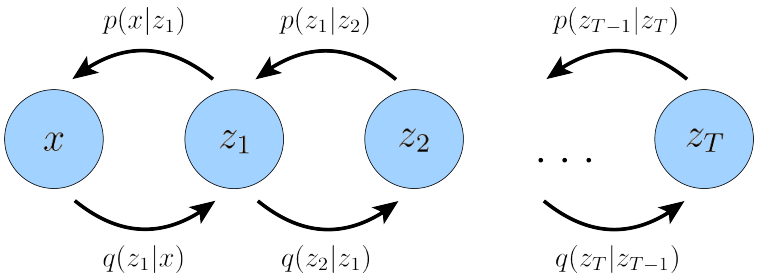
Understanding Diffusion Models: A Unified Perspective

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

August 26, 2022

Contents



Markovian HVAE:

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

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

==> ELBO :

$$\begin{aligned}\log p(\boldsymbol{x}) &= \log \int p(\boldsymbol{x}, \boldsymbol{z}_{1:T}) d\boldsymbol{z}_{1:T} \\ &\geq \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}_{1:T}|\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{x}, \boldsymbol{z}_{1:T})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}_{1:T}|\boldsymbol{x})}\right]\end{aligned} \tag{25}$$

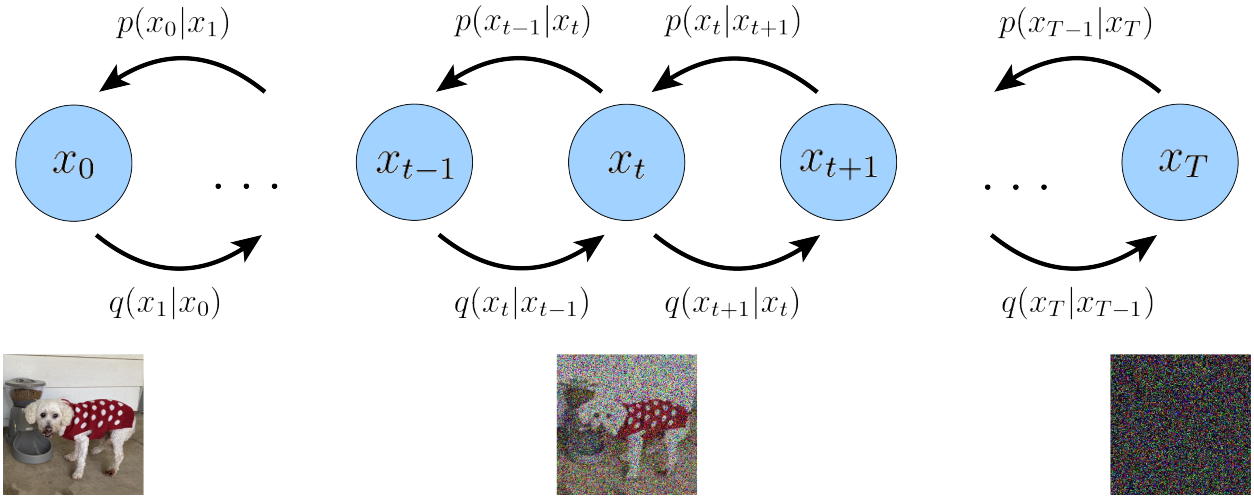
==>

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

Variational Diffusion Models

a VDM [4, 5, 6] i s a MHVAE wi th

- Th e latent di mensi on i s exactly equal to th e data di mensi on
- Th e structure of th e latent encoder at each ti mestep i s not learned; i t i s pre-defined as a li near Gaussi an model. In oth er words, i t i s a Gaussi an di stri b uti on centered around th e output of th e previ ous ti mestep
- Th e Gaussi an parameters of th e latent encoders vary over ti me i n such a way th at th e di stri b uti on of th e latent at final ti mestep T i s a standard Gaussi an



Furthermore, we explicitly maintain the Markov property between hierarchical transitions, from a standard HVAE

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 HVAE posterior (Eq 24):

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

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 time step is not learned; it is fixed as a linear Gaussian model, where the mean and standard deviation can be set before hand as hyperparameters or learned as parameters [5]. 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 the (potentially learnable) coefficient that can vary with the hierarchical depth t 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 evolves over time according to a fixed or learnable schedule structured such that the distribution of the final $p(\boldsymbol{x}_T)$ is a standard Gaussian. We can then update the joint distribution of a Markovian HVAE (Eq 23) to now 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 ϕ, 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 conditional $p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$, so that we can simulate new data. After optimizing the VDM, the sampling procedure is as simple as sampling Gaussian noise $p(\boldsymbol{x}_T)$ and iteratively running the denoising transition $p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ for T steps to generate a novel $\boldsymbol{x}_0$.

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

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

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

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

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

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

The derivedform of the ELBO can be interpreted:

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.

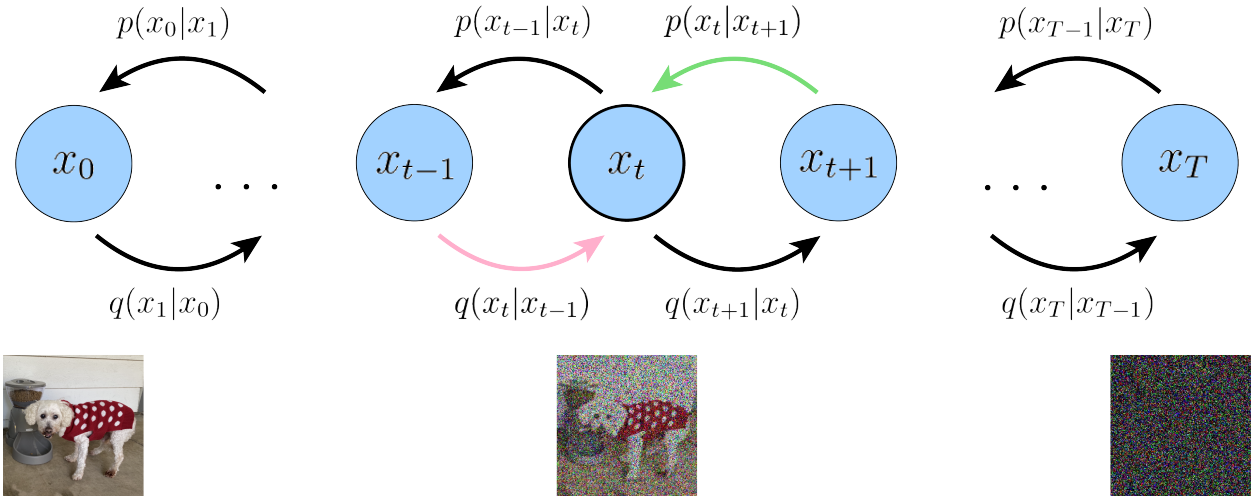


Figure 4: Under our first derivation, a VDM can be optimized by ensuring that for every intermediate, the posterior from the latent above, $p_\theta(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})$ matches the Gaussian corruption of the latent before it $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$. In this figure,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

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 consistency terms, the final estimated value of the ELBO may have high variance. For lower

Let us instead try to derive a form for our ELBO where each term is computed as an expectation over only one or two timesteps. The key insight that we can rewrite the 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:

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

Armed with (46), we can retry the derivation, resuming from the ELBO in Eq 37:

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

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

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

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

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

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 variable. This formulation also has an elegant interpretation:

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. With a sufficiently large T, this term will equal 0 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 within the process of both ELBO derivations (Equation 45 and 58), only the Markov assumption is used; as a result these formulae will hold true for any arbitrarily Markovian HVAE.

When $T=1$, both of the ELBO interpretations for our VDM exactly recreate the ELBO equation of a vanilla VAE, as in Eq(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 $D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))$ is difficult to minimize for arbitrary posteriors in arbitrarily complex MHVAEs due to the added complexity of simultaneously learning the encoder, in a VDM we can leverage the Gaussian transition assumption to make optimization tractable:

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

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 assumptions regarding encoder transitions (31), what remains is deriving forms of $q(\boldsymbol{x}|\boldsymbol{x}_0)_t$ 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 written:

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

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

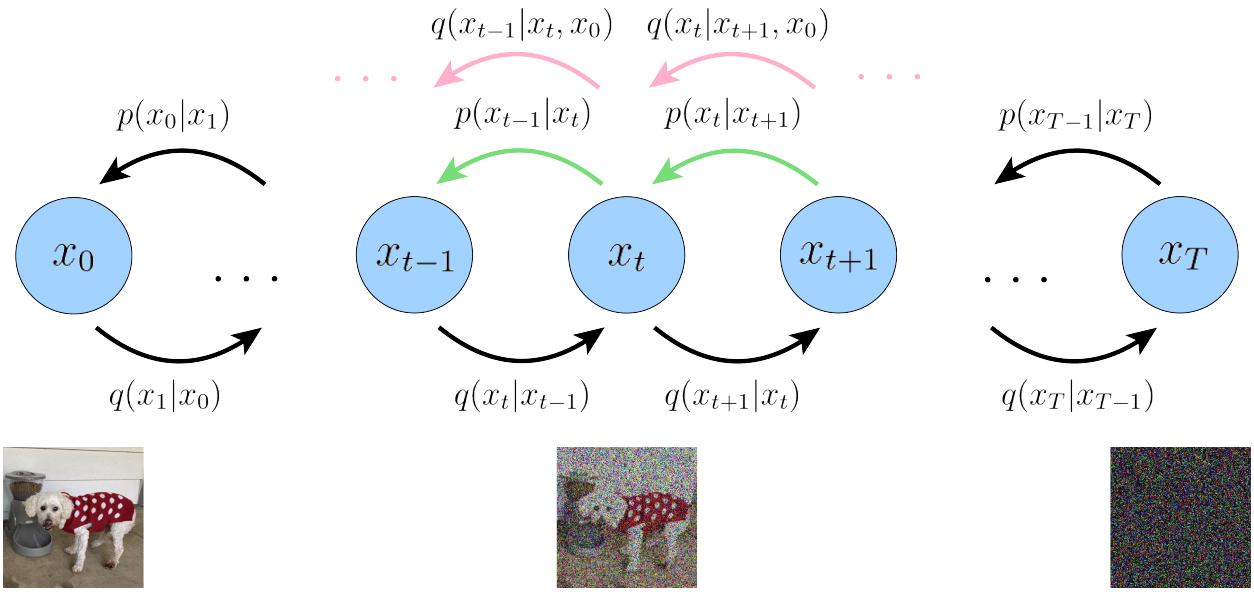


Figure 5: Depicted is an alternate, lower-variance method to optimize a VDM. In it, we compute the form of ground-truth denoising step $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$ using Bayes rule, and minimize its KL Divergence with our approximate denoising step $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$. This is shown 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 T 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:

$$\boldsymbol{x}_t = \sqrt{\alpha_t}\boldsymbol{x}_{t-1} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_{t-1}^* \tag{61}$$

$$= \sqrt{\alpha_t}\left(\sqrt{\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{1-\alpha_{t-1}}\boldsymbol{\epsilon}_{t-2}^*\right) + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_{t-1}^* \tag{62}$$

$$\tag{63}$$

$$= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\boldsymbol{\epsilon}_{t-2} \tag{64}$$

$$= \dots \tag{65}$$

$$= \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_0 \tag{66}$$

$$\sim \mathcal{N}(\boldsymbol{x}_t; \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0, (1-\bar{\alpha}_t)\,\mathbf{I}) \tag{67}$$

$$\tag{68}$$

$$\tag{69}$$

$$\tag{70}$$

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:

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

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

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

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

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

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

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 $\boldsymbol{\Sigma}_q(t) = \sigma_q^2(t)\mathbf{I}$, 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 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 KL Divergence between two Gaussian distributions

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

==>

$$\begin{aligned}
&\arg\min_{\boldsymbol{\theta}} D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)\parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))\\
&=\arg\min_{\boldsymbol{\theta}} D_{\mathrm{KL}}(\mathcal{N}(\boldsymbol{x}_{t-1};\boldsymbol{\mu}_q,\boldsymbol{\Sigma}_q(t))\parallel\mathcal{N}(\boldsymbol{x}_{t-1};\boldsymbol{\mu}_{\boldsymbol{\theta}},\boldsymbol{\Sigma}_q(t))) &(87)\\
&=\arg\min_{\boldsymbol{\theta}}\frac{1}{2}\left[(\boldsymbol{\mu}_{\boldsymbol{\theta}}-\boldsymbol{\mu}_q)^T\boldsymbol{\Sigma}_q(t)^{-1}(\boldsymbol{\mu}_{\boldsymbol{\theta}}-\boldsymbol{\mu}_q)\right] &(90)
\end{aligned}$$

$$=\arg\min_{\boldsymbol{\theta}}\frac{1}{2\sigma_q^2(t)}\left[\|\boldsymbol{\mu}_{\boldsymbol{\theta}}-\boldsymbol{\mu}_q\|_2^2\right] \tag{92}$$

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 $\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_t,t)$ that match $\boldsymbol{\mu}_q(\boldsymbol{x}_t,\boldsymbol{x}_0)$, 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 simplifies to:

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

$$=\arg\min_{\boldsymbol{\theta}}\frac{1}{2\sigma_q^2(t)}\frac{\bar{\alpha}_{t-1}(1-\alpha_t)^2}{(1-\bar{\alpha}_t)^2}\left[\|\hat{\boldsymbol{x}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t,t)-\boldsymbol{x}_0\|_2^2\right] \tag{99}$$

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. 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 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:

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

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

SNR $= \frac{\mu^2}{\sigma^2}$, we can write the SNR at each time step t:

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

Then, Eq 108 (and 99) ==>

Precise(t-1) - Precise(t)

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

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 110 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 noised $\boldsymbol{x}_t$ from input $\boldsymbol{x}_0$ using the reparameterization trick, as derived in Equation 69.

Three Equivalent Interpretations

As we previously proved, a VDM 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 sinusoidal time index t. However, $\boldsymbol{x}_0$ has two other equivalent parameterizations, which leads to two further interpretations of a VDM.

1, we can utilize the reparameterization trick $q(\boldsymbol{x}_t|\boldsymbol{x}_0)$, (69) ==>

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

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

$$\boldsymbol{\mu}_q(\boldsymbol{x}_t, \boldsymbol{x}_0) = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\boldsymbol{x}_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\boldsymbol{x}_0}{1-\bar{\alpha}_t} \tag{116}$$

$$= \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\boldsymbol{x}_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\frac{\boldsymbol{x}_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_0}{\sqrt{\bar{\alpha}_t}}}{1-\bar{\alpha}_t} \tag{117}$$

$$= \frac{1}{\sqrt{\alpha_t}}\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}\boldsymbol{\epsilon}_0 \tag{119}$$

2, we can set our approximated 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:

$$\arg\min_{\boldsymbol{\theta}} D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))$$

$$= \arg\min_{\boldsymbol{\theta}} D_{\mathrm{KL}}(\mathcal{N}(\boldsymbol{x}_{t-1}; \boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q(t)) \parallel \mathcal{N}(\boldsymbol{x}_{t-1}; \boldsymbol{\mu}_{\boldsymbol{\theta}}, \boldsymbol{\Sigma}_q(t))) \tag{126}$$

$$= \arg\min_{\boldsymbol{\theta}} \frac{1}{2\sigma_q^2(t)} \frac{(1-\alpha_t)^2}{(1-\bar{\alpha}_t)\alpha_t} \left[\|\boldsymbol{\epsilon}_0 - \hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)\|_2^2\right] \tag{130}$$

Here, $\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$ is a neural network that learns to predict the source noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\epsilon}; \mathbf{0}, \mathbf{I})$ 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].

3 common interpretation of VDMs, we appeal to Tweedie's Formula [8]. In English, Tweedie's Formula states that the true mean of an EFD, given samples drawn from it, can be estimated by the MLE 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, since if observed samples all lie on one end of the underlying distribution, the negative score becomes large and corrects the naive MLE of the samples towards the true mean.

for $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{z}; \boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z)$, Tweedie's Formula:

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

apply it to predict the true posterior mean of $\boldsymbol{x}_t$ given its samples. From Eq70, we know

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

Then, Tweedie's Formula ==>

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

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 Eq133 into our ground-truth denoising transition mean $\boldsymbol{\mu}_q(\boldsymbol{x}_t, \boldsymbol{x}_0)$ once again

==>

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

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

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

==> approximate denoising transition mean

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

==>

$$\arg\min_{\boldsymbol{\theta}} D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)) \tag{144}$$

$$= \arg\min_{\boldsymbol{\theta}} \frac{1}{2\sigma_q^2(t)} \left[\left\| \frac{1}{\sqrt{\alpha_t}}\boldsymbol{x}_t + \frac{1-\alpha_t}{\sqrt{\alpha_t}}\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) - \frac{1}{\sqrt{\alpha_t}}\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{\alpha_t}}\nabla \log p(\boldsymbol{x}_t) \right\|_2^2 \right] \tag{145}$$

$$= \arg\min_{\boldsymbol{\theta}} \frac{1}{2\sigma_q^2(t)} \frac{(1-\alpha_t)^2}{\alpha_t} \left[\|\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) - \nabla \log p(\boldsymbol{x}_t)\|_2^2 \right] \tag{148}$$

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.

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 (133) with the reparameterization trick (115):

$$\boldsymbol{x}_0 = \frac{\boldsymbol{x}_t + (1-\bar{\alpha}_t)\nabla \log p(\boldsymbol{x}_t)}{\sqrt{\bar{\alpha}_t}} = \frac{\boldsymbol{x}_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_0}{\sqrt{\bar{\alpha}_t}} \tag{149}$$

$$\therefore (1-\bar{\alpha}_t)\nabla \log p(\boldsymbol{x}_t) = -\sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_0 \tag{150}$$

$$\nabla \log p(\boldsymbol{x}_t) = -\frac{1}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_0 \tag{151}$$

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 time steps t and minimizing the norm of the predicted output with the ground truth target.

Score-based Generative Models

We have shown that a VDM 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 does not necessarily provide us with great intuition into what exactly the score function is or why it is worth modeling. Fortunately, we can look to another class of generative models, SBGMs [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 SbGM formulation, allowing us to flexibly switch between these two interpretations as we will

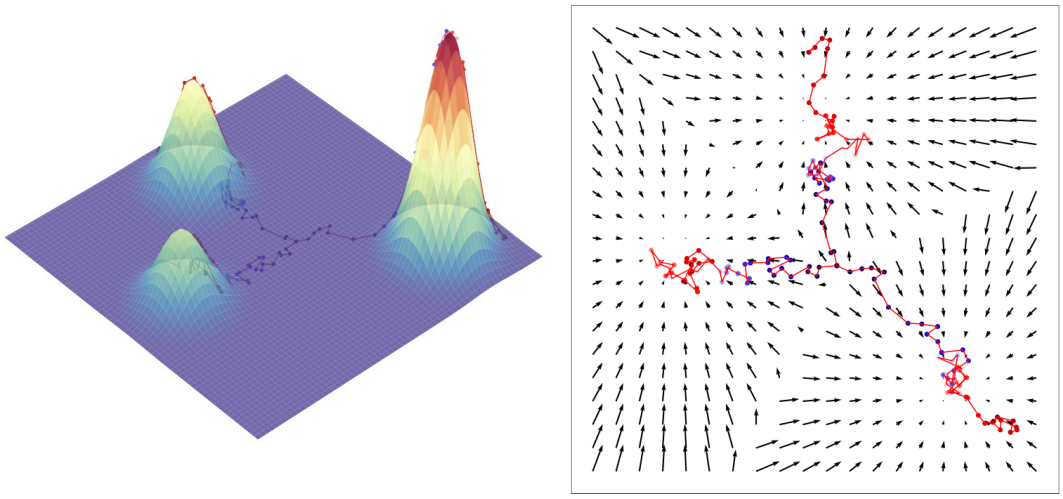


Figure 6: Visualization of three random sampling trajectories generated with ALD, all starting from the same initialization point, for a MoG. The left figure plots these sampling trajectories on a 3D contour, while the right figure plots these sampling trajectories against the ground-truth score function. From the same initialization point, we are able to generate samples from different modes due to the stochastic noise term in the ALD 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 with score function makes sense, we take a detour and revisit energy-based models [12, 13]:

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

neural network $\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x})$ to learn the score function $\nabla \log p(\boldsymbol{x})$:

$$\begin{aligned} \nabla_{\boldsymbol{x}} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}) &= -\nabla_{\boldsymbol{x}} f_{\boldsymbol{\theta}}(\boldsymbol{x}) \\ &\approx \boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}) \end{aligned} \tag{156}$$

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:

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

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 points 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 Eq 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 SM [14-17] have been derived to minimize this Fisher divergence without knowing the ground truth score, and can be optimized with SGB.

Collectively, learning to represent a distribution as a score function and using it to generate samples through MCMC techniques, such as Langevin dynamics, is known as SbGM [9, 10, 11].
There are three remaining 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 sub optimal as well, or require many more iterations to converge on an accurate output.

Lastly, LDs sampling may not mix, even if it is performed using the ground truth scores. Suppose that the true data distribution is a mixture:

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

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 two plots of Figure 6 is agnostic of the different weights between the three distributions; LD 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 MoGs.

It turns out that these three drawbacks can be simultaneously addressed by adding multiple levels of Gaussian noise to the data. 1, 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. 2, adding large Gaussian noise will increase the area each mode covers in the data distribution, adding more training signal in low density regions. 3, 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 SM 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 the noise level at time t. Note that this objective almost exactly matches the objective derived in Equation 18 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 $T, T-1, ..., 2, 1$ 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 time, as the step size over time, the samples eventually converge into a true mode. This is directly analogous to the sampling procedure performed in the MHVAE interpretation of a VDM, where a randomly initialized data vector is iteratively refined over decreasing noise levels.

Therefore, we have established an explicit connection between VDMs and SbGMs, 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 MHVAE 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 sbGM 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 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 SOTA 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 Eq 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 DM, 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 $\nabla \log p(\boldsymbol{x}_t|y)$, the score of the conditional model, at arbitrary noise level. Recall that ∇ is shorthand for $\nabla_{\boldsymbol{x}_t}$ in the interest of brevity. By Bayes rule, we can derive the following equivalent form:

$$\nabla \log p(\boldsymbol{x}_t|y) = \nabla \log \left(\frac{p(\boldsymbol{x}_t)p(y|\boldsymbol{x}_t)}{p(y)} \right) \tag{163}$$

$$= \nabla \log p(\boldsymbol{x}_t) + \nabla \log p(y|\boldsymbol{x}_t) - \nabla \log p(y) \tag{164}$$

$$= \underbrace{\nabla \log p(\boldsymbol{x}_t)}_{\text{unconditional score}} + \underbrace{\nabla \log p(y|\boldsymbol{x}_t)}_{\text{adversarial gradient}} \tag{165}$$

where $\text{grad } \log p(y) \mid \boldsymbol{x}_t = 0$.

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 :

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

Intuitively, when $\gamma = 0$ the conditional DM learns to ignore the conditioning information entirely, and when γ is large the conditional DM 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 adhocalongside the diffusion model.

Classifier-Free Guidance

In CFG [21], the authors ditch the training of a separate classifier model in favor of an unconditional DM and a conditional DM. To derive the score function under CFG, we can first rearrange Eq 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, Eq 166, ==>

$$\nabla \log p(\boldsymbol{x}_t|y) = \nabla \log p(\boldsymbol{x}_t) + \gamma \left(\nabla \log p(\boldsymbol{x}_t|y) - \nabla \log p(\boldsymbol{x}_t) \right) \tag{168}$$

$$= \nabla \log p(\boldsymbol{x}_t) + \gamma \nabla \log p(\boldsymbol{x}_t|y) - \gamma \nabla \log p(\boldsymbol{x}_t) \tag{169}$$

$$= \underbrace{\gamma \nabla \log p(\boldsymbol{x}_t|y)}_{\text{conditional score}} + \underbrace{(1-\gamma) \nabla \log p(\boldsymbol{x}_t)}_{\text{unconditional score}} \tag{170}$$

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 is 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 *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.