Project - Image Generation Models from a Probabilistic Perspective

Amro Alabsi Aljundi (nmm2uy) — Nate Kimball (tma5gv) — Patrick Soga (zqe3cg)

April 2024

1 Proposal

Diffusion models have demonstrated impressive image-generation abilities, exceeding those of techniques like generative adversarial networks (GAN) and autoregressive models. We will investigate Diffusion Models from a probabilistic standpoint. We will begin by explaining the class of Variational Bayesian Methods and their relationship with diffusion models. DPMs (diffusion models) can be viewed as a type of variational autoecoder (VAE) [Kingma and Welling, 2014, Rezende et al., 2014].

Specifically, we will first explain the inner workings of variational autoencoders (VAEs) [Kingma and Welling, 2014] for image generation, and then introduce the essential concepts and mathematical background necessary for diffusion models [Ho et al., 2020] as originally defined. Finally, we will connect VAEs with diffusion models from a Bayesian variational perspective, explaining and analyzing work by Kingma et al. [2021] on variational diffusion models and investigating the improvements on image generation quality that diffusion can provide.

2 Auto-Encoding Variational Bayes

Kingma and Welling introduce a stochastic variational inference and learning algorithm for approximating intractable posteriors. With a reparameterization of the variational lower bound, they produce a simpler and more stable lower bound that can be optimized with standard stochastic gradient methods. To do posterior inference on i.i.d. datasets with continuous latent variables per datapoint, they fit an approximate inference model to the intractable posterior using their lower bound estimator. Essentially, they are trying to approximate the distribution of latent variables, underlying lower-dimensional information in the data. They do this by finding an optimizable lower bound on the log-likelihood of the data. In optimizing this lower bound, they learn the parameters for a recognition model, an approximation of the latent posterior, and a generative model, the likelihood of the data given the latent variables. They also propose the Variational Autoencoder architecture for applying their loss function to image generation.

2.1 Problem Scenario

You have a dataset $\mathcal{D} = \{x^{(i)}\}_{i=1}^{n},\ x^{(i)}$ i.i.d., where each datapoint is associated with a continuous latent variable. We assume each datapoint is generated by the following random process:

1. a value $z^{(i)}$ is generated according to some prior distribution $p_{\theta^*}(z)$
2. a value $x^{(i)}$ is generated from some conditional distribution $p_{\theta^*}(x|z)$

We assume that the prior $p_{\theta^*}(z)$ and likelihood $p_{\theta^*}(x|z)$ come from parametric families of distributions of $p_\theta(z)$ and $p_\theta(x|z)$, and that their PDFs are differentiable almost everywhere w.r.t. both θ and z. Commonly, this means assuming our distribution can be represented by a specific neural network. This simplifies our search space. θ^* and $z^{(i)}$ are unknown to us.

We want to find an efficient algorithm that avoids the following obstacles:

1. Intractability: The integral of the marginal likelihood $p_\theta(x) = \int p_\theta(z)p_\theta(x|z)dz$ is intractable, and the true posterior density $p_\theta(z|x) = p_\theta(x|z)p_\theta(z)/p_\theta(x)$ is intractable so the EM algorithm cannot be used. Intractability is common when working with moderately complicated likelihood functions $p_\theta(x|z)$, e.g. a non linear neural network

2. Large dataset: With a large dataset, batch optimization is too costly, and sampling-based solutions, e.g. Monte Carlo EM, are too slow. Minibatch or single datapoint parameter updates are preferable.

The authors propose a solution to the following problems in this scenario:

1. $\hat{\theta}$: efficient approximate ML or MAP estimation for the parameters θ.

2. $p_\theta(z|x)$: efficient approximate posterior inference of the latent variable z given an observed value x and parameters θ, aka encoding.

3. $p_\theta(x)$: efficient approximate marginal inference of the variable x aka synthesis.

2.2 The Variational Method

Variational methods allow us to avoid the intractable integral by transforming the statistical inference problem into an optimization problem. We will learn an approximation of the true posterior, which we call a recognition model, denoted by $q_\phi(z|x)$. We introduce a method for learning the recognition model parameters ϕ jointly with the generative model parameters θ. Our goal is to find the best approximation of $p_\theta(z|x)$, so we start by minimizing the KL-divergence between $q_\phi(z|x)$. and $p_\theta(z|x)$.

$$\phi^*, \theta^* = \underset{\phi,\theta}{\operatorname{argmin}}\, D_{KL}(q_\phi(z|x)||p_\theta(z|x)) \tag{1}$$

We further reduce this term to derive what is known as the *evidence lower bound* (ELBO).

$$\begin{aligned}
D_{KL}(q_\phi(z|x^{(i)})||p_\theta(z|x^{(i)})) &= \int_z q_\phi(z|x^{(i)}) \log \frac{q_\phi(z|x^{(i)})}{p_\theta(z|x^{(i)})} dz && (2)\\
&= \int_z q_\phi(z|x^{(i)}) \log q_\phi(z|x^{(i)}) dz - \int_z q_\phi(z|x) \log p_\theta(z|x^{(i)}) dz && (3)\\
&= \mathbb{E}_{q_\phi(z|x)}[\log(q_\phi(z|x^{(i)}))] - \mathbb{E}_{q_\phi(z|x)}[\log(p_\theta(z|x^{(i)}))] && (4)\\
&= \mathbb{E}_{q_\phi(z|x)}[\log(q_\phi(z|x^{(i)}))] - (\mathbb{E}_{q_\phi(z|x)}[\log(p_\theta(x^{(i)}, z))] - \mathbb{E}_{q_\phi(z|x)}[\log p(x^{(i)}]) && (5)\\
&= \mathbb{E}_{q_\phi(z|x)}[\log(q_\phi(z|x^{(i)}))] - \mathbb{E}_{q_\phi(z|x)}[\log(p_\theta(x^{(i)}, z))] + \log p(x^{(i)}) && (6)\\
&= -\text{ELBO} + \log p_\theta(x^{(i)}) && (7)
\end{aligned}$$

The term $\log p(x^{(i)})$ is a constant for a particular data point, so it can be ignored during the optimization process. We can rewrite the evidence lower bound as follows:

$$\begin{aligned}
\text{ELBO} &= -\mathbb{E}_{q_\phi(z|x)}[\log(q_\phi(z|x^{(i)}))] + \mathbb{E}_{q_\phi(z|x)}[\log(p_\theta(x^{(i)}, z))] && \text{from eq. (6)} \quad (8)\\
&= -\mathbb{E}_{q_\phi(z|x)}[\log(q_\phi(z|x^{(i)}))] + \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(z)] + \mathbb{E}_{q_\phi(z|x)}[\log(p_\theta(x^{(i)}|z))] && (9)\\
&= -D_{KL}(q_\phi(z|x^{(i)})||p_\theta(z)) + \mathbb{E}_{q_\phi(z|x)}[\log(p_\theta(x^{(i)}|z))] && (10)
\end{aligned}$$

Using equation 7, we can show that the evidence lower bound is a lower bound on the marginal log-likelihood of the data $\log p_\theta(x^{(i)})$:

$$\log p_\theta(x^{(i)}) = \text{ELBO} + D_{KL}(q_\phi(z|x^{(i)})||p_\theta(z|x^{(i)})) \tag{11}$$

$$\log p_\theta(x^{(i)}) \geq \text{ELBO} \tag{12}$$

We can reformulate the original optimization process as:

$$\phi^*, \theta^* = \underset{\phi,\theta}{\text{argmin}}\ D_{KL}(q_\phi(z|x)||p_\theta(z|x)) \tag{13}$$

$$= \underset{\phi,\theta}{\text{argmax}}\ \text{ELBO} \tag{14}$$

$$= \underset{\phi,\theta}{\text{argmax}} \left[-D_{KL}(q_\phi(z|x)||p_\theta(z)) + \mathbb{E}_{q_\phi(z|x)}[\log(p_\theta(x^{(i)}|z))] \right] \tag{15}$$

2.3 Optimizing the Lower bound

We will define our likelihood estimator as the evidence lower bound.

$$\log p_\theta(x^{(i)}) \geq \mathcal{L}(\theta, \phi; x^{(i)}) = -D_{KL}(q_\phi(z|x)||p_\theta(z)) + \mathbb{E}_{q_\phi}[\log(p_\theta(x^{(i)}|z))] \tag{16}$$

We want to optimize the lower bound w.r.t. both the variational parameters ϕ and generative parameters θ. However, this requires sampling $q_\phi(z|x)$ to compute each latent. This process is generally not differentiable, so the gradient w.r.t. ϕ is problematic. One way to estimate this gradient is to use a Monte Carlo gradient estimator:

$$\nabla_\phi \mathbb{E}_{q_\phi(z)}[f(z)] = \mathbb{E}_{q_\phi(z)}[f(z)\nabla_{q_\phi(z)} \log q_\phi(z)] \simeq \frac{1}{L}\sum_{l=1}^{L} f(z)\nabla_{q_\phi(z^{(l)})} \log q_\phi(z^{(l)}) \tag{17}$$
$$\text{where } z^{(l)} \sim q_\phi(z|x^{(i)})$$

This gradient estimator exhibits very high variance and is impractical in many cases, because q_ϕ can be an arbitrarily complicated distribution that is hard to sample. Since we sample from $q_\phi(z|x^{(i)})$, the gradients of the MC estimate term in the loss can only be back-propagated up to the sampled latent variable $z^{(l)}$. The gradient of $z^{(l)}$ w.r.t. ϕ can not be computed. Thus, only the KL divergence term can optimize ϕ leading to an unstable suboptimal learning process.

2.3.1 The SGVB estimator and AEVB algorithm

To resolve this issue, we employ the *reparameterization trick*, which allows us to backpropagate loss w.r.t. ϕ through the decoder and encoder by treating q_ϕ as a differentiable deterministic function by isolating randomness to an auxiliary variable. Under certain mild conditions for $q_\phi(z|x)$, we can reparameterize each latent $\tilde{z} \sim q_\phi(z|x)$ using a differentiable transformation $g_\phi(\epsilon, x)$ of an auxiliary noise variable ϵ:

$$\tilde{z} = g_\phi(\epsilon, x) \text{ with } \epsilon \sim p(\epsilon) \tag{18}$$

We can now rewrite samples from $q_\phi(z|x)$ in terms of g_ϕ and form Monte Carlo estimates of expectations

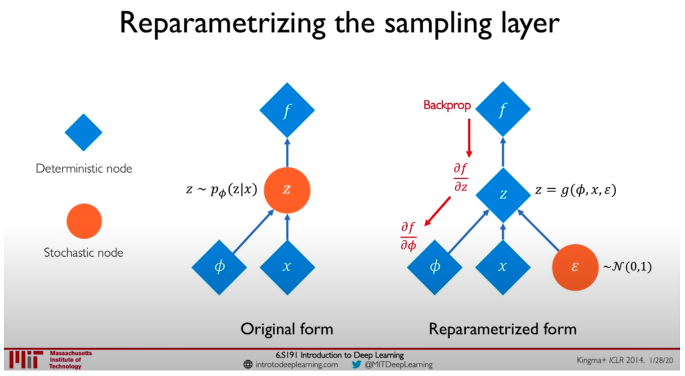
Reparametrizing the sampling layer
Deterministic node
Stochastic node
$z \sim p_\phi(z|x)$
z
f
ϕ
x
Original form
Backprop
$\frac{\partial f}{\partial z}$
$\frac{\partial f}{\partial \phi}$
$z = g(\phi, x, \varepsilon)$
ε
$\sim \mathcal{N}(0,1)$
Reparametrized form
Massachusetts Institute of Technology
6.S191 Introduction to Deep Learning
introtodeeplearning.com
@MITDeepLearning
Kingma+ ICLR 2014. 1/28/20

Figure 1: Reparameterization Trick Visualization

as follows:

$$\mathbb{E}_{q_\phi(z|x^{(i)})}[f(z)] = \mathbb{E}_{p(\epsilon)}[f(g_\phi(\epsilon, x^{(i)}))] \simeq \frac{1}{L}\sum_{l=1}^{L} f(g_\phi(\epsilon^{(l)}, x^{(i)}))$$
$$\text{where } \epsilon^{(l)} \sim p(\epsilon) \tag{19}$$

By applying this technique to the variational lower bound eq. (8), we get our generic Stochastic Gradient Variational Bayes (SGVB) estimator $\tilde{\mathcal{L}}^A(\theta, \phi; x^{(i)}) \simeq \mathcal{L}(\theta, \phi; x^{(i)})$:

$$\tilde{\mathcal{L}}^A(\theta, \phi; x^{(i)}) = \frac{1}{L}\sum_{l=1}^{L} \log p_\theta(x^{(i)}, z^{(i,l)}) - \log q_\phi(z^{(i,l)}|x^{(i)})$$
$$\text{where } \ z^{(i,l)} = g_\phi(\epsilon^{(i,l)}, x^{(i)}) \ \text{ and } \ \epsilon^{(l)} \sim p(\epsilon) \tag{20}$$

Often, the KL-Divergence $D_{KL}(q_\phi(z|x^{(i)})||p_\theta(z))$ from eq. (10) can be integrated analytically such that only the expected reconstruction error $\mathbb{E}_{q_\phi}[\log(p_\theta(x^{(i)}|z))]$ requires estimation by sampling. The KL-divergence term can then be interpreted as regularizing ϕ, encouraging the approximate posterior to be close to the prior $p_\theta(z)$. This leads to a new lower-variance SGVB estimator $\tilde{\mathcal{L}}^B(\theta, \phi; x^{(i)}) \simeq \mathcal{L}(\theta, \phi; x^{(i)})$:

$$\tilde{\mathcal{L}}^B(\theta, \phi; x^{(i)}) = -D_{KL}(q_\phi(z|x^{(i)})||p_\theta(z)) + \frac{1}{L}\sum_{l=1}^{L} \log p_\theta(x^{(i)}|z^{(i,l)})$$
$$\text{where } \ z^{(i,l)} = g_\phi(\epsilon^{(i,l)}, x^{(i)}) \ \text{ and } \ \epsilon^{(l)} \sim p(\epsilon) \tag{21}$$

2.4 Example: Variational Auto-Encoder

For VAEs, we will use a neural network for the probabilistic encoder $q_\phi(z|x)$. Let $p_\theta(x|z)$ be a multivariate Gaussian (for continuous data) or Bernoulli (for binary data) distribution whose parameters are computed from z with an MLP. We assume the true but intractable posterior $p_\theta(z|x)$ takes on an approximate Gaussian form with approximately diagonal covariance:

$$\log q_\phi(z|x^{(i)}) = \log \mathcal{N}(z; \mu^{(i)}, \sigma^{2(i)}\mathbf{I}) \tag{22}$$

where the mean and s.d. of the approximate posterior, $\mu^{(i)}$ and $\sigma^{(i)}$, are outputs of the encoding MLP. We sample from the posterior $z^{(i,l)} \sim q_\phi(z|x^{(i)})$ using $z^{(i,l)} = g_\phi(x^{(i)}) = \mu^{(i)} + \sigma^{(i)} \odot \epsilon^{(l)}$ where $\odot$ stands for element-wise multiplication. We assume our prior over the latent space is the standard normal distribution: $p_\theta(z) = \mathcal{N}(0, \mathbf{I})$. Since both the prior $p_\theta(z)$ and approximate posterior $q_\phi(z|x)$ are Gaussian, we can analytically compute and differentiate the KL divergence. The resulting estimator for VAEs is:

$$\mathcal{L}(\theta, \phi; x^{(i)}) \simeq D_{KL}(\mathcal{N}(\mu^{(i)}, \sigma^{2(i)}\mathbf{I}) \ || \ \mathcal{N}(0, \mathbf{I})) + \frac{1}{L}\sum_{l=1}^{L} \log p_\theta(x^{(i)}|z^{(i,l)})) \tag{23}$$

$$\mathcal{L}(\theta, \phi; x^{(i)}) \simeq \frac{1}{2}\sum_{j=1}^{J}\Big(1 + \log((\sigma_j^{(i)})^2) - (\mu_j^{(i)})^2 - (\sigma^{(i)})^2\Big) + \frac{1}{L}\sum_{l=1}^{L} \log p_\theta(x^{(i)}|z^{(i,l)}) \tag{24}$$
$$\text{where } \ z^{(i,l)} = \mu^{(i)} + \sigma^{(i)} \odot \epsilon^{(l)} \ \text{ and } \ \epsilon^{(l)} \sim \mathcal{N}(0, \mathbf{I})$$

In practice, we only predict $\mu_\theta(z^{(i)}) = \mathbb{E}_{p_\theta}[x|z^{(i)}]$ with the decoder. If we assume our likelihood is gaussian, $p_\theta(x|z) \sim \mathcal{N}(\mu_\theta(z), \Sigma)$, with a diagonal covariance, then the reconstruction loss can be implemented as mean squared error:

$$p_\theta(x|z) = \frac{1}{(2\pi)^{k/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu_\theta)^T \Sigma^{-1}(x - \mu_\theta)\right) \tag{25}$$

$$\log p_\theta(x|z) = -\frac{1}{2}\left[\log|\Sigma| + k\log(2\pi) + (x - \mu_\theta)^\top \Sigma^{-1}(x - \mu_\theta)\right] \tag{26}$$

Since k and Σ are constant with respect to μ_θ, we can ignore them in the optimization process, thus:

$$\mathcal{L}(\theta, \phi; x^{(i)}) \simeq \frac{1}{2}\sum_{j=1}^{J}\Big(1 + \log((\sigma_j^{(i)})^2) - (\mu_j^{(i)})^2 - (\sigma^{(i)})^2\Big) + (x - \mu_\theta(z^{(i)}))^\top (x - \mu_\theta(z^{(i)})) \tag{27}$$
$$\text{where} \;\; z^{(i,l)} = \mu^{(i)} + \sigma^{(i)} \odot \epsilon^{(l)}, \;\; \epsilon^{(l)} \sim \mathcal{N}(0, \mathbf{I}), \;\text{ and }\; \mu_\theta(z^{(i)}) = \mathbb{E}_{p_\theta}[x|z^{(i)}]$$

Intuitively, regardless of the variance of your prediction, you can maximize the likelihood of the correct answer by minimizing the square error between your predictions and the correct answer.

Alternatively, If we are working with black and white images with pixel values between 0 and 1, we can assume $p_\theta(x_m|z) \sim \text{Bernoulli}(\nu_\theta(z)_m)$. That is, each pixel is an independent Bernoulli random variable with mean $\nu_\theta(z)_m$, the outputs of the decoder. We can then represent the log-likelihood as binary cross-entropy.

$$p_\theta(x_m^{(i)}|z^{(i,l)}) = (\nu_\theta(z)_m)^{x_m^{(i)}} + (1 - \nu_\theta(z)_m)^{1 - x_m^{(i)}} \tag{28}$$
$$\log p_\theta(x_m^{(i)}|z^{(i,l)}) = x_m^{(i)} \cdot \log(\nu_\theta(z)_m) + (1 - x_m^{(i)}) \log(1 - \nu_\theta(z)_m) \tag{29}$$
$$\log p_\theta(x^{(i)}|z^{(i,l)}) = \sum_{m=1}^{k} \log p_\theta(x_m^{(i)}|z^{(i,l)}) \tag{30}$$

Thus, we can write the loss as:

$$\mathcal{L}(\theta, \phi; x^{(i)}) \simeq \frac{1}{2}\sum_{j=1}^{J}\Big(1 + \log((\sigma_j^{(i)})^2) - (\mu_j^{(i)})^2 - (\sigma^{(i)})^2\Big) \tag{31}$$
$$+ \sum_{m=1}^{k}\Big(x_m^{(i)} \cdot \log(\nu_\theta(z)_m) + (1 - x_m^{(i)}) \log(1 - \nu_\theta(z)_m)\Big)$$
$$\text{where} \;\; z^{(i,l)} = \mu^{(i)} + \sigma^{(i)} \odot \epsilon^{(l)}, \;\; \epsilon^{(l)} \sim \mathcal{N}(0, \mathbf{I}), \;\text{ and }\; \nu_\theta(z^{(i)}) = \mathbb{E}_{p_\theta}[x|z^{(i)}]$$

Intuitively, this process involves encoding images into a distribution over latent variables with a neural network, sampling from that distribution, then reconstructing the image with another neural network. This architecture resembles a U-net, except, in the bottleneck layer, we output the mean, $\mu^{(i)}$, and standard deviation, $\sigma^{(i)}$ of the latent posterior. This differs from a standard autoencoder which just outputs $\mathbb{E}[z^{(i)}|x^{(i)}]$ for the bottleneck layer. The parameters of these neural networks are trained with the loss function above which has a KL-divergence term that acts as regularization to keep the posterior close to the standard normal distribution. The second term acts as a reconstruction loss to learn the proper encoding. Now, to generate new images, aka sample from $p_\theta(x)$, we can sample from $z \sim \mathcal{N}(0, \mathbf{I})$, then sample from $p_\theta(x|z)$, the generative model.

3 Denoising Diffusion Probabilistic Models

Diffusion models were first introduced by Sohl-Dickstein et al. (2015), however the paper Denoising Diffusion Probabilistic Models (DDPM) made key theoretical and empirical contributions that significantly advanced the field. DDPM presents progress in diffusion probabilistic models, and demonstrates their effectiveness in high quality image synthesis. Diffusion probabilistic models, or diffusion models for short, are a class of latent variable models inspired by considerations from nonequilibrium thermodynamics. A diffusion model is a parameterized Markov chain trained using variational inference to reconstruct samples matching the data after finite time. This summary will explain how diffusion models work from a probabilistic perspective.

3.1 Diffusion Process

3.1.1 Forward Process

A diffusion process takes a data-point from a data distribution $x_0 \sim q(x)$ and adds Gaussian noise in each of T time steps. This is called the forward process. At each step of the Markov chain, we add Gaussian noise to x_{t-1} according to the variance schedule, β_t, producing a new latent variable x_t with the following posterior distribution:

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \mu_t = \sqrt{1-\beta_t}, \Sigma_t = \beta_t \mathbf{I}) \tag{32}$$

The authors of DDPM selected a linear variance schedule increasing from $\beta_1 = 10^{-4}$ to $\beta_T = 0.02$, although newer research has found better alternatives. By applying this transition multiple times, we get the posterior of all latent variables:

$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1}) \tag{33}$$

where $x_{1:T}$ is the states from timestep 1 to T, also called the trajectory.

3.1.2 The reparameterization trick

Similar to the reparameterization trick in Auto-Encoding Variational Bayes (2.3.1), if we isolate the randomness to auxiliary noise variables we can write x_t as a function of x_0 and auxiliary noise.
We define $\alpha_t = 1 - \beta_t,\ \bar{\alpha}_t = \prod_{s=0}^{t} \alpha_s$, and $\epsilon_0, \dots, \epsilon_{t-2}, \epsilon t - 1 \sim \mathcal{N}(0, \mathbf{I})$ and prove recursively that:

$$\begin{aligned}
x_t &= \sqrt{1-\beta_t} x_{t-1} + \sqrt{\beta_t} \epsilon_{t-1} && (34)\\
&= \sqrt{(1-\beta_t)(1-\beta_{t-1})} x_{t-2} + \sqrt{(1-\beta_t)\beta_{t-1}} \epsilon_{t-2} + \sqrt{\beta_t} \epsilon_{t-1} && (35)\\
&= \sqrt{\alpha_t \alpha_{t-1}} x_{t-2} + \sqrt{\alpha_t (1-\alpha_{t-1})} \epsilon_{t-2} + \sqrt{1-\alpha_t} \epsilon_{t-1} && (36)\\
&= \sqrt{\alpha_t \alpha_{t-1}} x_{t-2} + \sqrt{1 - \alpha_t + \alpha_t - \alpha_t \alpha_{t-1}} \epsilon_{t-2} && (37)\\
&= \sqrt{\alpha_t \alpha_{t-1}} x_{t-2} + \sqrt{1 - \alpha_t \alpha_{t-1}} \epsilon_{t-2} && (38)\\
&= \dots && (39)\\
&= \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t} \epsilon_0 && (40)
\end{aligned}$$

We can now sample x_t directly at any arbitrary timestep according to its posterior:

$$x_t \sim q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t} x_0, (1-\bar{\alpha}_t)\mathbf{I}) \tag{41}$$

3.1.3 Reverse Process

If we can learn the reverse distribution $q(x_{t-1}|x_t)$, we can sample x_T randomly and generate a new image. Since as $T \to \infty$, $x_T \to \mathcal{N}(0, \mathbf{I})$, we can sample x_T from the standard normal distribution, run the reverse diffusion process, and generate a sample from the data distribution $q(x_0)$. Similar to Auto-Encoding Variational Bayes, we will approximate q with a parametrized model p_θ (typically a neural network) and we will minimize KL-Divergence between these two distributions. Since $q(x_{t-1}|x_t)$ is Gaussian, we can let p_θ be Gaussian and just parameterize the mean and variance:

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \sum_\theta(x_t, t)) \tag{42}$$

Repeating the reverse process from x_T to x_0, we get the probability of a trajectory:

$$p_\theta(x_{0:T}) = p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t) \tag{43}$$

3.1.4 Evidence Lower Bound

We will try to minimize the divergence between the forward process $q(x_{1:T}|x_0)$ and backward process $p_\theta(x_{0:T})$:

$$\begin{aligned}
L &= D_{KL}(q(x_{1:T}|x_0)||p_\theta(x_{0:T})) && (44)\\
&= \int_{x_{1:T}} q(x_{1:T}|x_0)\log\frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})} && (45)\\
&= \mathbb{E}_{q(x_{1:T}|x_0)}\left[-\log\frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)}\right] && (46)\\
&= \mathbb{E}_q\left[-\log\frac{p_\theta(x_T)\prod_{t=1}^T p_\theta(x_{t-1}|x_t)}{\prod_{t=1}^T q(x_t|x_{t-1})}\right] && (47)\\
&= \mathbb{E}_q\left[-\log p_\theta(x_T)-\sum_{t=1}^T\log\frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1})}\right] && (48)\\
&= \mathbb{E}_q\left[-\log p_\theta(x_T)-\sum_{t=1}^T\log\frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1})}\right] && (49)\\
&= \mathbb{E}_q\left[-\log p_\theta(x_T)-\sum_{t=2}^T\log\frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1})}-\log\frac{p_\theta(x_0|x_1)}{q(x_1|x_0)}\right] && (50)\\
&= \mathbb{E}_q\left[-\log p_\theta(x_T)-\sum_{t=2}^T\log\left(\frac{p_\theta(x_{t-1}|x_t)}{q(x_{t-1}|x_t,x_0)}\cdot\frac{q(x_{t-1}|x_0)}{q(x_t|x_0)}\right)-\log\frac{p_\theta(x_0|x_1)}{q(x_1|x_0)}\right] && (51)\\
&= \mathbb{E}_q\left[-\log\frac{p_\theta(x_T)}{q(x_T|x_0)}-\sum_{t=2}^T\log\frac{p_\theta(x_{t-1}|x_t)}{q(x_{t-1}|x_t,x_0)}-\log p_\theta(x_0|x_1)\right] && (52)\\
&= D_{KL}(q(x_T|x_0)||p(x_T))+\sum_{t=2}^T\mathbb{E}_{q(x_t|x_0)}\left[D_{KL}(q(x_{t-1}|x_t,x_0)||p_\theta(x_{t-1}|x_t))\right]-\mathbb{E}_{q(x_1|x_0)}[\log p_\theta(x_0|x_1)] && (53)\\
&= L_T+\sum_{t=2}^T L_{t-1}-L_0 && (54)
\end{aligned}$$

Since each KL divergence in eq. (53) is between Gaussians, they can all be computed in closed-form expressions. Additionally, we can show that $-L$ is an evidence lower bound (ELBO) of $\log p(x)=\log q(x_0)$, so we can maximize $\log p(x)$ of our training data by minimizing L:

$$\begin{aligned}
D_{KL}(q(x_{1:T}|x_0)||p(x_{1:T}|x_0)) &= \int_{x_{1:T}} q(x_{1:T}|x_0)\log\frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})/p(x_0)} && (55)\\
&= \mathbb{E}_{q(x_{1:T}|x_0)}\left[-\log\frac{p_\theta(x_{0:T})}{p(x_0)q(x_{1:T}|x_0)}\right] && (56)\\
&= \mathbb{E}_q\left[\log p(x_0)-\log\frac{p_\theta(x_{0:T})}{q(x_0)q(x_{1:T}|x_0)}\right] && (57)\\
&= \mathbb{E}_q[\log p(x_0)]+\mathbb{E}_q\left[-\log\frac{p_\theta(x_{0:T})}{q(x_0)q(x_{1:T}|x_0)}\right] && (58)\\
&= \log p(x_0)+L && (59)\\
& && (60)
\end{aligned}$$

Thus,

$$\log p(x) = D_{KL}(p(x_0)q(x_{1:T}|x_0)||p(x_{0:T})) - L \tag{61}$$

$$\log p(x) \geq -L \tag{62}$$

$$\log p(x) \geq -D_{KL}(q(x_T|x_0)||p(x_T)) - \sum_{t=2}^{T} \mathbb{E}_{q(x_t|x_0)}\left[D_{KL}(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t))\right] + \mathbb{E}_{q(x_1|x_0)}[\log p_\theta(x_0|x_1)] \tag{63}$$

$$\log p(x) \geq -L_T - \sum_{t=2}^{T} L_{t-1} + L_0 \tag{64}$$

Let's break down this loss function

1. The first term of the loss, $L_T = D_{KL}(q(x_T|x_0)||p(x_T))$, represents the "distance" between the latent posterior at time T, $q(x_T|x_0)$, and our latent prior, $p(x_T)$, which we typically assume to be the standard normal distribution, $p(x_T) \sim \mathcal{N}(0, \mathbf{I})$. In the DDPM paper, the authors fix the variances β_t of the forward process to be constant, but they note that they could be learned via parameterization. This term has no trainable parameters when β_t is fixed, so it can be ignored in the optimization process.
2. The second term, $\sum_{t=2}^{T} L_{t-1} = \sum_{t=2}^{T} \mathbb{E}_{q(x_t|x_0)}\left[D_{KL}(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t))\right]$, represents the distance between the the approximated denoising step $p_\theta(x_{t-1}|x_t)$ and the true transition $q(x_{t-1}|x_t, x_0)$ at each time step between 1 and T-1.
3. The third term, $L_0 = \mathbb{E}_{q(x_1|x_0)}[\log p_\theta(x_0|x_1)]$ represents a reconstruction loss that rewards reproducing the original image, x_0, after diffusion. This reconstruction loss can be computed similar to Auto-Encoding Variational Bayes, 2.4).

3.1.5 Diffusion models and denoising autoencoders

We must condition on a reference image x_0 to learn the forward process posterior $q(x_{t-1}|x_t, x_0)$, because $q(x_{t-1}|x_t)$ is intractable. We can prove that:

$$q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}; \tilde{\mu}_t(x_t, x_0), \tilde{\beta}_t\mathbf{I}) \tag{65}$$

$$\text{where } \tilde{\mu}_t(x_t, x_0) := \frac{\sqrt{\bar{\alpha}_t}\beta_t}{1-\bar{\alpha}_t}x_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}x_t \text{ and } \tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t \tag{66}$$

We can further simplify the ELBO with the reparameterization from before, Eq (40):

$$x_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}(x_t - \sqrt{1-\bar{\alpha}_t}\epsilon) \text{ where } \epsilon \sim \mathcal{N}(0, \mathbf{I}) \tag{67}$$

Now we can rewrite the distribution $q(x_{t-1}|x_t, x_0)$ such that it only depends on x_t and ϵ

$$\tilde{\mu}_t(x_t) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon\right) \tag{68}$$

Recall, we parameterized the reverse process as $p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$. The authors decide to set $\Sigma_\theta(x_t, t) = \sigma_t^2\mathbf{I}$ to untrained time dependent constants. They find $\sigma_t^2 = \beta_t$ and $\sigma_t^2 = \tilde{\beta}_t$ produce similar results. The authors parameterize the reverse process this way so that they can write the ELBO/loss function as follows:

$$L_{t-1} = \mathbb{E}_{q(x_t|x_0)}\left[D_{KL}(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t))\right] = \mathbb{E}_q\left[\frac{1}{2\sigma_t^2}\left|\left|\tilde{\mu}_t(x_t, x_0) - \mu_\theta(x_t, t)\right|\right|^2\right] + C \tag{69}$$

where C does not depend on θ.
Using the reparameterization Eq. (67), and the forward process posterior Eq. (66), we can rewrite the loss:

$$L_{t-1} - C = \mathbb{E}_{x_0,\epsilon}\left[\frac{1}{2\sigma_t^2}\left|\left|\tilde{\mu}_t\left(x_t, \frac{1}{\sqrt{\bar{\alpha}_t}}(x_t(x_0,\epsilon) - \sqrt{1-\bar{\alpha}_t}\epsilon)\right) - \mu_\theta(x_t(x_0,\epsilon),t)\right|\right|^2\right] \tag{70}$$

$$= \mathbb{E}_{x_0,\epsilon}\left[\frac{1}{2\sigma_t^2}\left|\left|\frac{1}{\sqrt{\alpha_t}}\left(x_t(x_0,\epsilon) - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon) - \right) - \mu_\theta(x_t(x_0,\epsilon),t)\right|\right|^2\right] \tag{71}$$

We could directly predict $\tilde{\mu}_t$ with μ_θ or we could use the reparameterization trick to reduce the problem to learning the noise added to x_{t-1} to make x_t at timestep t. We might then "subtract" that noise to get to the prior timestep until the original image is recreated. We can approximate this noise with a neural network, $\epsilon_\theta(x_t,t)$, thus equivalently approximating the mean of the forward process posterior $q(x_{t-1}|x_t,x_0)$ in Eq. (68):

$$\tilde{\mu}_\theta(x_t,t) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t,t)\right) \tag{72}$$

With this parameterization, we can sample $x_{t-1} \sim p_\theta(x_{t-1}|x_t)$ using $x_{t-1} = \tilde{\mu}_\theta(x_t,t) + \sigma_t z$, where $z \sim \mathcal{N}(0,\mathbf{I})$. Furthermore, with parameterization (72), we can simplify our loss function to:

$$\mathbb{E}_{x_0,\epsilon}\left[\frac{\beta_t^2}{2\sigma_t^2\alpha_t(1-\bar{\alpha}_t)}\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)\|^2\right] \tag{73}$$

Intuitively, we want to minimize the difference between our prediction of the noise and the actual ϵ_t sampled at time t. The authors propose a simpler variational bound that they empirically found to improve sample quality:

$$L_{simple}(\theta) := \mathbb{E}_{t,x_0,\epsilon}\left[\left|\left|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)\right|\right|^2\right] \tag{74}$$

The authors found that learning the reverse process variances by incorporating a parameterized diagonal $\Sigma_\theta(x_t)$ into the variational bound, leads to unstable training and poorer sample efficiency, so they set a constant and isotropic Σ. They also found predicting ϵ performs approximately as well as predicting $\tilde{\mu}$ directly when trained on the variational bound with fixed variances, but much better when trained with our simplified objective.

4 Code and Experiments

To visualize the outputs of an example VAE and DDPM, we implemented basic baselines in PyTorch [Paszke et al., 2019] tasked with generating images from the MNIST dataset [LeCun and Cortes, 2010]. For the VAE, we used a simple 3-layer multi-layer perceptron (MLP) with a 2-dimensional latent vector for embedding μ and σ. [Kingma and Ba, 2015]. Regarding the DDPM, we used a simple convolutional network as our model for predicting ϵ. We follow Ho et al. [2020] and use a linear variance schedule increasing from $\beta_1 = 10^{-4}$ to $\beta_T = 0.02$. We trained both the VAE and the DDPPM for 100 epochs with a learning rate of 0.001 with the Adam optimizer. All of our code can be found at `https://github.com/AJB117/VAE-Example`.

In 2, we see three examples of images generated by the VAE compared to the original data points it was tasked with reconstructing. The leftmost image is an example of a degenerate output with the VAE unable to output an image resembling an 8. In the middle image, the VAE nearly produces a 9, instead generating a 7, which looks similar to a 9. During training, the VAE's latent representation of images of 7 is likely very close to those of 9. Finally, the rightmost image shows a faithful reconstruction of the training image of the number 0. In 3, we observe ten example of images generated by the DDPM. Since DDPMs do not generate images from sampling from a latent code produced by encoding a given image, we are only able to assess the quality of its outputs from randomly initialized noise rather than compare them with ground-truth images. We see that the DDPM generally outputs higher-fidelity outputs than the VAE and does not produce as many degenerate images. We see that there is a general lack of diversity in the samples generated, however,

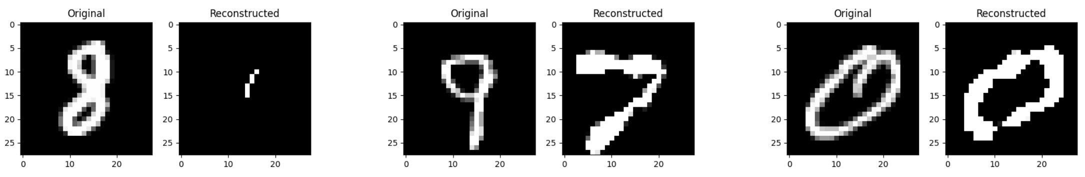

Figure 2: Three VAE generations of increasing faithfulness to the original training data.

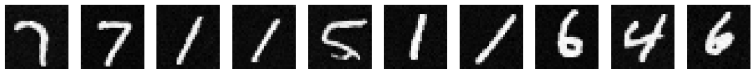

Figure 3: Ten example outputs of the DDPM.

although this may be mitigated by new work on reducing data bias in diffusion models. We note that DDPM took substantially longer to converge, taking over 3 hours, while the VAE took only around 45 minutes on a single machine with an Nvidia RTX 3090 GPU. While the differences in architecture are apparent, the diffusion process takes considerably longer than simply encoding an image and decoding its latent in a single step.

5 Conclusion

In summary, both variational autoencoders and diffusion models are latent variable models for encoding complex distributions. They both use variational inference methods to approximate intractable integrals and complex distributions more efficiently than prior approaches. To do this, both methods turn the integral into an optimization process by finding a lower bound on the data's likelihood which they can maximize directly. Both methods require reparametrization to avoid non-differentiable sampling operations in intermediate distributions. The two methods differ in how they represent the image generation process. Variational autoencoders model the process as sampling from the latent distribution, then sampling from the data's posterior. Diffusion models similarly model the latent distribution, but over several timesteps, each with their own latent variable, and increasing fidelity, until an image from the data distribution is created (reverse process). The many transition steps potentially allow for more expressiveness in the model, which may explain why diffusion models and their variants have become the state of the art for generative image models.

References

J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper_files/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf`.

D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, San Diega, CA, USA, 2015.

D. Kingma, T. Salimans, B. Poole, and J. Ho. Variational diffusion models. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 21696–21707. Curran Associates, Inc., 2021. URL `https://proceedings.neurips.cc/paper_files/paper/2021/file/b578f2a52a0229873fefc2a4b06377fa-Paper.pdf`.

D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

Y. LeCun and C. Cortes. MNIST handwritten digit database. 2010. URL `http://yann.lecun.com/exdb/mnist/`.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Z. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. *CoRR*, abs/1912.01703, 2019. URL `http://arxiv.org/abs/1912.01703`.

D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In E. P. Xing and T. Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1278–1286, Bejing, China, 22–24 Jun 2014. PMLR. URL `https://proceedings.mlr.press/v32/rezende14.html`.