A Tale of VAEs, WGANs, and DDPMs

Ashwin De Silva
EN.553.741 Machine Learning II (Project Report)
Johns Hopkins University
`ldesilv2@jhu.edu`

Abstract

The aim of this project is to describe the theoretical foundations and implement variational autoencoder (VAE), Wasserstein GAN (WGAN), and denoising diffusion probabilistic models (DDPM).

1 A Primer on Generative Modeling

Suppose that an observed datum x comes from a true but unknown distribution $p^*(x)$ such that $x \sim p^*(x)$. In generative modeling, our objective is to approximate $p^*(x)$ with a chosen model $p_\theta(x)$ parameterized by θ. We may attain this objective by searching for the best set of parameters θ^* such that $p_{\theta^*} \approx p^*(x)$.

Ler $\mathcal{D} = \{x^{(1)}, x^{(2)}, \ldots, x^{(N)}\}$ be a dataset of observations drawn independently and identically from $p^*(x)$. The log-likelihood of observing these samples under the model $p_\theta(x)$ is given by,

$$\log p_\theta(\mathcal{D}) = \sum_{i=1}^{N} \log p_\theta(x^{(i)}) \tag{1}$$

By maximizing the log-likelihood of the observations with respect to the parameters θ, we may find a better approximation to the true distribution under the model $p_\theta(x)$. The estimate $\hat{\theta}$ we obtain through this procedure is known as the maximum likelihood estimate (MLE) and it is given by,

$$\hat{\theta} = \arg\max_{\theta} \sum_{i=1}^{N} \log p_\theta(x^{(i)}) \tag{2}$$

This maximization problem can be generaly solved using stochastic gradient descent if we consider minimizing the negative log-likelihood. It can also be shown that maximing the likelihood is equivalent to minimizing the Kullback-Liebler divergence between the model and the true distribution,

$$\mathrm{KL}(p^* \| p_\theta) = \mathbb{E}_{x \sim p^*}\left[\log\left(\frac{p_\theta(x)}{p^*(x)}\right)\right].$$

In addition to maximum likelihood estimation, there are alternate ways of performing generative modeling. For instance, in the Bayesian framework, one may use the *maximum a priori* (MAP) estimate. However, in this work, we will only focus on MLE and related estimators moving forward. The rest of the report is organized as follows: we first discuss latent variable models, a popular class models with high expressivity that exploits the evidence lower bound as a proxy to maximize the log-likehood of the model. Variational auto-encoders (VAEs) and denoising diffusion probabilistic models (DDPMs) falls under this category. Next, we discuss the Wasserstein GAN which minimizes the Wasserstein distance instead of the implicit KL-divergence. Finally we describe the Frechet inception distance (FID) score that is used to evaluate the generated sample quality, and demonstrate the results obtained from the implementations of VAE, DDPM, and WGAN on the MNIST dataset. The code for the implementations is available at `https://github.com/Laknath1996/gen-models`.

2 Latent Variable Models

Latent variables are model variables that we do not observe. Consider a model $p_\theta(x, z)$ where x are the observed variables and z denote the latent variables. Therefore, $p_\theta(x, z)$ is the joint distribution of x and z under the paramters θ. We call p_θ a latent variable model (LVM). A common factorization of this joint distribution is $p_\theta(x|z)p_\theta(z)$ where $p_\theta(z)$ is the prior and $p_\theta(x|z)$ is the conditional (or the decoding model). Since, x is observed, we are naturally interested in the marginal likelihood of x given by,

$$p_\theta(x) = \int p_\theta(x, z)\, \mathrm{d}z$$

For instance, if z is discrete and $p_\theta(x|z)$ is a Gaussian distribution, then $p_\theta(x)$ denotes a Gaussian mixture model that is highly expressive.

Consider the LVM $p_\theta(x, z)$. Given a dataset $\mathcal{D} = \{x^{(1)}, x^{(2)}, \ldots, x^{(N)}\ \}$ drawn from a true but uknown distribution $p(x)$, we may maximize the marginal log-likelihood $p_\theta(x)$ to find the best θ using eq. (2). If we are using stochastic gradient descent for this problem, we need to take the gradient $\nabla_\theta \log p_\theta(x)$. However, since $\int p_\theta(x, z)\, \mathrm{d}z$, taking the gradient requires integrating over z, which is intractable. Alternatively, we may use the Bayes theorem to get $p_\theta(x) = p_\theta(x, z)/p_\theta(z|x)$ where $p_\theta(z|x)$ is the posterior. However, although computing $p_\theta(x, z)$ is tractable, computing $p_\theta(z|x)$ is not.

To overcome this problem, we can consider approximating the posterior $p_\theta(z|x)$ by $q_\phi(z|x)$ that is tractable. We refer to $q_\phi(z|x)$ as the approximate posterior or the encoding model. It is parameterized by ϕ which is commonly referred to as the variational parameters.

With the approximate posterior $q_\phi(z|x)$, we can write down the evidence lower bound for the marginal likelihood $p_\theta(x)$ as follow.

$$\log p_\theta(x) \geq \mathbb{E}_{z\sim q(\cdot|x)} \left[\log \left(\frac{p_\theta(x, z)}{q_\phi(z|x)} \right) \right]$$

Therefore, instead of maximizing $\log p_\theta(x)$ directly, we can maximize the evidence lower bound with respect to both θ and ϕ.

$$\max_{\theta,\phi} \sum_{i=1}^{N} \mathbb{E}_{z\sim q(\cdot|x^{(i)})} \left[\log \left(\frac{p_\theta(x^{(i)}, z)}{q_\phi(z|x^{(i)})} \right) \right] \tag{3}$$

This is the objective of the variational autoencoder.

3 Variational Autoencoder

We consider the following latent variable model explained in Kingma & Welling (2013) for the observed variable $x = (x_1, \ldots, x_m) \in \{0, 1\}^m$ and latent variable $z \in \mathbb{R}^d$.

$$\begin{aligned} p(z) &= \mathcal{N}(z; 0, I) \\ (u_1, \ldots . u_m) &= \text{DecoderNN}_\theta(z) \\ p(x|z) &= \prod_{i=1}^{m} x_i^{u_i}(1 - x_i)^{1-u_i} \end{aligned}$$

Note that conditional/decoding model $p(x|z)$ is a factorized Bernoulli distribution and we have let the latent variable z to distributed according to a spherical Gaussian. We ensure that $u_1, \ldots, u_m$ are within $[0, 1]$ by having a sigmoid non-linearlity at the end of the DecoderNN MLP.

The approximate posterior/encoding model $q_\phi(z|x)$ is set to be a Gaussian distribution as specified below.

$$\begin{aligned} (\mu, \log\sigma) &= \text{EncoderNN}_\phi(x) \\ q_\phi(z|x) &= \mathcal{N}(z; \mu, \text{diag}(\sigma)) \end{aligned}$$

The EncoderNN MLP is constructed to yield two outputs corresponding to μ and $\log\sigma$, which are the mean and the diagonal of the covariance matrix of $q_\phi(z|x)$ respectively.

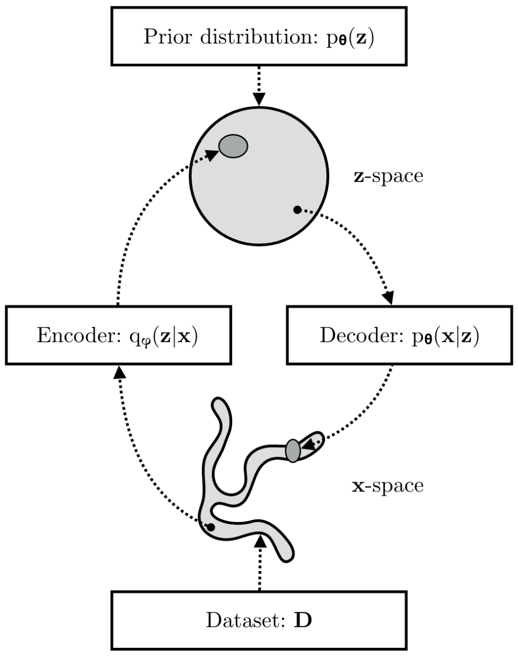
Prior distribution: $p_{\boldsymbol{\theta}}(\mathbf{z})$
$\mathbf{z}$-space
Encoder: $q_{\varphi}(\mathbf{z}|\mathbf{x})$
Decoder: $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$
$\mathbf{x}$-space
Dataset: $\mathbf{D}$

Figure 1: An illustration of stochastic mappings associated with the variational autoencoder

Rearranging the terms in eq. (3), we obtain the following loss function for training the variational autoencoder.

$$\ell(\theta, \phi) = \mathbb{E}_{q(z|x)}\left[\log p_\theta(x, z) - \log q_\phi(z|x)\right]$$

Although obtaining the gradient w.r.t θ is straightforward, it is not the case for ϕ. To overcome that, we employ the following reparameterization for z:

$$\begin{aligned}&\epsilon \sim \mathcal{N}(0, I)\\ &z = g(\phi, x, \epsilon) = \mu + \text{diag}(\sigma)\ \odot\ \epsilon\end{aligned}$$

This enables us to obtain a gradient of $\ell(\theta, \phi)$ w.r.t ϕ. Rearranging the loss function further, we obtain the following expression which leads to a better intuition behind the VAE objective.

$$\ell(\theta, \phi) = \text{KL}(q_\phi(z|x) \| p_\theta(z)) - \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)]$$

Since $q_\phi(z|x)$ and $p_\theta(z)$ are both Gaussians, the first term minimizes the KL-divergence between two Gaussians. This can be thought of as the regularizer term that does not let the approximate posterior drift too far from the prior distribution. Under the VAE formulation we described above, the second term becomes the binary cross entropy loss between original data x and the decoder output u. This represents a reconstruction error between the original data and the decoder outputs. Upon training the VAE with this objective, we can feed a noise vector to the decoder to obtain a generated sample.

4 Denoising Diffusion Probabilistic Models

Denoising diffusion probabilistic models (Ho et al., 2020) or "diffusion" models are also a type of latent variable models. With observed variable x_0 and latent variables $x_1, \ldots, x_T$, the joint distribution is given by $p_\theta(x_0, x_1, \ldots, x_T)$. The observed variable x_0 is distributed according to the true but unknown distribution $q(x_0)$.

The joint distribution $p_\theta(x_0, x_1, \ldots, x_T)$ is referred to as the reverse process as it is assumed to be a first order-Markov chain given by,

$$p_\theta(x_0, x_1, \ldots, x_T) = p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t),$$

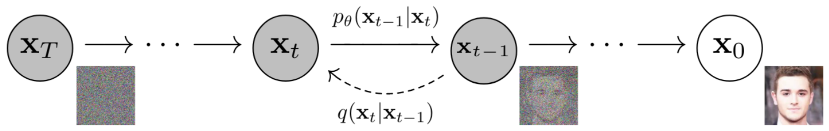
$\mathbf{x}_T$
$\mathbf{x}_t$
$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$
$\mathbf{x}_{t-1}$
$q(\mathbf{x}_t|\mathbf{x}_{t-1})$
$\mathbf{x}_0$

Figure 2: An illustration of the forward and reverse processes of a diffusion model

where the transition probability distributions $p_\theta(x_{t-1}|x_t)$ is modeled as a Gaussian distribution such that,

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

As we discussed above, we are interested in maximizing the marginal likelihood $p_\theta(x_0) = \int p_\theta(x_0, \dots, x_T) dx_{1:T}$ with respect to θ. Since we cannot maximize the marginal likelihood without a tractable posterior $p_\theta(x_1, \dots, x_T|x_0)$, we consider the approximate posterior $q(x_1, \dots, x_T|x_0)$ which is also assumed to be a first order Markov chain denoted as the forward process.

$$q(x_1, \dots, x_T|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1})$$

The transition probability distribution $q(x_t|x_{t-1})$ is also modeled as a Gaussian distribution such that

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

where, β_t is a "noise schedule" that controls the level of variance. An illustration of the forward and reverse processes is given in fig. 2.

As this is still a LVM problem, we can write down the corresponding evidence lower bound L as follows.

$$-\log p_\theta(x) \leq \mathbb{E}_q \left[-\log \left(\frac{p_\theta(x_0, x_1, \dots, x_T)}{q(x_1, \dots, x_T|x_0)} \right) \right] := L$$

Rearranging the terms, and simplifying using the Markov and Gaussian assumptions discussed above, L can simplified into the following objective function,

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

where $\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$ and ϵ is a noise vector drawn from $\mathcal{N}(0, I)$. The neural network ϵ_θ takes a uniformly randomly sampled time $t \sim$ Uniform($\{1, \dots, T\}$) and the noised input $x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$ and attempts to return noise vector ϵ that was added to the input x_0 that was drawn from the true distribution $q(x_0)$. In other words, the network ϵ_θ is trained to *denoise* the noised input x_t. Therefore, the training procedure of ϵ_θ can be summarized in the following algorithm.

Algorithm 1 Training procedure of DDPM

```
while not converged do
    $x_0 \sim q(x_0)$
    $t \sim$ Uniform($\{1, \dots, T\}$)
    $\epsilon \sim \mathcal{N}(0, I)$
    Take the gradient descent step on $\nabla_\theta \|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)\|^2$
```

What is the significance of training a denoising network ϵ_θ? To answer this question, let's consider the Langevin Monte Carlo stochastic difference equation (SDE) given below.

$$x_{t+1} = x_t - \eta \nabla \log p(x_t) + \sqrt{2\eta}\zeta_t \; ; \; \zeta_t \sim \mathcal{N}(0, I)$$

The stationary distribution of this SDE is $p(x)$. Therefore, if we had the term $\nabla \log p(x)$ also known as the score function of $p(x)$, we can run the Langevin MC equation until it converges to obtain samples from $p(x)$. However, when we do not know the true distribution $p(x)$ or its score function, how may we obtain an estimate for the score function? To this end, one could consider a neural network ϵ_θ to fit the score function $\nabla \log p(x)$ by training it to minimize the loss function

$\mathbb{E}_{x\sim p}[\|\nabla \log p(x) - \epsilon_\theta(x)\|^2]$. Interestingly, it can be shown that this loss function is approximately equivalent to a denoising loss function:

$$\mathbb{E}_{x\sim p}[\|\nabla \log p(x) - \epsilon_\theta(x)\|^2] \approx \mathbb{E}_{x\sim p, \epsilon\sim\mathcal{N}(0,I)}[\|\epsilon - \epsilon_\theta(x+\epsilon)\|^2]$$

In other words, this implies that a denoising objective leads the network ϵ_θ to approximate the score function $\nabla \log p(x)$.

Therefore, once the network ϵ_θ is trained we can employ the following sampling scheme to obtain generated samples from the diffusion model. Notice that the expression for x_{t-1} is related to the Langevin MC equation mentioned above.

Algorithm 2 Sampling from DDPM

```
$x_T \sim \mathcal{N}(0, I)$
for $t = T, \dots, 1$ do
    $z \sim \mathcal{N}(0, I)$ if $t > 1$, else $z = 0$
    $x_{t-1} = \frac{1}{\sqrt{\bar{\alpha}_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)\right) + \sigma_t z$
return $x_0$
```

Currently, diffusion models are the state-of-the-art baseline in generating images. However, it suffers from a high computational time taken during the sampling process. There are several ways to boost the sampling speed, but these methods are not discussed in this report.

5 Wasserstein GAN

Let $p(x)$ and $p_\theta(x)$ be the true and model distributions as before. The Wasserstein distance between them is defined by,

$$W(p, p_\theta) = \inf_{\gamma\in\Pi(p,p_\theta)} \mathbb{E}_{(x,y)\sim\gamma}\left[\|x-y\|^2\right], \tag{4}$$

where, $\Pi(p, p_\theta)$ denotes the set of all joint distributions $\gamma(x, y)$ whose marginals are $p(x)$ and $p_\theta(x)$ respectively. The joint distribution γ indicates how much mass should be transported from x to y in order to transform the $p_\theta(x)$ to $p(x)$. Then, Wasserstein distance is the cost of the optimal tranport plan.

Unlike other measures of differences between distributions such as total variation distance, Kullback-Liebler (KL) divergence, Jenson-Shannon (JS) divergence, the Wasserstein distance, under regularity assumptions, is continuous everywhere, is differentiable almost everywhere, and induces weak topologies (i.e. sequences of probability distributions are more likely to converge under the Wasserstein distance).

However, the infimum in eq. (4) is difficult to compute in reality. However, the Kantorovich-Rubinstein duality provides an alternate formulation to the Wasserstein distance:

$$W(p, p_\theta) = \sup_{\|f\|_L\leq 1} \mathbb{E}_{x\sim p(x)}[f(x)] - \mathbb{E}_{x\sim p_\theta(x)}[f(x)] \tag{5}$$

where the supremum is over all the real-valued 1-Lipschitz functions. If z is a random variable distributed according to some known density function $q(z)$ (e.g. the normal distribution), then, let the $p_\theta(x)$ be the distribution of the random variable $g_\theta(z)$ where g_θ is a function parameterized by θ. For instance, g_θ could be a neural network with parameters θ.

If we have a family functions $\{f_w\}_{w\in\mathcal{W}}$ parameterized by w such that all f_w are 1-Lipschitz, we can consider the following expression for the Wasserstein distance:

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \max_{w\in\mathcal{W}} \mathbb{E}_{x\sim p(x)}[f_w(x)] - \mathbb{E}_{z\sim q(z)}[f_w(g_\theta(z))] \tag{6}$$

Let's consider minimizing the Wasserstein distance between $p(x)$ and $p_\theta(x)$ w.r.t θ. Let θ^* be a solution to this minimization problem, that we may compute by running Gradient descent. Then, the samples we will be obtaining according to $g_{\theta^*}(z)$ (known as the "generator") where $z \sim q(z)$ are drawn from $p_{\theta^*}(x)$ which will be maximally closer to the desired distribution $p(x)$ under the Wasserstein distance.

However, the minimization of Wasserstein distance becomes a minimax problem through eq. (7), which essentially summarizes the generative modeling strategy of the Wasserstein GAN Arjovsky et al. (2017).

$$\min_\theta W(p, p_\theta) = \min_\theta \max_w \mathbb{E}_{x\sim p(x)}[f_w(x)] - \mathbb{E}_{z\sim q(z)}[f_w(g_\theta(z))] \tag{7}$$

To solve the inner maximization problem, we let f_w (known as the "critic") be a separate neural network parameterized by weights w and perform gradient ascent. In order to ensure the Lipschitzness of f_w, we can consider clamping the weights w within a specified fixed range after each gradient update. While this is a primitive way to ensure Lipschitzness, there are more advanced methods of achieving the same result (e.g. the gradient penalty method Gulrajani et al. (2017))

Based on this description, we can summarize the training procedure of the Wasserstein GAN as follows.

Algorithm 3 Training procedure of the Wasserstein GAN

```
Require: the learning rate $\alpha$, the clipping parameter $c$, batch size $m$, number of iterations $n_{\text{critic}}$ of
  the critic per generator iteration
Require: The initial critic parameters $w_0$, the initial generator parameters $\theta_0$
  while $\theta$ has not converged do
      for $t = 0, \dots, n_{\text{critic}}$ do
          Sample $m$ real data $x^{(1)}, \dots, x^{(m)} \sim p(x)$
          Sample $m$ latent realizations $z^{(1)}, \dots, z^{(m)} \sim q(z)$
          $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))\right]$
          $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
          $w \leftarrow \text{clip}(w, -c, c)$
      Sample $m$ latent realizations $z^{(1)}, \dots, z^{(m)} \sim q(z)$
      $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))$
      $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
```

6 Frechet Inception Distance Score

The Frechet Inception Distance (FID) (Heusel et al., 2017) score is used to evaluate the quality of the generated images from a generative model. Consider a true distribution $p(x)$ and a generative model $p_\theta(x)$. Then the FID score is computed using the following algorithm.

Algorithm 4 Computing the FID score

```
  Sample a set of true images $S_1 = x_1, \dots, x_N \sim p(x)$
  Sample a set of generated images $S_2 = y_1, \dots, y_N \sim p_\theta(x)$
  Obtain the latent feature vectors $f(S_1)$ and $f(S_2)$
  Fit a Gaussian distribution $\mathcal{N}_1, \mathcal{N}_2$ over $f(S_1)$,$f(S_2)$
  Compute the Wasserstein distance $W_2^2(\mathcal{N}_1, \mathcal{N}_2)$
  return $W_2^2(\mathcal{N}_1, \mathcal{N}_2)$
```

In the algorithms above, the function f is *InceptionV3*, a pre-trained convolutional architecture that is used to obtain a 2048 feature vector from the true and generated images. Then, we fit Gaussian distributions over the two feature vector ensembles, and compute the 2-Wasserstein distance between them to obtain the FID score. The higher the FID score, the lesser the sample quality as the two distributions are far from each other.

7 Implementation and Results

The implementations of the VAE, WGAN, and DDPM are available in `https://github.com/Laknath1996/gen-models`. The models were trained on a NVIDIA Quadro RTX 8000 GPU. Several generated images from the trained models are illustrated in fig. 3 alongside the true images.

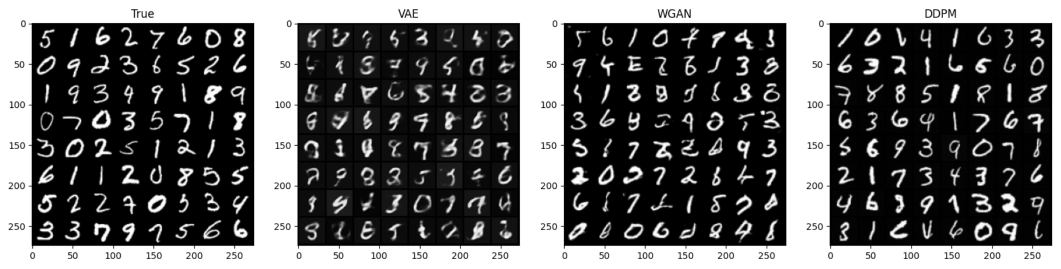

Figure 3: True images and generated images from VAE, WGAN, and DDPM

At a glance, it can be observed that the DDPM samples are of superior quality compared to the samples generated by VAE and WGAN.

The FID scores for VAE, WGAN, and DDPM were 16.3, 12.4 and 9.3 respectively. The details of the training metrics are available under the same heading in the repository mentioned above.

References

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan, 2017.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans, 2017.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

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