A Survey of Deep Learning Generative Models

Mahebub Aalam Khatri

Abstract

Deep Generative Models (DGMs) have become ubiquitous in today's world. Over the past decade, they have revolutionized fields such as computer vision, computer graphics, natural language processing, and even engineering design [4, 37] and are starting to become a staple in our day to day lives. Therefore, in this survey, our goal is to establish the basic framework of these DGMs and explain how three of the most popular DGMs fit into that framework to learn distributions and create samples. In particular, this survey will discuss Variational Autoencoders (VAEs) first introduced in 2013 by [24], Generative Adversarial Networks first introduced in 2014 by [16], and finally Denoising Diffusion Models (DDMs) first introduced in 2015 by [43]. In addition to their basic functionality, the major advantages and drawbacks of each of these models will be discussed in the context of the deep learning trilemma [56] followed by a discussion of the research being conducted that aims to mitigate the drawbacks or otherwise improve the performance of these DGMs.

1 Introduction

Generative modeling covers a broad area of machine learning which revolves around the idea of learning an approximation $P_g(X)$ to a true data distribution $P_t(X)$ over some set of data points X in a potentially high-dimensional space $\mathcal{X}$ [11]. Note that these models focus on *approximations* of $P_t(X)$ because for most cases, the explicit distribution $P_t(X)$ cannot be

determined. The aspect that makes these models *generative* is that new samples can then be generated from the approximate distribution $P_g(X)$ which simulate the characteristics of samples from the true data distribution $P_t(X)$.

1.1 Non Deep Learning Models

Historically, research on generative modeling has focused on non-deep learning methods like Gaussian Mixture Models (GMMs), Hidden Markov Models (HMMs), and Latent Dirichlet Allocation (LDA) [18]. GMMs try to approximate distributions of certain classes in data by modeling them as normal distributions and learn the means, covariance matrices, and mixture probabilities of each class that best approximate the true data distribution. HMMs can generate state sequences by learning sequences called Markov Chains which describe state-transition probabilities. Both GMMs and HMMs are trained using expectation maximization [59]. Finally, LDA is a probabilistic model for topic modeling, which can learn thematic structures within a collection of data. In LDA, each data point is assumed to be some mix of topics and the components of each data points are generated by one of those topics. For a more in-depth survey of these traditional methods, the reader is directed to [18]. These "traditional" approaches to learning data distributions have, however, recently been eclipsed in performance by the deep learning generative models covered in this survey.

1.2 Deep Learning Generative Models

Modern deep learning generative models (DGMs) were first introduced just over a decade ago at the time of writing this report and have, in that time, seen tremendous growth in performance and utility. These DGMs are now used in a large variety of tasks including artistic applications such as novel image generation [9, 16], image denoising and in-painting [14, 57], voice and music generation [2, 13], and recently, even video generation [21, 49]. They have also seen application in non-artistic domains such as in the discovery of new compounds for

chemistry [46], aid in engineering design [37], and medical imaging [45]. These DGMs follow the same basic principles as traditional generative methods in that they attempt to estimate the true data distribution. However, these models often achieve this task by creating a *mapping* from some latent distribution $P(z)$ to the approximation of the true data distribution $P_g(X)$. For DGMs, the function that maps a vector from the latent variable to the generated distribution is represented by some neural network. This neural network is trained on a loss function which is based on some metric which describes the difference between the generated distribution and the target distribution (see figure 1). The three most popular DGMs which include Variational Autoencoders (VAEs), Generative Adversarial Networks (GANs), and Denoising Diffusion Models (DDMs) formulate the training of this neural network and the corresponding loss function in different ways which we will explore in this survey.

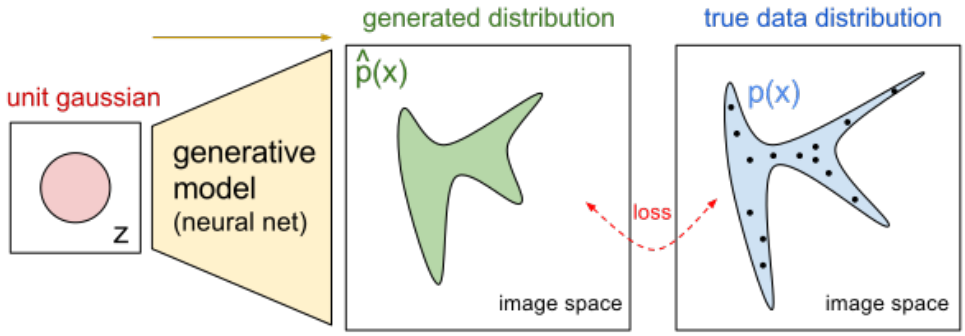

Figure 1: The general deep generative model framework. The goal of DGMs is to learn a mapping from some latent space to an approximation of the true target distribution. This mapping is learned via a neural network trained on a loss function that quantifies the difference between the generated and true data distributions. Figure courtesy of OpenAI.

1.3 The Deep Learning Trilemma

Each of these DGMs has their own strengths and weaknesses which enable their use in particular applications and none of the three model paradigms on their own provide a definitive solution for all problems that may require the use of a DGM. Specifically, each of these models suffers from one of three drawbacks that are handled well by the other two models. For example, VAEs are quick to train and create samples and they cover the distribution of the

training data well. However, they tend to produce poor results compared to the other two models. GANs on the other hand, produce high quality samples quickly, but are notoriously difficult to train and do a poor job of covering the variance of the target distribution. Finally, DDMs produce the highest quality samples that cover the modes of the distributions well but are by far the most expensive to train and generate samples. The trade-offs between these models is what is known in the literature as the *Deep Learning Trilemma* [56] (see figure 2). Consequently, much of the research in this field has focused on overcoming the drawbacks of each of these models in the hopes of creating the optimal DGM with good quality samples which cover all the modes of the data and is quick to train and create samples.

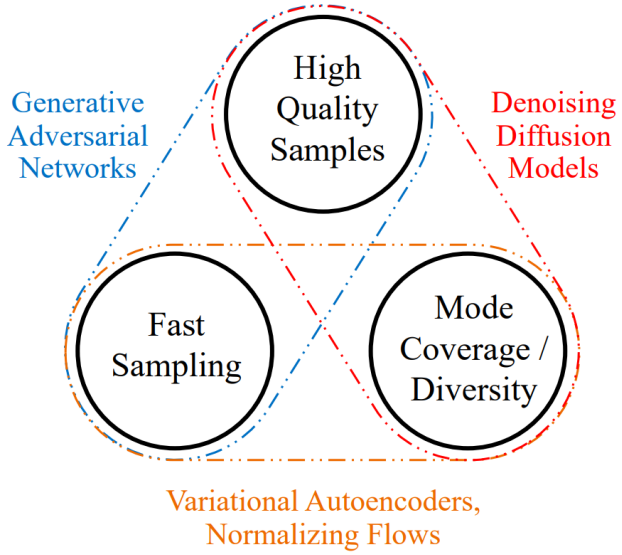

Figure 2: The generative learning trilemma. Figure courtesy of [56].

1.4 Survey Structure

The structure of the remaining sections of this survey are as follows. In §2, 3, and 4, we will cover the formulations of VAEs, GANs, and DDMs respectively and discuss each of their drawbacks, applications, and research trends. Notably, in §4.1, we take a detour to discuss evaluation metrics for DGMs. Then, in §5, we will discuss recent literature that tries to tackle the deep learning trilemma using a variety of methods including improving existing

DGMs or combining DGMs together. Finally, we will conclude in §6 by reviewing the content of the survey and discuss our expectations and desires for future research directions.

2 Variational Autoencoders

2.1 Autoencoders

VAEs can be thought of as an extension of the traditional *autoencoder* framework, so we will first discuss the autoencoder formulation. As the name suggests, an autoencoder is a neural network that can encode *itself*. Unlike what we will see in GANs, autoencoders start directly from the feature space representation of the data x with some dimensionality n and *encode* x down to a latent z space representation with some dimensionality m such that $m < n$ and $z = E(x)$ where E represents the encoder network. Then, the second half of the autoencoder known as the *decoder* takes z as input to create an approximation of the original input $\tilde{x} = D(z) = D(E(x))$ where D represents our decoder network. The loss function $\mathcal{L}$ is the *reconstruction loss* which can be expressed simply as $\mathcal{L} = \|x - \tilde{x}\|^2$. The simplest linear single-hidden-layer autoencoder is effectively the same as *Principle Component Analysis* (PCA) [55]. However, compared to traditional PCA methods like *Eigenvalue Decomposition* (EVD) or *Singular Value Decomposition* (SVD), the advantage of autoencoders is that we are not making any assumptions about the linearity, rank, or eigen/singular vectors of the latent space. Autoencoders are simply learning *some* non-linear mapping from the input space to the latent subspace that is capturing *something* meaningful about our data [54, 55]. Like in many machine learning models, we are giving up the clear mathematical foundation of methods like PCA for a black box that is learning some complex non-linear mapping.

Autoencoders have been shown to be very good for tasks like image denoising and image in-painting [14, 57]. However, on their own, they are typically not considered to be *generative models* in the usual sense because a random latent vector passed in to the decoder is not guaranteed to make a meaningful output. This is because the latent space of autoencoders

make no guarantees on structure. Choosing a latent variable that is even slightly different than a known latent variable which produces a valid output is likely to produce a random result (see figure 3). For this reason, a common extension of autoencoders is to include a regularization term to the loss function so that more complex mappings have higher loss. This in turn makes the autoencoder less sensitive to changes in the input x and the latent space representations z, making the autoencoder more generalizable and allows minimal traversal in the latent space. This regularization can be simply expressed as the sum of squared weights of our encoder and decoder networks [1].

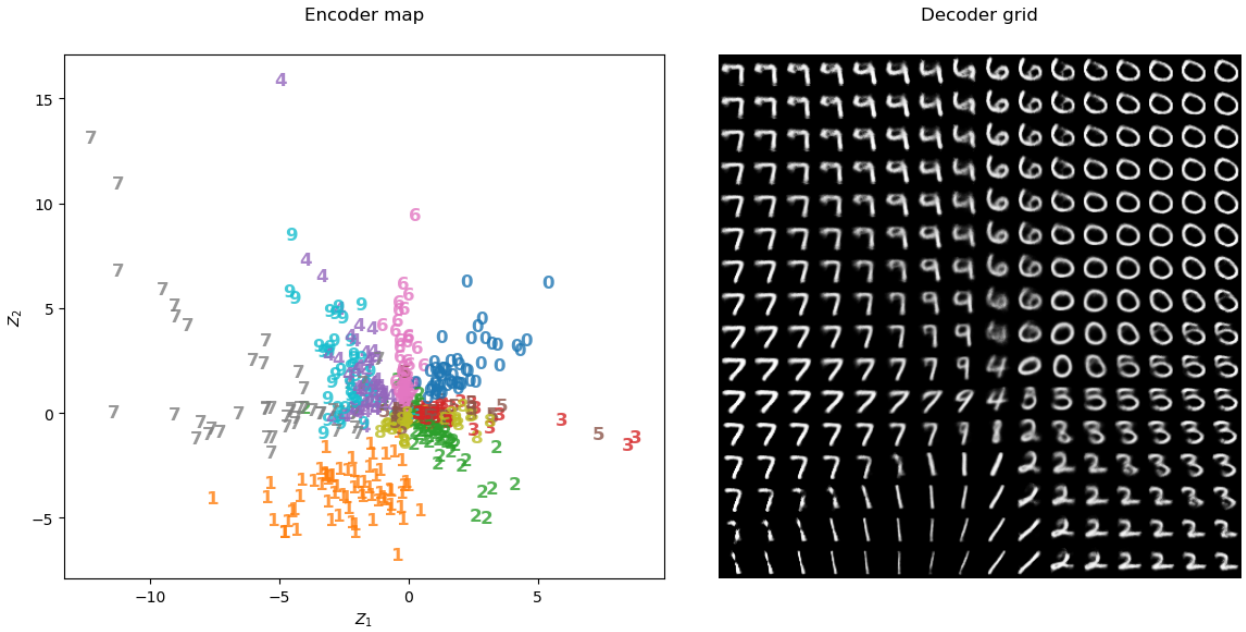

Figure 3: Left: A 2D latent space representation of encoded MNIST images for a simple autoencoder trained on a subset of MNIST. Right: The decoded outputs from evenly spaced locations in the latent space. We can see that with a simple autoencoder, the learned latent representation has some structure but the distinction between digits is not well defined and the sizes of regions for different digits is also not consistent.

2.2 Extending Autoencoders to VAEs

Variational Autoencoders take the idea of creating a more structured latent space a step further by adding additional constraints to the encoder which enforce the latent space mappings to be more Gaussian. With VAEs, we are trying to learn a latent representation $p(\mathbf{z})$

such that the mapping $p(\mathbf{x}|\mathbf{z})$ represented by our decoder is meaningful in some way, i.e., such that the resulting $p(\mathbf{z})$ allows us to approximate $p(\mathbf{x})$ which can be described as:

$$p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})dx. \tag{1}$$

But, to do that, we will need to learn a mapping $p(\mathbf{z}|\mathbf{x})$ which is generally intractable. So instead, we learn a simpler prior distribution which is a good approximation of our desired $p(\mathbf{z})$ which allows us to then approximate $p(\mathbf{z}|\mathbf{x})$ without having to learn it explicitly. In particular, we can decide before hand that we want our latent distribution $p(\mathbf{z})$ to be a normal (Gaussian) distribution with mean 0 and an identity covariance matrix. Then, our encoder $q_\phi(\mathbf{z}|\mathbf{x})$ which approximates $p(\mathbf{z}|\mathbf{x})$ should map, on average, $p(\mathbf{x})$ to a Gaussian distribution for $p(\mathbf{z})$ [23].

However, if we only focus on training our encoder to map to a Gaussian distribution, then we will not learn a meaningful latent space $p(\mathbf{z})$. Instead, in addition to the reconstruction loss used for autoencoders, we want to introduce a measure of the difference between our learned latent distribution and our desired $\mathcal{N}(\mathbf{0}, \mathbf{I})$ Gaussian distribution. The measure used is the Kullback-Leibler (KL) divergence [23, 26]. The KL divergence between real-valued distributions P and Q can be expressed as:

$$D_{\mathrm{KL}}(P\|Q) = \int p(x) \log \left(\frac{p(x)}{q(x)} \right) dx. \tag{2}$$

For VAEs, our encoder no longer maps down to a single point but rather to an n-dimensional Gaussian (where n is the dimensionality of our latent space) and the decoder takes a collection of points from the latent Gaussian distribution to produce multiple images which are compared to the original input to determine the reconstruction error. Due to this formulation, small deviations in the latent distribution will still produce good examples since they *have* to handle variations in the latent space.

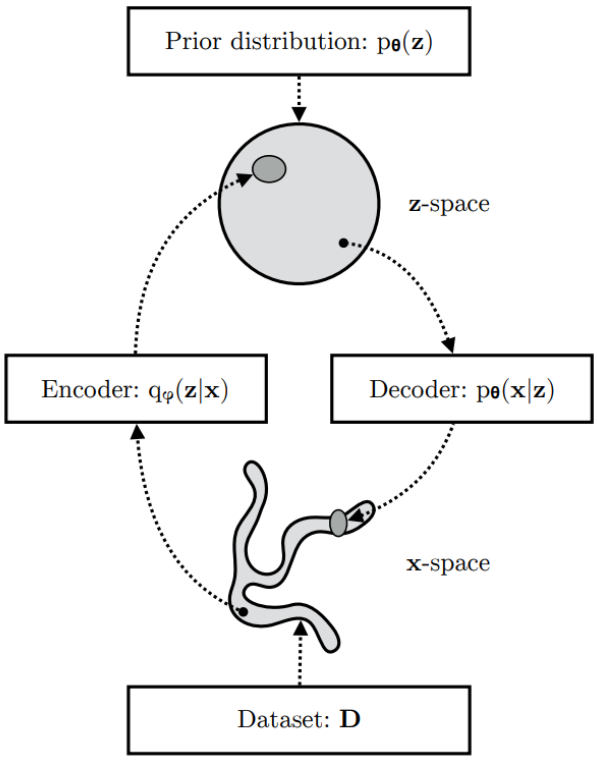

Figure 4: A VAE learns a stochastic mapping from some **x**-space with a complicated distribution to a relatively simple latent **z**-space, such as a spherical Gaussian. The generative model learns the joint distribution $p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{z})p_\theta(\mathbf{x}|\mathbf{z})$ where $p_\theta(\mathbf{z})$ is the prior distribution in the latent space and $p_\theta(\mathbf{x}|\mathbf{z})$ is the stochastic decoder. The encoder $q_\phi(\mathbf{z}|\mathbf{x})$ is an approximation of the intractable posterior $p_\theta(\mathbf{z}|\mathbf{x})$. Figure courtesy of [23].

2.3 The VAE Training Objective

Kingma et. al. ([23]) show that the training objective for VAEs (the log-likelihood of $p(\mathbf{x})$) can be expressed as the sum of the *evidence lower bound* (also known as the *variational lower bound*) (ELBO) and the KL divergence between our encoder's latent space mapping and the desired latent space mapping:

$$\log p_\theta(\mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log\left[\frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\right]\right] + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log\left[\frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})}\right]\right] \tag{3}$$

$$= \text{ELBO} + D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}|\mathbf{x})). \tag{4}$$

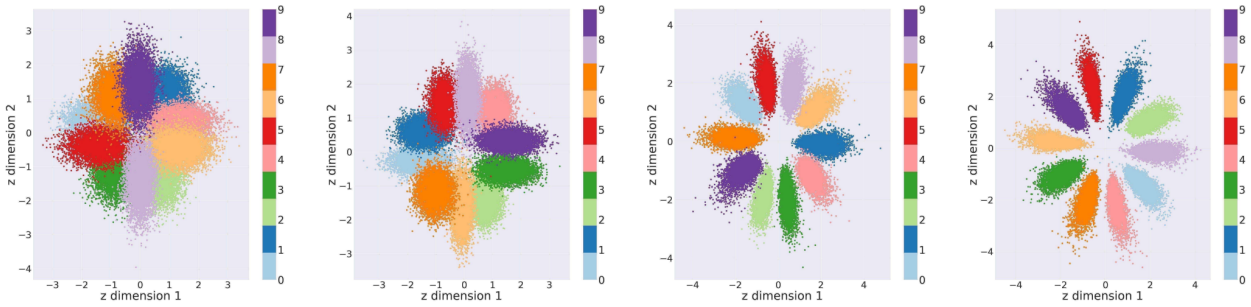

Figure 5: A 2D MNIST latent space visualization for VAEs with different β values from left to right: 1.0, 0.5, 0.1, 0.05. When $\beta >> 1$, there will be a collapse in any structure in the latent space and when $\beta << 1$, the latent space will approach a look-up table. Figure courtesy of [29].

Since the KL divergence is necessarily non-negative, the ELBO is also a lower bound on the log-likelihood of $p(\mathbf{x})$. Note that most VAEs incorporate a β parameter which determines weighting of the KL divergence in the loss function which effects how strict the adherence to a Gaussian distribution for the latent space representations should be (see figure 5). For a more thorough description of VAEs, readers are directed to the excellent tutorials given in [23] and [11].

2.4 VAEs in the Deep Learning Trilemma

Compared to normal autoencoders, VAEs actually allow us to create meaningful novel data due to its structured latent space. We can now generate examples using arbitrary (or even conditional) vectors in our latent spaces that are less likely to be noise. However, even in well-trained VAEs, the generated images from latent vectors interpolated between the latent representations of true examples still tend to be blurry like those seen in figure 3. However, autoencoders by their nature are very good at capturing the variation in the data distribution and compared to the other DGMs we will discuss are relatively quick to train and produce samples from since we only need to pass latent vectors through a single neural network. One of the key advantages of VAEs is that we can utilize their latent spaces for a variety of applications. For example, similar to how one could do classification on a reduced dimension representation of data with a process like PCA, VAEs can make classification tasks easier

by learning class boundaries in the latent space. Consider figure 5 for example where the MNIST digits are clearly differentiated. By extension, these latent representations can make creating captions or labels for images a lot easier as well [34].

3 Generative Adversarial Networks

One of the biggest challenges for DGMs is coming up with a good loss function for training the networks, i.e. a metric that can tell us how good the generated data is (see figure 1). In VAEs and, as we will see later for DDMs, the choice of loss is the reconstruction loss which directly compares reproduced images to their original counterparts using some L_p norm. However, this approach has a major drawback – namely that a perfectly reproduced image that is slightly shifted can have a higher loss than an incorrect image that happens to have more overlap. An alternative for the loss could be to use another *network* trained specifically to determine if a given sample is part of the training data set. This is the approach suggested by Goodfellow et. al. in their seminal work on *Generative Adversarial Networks* [16].

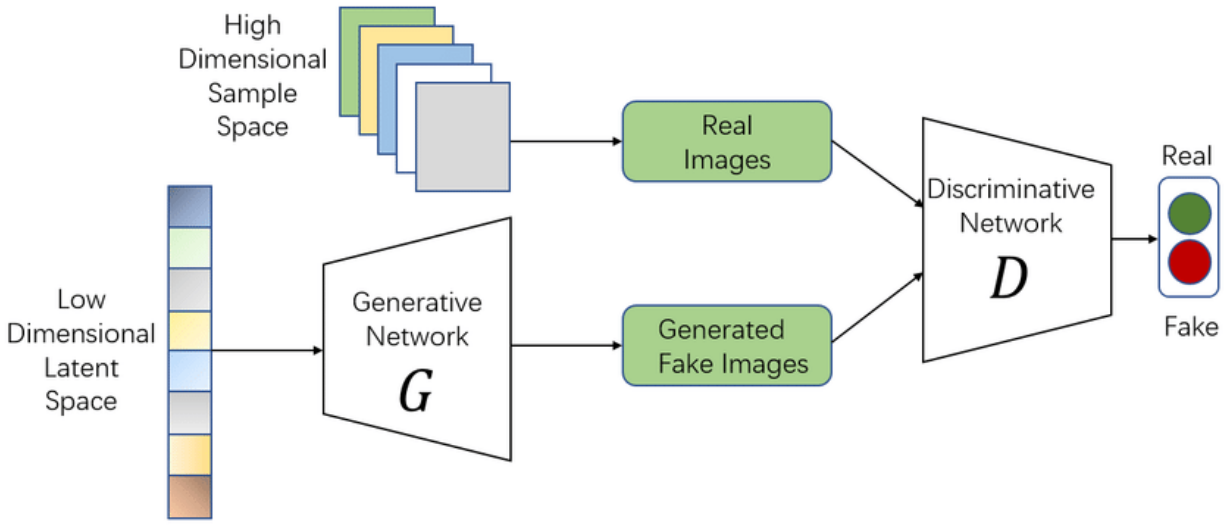

Figure 6: The GAN framework consists of the generator G which takes in a random low dimensional vector to produce an image and the discriminator which takes in either the generated image or a true image from the training data and determines if the produced sample is real or fake. Figure courtesy of [3].

3.1 The Two Player Formulation

GANs learn to produce examples that closely resemble the training data using an adversarial two player game where each player is a neural network. The first network is the *generator* G which takes in a low-dimensional random noise vector z and produces a sample $G(z)$ to try to fool the second network, the *discriminator* D which tries to determine if a given sample is from the real data or if it is a fake sample produced by the generator. This game can be represented as a minimax formulation with a value function $V(G, D)$ defined by:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data(x)}}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]. \tag{5}$$

Each network then has its own goal. The generator is trying to maximize its probability of fooling the discriminator or equivalently, minimize the loss function $\log(1 - D(G(z))$. Note that $D(G(z))$ is the discriminators prediction of whether the generated sample $G(z)$ is a true or fake with 0 corresponding to a prediction that $G(z)$ is definitely fake, and 1 corresponding to a prediction that $G(z)$ is definitely real. Meanwhile, the generator is trying to maximize its probability of assigning the right label to real samples $D(x)$ and the fake, generated samples $D(G(z))$. Note that in equation 5, $x \sim p_{data(x)}$ and $z \sim p_z(z)$ denote that x is a sample drawn from the true data distribution, and z is a sample drawn from the latent distribution. Ideally when training these networks, the two should improve in lock-step with each other. If the discriminator is too good initially, the generator will never be able to learn what it should change in its network to fool the discriminator. Ultimately, by the time the generator has completed training, the discriminator should have a 50/50 chance of correctly identifying any given image – i.e., the generated images should be indistinguishable from the true images. Once training is complete, the discriminator is usually thrown away. Note that we cannot use the discriminator as a classifier to determine real or fake images because we have created the generator which was explicitly designed to fool it.

3.2 Overcoming Mode Collapse

In practice, GANs have been able to achieve high fidelity and even pass visual Turing tests for image generation where humans were also unable to distinguish between real and generated samples [41]. However, GANs suffer a major drawback. Let's consider again that in the GAN formulation, we are not predefining what a "good" image looks like for the generator. Instead, we are simply training the generator to fool the discriminator. The generator notably *never sees the actual images*. Consequently a common side-effect of this framework is that the generator can learn a few examples which consistently fool the discriminator, causing the generator to repeatedly produce the same output for various random input latent vectors. This phenomenon is known as *mode collapse* and can be more generally described as when the GAN can only produce one/a few modes of a distribution such as only producing one type of cat or producing an image of a cat with only one type of framing (e.g., a close up shot of the cat's head) [30, 41, 47]. Various approaches to solve this issue have been proposed, all of which involve an adjustment or addition to the basic two-player formulation.

The first approach is *minibatch discrimination* as proposed by [41]. They clarify that the reason why mode collapse occurs is because, for a given batch, it is possible for the gradient of the discriminator to point in the same direction for many similar inputs. Since the discriminator in the standard GAN formulation handles each example independently, there is no way to tell the generator to produce more variety in its outputs. This results in the generator re-producing the same output that the discriminator currently believes is real. But, even after the discriminator learns that the result is fake, gradient descent is unable to separate the identical outputs of the generator and learning stops. With this in mind, [41] propose a modification to the discriminator, allowing it to evaluate multiple examples (i.e., a minibatch) at once and measure the closeness of the samples in that minibatch. The discriminator, then, still evaluates the probability that a given sample is real, but now can use the closeness metric to adjust its gradients. The authors of [41] show that this adjustment allows them to train a GAN to create realistic and visually appealing samples much more

quickly.

However, [30] point out that the minibatch discrimination "trick" is too computationally expensive, preventing GANs that use it from scaling up to larger datasets like ImageNet. Instead, they propose a different modification to the GAN framework to mitigate mode collapse which they call *Dual Discriminator GANs* (D2GANs). Instead of the standard two-player GAN formulation, they propose a *three-player* formulation consisting of one generator and two discriminators, one which favors outputs from the data and another which favors outputs from the generator. They base their argument on combining the statistical properties of minimizing the asymmetric KL divergence (i.e., between data and model) which they claim has been shown to cover multiple modes of the data but is likely to produce poor samples, and minimizing reverse KL divergence (i.e., between model and data) which tends to produce better samples but can lead to mode collapse. They argue that the standard GAN formulation leads to mode collapse because it minimizes *Jensen-Shannon* (JS) divergence (a symmetric metric that combines KL and reverse KL divergences), which has been empirically shown to mimic the effect of minimizing reverse KL divergence. Instead, they show that training a generator to fool their dual discriminators (which act as proxies to KL and reverse KL divergences) results in the best of minimizing both KL divergences and avoids mode collapse.

When it came to measuring performance, [41] focused on general perception tests and did not evaluate mode collapse specifically while [30] extensively showcase how their method overcomes mode collapse. In particular, [30] present a convincing mathematical foundation for their approach and their presentation of D2GANs performance on a synthetic data set was particularly compelling. Granted, the objectives of the two papers were different and [30] were specifically targeting the issue of mode collapse. Regardless, the dual discriminator approach presented in [30] is also more intuitive and may be easier for readers to implement as a simple architecture modification compared to the minibatch discrimination technique described by [41].

3.3 Deep Convolutional GAN

In terms of high performance GAN models, one of the most popular is *Deep Convolutional GAN* (DCGAN) introduced by [35]. Convolutional Neural Networks (CNNs) are known for their ability to learn spatial information for data like images by learning convolutional filters (also known as kernels) that can describe patterns in images. As a result, CNNs have often been used for image classification tasks [31]. DCGAN adapts CNNs for the discriminator (which is effectively just a classifier) but also utilizes a CNN for the generator. However, instead of the usual convolution process of reducing higher dimensional data down to lower dimensions by pooling the results of convolutions, [35] reverse the process for the generator, taking in a 100 dimensional z vector and "up-convolving" to the original image's dimensionality.

One particularly interesting thing that [35] show is that, like for VAEs, it is possible to explore the latent space of GANs to produce examples that mix properties of different types of images. Note that this includes an extension to DGMs for conditional generation which is not really discussed in this report, but it is too interesting not to include. Specifically, [35] show that they can add and subtract the latent space vectors of different classes to produce new examples that combine features of the classes (see figure 7). This in turn suggests a highly-structured latent space that can be traversed similar to those seen for VAEs. In fact, [35] also depicts the effect of interpolation between examples in the latent space, successfully showing, for example, interpolating between lighting in a room in the morning to a very similar room being lit in the evening.

4 Denoising Diffusion Models

While denoising diffusion models were first introduced in 2015 by [43], they did not take over GANs in popularity until 2021 when [9] showed that DDMs were able to overtake the performance of GANs in image generation. Since then, DDMs have been the primary models

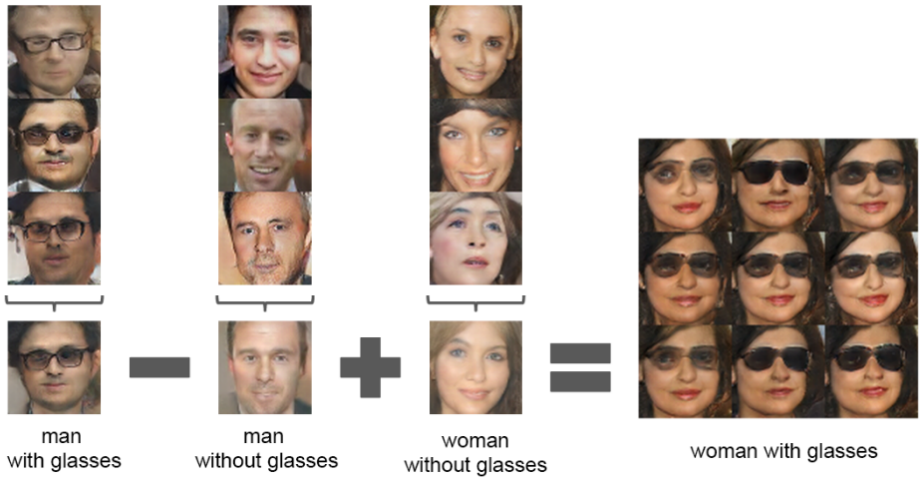

Figure 7: The authors of [35] were able to perform vector arithmetic in the latent space of DCGAN to produce examples which combine features of different classes. Figure courtesy of [35].

used for commercial AI image generation including OpenAI's *Dall-E*, Google's *Imagen*, and Stability AI's *Stable Diffusion*. Compared to GANs, diffusion models have a number of preferred qualities. For example, DDMs have more stable training since they do not need to balance two networks simultaneously and are not prone to mode collapse. Additionally, unlike GANs, DDMs are *maximum likelihood estimators*. This means that DDMs can directly maximize the likelihood that the learned distribution approximates the true data distribution and hence are, similar to VAEs but opposed to GANs, good at learning the diversity of the data.

4.1 Evaluating Generative Models

In this section, we will take a brief detour to explore how generative models are evaluated inspired by the claim in [9] that diffusion models beat GANs. Coming up with a metric for evaluating the performance of generative models is difficult since there is no real way to *quantify*, for example, artistic intent or style. Many papers which discuss the performance of DGMs often end up relying on qualitative measurements and resort to manual inspection of visual fidelity. However, this approach is subjective, time consuming, and potentially misleading. This has led to the creation of several metrics which aim to quantify the performance

of DGMs [58].

A common approach for determining good metrics is to determine which metrics closely match with manual inspection (i.e., metrics that align closely with human judgement). Of the many metrics proposed, the two most common are the *Inception Score* (IS) introduced in [41] and the *Fréchet Inception Distance* (FID) score introduced in [19]. Note that FID score was the metric used to support the claim that diffusion models beat GANs on image synthesis in [9]. The IS is among the most used metrics in the literature. It uses a pretrained inception network $\mathcal{M}$ from [48] trained on ImageNet [8] to compute

$$\text{IS}(\mathbb{P}_g) = e^{\mathbb{E}_{x\sim\mathbb{P}_g}[\text{KL}(p_{\mathcal{M}}(y|\mathbf{x})\|p_{\mathcal{M}}(y))]} \tag{6}$$

where $p_{\mathcal{M}}(y|\mathbf{x})$ is the label distribution of the data x as predicted by $\mathcal{M}$, $p_{\mathcal{M}}(y)$ is the marginal of $p_{\mathcal{M}}(y|\mathbf{x})$ over the probability measure $\mathbb{P}_g$ [58]. Notably, IS only works for small, square images and can return artificially high scores for small sample sizes and artificially low scores when produced samples include unusual images that were not part of the original data set. Similar to IS, FID also utilizes the inception network from [48]. FID is based on the idea that some real distribution $p_r(.)$ and some generated distribution $p_g(.)$,

$$p_r(.) = p_g(.) \iff \int_x p_r(x)f(x)dx = \int_x p_g(x)f(x)dx \tag{7}$$

where basis functions $f(.)$ span the space in which $p_r(.)$ and $p_g(.)$ live. If we let $f(x)$ be first and second degree polynomials, we can determine the first and second moments of the distributions which can give us the mean and covariances of the Gaussians which approximate the distributions. Then, if we let μ_r, C_r be the mean and covariance of the real distribution and let μ_g, C_g be the mean and covariance of the generated distribution, the FID of these

distributions can be calculated as:

$$d^2((\mu_r, C_r), (\mu_g, C_g)) = \|\mu_r - \mu_g\|_2^2 + \mathrm{Tr}(C_r + C_g - 2(C_r C_g)^{1/2}). \tag{8}$$

For more information regarding different evaluation metrics for DGMs, we refer the reader to [58]. Note that even these popular metrics have bias and are not perfect measures. There continues to be research in improving these metrics as seen in [5].

4.2 The DDM Formulation

Diffusion models can be described in one sentence as *reversing a thermodynamic process*. Consider the analogy of a drop of paint in water. When that paint is first dropped into the water, it still holds most of its initial shape but over time, the paint droplet *diffuses* throughout the water until it is completely mixed. Now, with a diffusion model, what we are trying to learn is the process of transforming that fully mixed water back into *one* of the potential original paint droplets that could have produced that fully-mixed solution. In practice, for images, this process is done by adding random Gaussian noise to an image until it is entirely white noise then learning a reverse process which iteratively removes the noise until we get back to some noise-free image (see figure 8). Note that a good portion of the following section is based on the excellent tutorial given in [25] and we recommend this resource for readers interested in a more thorough description. This section is also based on [20] which describes the particular probabilistic denoising diffusion process shown below.

The forward diffusion process adds noise to an image in T steps where the image $\mathbf{x}_t$ can be generated from $\mathbf{x}_{t-1}$ using the kernel defined by:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_t - 1, \beta_t \mathbf{I}). \tag{9}$$

Notice that we are also slightly downscaling the image at each step by the β_t parameter.

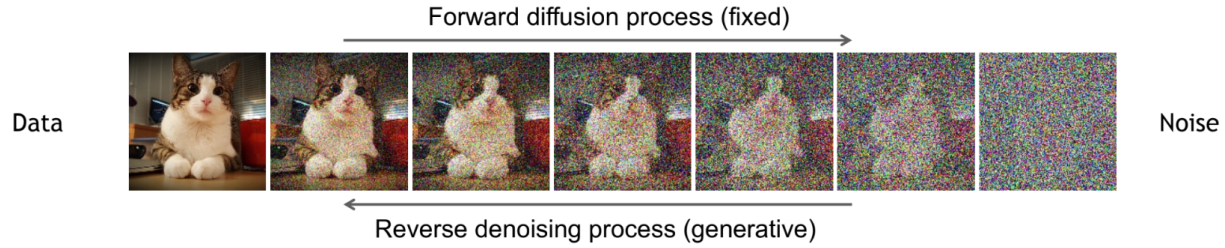

Figure 8: DDMs consist of two processes, the forward diffusion process and the learned reverse denoising process. For the forward process, we iteratively add noise until our image is entirely white noise and in the reverse process, we use neural networks to learn how to produce iteratively less noisy images until we return to a noise-free image. Figure courtesy of [25].

Since we are dealing with Gaussian distributions, we can express the *total* noise from x_0 to x_T as a product of each individual step:

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}). \tag{10}$$

For the forward process, to get an image at step t, we do not need to generate each intermediate sample. Instead, again since we are using a simple Gaussian kernel, we can define a scalar $\bar{\alpha}_t$

$$\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s) \tag{11}$$

such that the diffusion kernel for time step t can be expressed in terms of that scalar as

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}). \tag{12}$$

Therefore, to get the diffused image at some time step t, we can just apply the diffusion kernel for time step t to the initial input image x_0:

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{(1-\bar{\alpha}_t)}\epsilon \tag{13}$$

where $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Note that we set the β_t values such that $\bar{\alpha}_T \to 0$ so

$$q(\mathbf{x}_T|\mathbf{x}_0) \approx \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I}). \tag{14}$$

This way, at the end of the diffusion process, the diffused data will have a normal Gaussian distribution. As a side note, one thing that we have control over is how much noise we add in each step. The determination of how much noise is added for each iteration is known in the literature as the *noise schedule* and the particular decision for what noise schedule to use is an engineering design choice for any particular model. Note that similar to equation 1, we can express the diffused data distribution $q(\mathbf{x}_t)$ in terms of the input data distribution $q(\mathbf{x}_t)$ as:

$$q(\mathbf{x}_t) = \int q(\mathbf{x}_0, \mathbf{x}_t) d\mathbf{x}_0 = \int q(\mathbf{x}_0) q(\mathbf{x}_t|\mathbf{x}_0) d\mathbf{x}_0. \tag{15}$$

Therefore, we can sample $\mathbf{x}_t \sim q(\mathbf{x}_t)$ by first sampling $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ then sampling $\mathbf{x}_t \sim q(\mathbf{x}_t|\mathbf{x}_0)$.

Now, for the reverse process, we begin with a noisy sample $\mathbf{x}_T \sim \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$ and we want to iteratively sample $\mathbf{x}_{t-1} \sim q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ where $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ is the true denoising distribution. However, determining the true denoising distribution is generally intractable. So, instead, if β_t is small, we can approximate the reverse process by assuming each reverse step is also approximately normal with some mean and variance. Then, we can train some neural network to learn the mapping from a noisy image to a slightly less noisy image. More specifically, we are learning the approximate denoising distribution

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \sigma_t^2 \mathbf{I}) \tag{16}$$

where $\mu_\theta(\mathbf{x}_t, t)$ is our trainable neural network such as a U-net or denoising autoencoder. Also, we are often not learning how to necessarily produce a less noisy image directly, but

rather trying to determine the noise *itself* that was added to the image at each step. To actually train these neural networks, we can utilize variational inference, i.e., the *variational upper bound*, similar to what is done for training VAEs. However, the mathematics that describe this process and the loss function used are rather complicated, so the reader is referred to [25] for more details. Note also, that this formulation makes diffusion models inherently slow because to go from random noise to an image, we can only take small steps. Consequently, training DDMs is expensive since we have to train multiple individual neural networks, one for each step. This also makes creating samples from DDMs substantially more expensive since we need to pass our initial noisy image through several denoising neural networks instead of the single neural networks used to create samples for VAEs and GANs.

5 Tackling the Deep Learning Trilemma

As mentioned in §1.3 and throughout this report, none of the current DGM frameworks encapsulate all three of the desiderata for generative models (see figure 2). In this section, we will review some of the recent literature that tries to tackle the trilemma by either improving existing model architectures or combining architectures together. Note that we already mentioned a few techniques for improving the mode collapse problem of GANs in §3.2 and many more research papers have demonstrated other techniques not mentioned in the sections below including [7, 22, 44, 50].

5.1 Vector Quantized VAEs and GANs

Vector quantized VAEs (VQVAE) first introduced in [52] and later improved by [36], update the VAE formulation to create higher quality samples which can rival the fidelity of GAN models but avoid the mode collapse and low diversity issues present in GANs. The way they achieve this is by using discrete latent variables instead of continuous normal distributions.

In normal VAEs, we assume the posterior $q(\mathbf{z}|\mathbf{x})$ and prior $p(\mathbf{z})$ are normally distributed with diagonal variance and the encoder predicts the mean and variances of the posterior. However, in VQVAEs, the posterior and prior distributions are categorical. A new vector quantization layer is added to the framework in place of the standard latent space in VAEs which represents a dictionary of embeddings. The output of the encoder is passed to this vector quantization layer which then calculates the distance of the encoded latent representations to each of the dictionary embedding to determine the closest embedding. The chosen embedding is then used to generate samples through the decoder. For training VQVAEs, we are training the encoder and decoder like in normal VAEs but now also training the dictionary embeddings in the vector quantization layer. With this discretization formulation, we lose the ability to interpolate between categories like we could with traditional VAEs. However, many real-world objects are discrete and interpolating between them does not make much sense. For example, we do not really want to interpolate between, say, a "car" and a "cat". Therefore, by discretizing these categories, we can more easily model each category instead of needing to learn the potentially complex interdependencies between categories. As a side note, the authors of the paper also claim that their VQVAE model does not suffer from *posterior collapse* which normally is an issue for VAEs where the signal from an input $\mathbf{x}$ is too weak or noisy causing the $\mathbf{z}$ samples drawn from the posterior $q(\mathbf{z}|\mathbf{x})$ to be ignored by the decoder. VQVAEs have also since been extended to video generation [53, 60], and text-to-speech generation [62].

Vector quantized GAN (VQGAN) [12, 63] is an improved version of VQVAEs that introduce an adversarial loss to improve reconstruction even further. Like in VQVAEs, VQGANs first encode images down to lower dimensional discrete latent codes. Then, they use non-local attention blocks which are similar to transformers which allow them to capture more distant interactions in the image using fewer layers. Notably, both VQVAEs and VQGANs use a CNN framework. For VQGANs, the reason they use transformers is to model distant relationships in *pixel space* for high resolution images. They use the transformers to model

the composition of a high-resolution image in terms of many smaller CNNs so they train those many small CNNs instead of a single large CNN.

5.2 Latent Diffusion Models

Latent Diffusion Models (LDMs) introduced in [39] are very similar to DDMs except that they apply the diffusion process to a latent representation of images instead of working in the original pixel space. Hence, they can be considered a combination of diffusion models with AEs. By performing diffusion in the latent space, LDMs are far more computationally efficient to train and generate samples, allowing for cheaper training and even higher-resolution image generation. They also make conditional generation much easier since those conditions can be applied to the lower dimensional latent space. The architecture for LDMs is very similar to traditional DDMs except that they begin the diffusion process by first passing the image through an encoder which creates the initial mapping to the latent space. Then, the forward and reverse diffusion processes occur in the latent space along with and conditioning. To incorporate conditioning, LDMs utilize cross-attention in each denoising U-net in the reverse process (see figure 9).

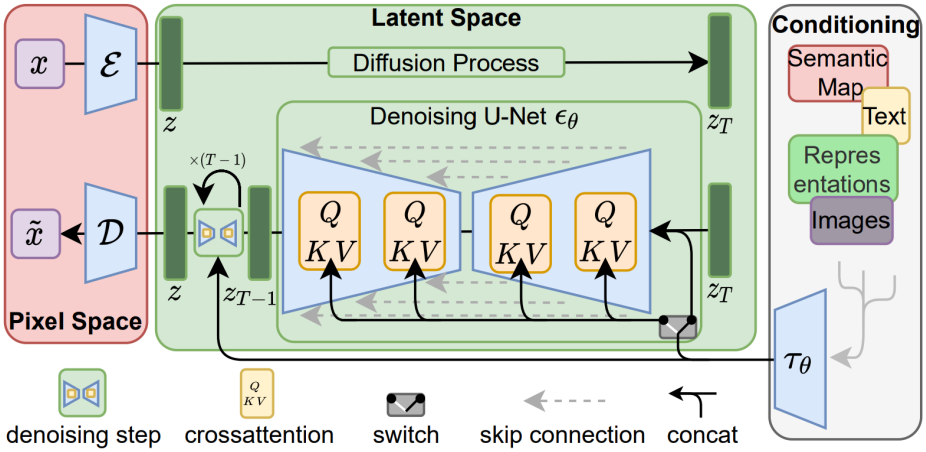

Figure 9: The architecture for latent diffuse models which perform the diffusion process and conditioning in the latent space representation of the images. Figure courtesy of [39].

5.3 Denoising Diffusion GANs

Denoising Diffusion GANs (DDGANs) first introduced in [25] aim to overcome the slow sampling of DDMs by replacing the multiple denoising U-Nets or autoencoders in the reverse diffusion process with fewer conditional GANs. They argue that the slow sampling of DDMs is caused by the Gaussian assumption in the denoising step which is justified only for small steps. This necessitates a very large number of neural nets in the reverse process. To enable larger denoising with larger steps and consequently reduce the total number of denoising steps, they introduce the use of multi-modal conditional GANs. Due to the drastically reduced number of reverse steps, their model boasts up to a 2000x improvement in sample generation while outperforming traditional GANs in mode coverage and sample diversity. They incorporate the GANs by training a generator to reproduce an approximation of the original $\mathbf{x}'_0$ from a noisy image $\mathbf{x}_t$ to then create a fake less noisy image $\mathbf{x}'_{t-1}$ which is then compared with a discriminator to the actual less noisy image $\mathbf{x}_{t-1}$ (see figure 10).

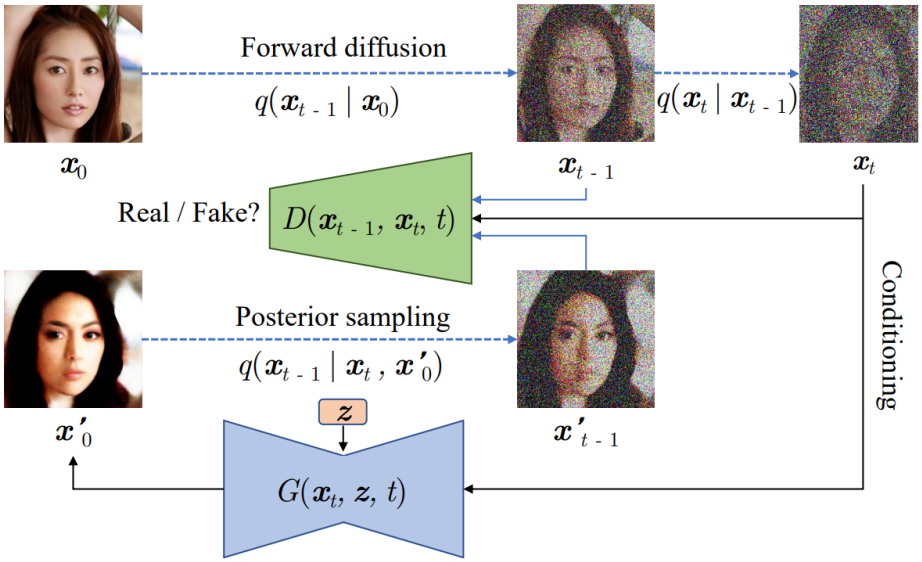

Figure 10: The model architecture for DDGANs which improve the sampling speed of DDMs by replacing the denoising U-Nets with fewer reverse steps utilizing a GAN architecture to learn the noise. Figure courtesy of [25].

6 Conclusion

With the ubiquity of DGMs today, this survey aimed to provide a basic foundation of the three most popular DGMs along with their limitations and current research trends. In this survey, we briefly explored historic approaches for generative learning before introducing the essential deep learning formulation used by today's DGMs. We also introduced the deep learning trilemma which provides a framework for putting into perspective the advantages and disadvantages of each of the DGMs. Then, we described in some details the formulations for Variational Autoencoders, Generative Adversarial Networks, and Denoising Diffusion Models. Along the way, we described how each of these formulations contribute to their placement in the deep learning trilemma and discussed some of the specific research which aims to reduce the negative aspects of these formulations. We also took a detour to describe DGMs are evaluated empirically. Finally, we discussed some recent literature which aims to improve existing DGMs through small adjustments of the formulation such as by introducing vector quantization or by combining different DGMs into a single model to reap the benefits of each.

Looking at the current state of research in this field, we can obviously expect the trend of trying to overcome the deep trilemma to continue. We can expect to see many more combinations and adjustments to existing models in the near future. We can also expect that the training objectives will get increasingly complex. We have already seen the improvement of generative networks in image resolution starting from the smallest data sets like MNIST and CIFAR-10 to larger data sets including ImageNet. A lot of recent research has also focused on complex and high-resolution conditional video generation tasks [21, 49, 53, 60].

In addition to the continual improvement of DGMs, one very interesting future research direction is in the creation of *smaller* network architectures that are far cheaper to train while still keeping up with the performance of state-of-the-art models. Recently, in 2023, a leaked report from Google entitled "We have no moat, and neither does OpenAI" [33]

explained how open-source models that cost on the order of hundreds of dollars are starting to become competitive against the expensive and extremely large AI models being trained by companies like Google and OpenAI for millions of dollars. While this particular memo was discussing the large language models like Google Bard and ChatGPT being caught up with open source models like LLaMa and Alpaca, we can imagine a similar trend in image generation models and other generative AI models. Besides the obvious cost-saving benefits of smaller models, there are also various environmental benefits which reduce the carbon emission impact of large GPU clusters being used to train expensive models.

References

[1] Guillaume Alain and Yoshua Bengio. "What regularized auto-encoders learn from the data-generating distribution". In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 3563–3593.

[2] Jean-Pierre Briot, Gaëtan Hadjeres, and François-David Pachet. "Deep learning techniques for music generation–a survey". In: *arXiv preprint arXiv:1709.01620* (2017).

[3] Likun Cai et al. "Utilizing Amari-Alpha Divergence to Stabilize the Training of Generative Adversarial Networks". In: *Entropy* 22 (Apr. 2020), p. 410.

[4] Hanqun Cao et al. "A survey on generative diffusion models". In: *IEEE Transactions on Knowledge and Data Engineering* (2024).

[5] Min Jin Chong and David Forsyth. "Effectively unbiased fid and inception score and where to find them". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 6070–6079.

[6] Lucas Pinheiro Cinelli et al. *Variational methods for machine learning with applications to deep networks*. Springer, 2021.

[7] Bin Dai and David Wipf. "Diagnosing and enhancing VAE models". In: *arXiv preprint arXiv:1903.05789* (2019).

[8] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[9] Prafulla Dhariwal and Alexander Nichol. "Diffusion models beat gans on image synthesis". In: *Advances in neural information processing systems* 34 (2021), pp. 8780–8794.

[10] Tim Dockhorn, Arash Vahdat, and Karsten Kreis. "Score-based generative modeling with critically-damped langevin diffusion". In: *arXiv preprint arXiv:2112.07068* (2021).

[11] Carl Doersch. "Tutorial on variational autoencoders". In: *arXiv preprint arXiv:1606.05908* (2016).

[12] Patrick Esser, Robin Rombach, and Bjorn Ommer. "Taming transformers for high-resolution image synthesis". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 12873–12883.

[13] Yang Gao, Rita Singh, and Bhiksha Raj. "Voice impersonation using generative adversarial networks". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2018, pp. 2506–2510.

[14] Lovedeep Gondara. "Medical image denoising using convolutional denoising autoencoders". In: *2016 IEEE 16th international conference on data mining workshops (ICDMW)*. IEEE. 2016, pp. 241–246.

[15] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[16] Ian Goodfellow et al. "Generative adversarial nets". In: *Advances in neural information processing systems* 27 (2014).

[17] Jie Gui et al. “A review on generative adversarial networks: Algorithms, theory, and applications”. In: *IEEE transactions on knowledge and data engineering* 35.4 (2021), pp. 3313–3332.

[18] GM Harshvardhan et al. “A comprehensive survey and analysis of generative models in machine learning”. In: *Computer Science Review* 38 (2020), p. 100285.

[19] Martin Heusel et al. “Gans trained by a two time-scale update rule converge to a local nash equilibrium”. In: *Advances in neural information processing systems* 30 (2017).

[20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. “Denoising diffusion probabilistic models”. In: *Advances in neural information processing systems* 33 (2020), pp. 6840–6851.

[21] Jonathan Ho et al. “Imagen video: High definition video generation with diffusion models”. In: *arXiv preprint arXiv:2210.02303* (2022).

[22] Diederik Kingma et al. “Variational diffusion models”. In: *Advances in neural information processing systems* 34 (2021), pp. 21696–21707.

[23] Diederik P Kingma, Max Welling, et al. “An introduction to variational autoencoders”. In: *Foundations and Trends in Machine Learning* 12.4 (2019), pp. 307–392.

[24] Diederik P Kingma and Max Welling. “Auto-encoding variational bayes”. In: *arXiv preprint arXiv:1312.6114* (2013).

[25] Karsten Kreis, Ruiqi Gao, and Arash Vahdat. “Denoising diffusion-based generative modeling: Foundations and applications”. In: *CVPR*. 2022.

[26] Solomon Kullback and Richard A Leibler. “On information and sufficiency”. In: *The annals of mathematical statistics* 22.1 (1951), pp. 79–86.

[27] Chongxuan Li et al. “Triple generative adversarial nets”. In: *Advances in neural information processing systems* 30 (2017).

[28] Mehdi Mirza and Simon Osindero. “Conditional generative adversarial nets”. In: *arXiv preprint arXiv:1411.1784* (2014).

[29] Martin Mundt et al. “Unified Probabilistic Deep Continual Learning through Generative Replay and Open Set Recognition”. In: *Journal of Imaging* 8 (Mar. 2022), p. 93.

[30] Tu Nguyen et al. “Dual discriminator generative adversarial nets”. In: *Advances in neural information processing systems* 30 (2017).

[31] Keiron O’shea and Ryan Nash. “An introduction to convolutional neural networks”. In: *arXiv preprint arXiv:1511.08458* (2015).

[32] Zhaoqing Pan et al. “Recent progress on generative adversarial networks (GANs): A survey”. In: *IEEE access* 7 (2019), pp. 36322–36333.

[33] Dylan Patel and Afzal Ahmad. “Google “We Have No Moat, And Neither Does OpenAI.””. In: *SemiAnalysis. May* 4 (2023), p. 2023.

[34] Yunchen Pu et al. “Variational autoencoder for deep learning of images, labels and captions”. In: *Advances in neural information processing systems* 29 (2016).

[35] Alec Radford, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks". In: *arXiv preprint arXiv:1511.06434* (2015).

[36] Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. "Generating diverse high-fidelity images with vq-vae-2". In: *Advances in neural information processing systems* 32 (2019).

[37] Lyle Regenwetter, Amin Heyrani Nobari, and Faez Ahmed. "Deep generative models in engineering design: A review". In: *Journal of Mechanical Design* 144.7 (2022), p. 071704.

[38] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. "Stochastic backpropagation and approximate inference in deep generative models". In: *International conference on machine learning*. PMLR. 2014, pp. 1278–1286.

[39] Robin Rombach et al. "High-resolution image synthesis with latent diffusion models". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 10684–10695.

[40] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*. Springer. 2015, pp. 234–241.

[41] Tim Salimans et al. "Improved techniques for training gans". In: *Advances in neural information processing systems* 29 (2016).

[42] Jürgen Schmidhuber. "Deep learning in neural networks: An overview". In: *Neural networks* 61 (2015), pp. 85–117.

[43] Jascha Sohl-Dickstein et al. "Deep unsupervised learning using nonequilibrium thermodynamics". In: *International conference on machine learning*. PMLR. 2015, pp. 2256–2265.

[44] Jiaming Song, Chenlin Meng, and Stefano Ermon. "Denoising diffusion implicit models". In: *arXiv preprint arXiv:2010.02502* (2020).

[45] Yang Song et al. "Solving inverse problems in medical imaging with score-based generative models". In: *arXiv preprint arXiv:2111.08005* (2021).

[46] Tiago Sousa et al. "Generative deep learning for targeted compound design". In: *Journal of Chemical Information and Modeling* 61.11 (2021), pp. 5343–5361.

[47] Akash Srivastava et al. "Veegan: Reducing mode collapse in gans using implicit variational learning". In: *Advances in neural information processing systems* 30 (2017).

[48] Christian Szegedy et al. "Rethinking the inception architecture for computer vision". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2818–2826.

[49] Sergey Tulyakov et al. "Mocogan: Decomposing motion and content for video generation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 1526–1535.

[50] Arash Vahdat and Jan Kautz. *NVAE: A Deep Hierarchical Variational Autoencoder*. 2021. arXiv: `2007.03898 [stat.ML]`.

[51] Arash Vahdat, Karsten Kreis, and Jan Kautz. "Score-based generative modeling in latent space". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 11287–11302.

[52] Aaron Van Den Oord, Oriol Vinyals, et al. "Neural discrete representation learning". In: *Advances in neural information processing systems* 30 (2017).

[53] Jacob Walker, Ali Razavi, and Aäron van den Oord. "Predicting video with vqvae". In: *arXiv preprint arXiv:2103.01950* (2021).

[54] Wei Wang et al. "Generalized autoencoder: A neural network framework for dimensionality reduction". In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 2014, pp. 490–497.

[55] Yasi Wang, Hongxun Yao, and Sicheng Zhao. "Auto-encoder based dimensionality reduction". In: *Neurocomputing* 184 (2016), pp. 232–242.

[56] Zhisheng Xiao, Karsten Kreis, and Arash Vahdat. "Tackling the generative learning trilemma with denoising diffusion gans". In: *arXiv preprint arXiv:2112.07804* (2021).

[57] Junyuan Xie, Linli Xu, and Enhong Chen. "Image denoising and inpainting with deep neural networks". In: *Advances in neural information processing systems* 25 (2012).

[58] Qiantong Xu et al. "An empirical study on evaluation metrics of generative adversarial networks". In: *arXiv preprint arXiv:1806.07755* (2018).

[59] Guorong Xuan, Wei Zhang, and Peiqi Chai. "EM algorithms of Gaussian mixture model and hidden Markov model". In: *Proceedings 2001 International Conference on Image Processing*. Vol. 1. 2001, 145–148 vol.1.

[60] Wilson Yan et al. "Videogpt: Video generation using vq-vae and transformers". In: *arXiv preprint arXiv:2104.10157* (2021).

[61] Ling Yang et al. "Diffusion models: A comprehensive survey of methods and applications". In: *ACM Computing Surveys* 56.4 (2023), pp. 1–39.

[62] Yusuke Yasuda, Xin Wang, and Junichi Yamagishd. "End-to-end text-to-speech using latent duration based on vq-vae". In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2021, pp. 5694–5698.

[63] Jiahui Yu et al. "Vector-quantized image modeling with improved vqgan". In: *arXiv preprint arXiv:2110.04627* (2021).