A Comparative Study Of Image Generation Models

Srijita Chandra, Haniyeh Fekrmandi, Benjamin Lee
Department of Computer Science
Iowa State University
Ames, Iowa

Abstract

The need for generated images have increased drastically in the past few years with new use cases being discovered. With increased use cases comes the need to find models and algorithms for image generation which are optimized to handle large amounts of data while also producing accurate images. In recent years there has been extensive research and the generative models that stand out the most include variational autoencoders, generative adversarial networks and diffusion models. For this paper our objective was to compare these three image generative models and gauge their performances in terms of the time taken to train and test as well as the quality of the images generated. We also talk about the advantages and disadvantages of each of these models compared to each other and in doing so we get a overall understanding of the models and determine which model to use according to our needs.

Introduction

Image generators, are a family of powerful tools that are trained to generate new images similar to the input data the have received. They are alternated autoencoders which are originally defined as a form of Neural Networks to manipulate and reduce the dimensionality of data. The architecture of an autoencoder, as shown in Figure 1, consists three major parts; an encoder, a decoder and a sampler containing a compressed, lossy version of input image. It can be claimed that autoencoders are a sort of compression algorithms, with relative qualities of being (i) data-dependent, (ii) being lossy and (iii) being trained automatically. The training data is fed to the encoder and the generated (or reconstructed) output, is given from the decoder. After the training phase is done, the architecture can be altered to be used based on our required task. If both the encoder and decoder are used, we call the network a *data manipulator*, if we keep only the decoder and not the encoder, we have a *dimensionality reduction* network and if we keep the encoder and not the decoder, we have ourselves a *data generator*. There are three well-known, stable deep learning image generative networks namely Variational AutoEncoders, Generative Adversarial Networks and Diffusion Models. This project aims to implement these three networks and compare them while trained on a uniform image dataset, based on their training speed, prediction accuracy and the convergence power of their loss function.
The remainder of this report, focuses on the related work, dataset used, network structures and details of them, results and experiments which includes a discussion.

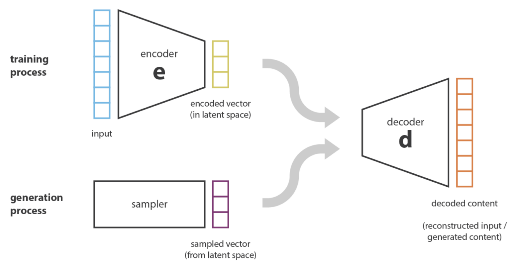
training process
encoder
e
encoded vector (in latent space)
input
decoder
d
generation process
sampler
sampled vector (from latent space)
decoded content
(reconstructed input / generated content)

Figure 1: AutoEncoder (AE) network architecure

Related Work

There has been extensive study about the effects of different image generation models. A similar work (Kırbıyık, Simsar, and Cemgil 2019) talks about the possibility to produce not only images similar to the ones in the training set but also novel images that belong to a new class.

The emergence of Variational AutoEncoders (VAEs) was after the introduction of autoencoders due to their similarity in math. VAEs are closely related to sparse autoencoders (Olshausen and Field 1996) except that unlike sparse autoencoders, they do not need to tune the sparsity penalty parameters. They are also extensively used in image denoising (Lee et al. 2006) with an additional capability to sample directly from prior distribution without performing Markov Chain Monte Carlo method as in (Bengio et al. 2013). So, what a VAE network does technically, is to maximize a maximum likelihood of the generated image given that is sampled from a latent space which is trained in the training process (Vincent et al. 2008). VAE has to deal with two things, first, choosing the right latent variables z that is the information they represent and taking the integral over the latent variable z. The details of how VAE deals with these complications is presented later in this report.

Generative adversarial networks (Goodfellow et al. 2014) have had a number of variations over the years with the newer models building on top of the convolutional GAN model or making changes to the primary structure of the model to get better performance. One such model is deep convolutional GAN (Radford, Metz, and Chintala 2015) and it discusses ways of making the latter more stable by setting constraints on the architectural constraints. Another variation of GAN, the conditional GAN (Mirza and Osindero 2014) talks about multi modal deep learning.

Diffusion model (Jonathan Ho 2020) might resemble flows and are optimized from Variational AutoEncoders, but diffusion models are designed so that approximate posterior has no parameters and top level latent has nearly zero mutual information with the data. Diffusion models for learning transition operators of Markov chains include infusion training, variational walkback, generative stochastic networks and others. the progressive decoding argument can also be seen in convolutional and related models and my lead to better design for subscale orderings or sampling strategies for autoregressive models.

Our goal for the project was to stick to the most basic use case of these models and see a comparison of their performance. This helped us get a deeper level understanding of the models and their functions.

Dataset

The MNIST dataset is a collection of 60,000 training images and 10,000 testing images and is a part of a larger NIST dataset (LeCun, Cortes, and Burges 2010). The images represent handwritten digits from 0 to 9. The images were modified from the original NIST dataset to fit in a 20x20 pixel box while preserving aspect ratio. The resulting images contain grey levels as a result of the anti-aliasing technique used by the normalization algorithm. The images are centered in a 28x28 image by computing the center of mass of the pixels, and translating the image so as to position this point at the center of the 28x28 field (LeCun, Cortes, and Burges 2010). The dataset is widely used in image processing models and some models have reached human level results using this dataset.

Variational Auto Encoders

Variational AutoEncoders (VAEs) are a particular type of autoencoders. A simple autoencoder looks like an hour glass and consists of an encoder, a decoder and at the bottleneck, there is a layer, the latent presentation of the input. After being fed to the encoder and going through the alternations, the input image becomes a compressed, low dimentional version by the time it reaches the latent representation layer. In a VAE however, this layer has the capability of keeping a distribution defined by a mean μ and a standard deviation σ rather than a fixed vector. This distribution is of the form of a normal distribution. The architecture of a VAE is depicted in Figure 2. The decoder, tries to generate an output that is as close to the input image as possible. This similarity is measured by the loss function, which is calculated as the

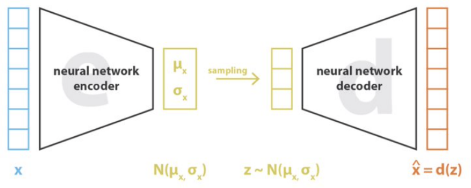

Figure 2: Variational AutoEncoder (VAE) network architecure

sum of two terms, reconstruction term and KL-divergence[1] term. Now, let's look at the problem from probabilistic point of view, since we want to train to get the maximum likelihood directly. Take the latent representation z belonging to a prior distribution $p(z)$. Having a data point x is sampled from the conditional likelihood distribution of $P(x|z)$ (as shown in Figure 3), we define encoder and decoder in terms of a probabilistic model

$$\begin{aligned} p(z) &\equiv \mathcal{N}(0,1) \\ p(x|z) &\equiv \mathcal{N}(f(z), cI) \qquad\qquad f \in Fc > 0 \end{aligned}$$

where the given probabilities belong to a normal distribution and f is a function and c is a constant. For a given data point x, maximizing the probability means $\hat{x} = x$ where $\hat{x}$ is a sample from the distribution of $p(x|z)$. So, the loss function

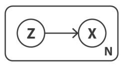

Figure 3: Probability model of the problem.

is formulated as follows

$$\begin{aligned} Loss &= \|x - \hat{x}\|^2 + KL[N(\mu_x, \sigma_x), N(0,1)] \\ &= \|x - d(z)\|^2 + KL[N(\mu_x, \sigma_x), N(0,1)] \end{aligned}$$

Now the whole problem becomes an optimization problem that should be solved to produce the optimal f^* using the formula

$$\begin{aligned} f^* &= argmax_{f \in F} \mathbf{E}_{z \sim q_x^*}(log\, p(x|z)) \\ &= argmax_{f \in F} \mathbf{E}_{z \sim q_x^*}(-\frac{\|x - f(z)\|^2}{2c}) \end{aligned}$$

which maximizes the likelihood probability of $\hat{x}$ being as close to x as possible by making the loss function better throughout the training process.

Generative Adversarial Network

Generative adversarial networks belong to a class of models that are built on the idea of having two networks compete against each other to outperform the other. Both these neural networks are assumed to be multilayer perceptrons.

[1]Kullback Leibler divergence is a natural measure of distance between distributions.

The first network is the generator network which takes a D-dimensional noise vector as input and produces an output image by passing the noise vector through a number of layers. The discriminator on the other hand is responsible for taking an input image and put a label of fake or real image after analysis as seen in figure 4. Both models are trained using only the highly successful backpropagation and dropout algorithms and sampled from the generative model using only forward propagation (Goodfellow et al. 2014).

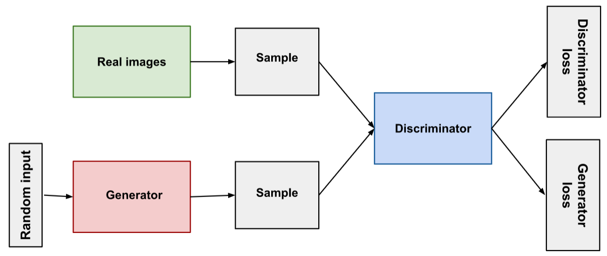

Figure 4: Generative Adversarial Network (GAN)

The standard GAN loss function, also known as the min-max loss, was first defined by Goodfellow et al. in 2014.

$$\mathbf{E}_{\mathbf{x}\sim\mathbf{p_{data}(x)}}[logD(x)] + \mathbf{E}_{\mathbf{x}\sim\mathbf{p_z(z)}}[log(1 - D(G(z)))]$$

Here, $logD(x)$ refers to the discriminator rightly classifying the real image from the fake. In essence, maximizing $log(1 - D(G(z)))$ helps to correctly label the image coming from the generator since it indicates the generator loss. The concept of penalizing is also prevalent in the two networks since every time the generator 'fools' the discriminator it gets rewarded, otherwise penalized. The same happens with the discriminator every time with the network getting rewarded when it is correctly able to distinguish a fake from a real image and penalized otherwise. The models compete against each other to outperform the opponent while increasing its own efficiency.

Diffusion Models

The essential idea of diffusion model is to systematically and slowly destroy structure in a data distribution through an iterative forward diffusion process. We then learn a reverse diffusion process that restores structure in data, yielding a highly flexible and tractable generative model of the data. (Jonathan Ho 2020) Diffusion model is a 2 process model, the first process is a forward process that gradually adds Gaussian noise to an image, until you end up with a pure noise image. The second process is a learn reverse diffusion process where a neural network is trained to gradually denoise an image starting from pure noise until you end up with an actual image. Figure 5 shows the illustration of the diffusion model where it starts from X_o, which is an actual image and gradually add Gaussian noise till time step z, where z will have a pure noise image. Figure 6 shows the illustration of the forward and reverse diffusion process. Both diffusion process is a markov chain, where a sequence of stochastic event where each step depends on the previous step. The forward process is considered a noise scheduler.

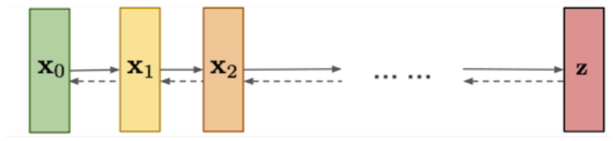

Figure 5: Diffusion model

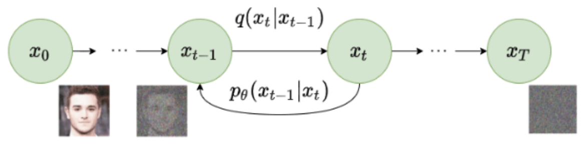

Figure 6: Forward and reverse diffusion process

There are many different ways to add noise into an image, but for simplicity, the experiment only look at adding noise linearly. Figure 7 shows the illustration of the U-net neural

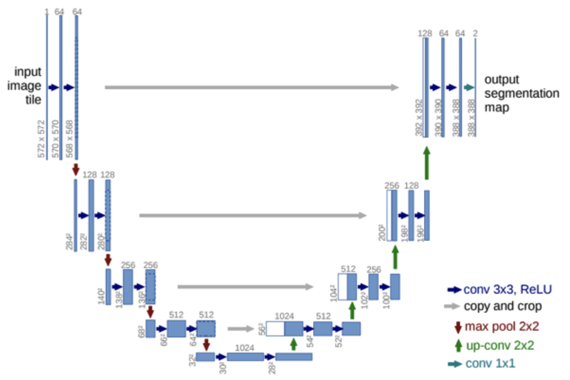

Figure 7: U-Net Neural Network

network. U-net neural network is a popular model for image segmentation. A characteristic of the U-net is that the output have the same shape as the input. The input image passes through a series of convolutional down-sampling layers until a bottleneck is reach, then it passes through a series of convolutional up-sampling layers. The reason U-net is widely used in diffusion model is because diffusion model have a constraint where the image needs to have the same dimension for input and output, making U-Net one of the best option. Figure 8 shows the loss function of the diffusion model,

$$L_{\text{simple}}(\theta) := \mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\right\|^2\right]$$

Figure 8: Diffusion loss function

which calculates the difference between the actual noise and the predicted noise.

Experimental Setup

The MNIST dataset was used for this experiment. The three different generative models were trained on the MNIST dataset using 4 different optimizers to get the best results and compare the models with the best results. The 4 different optimizer used in this experiment is Adam, SGD, RMSprop and Adagrad. For this experiment the models were training on only 50 epochs because there was not enough computational power to run more than 50 epochs as it was taking a much longer time. The tensorflow and keras python libraries was used in this experiment. For this experiment, google colab was used to run the three different models and the GPU runtime was used to train the model because that accelerated the speed of the training as compared to not using the GPU runtime.

Results

After comparing the best models generated after using different optimizers, we compared the models in terms of clarity of the images generated. The figures 9, 10, 11 are generated at the end of 50 epochs. We see that there is an increased clarity in the images created as the number of epochs progress for each of the models. But when we compare the last images of each of these models, we see that the clarity of the images of VAE is least as it appears to be hazy and ineligible. Diffusion model however performs the best out of the three models and we see that it attains readable numbers in less number of epochs. It then trains itself to perfect the images that it generates. GAN, on the other hand, does create legible images of the numbers but it requires a longer time compared to VAE to get to such results.

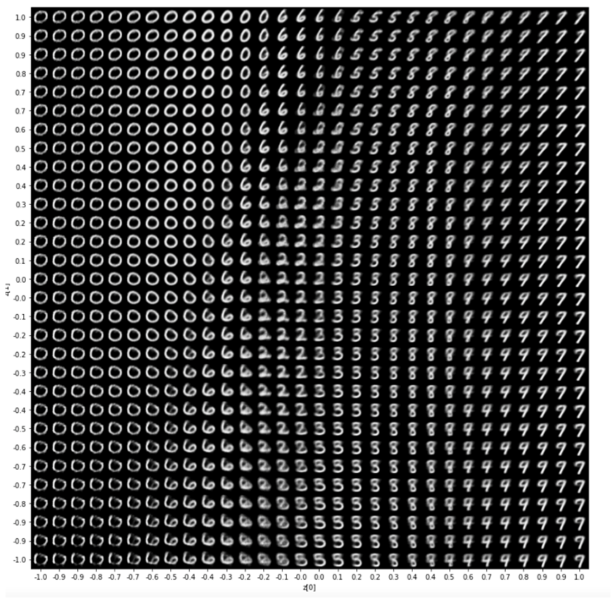

Figure 9: Images generated by VAE

One noticeable aspect while executing our code was time difference in the execution time for each of these models during the training process as seen in table .

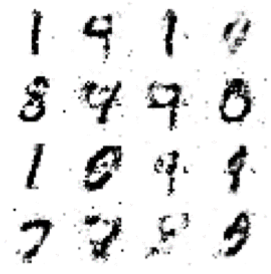

Figure 10: Images generated by GAN at end of 50 epoch

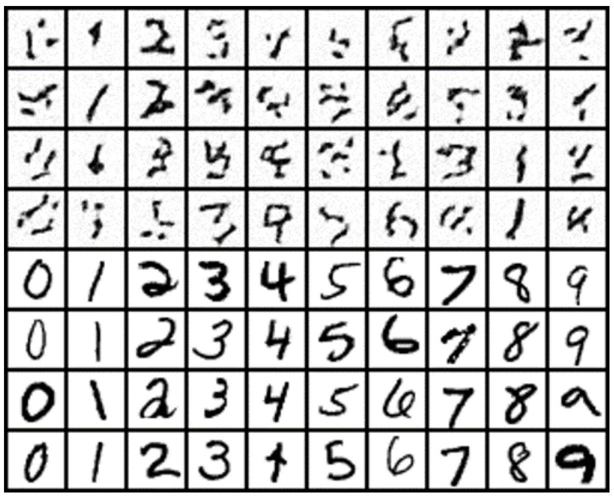

Figure 11: Images generated by Diffusion Model

Table 1: Execution time during training

Model	1st epoch time	50th epoch time
VAE	14.01 s	4.15 s
GAN	28.11 s	34.27 s
Diffusion Model	110.29 s	107.45 s

VAE has a similar time for all epochs since the basic implementation of VAE is to deconstruct and reconstruct an image but learn in every epoch to di it faster. The increase in the execution time from the first to the 50th epoch for the GAN model can be attributed to increase in the complexity of the images being generated and discriminated. With every epoch the model is expected to perform better to generate higher quality images. Diffusion models, on the other hand, have almost the same execution time throughout all the epochs but it comparatively much greater than VAE and GAN. This can be attributed to the multiple convolutional layers that it goes through in both forward and backward diffusion process since these individual layers take time to process the pixels. In every epoch the model is trying to break down an image and trying to build it up and, hence learning and training in the process. There is also a greater difference between the training time of VAE, GAN and diffusion model which attributes to the difference in the quality of the images generated.

Conclusion and Future Work

The project shed a light on the trade-off that exists while implementing these models. These models were built to outperform the previously existing models. But with increased clarify and accuracy in the images created came increase in the amount of time taken to implement and train the models. Considering diffusion models have the best performance in terms of the images generated, the trade-off is the time taken and the computational requirements to train it is much larger compared to the other two models. The opposite can be said about VAEs which require a much shorter time but also do not have the best generated images.

A viable future work would be to test the capabilities of these models would be to train them as parts of an ensemble model. This might have a better result in terms of both quality of image produced as well as time taken for training the model.

Repositories

For our project we referred to the following repositories for the model codes

1. https://github.com/keras-team/keras-io/vae.py
2. https://github.com/Zackory/Keras-MNIST-GAN
3. https://github.com/TeaPearce/Conditional_Diffusion_MNIST

References

[Bengio et al. 2013] Bengio, Y.; Yao, L.; Alain, G.; and Vincent, P. 2013. Generalized denoising auto-encoders as generative models. *ArXiv* abs/1305.6663.

[Goodfellow et al. 2014] Goodfellow, I. J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*.

[Jonathan Ho 2020] Jonathan Ho, Ajay Jain, P. A. 2020. Denoising diffusion probabilistic models.

[Kırbıyık, Simsar, and Cemgil 2019] Kırbıyık, Ö.; Simsar, E.; and Cemgil, A. T. 2019. Comparison of deep generative models for the generation of handwritten character images. In *2019 27th Signal Processing and Communications Applications Conference (SIU)*, 1–4.

[LeCun, Cortes, and Burges 2010] LeCun, Y.; Cortes, C.; and Burges, C. 2010. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist* 2.

[Lee et al. 2006] Lee, H.; Battle, A.; Raina, R.; and Ng, A. 2006. Efficient sparse coding algorithms. In Schölkopf, B.; Platt, J.; and Hoffman, T., eds., *Advances in Neural Information Processing Systems*, volume 19. MIT Press.

[Mirza and Osindero 2014] Mirza, M., and Osindero, S. 2014. Conditional generative adversarial nets.

[Olshausen and Field 1996] Olshausen, B. A., and Field, D. J. 1996. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature* 381(6583):607–609.

[Radford, Metz, and Chintala 2015] Radford, A.; Metz, L.; and Chintala, S. 2015. Unsupervised representation learning with deep convolutional generative adversarial networks.

[Vincent et al. 2008] Vincent, P.; Larochelle, H.; Bengio, Y.; and Manzagol, P.-A. 2008. Extracting and composing robust features with denoising autoencoders. In *Extracting and Composing Robust Features with Denoising Autoencoders*. Association for Computing MachineryNew YorkNYUnited States.