Deep Generative Models: Theory and Practice

An Zhang[*] [1] **Xinwei Zhang**[*] [1]

Abstract

Deep generative models have shown promising applications in various domains, including computer vision and natural language processing. While they have achieved significant practical success, the theoretical foundation of deep generative models still requires further development. In this paper, we aim to address this gap by presenting novel results in learning theory pertaining to several deep generative models. Moreover, we conduct a series of experiments to validate and support some of the proposed theoretical findings.

1. Introduction

Deep generative models have revolutionized the field of machine learning by enabling the generation of realistic and diverse data samples. These models, built upon the foundations of deep learning, have demonstrated remarkable capabilities in generating images, music, text, and even complex 3D structures. The development and advancement of deep generative models have opened up new avenues for creativity, data augmentation, and data analysis.

The success of deep generative models can be attributed to the powerful representation learning capabilities of deep neural networks. Deep architectures with multiple layers can capture complex hierarchical patterns in the data, allowing the models to generate samples that exhibit intricate details and high-level semantics. Additionally, deep generative models are often trained in an unsupervised or semi-supervised manner, which makes them versatile and applicable to a wide range of domains where labeled data may be limited or expensive to obtain.

This paper aims to provide a comprehensive overview of deep generative models, focusing on both the theoretical foundations and practical aspects of their application. We will delve into the underlying principles and key components of popular deep generative models such as GANs and VAEs, discussing their training procedures, evaluation metrics, and architectural variations. Furthermore, we will explore various applications of deep generative models in diverse domains, showcasing their potential in generating high-quality, realistic data.

[*]Equal contribution [1]School of Electronics Engineering and Computer Science, Peking University.

Our contributions could be summarized as follows:

- Firstly, we present novel results in the field of learning theory pertaining to Variational Autoencoders (VAEs), Generative Adversarial Networks (GANs), Wasserstein GANs (WGANs), and Diffusion Models. Our research delves into the theoretical foundations of these models, exploring their capabilities, limitations, and potential advancements.
- Secondly, we conduct extensive numerical experiments utilizing real-world datasets, specifially the MNIST training dataset, employing VAEs, GANs, WGANs, and Diffusion Models. The dataset contains 60000 figures of the digits. Each figure is 28×28 in size and has only one color channel. Through these experiments, we aim to investigate the practical performance of these models in real-world scenarios, providing empirical evidence to support their efficacy and uncovering insights into their behavior under various conditions.
- Lastly, we bridge the gap between theory and practice by verifying and validating some of the theoretical findings through our practical experiments. By corroborating the theoretical results with real-world data, we strengthen the credibility and applicability of our findings, offering practical insights and potential applications for the studied models.

2. Variational Autoencoders

Variational autoencoders (VAEs) (Kingma & Welling, 2022) are deep latent variable models that typically use Gaussian priors and Gaussian posteriors in their latent spaces.

2.1. theory

Using $x \in \mathcal{X}$ to denote data and $z \in \mathcal{Z}$ to denote the latents with associated prior $p(z)$, a VAE simultaneously learns both a forward generative model, $p_\theta(x|z)$, and an amortised approximate posterior distribution, $q_\phi(z|x)$ (where θ and ϕ correspond to their respective parameters) which are typically implemented using neural networks.

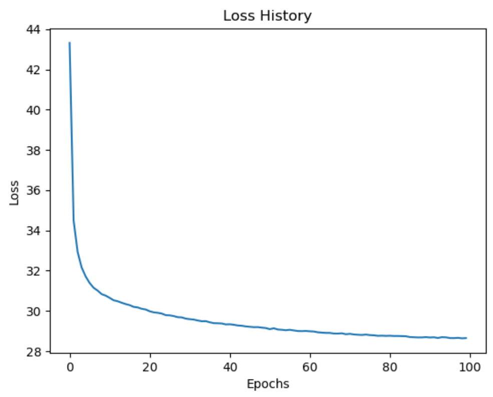


Figure 1. The loss history of the VAE model.

A VAE is trained by maximizing the evidence lower bound (ELBO) $\mathcal{L} = \mathbb{E}_{p_{\mathcal{D}}(\mathbf{x})}[\mathcal{L}(\mathbf{x})]$, where

$$\mathcal{L}(\mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - \mathrm{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))$$

and $p_{\mathcal{D}}(\mathbf{x})$ is the empirical data distribution.

2.2. numerical experiments

In our experiment on VAE, we've implemented a VAE model with a two-layer encoder and a two-layer decoder. The loss function consists of the reconstruction loss (MSE loss) and the regularization term (KL divergence). The model was trained for 100 epochs with CUDA 11.8 powered by NVIDIA GeForce MX450.

Figure 1 shows the history of the loss function. The reconstructed samples and samples decoded from random codes are shown in Figure 2.

2.3. reflections

VAEs typically assume that the latent space follows a simple distribution, such as a multivariate Gaussian. This assumption may result in a smooth and continuous latent space representation. However, in complex data distributions, the true latent space structure may not be perfectly smooth, leading to potential challenges in accurately modeling and generating intricate patterns.

At the same time, VAEs often generate samples that exhibit blurriness or lack crispness, especially when dealing with high-resolution images. This blurriness arises due to the reconstruction objective used in VAE training, which prioritizes capturing the overall structure rather than fine-grained details. As a result, VAE-generated samples may lack sharpness and high-frequency details compared to the original data.

3. Generative Adversarial Networks

GAN (Goodfellow et al., 2014) consists of two models:

- A discriminator D estimates the probability of a given sample coming from the real dataset. It works as a critic and is optimized to tell the fake samples from the real ones.
- A generator G outputs synthetic samples given a noise variable input z (z brings in potential output diversity).

It is trained to capture the real data distribution so that its generative samples can be as real as possible, or in other words, can trick the discriminator to offer a high probability.

3.1. theory

When combining both aspects together, D and G are playing a minimax game in which we should optimize the following loss function:

$$\begin{aligned}&\min_G\max_D L(D,G)\\ &= \mathbb{E}_{x\sim p_r(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))]\\ &= \mathbb{E}_{x\sim p_r(x)}[\log D(x)] + \mathbb{E}_{x\sim p_g(x)}[\log(1-D(x)]\end{aligned}$$

When both G and D are at their optimal values, we have $p_g = p_r$ and $D^*(x) = 1/2$ and the loss function becomes:

$$\begin{aligned}&L(G,D^*)\\ &= \int_x \Big(p_r(x)\log(D^*(x)) + p_g(x)\log(1-D^*(x))\Big)dx\\ &= \log\frac{1}{2}\int_x p_r(x)dx + \log\frac{1}{2}\int_x p_g(x)dx\\ &= -2\log 2\end{aligned}$$

JS divergence between p_r and p_g can be computed as:

$$\begin{aligned}&D_{JS}(p_r\|p_g)\\ &= \frac{1}{2}D_{KL}(p_r||\frac{p_r+p_g}{2}) + \frac{1}{2}D_{KL}(p_g||\frac{p_r+p_g}{2})\\ =&\frac{1}{2}\Big(\log 2 + \int_x p_r(x)\log\frac{p_r(x)}{p_r+p_g(x)}dx\Big)+\\ &\frac{1}{2}\Big(\log 2 + \int_x p_g(x)\log\frac{p_g(x)}{p_r+p_g(x)}dx\Big)\\ &\frac{1}{2}\Big(\log 4 + L(G,D^*)\Big)\end{aligned}$$

Thus,

$$L(G,D^*) = 2D_{JS}(p_r\|p_g) - 2\log 2$$

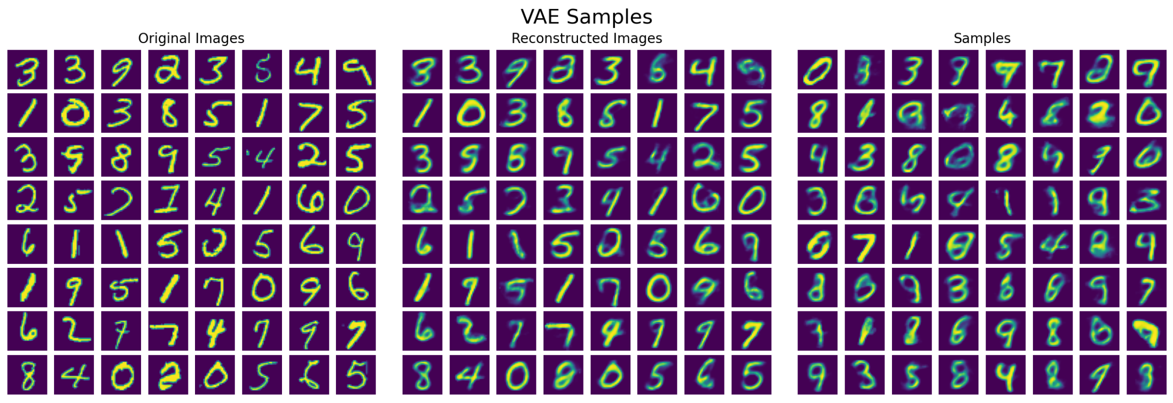


Figure 2. Samples generated by the VAE model.

Essentially the loss function of GAN quantifies the similarity between the generative data distribution p_g and the real sample distribution p_r by JS divergence when the discriminator is optimal. The best G^* that replicates the real data distribution leads to the minimum $L(G^*, D^*) = -2\log 2$ which is aligned with equations above.

3.2. numerical experiments

In our experiments on GAN, the generator and discriminator are both implemented as neural networks with 4 linear layers. The latent space is set to be 100-dimensional. We compared two GANs trained under different critic iterations (how many steps the discriminator is trained per step the generator goes forward). Each instance was trained for 100 epochs with CUDA 11.8 powered by NVIDIA GeForce MX450.

Figure 3 presents the loss histories of two instances (with different critic iterations) and shows some samples they generated from random noise. These samples revealed GANs' preference towards numbers with simple structures such as 1 and 7, which is undesirable as we want the generated dataset to be balanced. The balance of samples gets significantly worse as the value of critic iterations becomes greater.

3.3. reflections

In GAN, two models are trained simultaneously to find a Nash equilibrium to a two-player non-cooperative game. However, each model updates its cost independently with no respect to another player in the game. Updating the gradient of both models concurrently cannot guarantee a convergence.

4. Wasserstein GAN

In the modified Wasserstein-GAN (Arjovsky et al., 2017), the "discriminator" model is used to learn w to find a good f_w and the loss function is configured as measuring the Wasserstein distance between p_r and p_g.

$$\begin{aligned} L(p_r, p_g) &= W(p_r, p_g) \\ &= \max_{w\in W} \mathbb{E}_{x\sim p_r}[f_w(x)] - \mathbb{E}_{z\sim p_r(z)}[f_w(g_\theta(z))] \end{aligned}$$

4.1. theory

Wasserstein Distance is a measure of the distance between two probability distributions. It is also called Earth Mover's distance, short for EM distance, because informally it can be interpreted as the minimum energy cost of moving and transforming a pile of dirt in the shape of one probability distribution to the shape of the other distribution.

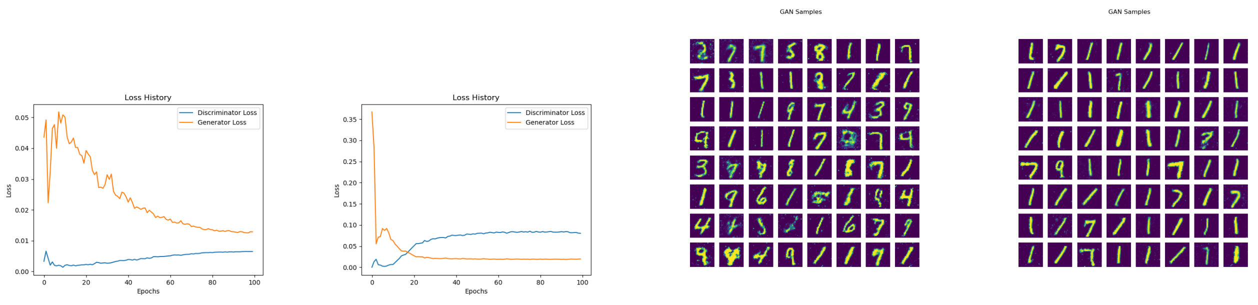
When dealing with the continuous probability domain, the distance formula becomes:

$$W(p_r, p_g) = \inf_{\gamma\sim\Pi(p_r, p_g)} \mathbb{E}_{(x,y)\sim\gamma}[\|x - y\|]$$

Even when two distributions are located in lower dimensional manifolds without overlaps, Wasserstein distance can still provide a meaningful and smooth representation of the distance in-between.

It is intractable to exhaust all the possible joint distributions in $\Pi(p_r, p_g)$ to compute. Thus the authors proposed a smart transformation of the formula based on the Kantorovich-Rubinstein duality to:

$$W(p_r, p_g) = \frac{1}{K} \sup_{\|f\|_L \le K} \mathbb{E}_{x\sim p_r}[f(x)] - \mathbb{E}_{x\sim p_g}[f(x)]$$



(a) critic_iterations = 1 (b) critic_iterations = 5 (c) critic_iterations = 1 (d) critic_iterations = 5

Figure 3. The loss history of the GAN models and samples generated.

Thus the "discriminator" is not a direct critic of telling the fake samples apart from the real ones anymore. Instead, it is trained to learn a K-Lipschitz continuous function to help compute Wasserstein distance. As the loss function decreases in the training, the Wasserstein distance gets smaller and the generator model's output grows closer to the real data distribution.

4.2. numerical experiments

The WGAN models we implemented consist of a 3-layer generator and a 3-layer discriminator. Our WGAN models are shallower than our GAN models because we want to prevent the gradients from vanishing. The models were trained for 100 epochs with CUDA 11.8 powered by NVIDIA GeForce MX450. Two set of comparisons are made, namely the critic iterations and the clipping window.

The loss histories and samples are shown in Figure 4. Bigger clipping windows make it harder for the model to reach convergence, while smaller ones have negative impacts on the renewal of weights. It seems that more critic iterations per training step of the generator can enhance the performance of WGAN models, though the generated samples are still lower in quality than GANs' samples. On the other hand, the samples generated by WGANs are much more balanced than those by GANs, indicating that WGANs've addressed this big issue of GANs. Besides, the fact that bigger clipping windows might bring about slow convergence and smaller clipping windows can cause gradients to vanish is verified by our experiments.

4.3. reflections

One big problem is to maintain the K -Lipschitz continuity of f_w during the training in order to make everything work out. The paper presents a simple but very practical trick: After every gradient update, clamp the weights w to a small window, such as $[-0.01, 0.01]$, resulting in a compact parameter space W and thus f_w obtains its lower and upper bounds to preserve the Lipschitz continuity.

Sadly, Wasserstein GAN is not perfect. Even the authors of the original WGAN paper mentioned that "Weight clipping is a clearly terrible way to enforce a Lipschitz constraint". WGAN still suffers from unstable training, slow convergence after weight clipping (when clipping window is too large), and vanishing gradients (when clipping window is too small).

5. Diffusion Models

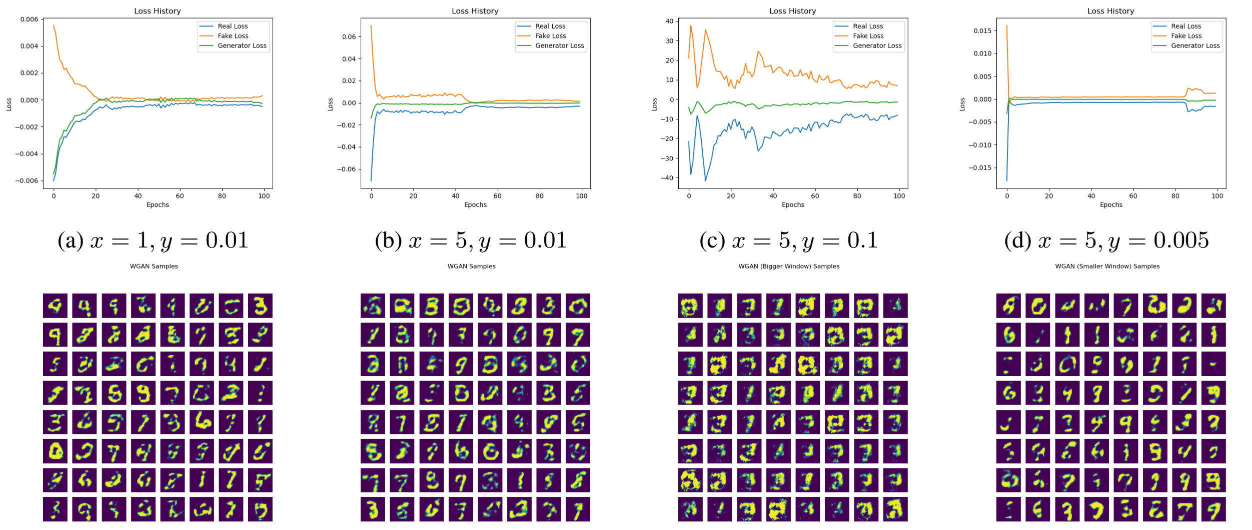
The diffusion model in deep learning is a generative model that iteratively refines a probability distribution over the data, allowing for high-quality synthesis and realistic generation of complex images, audio, and text.

5.1. theory

We would show the result presented in (Chen et al., 2023). In the forward process, starting from an initial data point x_0, Gaussian noise is gradually added to generate subsequent points $x_1, \dots, x_T$. This process is characterized by the conditional probability $p(x_t|x_{t-1}) = \mathcal{N}(\sqrt{1-\beta_t}x_{t-1}, \beta_t I)$, where β_t controls the noise level. The magnitude of each data point is then shrunk by $\sqrt{1-\beta_t}$, followed by the addition of noise with variance β_t. Specifically, the relationship $x_t|x_0 \sim \mathcal{N}(\alpha_t x_0, h_t I)$ holds, where α_t and h_t are parameters.

In the backward process, the objective is to approximate the posterior distribution using a conditional Gaussian, given by $p(x_{t-1}|x_t) \approx \mathcal{N}(\mu_\theta(x_t, t), \Sigma_t)$. The function $\mu_\theta(x_t, t)$ is represented by a neural network. As the noise level β_t approaches zero, the error tends to zero as well. Variational inference is employed to optimize the objective $\mathbb{E}|\mu_\theta(x_t, t) - \mathbb{E}[x_{t-1}|x_t]|^2$, aiming to learn a denoising network that predicts $\mathbb{E}[x_0|x_t]$ from x_t. The conditional distribution is given by $x_t|x_0 \sim \mathcal{N}(\alpha_t x_0, h_t I)$, allowing us to compute the gradient $\nabla_{x_t} \log p(x_t|x_0) = -\frac{x_t - \alpha_t x_0}{h_t}$.

By applying Bayes' Rule, we obtain the following expres-



(a) $x = 1, y = 0.01$ (b) $x = 5, y = 0.01$ (c) $x = 5, y = 0.1$ (d) $x = 5, y = 0.005$

(e) $x = 1, y = 0.01$ (f) $x = 5, y = 0.01$ (g) $x = 5, y = 0.1$ (h) $x = 5, y = 0.005$

Figure 4. The loss history and generated samples of the WGAN models. x and y represent the critic iterations and clipping windows respectively, for simplicity.

sion:

$$\begin{aligned}&\mathbb{E}[\nabla_{x_t} \log p(x_t|x_0)|x_t] \\ &= \mathbb{E}\left[\nabla_{x_t}\left(\log p(x_t) + \log p(x_0|x_t) - \log p(x_0)\right)|x_t\right] \\ &= \nabla_{x_t} \log p(x_t) + \mathbb{E}[\nabla_{x_t} \log p(x_0|x_t)|x_t].\end{aligned}$$

However, upon further analysis, it is determined that $\mathbb{E}[\nabla_{x_t} \log p(x_0|x_t)|x_t] = 0$. This result is obtained by integrating the expression $p(x_0|x_t)\nabla_{x_t} \log p(x_0|x_t)$, which simplifies to $\nabla_{x_t} \int p(x_0|x_t)dx_0 = \nabla_{x_t} 1 = 0$.

In summary, We connect score matching with denoising: Denoising is learning score function

$$\nabla \log p(x_t) = -\frac{x_t - \alpha_t \mathbb{E}[x_0|x_t]}{h_t}.$$

Then, we come to analyze the estimation capacity of diffusion models. Our target questions are the following:

- Can neural network accurately estimate score function?
- Can diffusion models estimate the data distribution using the learned score functions?

In practical scenarios, the data often exhibits low dimensions, and we make the assumption that $\psi^{-1}(\mathbf{x}) = A\mathbf{z}$, where $\mathbf{z} \sim P_z$ and $A \in \mathbb{R}^d$ is a latent variable. Here, ψ represents a known invertible transformation, such as the feature map of a Reproducing Kernel Hilbert Space (RKHS) or the Fourier transform. For simplicity, we consider ψ as the identity mapping in the following discussion. We define D as the ambient dimension and d as the intrinsic dimension.

The forward stochastic differential equation (SDE) governing the dynamics is given by:

$$\mathrm{d}\mathbf{X}_t = -\frac{1}{2}\mathbf{X}_t \mathrm{d}t + \mathrm{d}\mathbf{W}_t.$$

where $\mathbf{W}_t$ represents the Wiener process. The score function is defined as:

$$\nabla \log p_t(\mathbf{x}) = \frac{\int \nabla \phi_t(\mathbf{x}|A\mathbf{z}) p_z(\mathbf{z}) \mathrm{d}\mathbf{z}}{\int \phi_t(\mathbf{x}|A\mathbf{z}) p_z(\mathbf{z}) \mathrm{d}\mathbf{z}}.$$

where $\phi_t(\mathbf{x}|A\mathbf{z})$ represents the density of $\mathbf{X}_t$ conditioned on $A\mathbf{z}$ and $p_z(\mathbf{z})$ denotes the distribution of $\mathbf{z}$.

It can be observed that:

$$\begin{aligned}\phi_t\left(x|Az\right) &= \phi_t(AA^\top \mathbf{x} \mid A\mathbf{z})\phi_t((I_D - AA^\top)\mathbf{x}) \\ &= \phi_t(A^\top \mathbf{x} \mid \mathbf{z})\phi_t((I_D - AA^\top)\mathbf{x}).\end{aligned}$$

Based on this observation, the score function can be decomposed into two parts:

$$\nabla \log p_t(\mathbf{x}) = \underbrace{A\nabla \log p_t^{LD}(A^\top \mathbf{x})}_{\mathrm{s}_{\|}(A^\top \mathbf{x}, t);\text{on-support}} - \underbrace{\frac{1}{1-e^{-t}}\left(I_D - AA^\top\right)\mathbf{x}}_{\mathrm{s}_{\perp}(\mathbf{x}, t);\text{ortho.score}},$$

where

$$p_t^{\mathrm{LD}}(\mathbf{z}') = \int \phi_t(\mathbf{z}'|\mathbf{z})p_z(\mathbf{z})\mathrm{d}\mathbf{z} \quad \text{with} \quad \mathbf{z}' = A^\top \mathbf{x}.$$

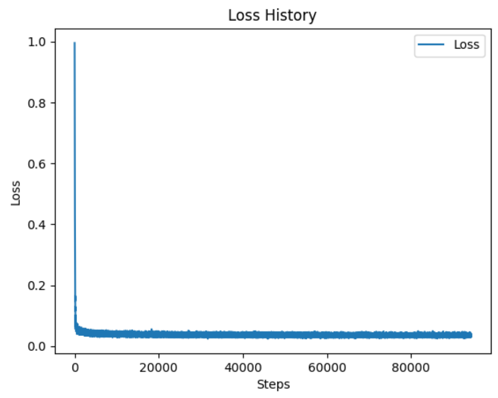


Figure 5. The loss history of the Diffusion model.

$\mathrm{s}_\perp$. The term $\mathrm{s}_\perp(\mathbf{x}, t)$ dominates as $t \to 0$, thereby enforcing accurate estimation of A.

Finally, the theorem can be summarized as follows:

Theorem 1 ((Chen et al., 2023)). *Under mild assumptions, with high probability, score function is estimated at a rate of*

$$\frac{1}{T - t_0} \int_{t_0}^{T} \mathbb{E}_x \left[\|\nabla \log p_t(\mathbf{x}) - \widehat{\mathbf{s}}_{V;\boldsymbol{\theta}}(\mathbf{x}, t)\|_2^2 \right] \mathrm{d}t \leq \widetilde{\mathcal{O}} \left(\frac{1}{t_0} \left(n^{-\frac{2}{d+5}} + D n^{-\frac{d+3}{d+5}} \right) \log^3 n \right)$$

in terms of the squared L^2 error.

5.2. numerical experiments

The implementation of the diffusion model is based on VSehwag's minimal-diffusion. The training (201 epochs) is done with CUDA 11.8 powered by NVIDIA GeForce RTX 2080 Ti.

The loss history and samples are shown in Figure 5 and 6 respectively. The capability of the diffusion model far exceeds that of any other deep generative model. Meanwhile, the diffusion model reaches convergence rapidly.

5.3. reflections

One significant limitation of this paper is its assumption that the data distribution can be represented as a linear subspace, while in practice it is more likely to be a low-dimensional manifold.

Another important consideration is the optimization method used in the proposed approach. Since the data distribution is assumed to be a low-dimensional manifold, traditional gradient descent may not be the most suitable optimization

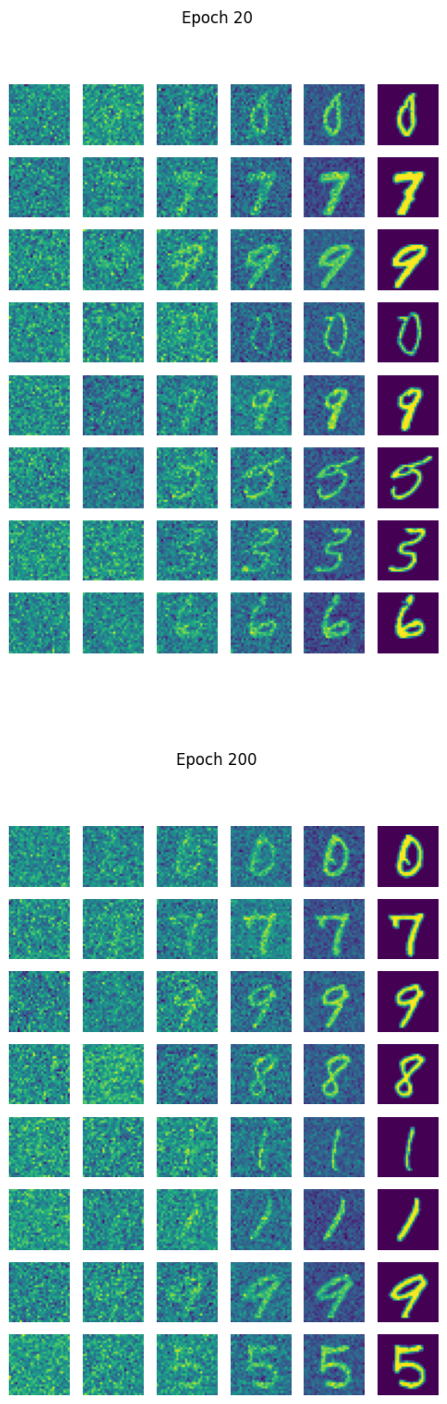


Figure 6. The denoising samples of the Diffusion model after 20 or 200 epochs of training.

technique. Investigating and proposing a specialized optimization method tailored for manifold optimization would further strengthen the paper's methodology.

Furthermore, the paper assumes that the optimization method employed during the training process can find a global minimum. However, in practice, finding a global minimum is often challenging and not always guaranteed.

6. What we have done

The contributions of our work are divided among the team members as follows: An Zhang conducted all the numerical experiments, while Xinwei Zhang focused on the theoretical aspects. We collaborated in comparing our results and providing insights in the reflections section. The paper was written collectively, with input and revisions from both team members. For the presentation of our work, An Zhang took the lead and Xinwei Zhang prepared the slides.

The codes consist of three parts: implementations of the models, training and sampling. The implementations and training routines of VAE, GAN and WGAN (models.py and train.ipynb) are done by ourselves, while those of Diffusion Models (unets.py, diffusion.py) are based on minimal-diffusion. However, the training routine and sampling of Diffusion Models (Diffusion.ipynb) were written by ourselves to fit for our specific needs. Besides, anything about sampling is totally written by us and tailored for our own sake.

Acknowledgement

This work was completed as part of the Computer Vision and Deep Learning course in the Spring of 2023 at Peking University. We would like to express our sincere gratitude to our professor and the teaching assistants for their invaluable support, guidance, and assistance throughout the project.

References

Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein gan, 2017.

Chen, M., Huang, K., Zhao, T., and Wang, M. Score approximation, estimation and distribution recovery of diffusion models on low-dimensional data, 2023.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial networks, 2014.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes, 2022.