IMAGE GENERATION USING ONLY A DISCRIMINATOR NETWORK WITH GRADIENT NORM PENALTY

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

CANSU CEMRE YEŞILÇIMEN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

SEPTEMBER 2022

Approval of the thesis:

IMAGE GENERATION USING ONLY A DISCRIMINATOR NETWORK WITH GRADIENT NORM PENALTY

submittedb **CANSU CEMRE YEŞILÇIMEN** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** ______________

Prof. Dr. Halit Oğuztüzün
Head of Department, **Computer Engineering** ______________

Assist. Prof. Dr. Emre Akbaş
Supervisor, **Computer Engineering, METU** ______________

Examining Committee Members:

Assist. Prof. Dr. Ramazan Gökberk Cinbiş
Computer Engineering, METU ______________

Assist. Prof. Dr. Emre Akbaş
Computer Engineering, METU ______________

Assist. Prof. Dr. Cemil Zalluhoğlu
Computer Engineering, Hacettepe University ______________

Date: 02.09.2022

ABSTRACT

IMAGE GENERATION USING ONLY A DISCRIMINATOR NETWORK WITH GRADIENT NORM PENALTY

Yeşilçimen, Cansu Cemre
M.S., Department of Computer Engineering
Supervisor: Assist. Prof. Dr. Emre Akbaş

September 2022, 50 pages

This thesis explores the idea of generating images using only a discriminator network by extending a previously proposed method (Tapli, 2021) in several ways. The base method works by iteratively updating the input image, which is pure noise at the beginning while increasing the discriminator's score. We extend the training procedure of the base network by adding the following new losses: (i) total variation, (ii) N-way classification (if labels are available), and (iii) gradient norm penalty on real examples. Our experiments show that while the total variation and N-way classification do not significantly improve the performance, the gradient norm penalty results in better generative examples and faster convergence. Combining all three modifications yield the best model. Using a small convolutional network, we achieve an FID score of 25.26 on the MNIST dataset. We demonstrate additional generation results on the EMNIST and Yale Face datasets and present scores for out-of-distribution detection on FashionMNIST, EMNIST, and KMNIST datasets.

Keywords: computer vision, gradient penalty, convolutional neural network, image

generation

ÖZ

YALNIZCA AYIRICI AĞ KULLANARAK GRADYAN BÜYÜKLÜĞÜ CEZASI İLE GÖRÜNTÜ ÜRETİMİ

Yeşilçimen, Cansu Cemre
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Dr. Öğr. Üyesi. Emre Akbaş

Eylül 2022 , 50 sayfa

Bu tez, daha önce Taplı (2021) tarafından önerilmiş olan sadece bir ayırıcı ağ ile görüntü üretme fikrini çeşitli iyileştirmeler ile genişletmektedir. Sözedilen yöntem, ayırıcı ağın çıktı puanlarını arttırarak başlangıçta saf gürültü olan görüntüleri yinelemeli şekilde günceller. Eğitim prosedürünü, şu yeni kayıp fonksiyonları kullanarak genişletmekteyiz: (i) toplam değişimsel kayıp, (ii) N-sınıflı ayırma (eğer sınıf etiketlerine erişilebiliyorsa) ve (iii) veri kümesindeki görüntüler için hesaplanan gradyan büyüklüğü cezası. Deneylerimiz gösteriyor ki, toplam değişimsel kayıp ve N-sınıflı ayırma üretim performansını önemli ölçüde değiştirmemesine rağmen, gradyan büyüklüğü cezası daha iyi görüntü üretimine ve daha hızlı yakınsamaya sahip olmasını sağlar. Bahsedilen üç değişikliği birleşik şekilde uygulamak ise en iyi sonuç çıktılarını oluşturur. Küçük bir evrişimli sinirsel ağ kullanarak MNIST veri kümesi üzerinde 25.26 FID puanına erişiyoruz. EMNIST ve Yale Face veri kümeleri üzerinde görüntü üretimi sonuçları ile belirsizlik kestirimi problemi için FashionMNIST, EMNIST ve KMNIST veri kümeleri üzerinde alınmış ekstra sonuçlar sunuyoruz.

Anahtar Kelimeler: bilgisayarlı görü, gradyan cezası, evrişimli sinirsel ağ, görüntü üretimi

To my beloved mother

ACKNOWLEDGMENTS

Above all, I would like to express my gratitude to my supervisor, Assist. Prof. Dr. Emre Akbaş for his unwavering support throughout this process. I am able to accomplish this task with his treasured wisdom and guidance.

I would like to thank my committee members, Assist. Prof. Dr. Ramazan Gökberk Cinbiş and Assist. Prof. Dr. Cemil Zalluhoğlu for their time and valuable advice.

I am grateful to my dear friend Melisa Idil Şener, for the love and tireless support she has shown me through the years.

Finally, I would like to thank my classmate and friend Merve Taplı for her endless passion on the research topic and our precious debates.

The numerical calculations reported in this paper were partially performed at TUBITAK ULAKBIM, High Performance and Grid Computing Center (TRUBA resources).

We also gratefully acknowledge the computational resources kindly provided by METU-ROMER, Robotics and Artificial Intelligence Technologies Applications and Research Center.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABBREVIATIONS	
2D	2 Dimensional
3D	3 Dimensional
AE	Autoencoder
AUROC	Area Under the Receiver Operating Characteristics
DCGAN	Deep Convolutional Generative Adversarial Network
DUQ	Deterministic Uncertainty Quantification
ELBO	Evidence lower bound
FID	Fréchet Inception Distance
GAN	Generative Adversarial Network
GPU	Graphics Processing Unit
IS	Inception Score
KL	Kullback-Leibler
PCA	Principal Component Analysis
PresGAN	Prescribed Generative Adversarial Network
VAE	Variational Autoencoder

CHAPTER 1

INTRODUCTION

1.1 Motivation and Problem Definition

Image generation is one of computer vision's most fascinating and prominent research areas. Novel loss functions, architectures, and models are currently being researched in the scientific community to improve generation quality and diversity. Generative models can be grouped into two categories, likelihood-based [2] and implicit generative models [3]. GANs [4] have been a very active research topic in implicit generative models. Likelihood-based models explicitly calculate the likelihood, which leads to either maximizing likelihood or, in other words minimizing Kullback-Leibler (KL) distance. They usually need a specialized architecture for each problem, and non-symmetrical divergence between distributions can create issues like mode-dropping. GANs are proven to behave unstable during training, and like other generation models, metrics for evaluation are still open to discussion and further research. GANs require two distinct networks, a generator, and a discriminator. Balancing these two networks in the training process is a challenge, and if the discriminator is too successful at classification, the generation network tends to fail due to vanishing gradients. They also show tendencies towards mode collapse and often require additional adversarial attacks, such as injecting noise into input data and batch statistic modifications to stabilize training.

Considering the complexity and the meticulous nature of balancing two networks in GANs, rather than stabilizing these individual networks, this thesis focuses and improves on the method introduced by Taplı [1], where, unlike GANs, only a discriminator network is used for generating images. We aim to achieve comparable

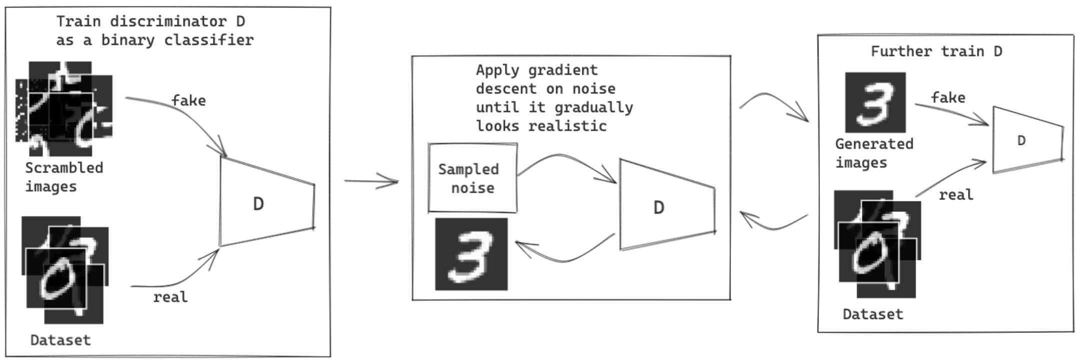


Figure 1.1: Flow of the proposed method.

qualitative results on MNIST, EMNIST and Yale Face datasets using a smaller architecture while adding regularization terms such as total variation loss, N-way classification (cross-entropy) loss, and gradient norm penalty on the discriminator. In this study, we also explore the idea of using the discriminator as an uncertainty estimator for out-of-distribution point detection.

1.2 Proposed Methods and Models

Contrary to GANs, we use only a discriminator to generate images. We train our discriminator network on the dataset prior to the generation process. Initial training consists of training the discriminator as a binary classifier. This is achieved by inputting images from the dataset as real labeled and pixel scrambled dataset images as fake labeled while minimizing hinge loss on inputs. Later, we sample a random noise from a multivariate normal distribution where its standard deviation and mean are set to be the same as the dataset. This random image is then fed through the discriminator network, and gradient descent is applied to the image based on the calculated gradients that maximize the discriminator's output scores. This is a type of adversarial attack applied to the input image. We continue this loop until a specific stopping point, or a maximum number of updates is reached. After we perturb the data in several iterations, these images are labeled as fake, and the discriminator is further trained (fine-tuning) on these examples. For the training of the discriminator, we explore three regularization terms in this thesis: total variation loss, N-way classification loss, and the gradient norm penalty on real examples. Figure 1.1 summarizes the flow of our proposed method and all qualitative and quantitative results are generated through this model.

1.3 Contributions and Novelties

Our contributions can be summarized as follows:

- This thesis expands and explores the idea of using only a discriminator network introduced by Taplı [1] for generation purposes.
- In this study, we explore the idea that as the network discovers the correct boundaries on data space while generating new images, the gradient norms of discriminator output with respect to its inputs should be close or get close to zero with each iteration in process. We show that this restriction on gradient norms improves the generated image qualities.
- We show that our discriminator has a high capability of rejecting out-of-distribution data points and has comparable AUROC scores with state-of-the-art uncertainty estimators.
- We add N-way classification (cross-entropy) loss in fine-tuning step as a measure of the divergence from semantic labels to improve the generation process.
- This thesis includes the addition of total variation loss in the generation objective function to add a measure to the spatial pixel complexity of generated images.
- We produce on-par qualitative generation results to previous work and state-of-the-art GANs on the MNIST dataset with a smaller network architecture both in the number of parameters and layers.
- We reach an FID score of 25.26 on MNIST, 35.37 on EMNIST, and 45.85 on Yale Face datasets outperforming previous work by Taplı [1].

- Our method has a high capability of detecting out-of-distribution data points with an AUROC score of 0.998 on FashionMNIST, 0.999 on KMNIST, and 0.938 on EMNIST datasets.

1.4 The Outline of the Thesis

The remaining sections of this thesis will follow the outline described below.

Chapter 2 explains previous research and related work on the thesis subject.
Chapter 3 describes the proposed method in detail with explanatory figures and tables.
Chapter 4 demonstrates the experiment setup and process and presents numeric and visual results.
In Chapter 5, we provide the conclusion to this thesis and general commentary.

CHAPTER 2

BACKGROUND AND RELATED WORK

In this chapter, we provide a background for generative models and describe related research in the literature. Generative models have many applications in the field of computer vision such as high fidelity image generation [5, 6, 7], style transfer [8, 9], text to image translation [10, 11], 3D modelling [12] and artistic creation [13]. Likelihood-based models include VAEs [14, 15, 16] and autoregressive models [17, 18]. Implicit models include GANs [4], one of the most influential generative models in recent years. We also provide background information on score networks and diffusion models, explaining the intuitive ideas of gradient norm restrictions and the behavior of scores of the inputs.

2.1 Autoencoders

Autoencoders were first introduced by Rumelhart *et al.* [19] in the late 80s as a means to offer a solution to the problem of achieving back-propagation in an unsupervised manner. A general autoencoder framework consists of an encoder and a decoder. The main goal is to map the input data into a compressed representation on a latent space and then decode it as similar as it can get to the original data. This conceptually simple idea has many application fields in the machine learning area. Autoencoders are used by Tomczak and Welling [20] as a method to perform space reduction, denoising images [21], anomaly detection [22, 23] and for clustering purposes [24]. Later on formally defined by Baldi [25], autoencoders attempt to solve the objective function defined in Eq. (2.1). Where $A : \mathbb{R}^n \to \mathbb{R}^p$ is defined as the encoder and $B : \mathbb{R}^p \to \mathbb{R}^n$ is defined as the decoder. The objective function tries to minimize the expectation over input x and Δ is the dissimilarity function between the input and output. Divergence functions are usually chosen to be in the form of L_p norm or Hamming distance.

$$\min_{A,B} \left[\sum_{t=1}^{m} \Delta(A \circ B(x_t), x_t) \right] \tag{2.1}$$

A and B are usually set as neural networks. However, if they are linear operations, the autoencoder can be defined as a linear autoencoder. Flaut [26] shows that when linear autoencoders are used for space reduction, the reduction is equivalent to using Principal Component Analysis (PCA).

The bottleneck layer between the encoder and the decoder captures a compressed representation of the latent space. This bottleneck creates some setbacks in the process of autoencoder training. For example, suppose the bottleneck's hidden nodes are equal to or greater than the input data size. In that case, the encoder might not learn any valid information and become an identity function. On the other hand, if the hidden node size is too small, the network still has the chance to overfit. Furthermore, the bias-variance trade-off is one of the acute issues to consider while tackling autoencoder training.

2.2 Variational Autoencoders

Kingma and Welling [14] proposed significant progress in the abilities of autoencoders where the latent variables are represented by a probability distribution rather than a fixed vector. When we consider the marginal likelihood defined in Eq. (2.2) it can be seen from Eq. (2.3) that the true posterior is intractable, which creates a problem, especially in learning models with non-linear layers. VAEs answer the question of how to achieve learning where the continuous latent variables have intractable posterior distributions.

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{z}) p_\theta(\mathbf{x}|\mathbf{z}) \tag{2.2}$$

$$p_\theta(\mathbf{z}|\mathbf{x}) = \frac{p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z})}{p_\theta(\mathbf{x})} \tag{2.3}$$

Let us define the dataset $\mathbf{X} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}$ where the N is the number of samples from variable $\mathbf{x}$. The process has 2 main steps. It starts by sampling $\mathbf{z}^{(i)}$ from p_{θ^*}, then $\mathbf{x}^i$ is generated from the conditional distribution $p_{\theta^*}(\mathbf{x}|\mathbf{z})$. As shown by Kingma and Welling [14], the aforementioned prior and likelihood are assumed to be members of the same parametric families as p_θ and $p_\theta(\mathbf{x}|\mathbf{z})$ but as with likelihood maximization tasks the true values of θ^* and $\mathbf{z}^{(i)}$ are unknown. To ease the maximum likelihood calculation of $p_\theta(\mathbf{x}^i)$, a new approximation of intractable posterior defined in Eq. (2.3) can be employed. Let us assign θ to the probabilistic encoder variable and ϕ to the probabilistic decoder variable as parameters to be learned. Then it can be assumed that $q_\phi(\mathbf{z}|\mathbf{x})$ recognition model can solve the probabilistic distribution of $\mathbf{z}$ from which the $\mathbf{x}$ could have been generated. With this perspective, the main objective function can be summarized as maximizing ELBO on the marginal log-likelihood using the approximated posterior distribution $q_\phi(\mathbf{z}|\mathbf{x})$.

$$\log p_\theta(\mathbf{x}^i) = \mathbf{D}_{KL}(q_\phi(\mathbf{z}|\mathbf{x}^i)||p_\theta(\mathbf{z}|\mathbf{x}^i)) + \mathcal{L}(\theta, \phi; \mathbf{x}^i) \tag{2.4}$$

can be restated as

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

which can also be written as

$$\mathcal{L}(\theta, \phi; \mathbf{x}^i) = -\mathbf{D}_{KL}(q_\phi(\mathbf{z}|\mathbf{x}^i)||p_\theta(\mathbf{z}|\mathbf{x}^i)) + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}^i)}\left[\log p_\theta(\mathbf{x}^i|\mathbf{z})\right]$$

The first term of the objective function defined in Eq. (2.4) is the KL divergence between the approximate model and the true posterior of the term $\mathbf{z}$. The objective function tries to minimize the divergence term with respect to parameter ϕ. The second term is defined as the marginal lower bound and aimed to be optimized for the parameters θ and ϕ. While composing the Monte Carlo expectations we can state Eq. (2.5) with the purpose of using differentiable function $g_\phi(\epsilon, \mathbf{x}^i)$. The main motivation in the reparameterization trick is to solve the problem created by our density being parameterized by θ.

$$\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}^\mathbf{i})} = \mathbb{E}_{p_\epsilon}\left[f(g_\phi(\epsilon, \mathbf{x}^i))\right] \simeq \frac{1}{L}\sum_{l=1}^{L} f(g_\phi(\epsilon^l, \mathbf{x}^i)) \text{ where } \epsilon^l \sim p(\epsilon) \tag{2.5}$$

The use of Gaussian posterior approximation creates some limitations on the abilities of VAEs. The forenamed models show tendencies to explore the spaces with lower dimensionality compared to their allowed capacity, as shown by Tomczak and Welling [27], and by Burda *et al.* [16]. Furthermore, VAEs tend to ignore some latent variables in these space reductions and might generate blurry outputs. Gregor *et al.* [28] shows that vanilla VAEs require manual annealing of KL divergence term. Expanding the posterior and using complex posteriors to overcome these issues create problems compared to vanilla VAEs during training with generalization and optimization.

2.3 Generative Adversarial Networks

GANs were proposed by Goodfellow *et al.* [4] in 2014 as a new implicit generative model and are one of the most popular and prominent developments in generative modeling. GANs are utilized in many fields in the computer vision area such as image synthesis [5, 6], image inpainting [29, 30], style transfer [31, 32] and text to image translation [33, 34]. They work by sampling from data distributions assembled via Gaussian transformations utilized by deep neural networks. Given a class of models $\mathbf{Q}$, GANs try to find the best model $\mathbf{q}$ that belongs to this family by matching it to the input data. The main logic behind GANs is to train two models simultaneously and improve them via minimizing opposing objective functions, which forces the networks to compete. The first model, namely the generator $\mathbf{G}$, tries to generate data $\mathbf{G}(\mathbf{z})$ from noise $\mathbf{z}$ as close to the real dataset as possible. Meanwhile, the second network, the discriminator $\mathbf{D}$, tries to distinguish whether the input is taken from the dataset or was generated via the generator. Both of these models usually are chosen to be neural networks for differentiability.

$$\min_{\mathbf{G}} \max_{\mathbf{D}} \mathbf{V}(\mathbf{D}, \mathbf{G}) = \mathbb{E}_{x \sim p_{data}(x)} \left[\log \mathbf{D}(x)\right] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} \left[\log(1 - \mathbf{D}(\mathbf{G}(\mathbf{z})))\right] \quad (2.6)$$

Vanilla GANs try to minimize the objective function defined in Eq. (2.6). The latent variable $\mathbf{z}$ is sampled from the latent distribution $p_{\mathbf{z}}(\mathbf{z})$. The generator $\mathbf{G}$ offers a mapping from random latent variable $\mathbf{z}$ to data space, and the discriminator $\mathbf{D}$ outputs a scalar probability of input being a genuine dataset sample. Because the GAN training requires finding Nash equilibrium in high dimensional space, they tend to fail to converge, especially when back-propagation is used for learning. GANs also tend to experience mode collapse when trained with fewer input data and create fewer modes than the multi-modal input dataset. Mode collapse pushes the discriminator $\mathbf{D}$ to overfit and removes the ability to create significant gradients to improve the generator $\mathbf{G}$, reducing the quality of generated data. They also show inclinations to encounter vanishing gradient and unstable gradient behavior based on the chosen distance function. Several techniques have been introduced to stabilize GAN training and overcome well-known problems faced in the training process [35]. Jacobian regularization for local convergence, gradient penalty and weight normalization for Lipschitz continuity, and data augmentation to reduce overfitting can be listed as some of these techniques [36]. By utilizing Lipschitz continuity, Arjovsky *et al.* [37] proposed a key improvement in the training of GANs. Other research followed enforcing Lipschitzness [38, 39], and this constraint motivated improvements like gradient penalty and spectral normalization. It achieved state-of-the-art generation results [7, 5].

Our model employs an idea similar to GANs, but in our method, we use only a discriminator network to overcome issues faced in training processes. We aim to reach equilibrium among discriminator states by balancing the gradients by further training the discriminator after the image generation loop.

2.4 Diffusion Models

Sohl-Dickstein *et al.* [40] proposed a variation to the latent variable methods named diffusion models, and they have shown great success in image generation. Diffusion models create new data by slowly adding Gaussian noise to input and learning to reverse this process through a fixed Markov chain. Diffusion models employ two processes: forward diffusion and reverse diffusion. In the forward diffusion, data $\mathbf{x}_0$ is sampled from its prior distribution $\mathbf{q}(\mathbf{x})$ and is destroyed by injecting Gaussian noise. Eq. (2.7) summarizes the forward diffusion process where the conditional Gaussian is combined with Markov formulation leads to. β represents the variance schedule.

$$\begin{aligned} q(\mathbf{x}_{1:T}|\mathbf{x}_0) &= \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}) \\ &= \prod_{t=1}^{T} \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I}) \end{aligned} \tag{2.7}$$

The reverse diffusion process model learns a variational decoder where it reverses the diffusion process and perturbs noise into data. By starting from pure noise $p_{\mathbf{x}_T} \sim \mathcal{N}(\mathbf{x}_T, 0, \mathbf{I})$ the model learns a joint distribution defined in Eq. (2.8). The equation being dependent on time affirms that each iteration of the distribution depends only on the previous.

$$\begin{aligned} p_\theta(\mathbf{x}_{0:T}) &= p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \\ &= p(\mathbf{x}_T) \prod_{t=1}^{T} \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \textstyle\sum_\theta(\mathbf{x}_t, t)) \end{aligned} \tag{2.8}$$

The reverse diffusion process model learns a variational decoder where it reverses the diffusion process and perturbs noise into data by starting from pure noise where $p_{\mathbf{x}_T} \sim \mathcal{N}(\mathbf{x}_T, 0, \mathbf{I})$.

2.5 Score Matching

Probabilistic models in many cases have an unknown normalization term that is intractable. Let's assume we want to estimate parameter θ from a random vector $\mathbf{x} \in \mathbb{R}^n$, in other words we want to approximate the probability density function of $\mathbf{x}$ denoted by $p_x(.)$ by using estimated parameter $\hat{\theta}$. The setback, in this case, is that we can only compute the pdf defined in Eq. (2.9). The distribution $p(\xi;\theta)$ can be learned by maximizing the log-likelihood of the training data though this would require it to be a normalized function. The normalizing constant $\mathbf{Z}(\theta)$ can usually be defined as an intractable integral defined in Eq. (2.10), hence calculating the log-likelihood would be an infeasible choice for our problem scope. By solving the problem of estimating non-normalized probabilistic models from unknown distributions, score matching was introduced by Hyvärinen [41]. This method builds on the general probabilistic tool used for obtaining the bounds of distances among distributions presented by Stein [42]. Score-based models can be utilized to learn normalizing constants directly. Score networks utilizing score matching approaches are employed in many ways in generative learning, such as image generation [43, 44] and audio synthesis [45].

$$p(\xi;\theta) = \frac{1}{\mathbf{Z}(\theta)} q(\xi;\theta) \tag{2.9}$$

$$\mathbf{Z}(\theta) = \int_{\xi\in\mathbb{R}^n} q(\xi;\theta) d\xi \tag{2.10}$$

The main idea of score matching is to minimize the divergence between the gradient of log density of ground-truth data and the gradient of log density calculated by the model. For a distribution $p(\mathbf{x})$ score function can be summarized as $\nabla_{\mathbf{x}} \log p(\mathbf{x})$. Song and Ermon [43] suggested that a score network $\mathrm{s}_\theta(\mathbf{x})$ can be trained for estimating $\nabla_{\mathbf{x}} \log p(\mathbf{x})$ without the need to train a model for the term $p(\mathbf{x})$ and this model is called a score-based model. The objective function Eq. (2.11) for score matching minimizes Fisher divergence between the model and score network. After the score network is trained, Langevin dynamics [46] can be used for sampling from its distribution. Langevin sampling simulates sampling from prior distribution $p(\mathbf{x})$ by sampling from $\log p(\mathbf{x})$.

$$\mathbb{E}_{p(\mathbf{x})} \left[||\nabla_{\mathbf{x}} \log p(\mathbf{x}) - \mathrm{s}_\theta(\mathbf{x})||_2^2 \right] \tag{2.11}$$

Song *et al.* [47] presents state-of-art inception scores on CIFAR dataset by perturbing data with different scales of noise. This approach initiated many studies that tie diffusion models and score networks. Ho *et al.* [48] showed that the ELBO term in diffusion models is analytically equivalent to score-network objective functions.

Our model employs a similar idea to score networks where we calculate the gradients of the real data on the discriminator. Rather than training a separate network for score-matching, we use the norm of the gradient values as a gradient penalty term similar to the idea proposed by Gulrajani *et al.* [49] to restrict the gradient update.

CHAPTER 3

PROPOSED METHOD

In this chapter, we describe our proposed image generation method that involves only a discriminator network. The method can be summarized in three main steps, initial training, image generation, and fine-tuning of the network. Generation and fine-tuning steps are repeated until convergence criteria. In the sections below, we explain the implementation details of these steps, the choice of loss functions, and the overall network architecture.

3.1 Network Architecture

Discriminator network in this thesis, illustrated in Figure 3.1, is chosen similar to a minimalistic version of VGG-Net [50] where filters of size 3×3 with small receptive fields are made use of. It takes a 28×28 image and outputs a scalar score. Architectural structure has 4 convolutional layers, followed by a concatenation layer and three

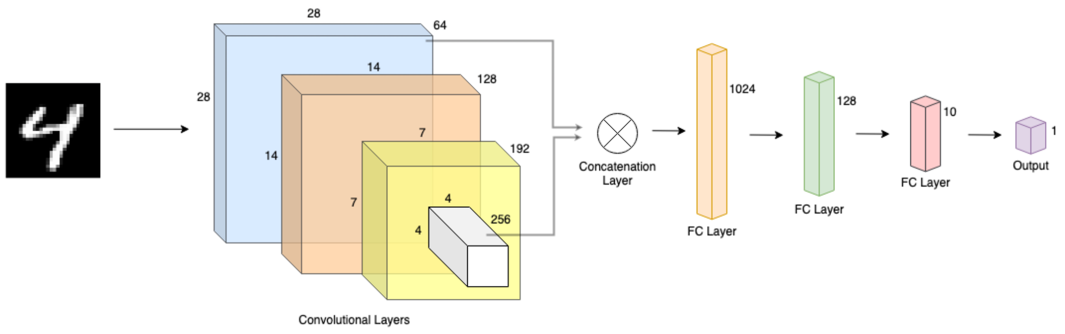


Figure 3.1: The network architecture of our model.

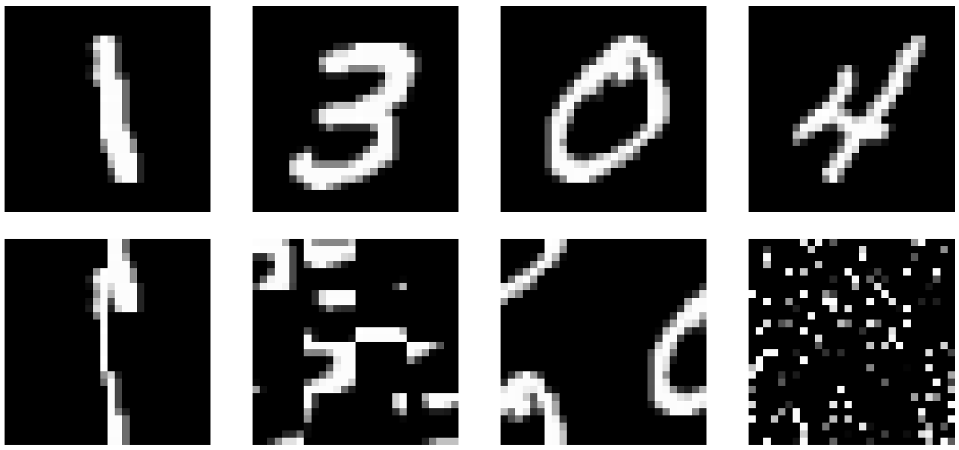


Figure 3.2: Examples illustrating the scrambling process. The first row shows original samples from the dataset. The second row is the respective original image's scrambled versions based on different patch sizes.

fully connected layers. The concatenation layer flattens and concatenates outputs of the first and last convolutional layer to pass information through. We use the third fully connected layer's output as class energies in cross-entropy loss for classification purposes. The output layer is used as a score value of the input images.

3.2 Initial Training

Before the generation process is initiated, the network is trained as a discriminator to distinguish real vs fake images. By previously training, we aim to start the generation process where the network has already started to explore data space. Real labeled images are chosen to be from the dataset. Fake labeled images are then created by a scrambling algorithm. The algorithm randomly chooses an integer for each image from a predefined range, then this value is used as the number of adjacent pixels that will be moved through the horizontal axis. Figure 3.2 illustrates examples of scrambled images created by this method using different patch sizes. These scrambled images are labeled fake and fed through the network with real labeled images.

$$L_{init} = \mathbb{E}_{x\sim\mathbb{X}}\left[\max(0, 1 - D(x))\right] + \mathbb{E}_{f\sim\mathbb{F}}\left[\max(0, 1 + D(f))\right] \tag{3.1}$$

Eq.(3.1) represents the main objective function minimized during the initial training process. $\mathbb{X}$ is the dataset and x is a sample image taken from it. $\mathbb{F}$ represents images labeled fake, scrambled images in this case, and f is an image taken from this set. It can be seen that Eq.(3.1) is a modified version of minimax loss defined by Goodfellow *et al.* [4] and this equation will be made use of in the upcoming sections as well.

3.3 Image Generation

After initial training, our method continues from the remaining two main steps, first applying gradient descent on images and then further training the network. Applying gradient descent on randomly sampled data to generate realistic-looking images and further training (fine-tuning) the discriminator where its weights are updated based on generated images.

The flow starts by sampling random noise data from a multivariate normal distribution where its mean and variation are chosen to be the same as the dataset. Sampled noise is then fed through the discriminator to get score results. These scores are used for calculating objective function and by utilizing the loss values, back-propagation is applied to gather gradients. Based on the outputs of back-propagation, the appropriate amount of gradient descent is then applied to sampled noise which is equivalent to subtracting the gradient vector from images at each iteration. Gradually, randomly sampled noise data starts to transform into images resembling the input dataset. The idea of applying gradient descent on data and its effects are investigated and substantiated in Chapter 4.

After certain stopping conditions are met or a certain number of iterations is reached the image update loop is broken and generated images are labeled as ‘fake’. These images are then passed through a process called fine-tuning. By labeling generated images as fake we aim to challenge the discriminator to generate higher quality images that correlate to higher scores on fake images. Figure 3.3 summarizes generation

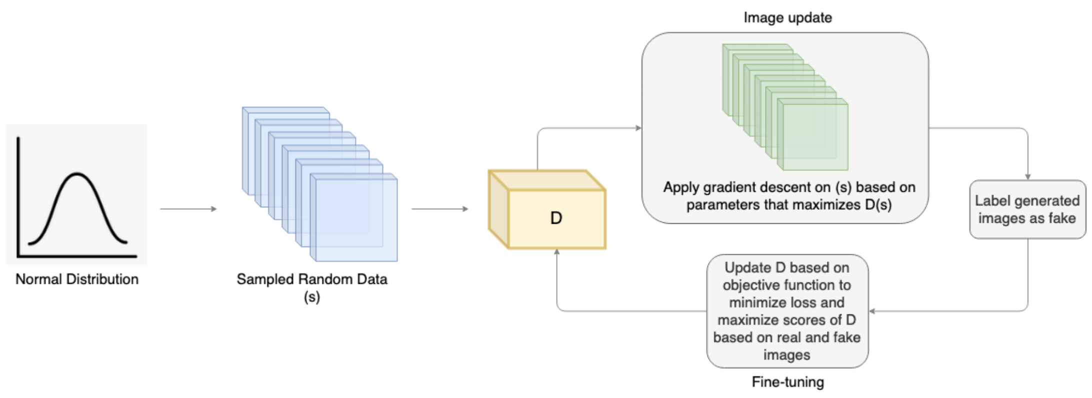


Image update
Normal Distribution
Sampled Random Data (s)
D
Apply gradient descent on (s) based on parameters that maximizes D(s)
Label generated images as fake
Update D based on objective function to minimize loss and maximize scores of D based on real and fake images
Fine-tuning

Figure 3.3: Steps of the generation pipeline.

and fine-tuning steps after the discriminator had been initially trained.

3.3.1 Total Variation Loss

Total variation loss is used in the overall generation objective as a measure of the complexity of images with respect to the spatial variation of pixel values. We add this term by summing the squares of differences on horizontally and vertically adjacent pixels. The division is used as a batch and image-wise normalization term. In Eq. (3.2), x represents individual images taken from dataset $\mathbb{X}$. H and W represent the height and width of images respectively. C is the channel size of input images and B is the batch size of the algorithm. λ_{ttl} is used as a multiplication factor of the total variation loss term.

$$L_{ttl} = \lambda_{ttl} \left[\frac{\sum_{i=1}^{W-1} \sum_{j=1}^{H} (x_{i+1,j,c} - x_{i,j,c})^2 + \sum_{i=1}^{W} \sum_{j=1}^{H-1} (x_{i,j+1,c} - x_{i,j,c})^2}{(C * H * W * B)} \right] \tag{3.2}$$

3.3.2 Objective Function of Image Generation

In addition to Eq. (3.2), hinge loss and dataset statistics are added to the general objective function of the image generation loop. Eq. (3.3) is the hinge loss of real images. Eq. (3.4) is added as a term to enforce similar batch statistics of the generated images and dataset. $\mathbb{X}$ is the original images and $\hat{\mathbb{X}}$ is generated images. λ_{stat} term is used as a scale factor to adjust the overall value and its effect on the objective function. The first term in Eq. (3.4) is the L_2 distance between the mean of the dataset and the mean of the generated image set. The second term is the distance between the variances of two sets.

$$L_{update} = \max_{x \in \mathbb{X}}(0, 1 - D(x)) \tag{3.3}$$

$$+ \lambda_{stat} \left[\sum_{i=1}^{W} \sum_{j=1}^{H} [\mu_{\mathbb{X}} - \mu_{\hat{\mathbb{X}}}]_{ij}^2 + \sum_{i=1}^{W} \sum_{j=1}^{H} \left[\sigma_{\hat{\mathbb{X}}}^2 - \sigma_{\hat{\mathbb{X}}}^2\right]_{ij}^2 \right] \tag{3.4}$$

The overall objective function of the image update step can be summarized as Eq. (3.5). Gradients are calculated to minimize this function and the gradient vector is then subtracted from the sampled noise as a way to apply gradient descent to the image itself.

$$L_{gen} = L_{update} + L_{ttl} \tag{3.5}$$

3.4 Fine-tuning of Network

The main goal of fine-tuning the discriminator is by labeling generated images as ‘fake’; we aim to force the network in a way that after each iteration it is forced to generate higher-scored images and will try to balance out the effect created by applying gradient descent on data.

3.4.1 Gradient Norm Penalty on Real Images

We present a similar idea to score matching, where we calculate the gradients of input images. Rather than training a separate network to learn these gradients we approach from a different perspective and use these gradients on dataset images as a penalty term. Assuming the discriminator learns the correct decision boundaries while generating images, in this thesis we propose that the update gradient on a real image chosen randomly from a dataset should be close to or approach zero while updating discriminator weights at each iteration. The main motivation of this standpoint is limiting the change in discriminator weights as the input changes. Hence, enforcing no action in the case of inputting dataset images while applying parameter update. In other words for an ideal world where the discriminator generates authentic images, its gradient norms on these dataset images should be zero. The objective function presented in Eq. (3.6) is based on minimizing the gradient norms of outputs of the discriminator D with respect to its inputs. λ_{gnorm} term is used as a scale factor in the overall objective function.

$$L_{gnorm}(\mathbb{X}) = \lambda_{gnorm}||\nabla_{x\in\mathbb{X}}D(x)|| \tag{3.6}$$

3.4.2 Cross Entropy Loss on Real Images

We add multi-class cross-entropy loss as a regularization term to the objective function to calculate the divergence from original labels to predicted labels. In Figure 3.1 the third fully connected layer has 10 nodes as output. This layer is then used as the class energies of the classification model. In Eq. (3.7) loss for each class is calculated for each image and then summed. $y_c(x)$ is the true label of x and $p_c(x)$ is the softmax probability of image x belonging to class c.

$$L_{ce}(\mathbb{X}) = \lambda_{ce}\left[-\sum_{c=1}^{C} y_c(x)\log(p_c(x))\right] \tag{3.7}$$

3.4.3 Objective Function of Fine-tuning

Eq. (3.8) utilizes a similar approach used in Eq. (3.1) where the hinge loss between real and fake images is calculated similarly to minimax loss. In fine-tuning step discriminator parameters are updated based on the minimization of the equation defined below.

$$L_{ftune} = \max_{x \in \mathbb{X}}(0, 1 - D(x)) + \max_{f \in \hat{\mathbb{X}}}(0, 1 + D(f)) \quad (3.8)$$

$$+ L_{gnorm}(\mathbb{X}) + L_{ce}(\mathbb{X}) \quad (3.9)$$

3.5 Summary

This chapter explains the details of objective functions used in the proposed method and gives an intuitive understanding of some of the approaches utilized. Figures and explanatory sections are provided to explain network architecture and how we generated fake labeled data.

In the next section, we give concrete results of the experimentation of the proposed method.

CHAPTER 4

EXPERIMENTS

4.1 Method Simulation on 2D

To substantiate the proposed method, we implemented a flow that simulates the overall process. The main purpose of this simulation is to show the effects of gradient descent on data and the effect of adding gradient norm restriction as a loss term to fine-tuning flow on the results of Taplı [1]. We generated a mock dataset consisting of points distributed along a spiral shape on a 2D plane ranging through $[-1, 1]$. This dataset is then fed through a simplified process that encapsulates initial training, data generation, and fine-tuning steps. Gradient norms are added as a regularization term in the fine-tuning objective function explained in Chapter 3. The discriminator network used in this simulation consists of 3 linear layers, $Tanh$ as an activation layer in between linear layers and sigmoid is used for predicting 2-class probability on the last layer as the output of the simulation network.

The network is first trained as a discriminator on the generated dataset for 25 epochs and after this initial training step, the accuracy of the discriminator was 0.84. Figure 4.1a represents the network's boundaries after initial training on real labeled data from the dataset. The decision boundary on each round is decided where the probability on a certain point is bigger than 0.5. Figure 4.1b represents the probability distribution of the model on data space. Red and light areas have a higher probability than blue-colored areas.

After discriminator training, at each loop, 1 batch which consists of 75 randomly generated data points, is fed through the 'generation' process where back-propagation is applied to calculate gradients and then gradient descent is applied on data points to

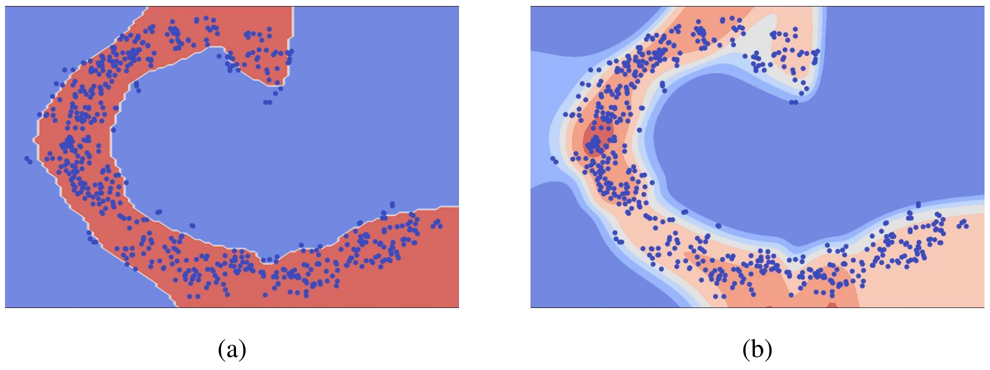


Figure 4.1: Original data represented as blue points visualized on a 2D plane ranging within $[-1, 1]$. The decision boundary divides the data space where red areas are labeled as real and blue area is labeled as fake. a) Original data consisting of 400 data points visualized on top of the decision boundary right after the initial training. b) Original data consisting of 400 data points visualized on top of the decision boundary represented as probability ranges, right after the initial training process. Blue areas have lower probability and red areas have higher probability values.

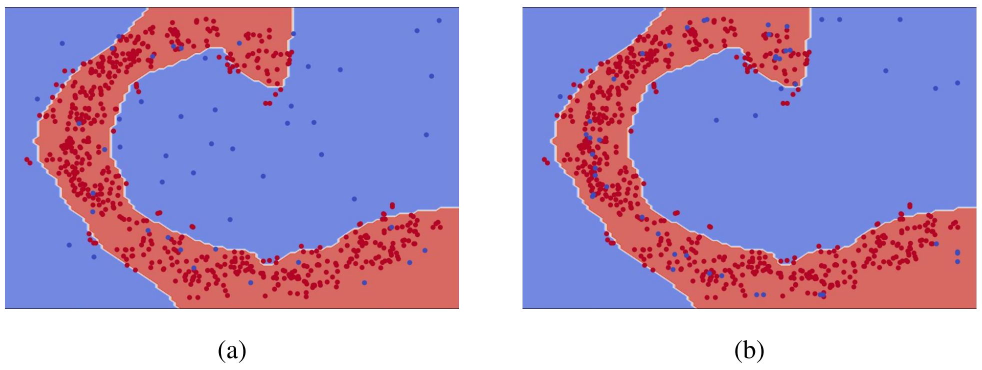
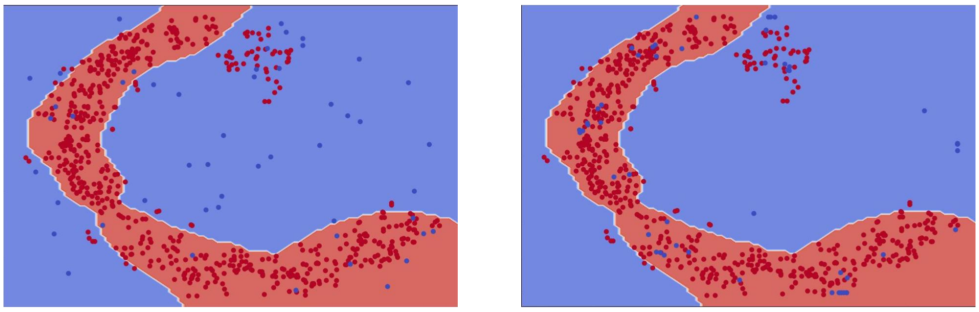
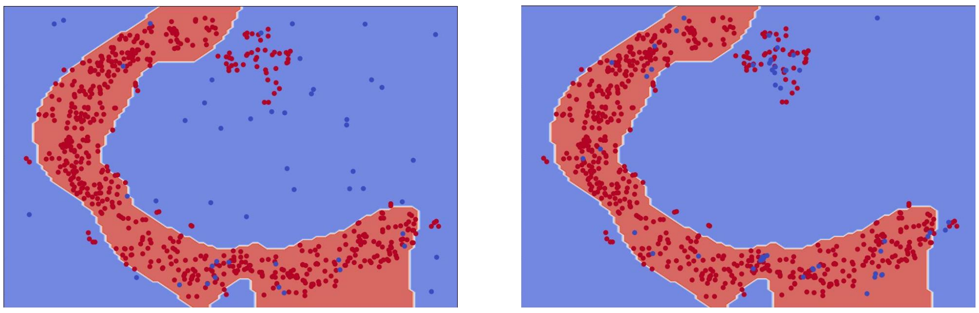


Figure 4.2: Data distribution of dataset and randomly generated data before and after applying first gradient descent on generated data. Red points represent dataset entries labeled real and blue points represent randomly generated data labeled fake. a) Original data and 1 batch of randomly generated data on top of the decision boundary. b) Original data and 1 batch of generated data after gradient descent had been applied and fake labeled generated data points moved into the decision boundary.



(a) before (b) after

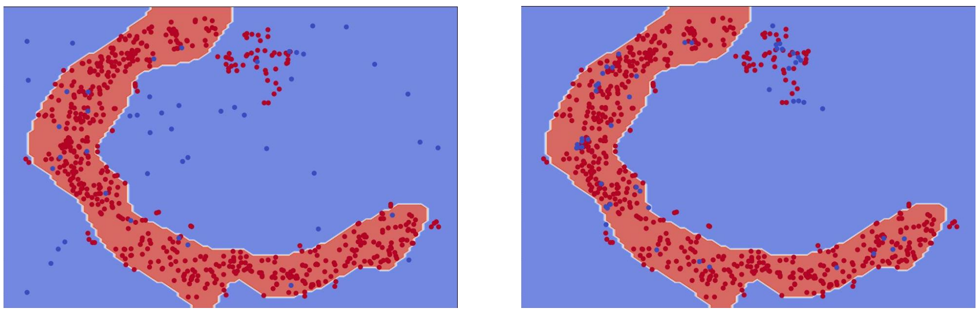
Figure 4.3: Data distribution of original and randomly generated data before and after round 50.



(a) before (b) after

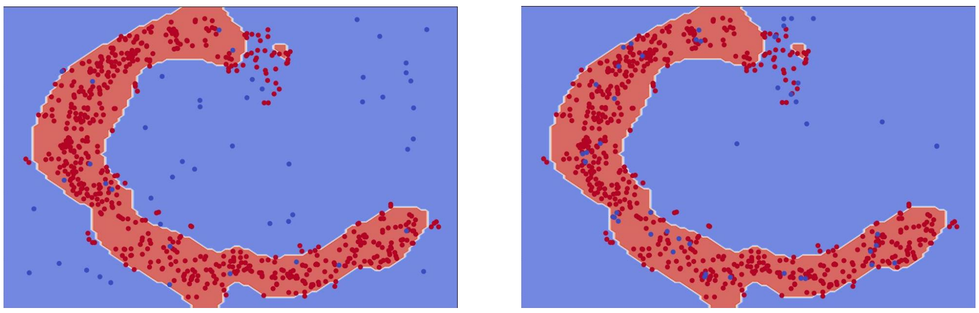
Figure 4.4: Data distribution of original and randomly generated data before and after round 100.

move them into decision boundary, where real labeled data resides. At each update round, data points are clamped into the range $[-1, 1]$ to ease comparison with initial data. Figure 4.2a visualizes randomly generated data and original data on top of the decision boundary right before the first round of the process is started. The process follows the same algorithm as the main method, first, back-propagation is applied to data, then data points that were moved in the plane are labeled as ‘fake’ and fed through the fine-tuning process. Back-propagation effect on data after 25 updates is represented on Figure 4.2b. This process has been repeated a total of 250 times. At each round which is a multiplication of 50 the state of the decision boundary and the data points are visualized.



(a) before (b) after

Figure 4.5: Data distribution of original and randomly generated data before and after round 200.



(a) before (b) after

Figure 4.6: Data distribution of original and randomly generated data before and after round 250.

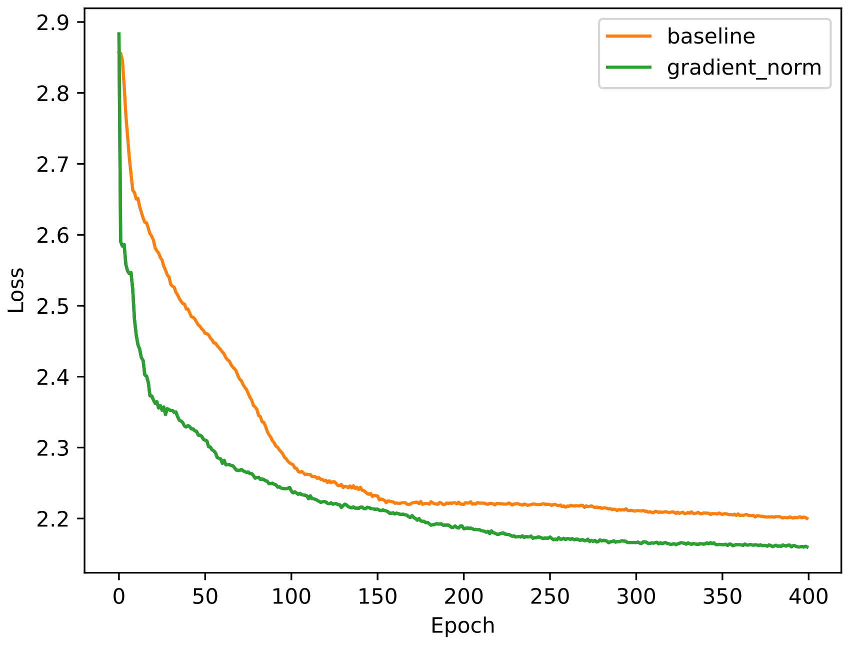


Figure 4.7: Overview of the loss values of the simulation network experiment.

By comparing Figure 4.2a and Figure 4.6b we conclude that the decision boundary is increasing to tightly squeeze distributed data into the space and randomly generated data are successfully being moved into the decision boundary.

We then implemented a baseline version where the gradient norm term is removed from fine-tuning loss and we run these versions for 400 epochs. Figure 4.7 illustrates the effect of gradient norm term on loss values at each epoch which is represented by the vertical axis. As can be observed from the figure, our version with the gradient norm penalty converges both faster and to a lower loss value than the baseline version in 2D plane simulation experiments.

4.2 Main Method

4.2.1 Setup

Experiments were partially run on TRUBA resources with Ubuntu 18.4 operating system using CUDA 11.3. Workers with multiple GPU slot capacities ranging from

4 to 8 resided mainly in partitions akya-cuda and barbun-cuda were made use of.

4.2.2 Datasets

In the upcoming experimental sections, we provide qualitative and quantitative results from networks trained on MNIST [51], EMNIST [52] and Yale Face [53] datasets. MNIST dataset consists of 28×28 gray-scale handwritten digit images. EMNIST is a dataset similar to MNIST and consists of 32×32 gray-scale handwritten letters. Yale Face includes 32×32 gray-scale human face images, taken in different illumination conditions. For our out-of-distribution experiments addition to the previously mentioned datasets, we also use FashionMNIST [54] and KMNIST [55].

4.2.3 Metrics

In the case of out-of-distribution experiments, we use the Area Under the Receiver Operating Characteristics (AUROC) score to compute how well our discriminator distinguishes between classes. It gives a measure of the degree of separability. Higher scores in AUROC mean the discriminator can reject out-of-distribution points better. Loss plots are also provided for tracking network performance and plateau points. We present image sets sampled from models trained on different datasets by utilizing our method.

4.2.3.1 Fréchet Inception Distance

Fréchet Inception Distance (FID) score is used to measure the quality of generated images. Images are first encoded into a feature space by Inception Net, then regarding this latent layer as a multivariate Gaussian distribution, the Fréchet distance is calculated by comparing the mean and variance of generated images and dataset images. Heusel *et al.* [56] shows that FID is more robust to noise injection than the Inception Score (IS) and more similar to human judgment. Numerically smaller values correlate to a more similar distribution between two sets when using FID scores. We use the same Inception Net provided with FID calculation script [57] for all experiments to have a more stable comparison among previous work Taplı [1] and related experiments.

4.2.4 Implementation and Parameter Details of Proposed Method

We implemented our method using Python programming language and made use of PyTorch [58] as the Graphics Processing Unit (GPU) acceleration package. Datasets are loaded using TorchVision [59] package. We use the following Alg. 1 for the implementation of our method. Images are computed as batches and batch size is 128. Before the generation flow is started, the discriminator is initially trained as a binary classifier for 10 epochs. Samples from the dataset are labeled real, and scrambled images are labeled fake. The training step uses Adam [60] as the optimizer for discriminator parameters and the learning rate is chosen to be 0.0001.

For generation purposes, the Adam optimizer is used for wrapping the random data sampled from a multivariate normal distribution and we use a learning rate of 0.01. In the training procedure, we use the ReduceLROnPlateau optimizer from the Pytorch package, which automatically reduces the learning rate depending on the stagnation of learning metrics. The optimizer has a factor of reduction by 0.5, patience is 10, and the minimum learning rate that can be reduced to is 0.00001. The main image generation loop is run for 40 epochs. Lambda values related to the generation objective function in Eq. (3.2) and Eq. (3.4) are chosen to be 0.001 and 100 respectively. The inner loop is controlled by the mean of the output scores. By this criteria, we aim to break the loop when the overall scores of images within a batch have reached a point where they are closer to real images in data space.

At each iteration of the inner loop, gradient descent is applied to random noise. After, the mean of output scores is updated. If the mean has not reached 1 we manually break the loop and the max update number for this is 500. On line 11 in Alg. 1, after the image generation loop, updated images via back-propagation are labeled as fake and appended to the dictionary that stores generated images. We then check the scores of images in $\mathbb{F}$. If the amount of generated images exceeds 3 times the size of the dataset, we then collect the top 3 scored images from $\mathbb{F}$. Our aim with this approach is to feed the discriminator higher-scored images as fake, therefore further forcing the network in a way where it tries to discover boundaries between fake and real images. The discriminator is then further trained (fine-tuned) and model parameters are updated.

In the fine-tuning step of the proposed method, we calculate Eq. (3.8) with 1 batch of real labeled data taken from the dataset and 3 batches of fake labeled data taken from $\mathbb{F}$. We use 50 for λ_{gnorm} in Eq. (3.6) and 0.0001 for λ_{ce} in fine-tuning objective function. Based on the structure of our algorithm, images are generated in an intertwined way, where image generation updates noise data and fine-tuning applies parameter updates on the discriminator. Because of this flow, it can be said that our discriminator generates 1 batch of images after fine-tuning.

Regarding the FID score calculation, we call the FID script for the difference between two sets, the generated images, and the dataset to have a more consistent approach. We randomly collect images from the dataset consisting of an equal number of images as the generated set for our problem scope.

Algorithm 1 Proposed Method

```
Require: Discriminator $D$, Dataset $\mathbb{X}$, Normal distribution $\mathbb{N}$, Scrambled data $\mathbb{S}$
 1: $s \leftarrow \mathbb{S}, x \leftarrow \mathbb{X}, L \leftarrow |\mathbb{X}|$
 2: Train($D$) with $s$ as fake and $x$ as real
 3: for predefined number of rounds do
 4:     $n \leftarrow \mathbb{N}, mean \leftarrow 0, count \leftarrow 0$
 5:     while $mean < 1$ and $count <$ max update number do
 6:         $out \leftarrow D(n)$
 7:         $n \leftarrow$ Apply gradient descent on $n$
 8:         $mean \leftarrow$ Calculate mean of $out$
 9:         $count \leftarrow count + 1$
10:     end while
11:     $\mathbb{F} \leftarrow \mathbb{F} \oplus n$
12:     if $|\mathbb{F}| \geq 3 \cdot \mathbb{L}$ then
13:         $\mathbb{F} \leftarrow$ Get top $3 \cdot \mathbb{L}$ from $\mathbb{F}$
14:     end if
15:     $D \leftarrow$ Fine-tune($D$)
16: end for
```

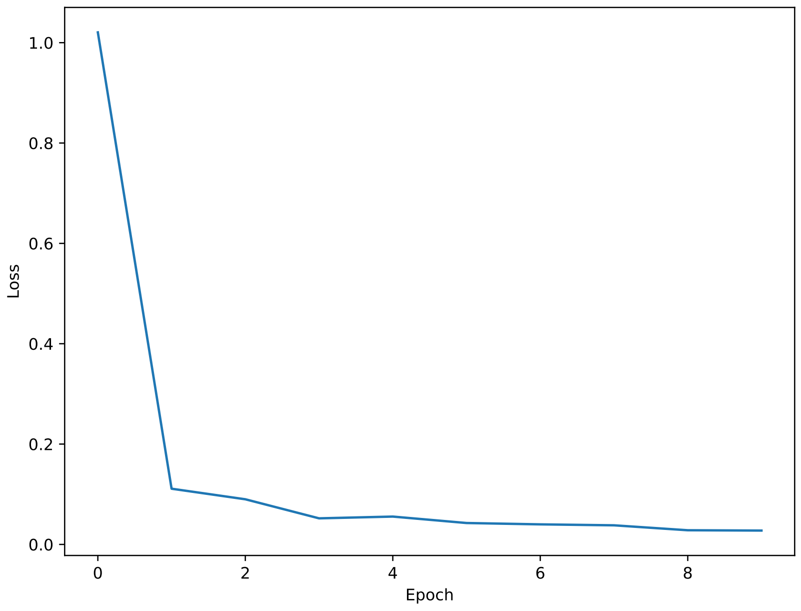


Figure 4.8: Overview of the loss values in initial training.

4.2.5 Results

Figure 4.8 shows the loss values of Eq. (3.1). Our main goal of initial training is to start the generation flow from a point where the discriminator already understands the data distribution in the data space. After the initial training, we calculated the accuracy of our discriminator to be 0.9945.

To compare our proposed improvements onto Taplı [1], we first implement a model called the baseline version and apply additional modifications to have a more coherent cumulative progression. In our experimentation results, this version will be named the baseline. Therefore, any other version can be assumed to be additions made on top of this initial implementation. The baseline version has the architecture visualized in Figure 3.1. All parameters from Taplı [1], are modified to be the same as our method, which depends on the discriminator's structure and is tuned for our method.

We first start by identifying the individual effects of total variation loss, cross-entropy loss, gradient norm restriction, and all 3 terms combined as additions to the baseline version. To achieve distinction among these modifications, we train 5 discriminators with dataset MNIST, each having the respective change applied. Figure 4.9 represents the behavior of the FID score of the baseline version and a second version where the total variation loss is added as a regularization term. It can be seen that the baseline version can reach a numerically smaller score value around epoch number 41, and the version with total variation loss seems to be reaching a plateau quicker than the baseline version. We present the score value change created by adding cross-entropy loss in the fine-tuning objective function in Figure 4.10. The addition of a cross-entropy term to the baseline version does not seem to improve the scores of the generation measurably, but when compared to each other, both reach numerically close score values in the generation process. Last but not least, we investigate the gradient norm restriction effect on the baseline version in Figure 4.11. Compared to the other modifications proposed, the gradient norm penalty affects the score values in a more quantifiable manner. The gradient norm version reaches a much smaller FID score than the baseline version but seems unstable in behavior and starts to increase after reaching a minima point after epoch 20. We illustrate the FID scores from all trained models in Figure 4.12. Baseline, total variation loss, and cross-entropy loss versions seem to be converging at the same rate, but Table 4.1 shows that cross-entropy loss reduces the FID score much faster compared to the other modifications. We also show the best FID scores and which epoch it is gathered from in Table 4.1. As previously mentioned, gradient norm restriction on the fine-tuning loss is numerically more effective than the aforementioned.

Model	FID Score	Epoch
Baseline	57.16	41
Total Variation	63.29	45
Cross Entropy	60.38	29
Gradient Norm	30.96	20
Combined	25.26	23

Table 4.1: Comparison of different discriminators with varying modifications trained on MNIST dataset following our image generation method. FID score and from which epoch the best score is gathered are also presented.

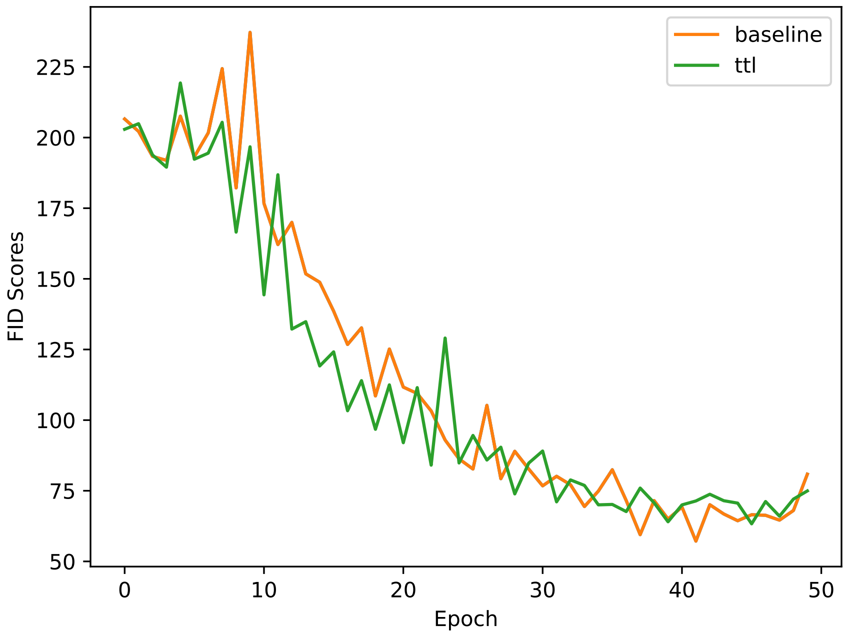


Figure 4.9: FID score comparison of baseline version and our addition of total variation loss as a regularization term in the image update loop.

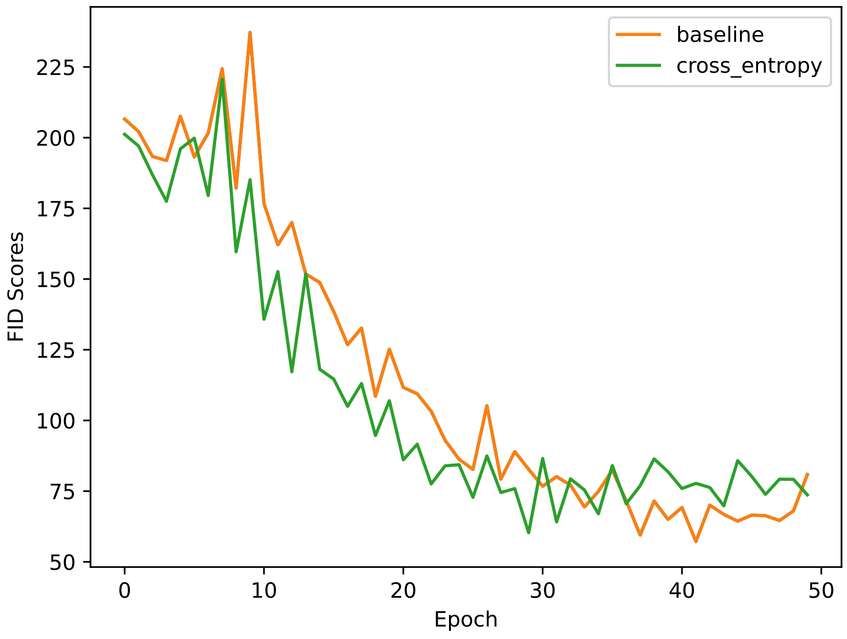


Figure 4.10: FID score comparison of baseline version and our addition of cross-entropy loss as a regularization term in the fine-tuning loop.

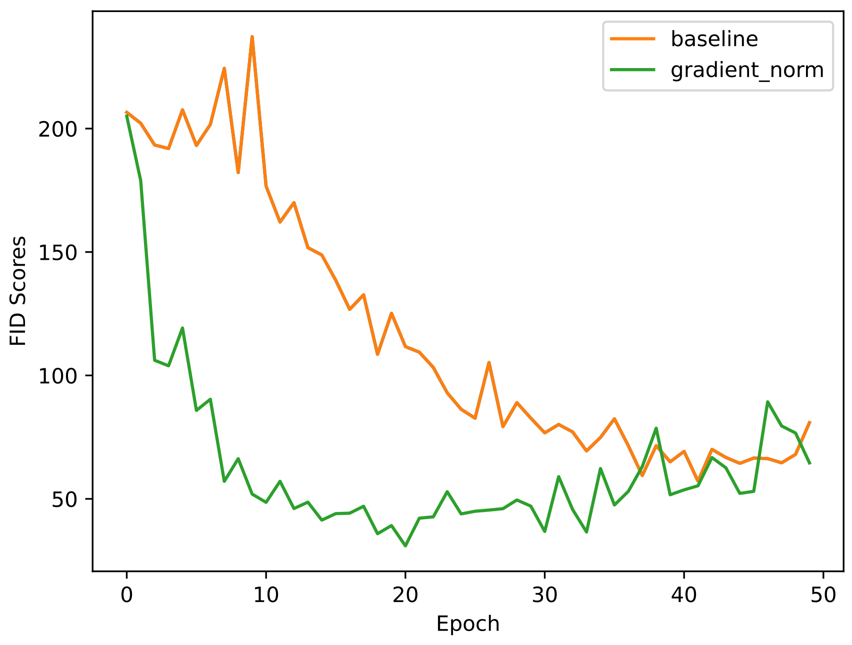


Figure 4.11: FID score comparison of baseline version and our addition of gradient norm restriction in the fine-tuning loop.

When we pay attention to the version named ‘all’ in Figure 4.12, which consists of all improvements formerly noted, it can be seen that the FID scores are stabilizing and reaching a plateau after epoch 23. This points to the conclusion that while gradient norm restriction improves generation’s quantitative and qualitative results, other regularization terms such as total variation loss and cross-entropy loss are meaningful additions for a more stable generation process. To see whether an individual addition of total variation loss or cross-entropy loss is enough to stabilize and improve the training of a gradient norm restricted version, we train two new discriminators. These new discriminators have gradient norm restriction and an individual additional modification of either total variation loss or cross-entropy loss. Figure 4.13 represents the comparison of individual affects made by these additions. The total variation loss version has a minimum FID score of 34.60, and cross-entropy loss has a minimum FID score of 27.69. It can be concluded that the individual regularization effects contributed by total variation loss and cross-entropy loss stabilize the training process of a version where real sample gradient norms are restricted. We conclude that in a case where the semantic labels of the dataset samples are not available, only total variation

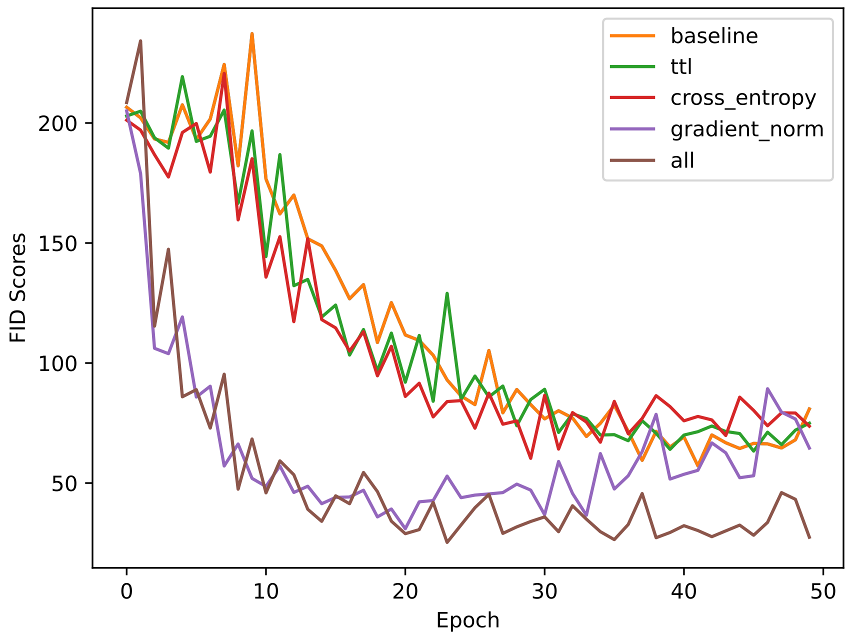


Figure 4.12: Overview of the FID scores of different discriminators with varying modifications trained on MNIST dataset following our image generation method.

loss can be used to stabilize training. By combining all three modifications we obtain the most successful model, but still, the advantages and disadvantages of adding both total-variation loss and cross-entropy loss are open to discussion.

Figure 4.14 presents the qualitative results of the generation process from different discriminators taken from the best-scored epoch detailed in Table 4.1. Figure 4.15 presents successful qualitative results chosen within a 1 batch of generated data created from a discriminator trained on MNIST using our combined method. The FID score of this result set is 25.26. In Figure 4.16, we give qualitative results from the discriminator trained using the EMNIST dataset, and the FID score of this result set is 35.37. Figure 4.17 presents sampled generation results from a discriminator trained in the Yale Face dataset and the FID score of this model is 45.85. Our method outperforms previous work by Taplı [1] which has an FID score of 70.82 on the Yale Face dataset and 115.68 on the EMNIST dataset (See Table 4.4).

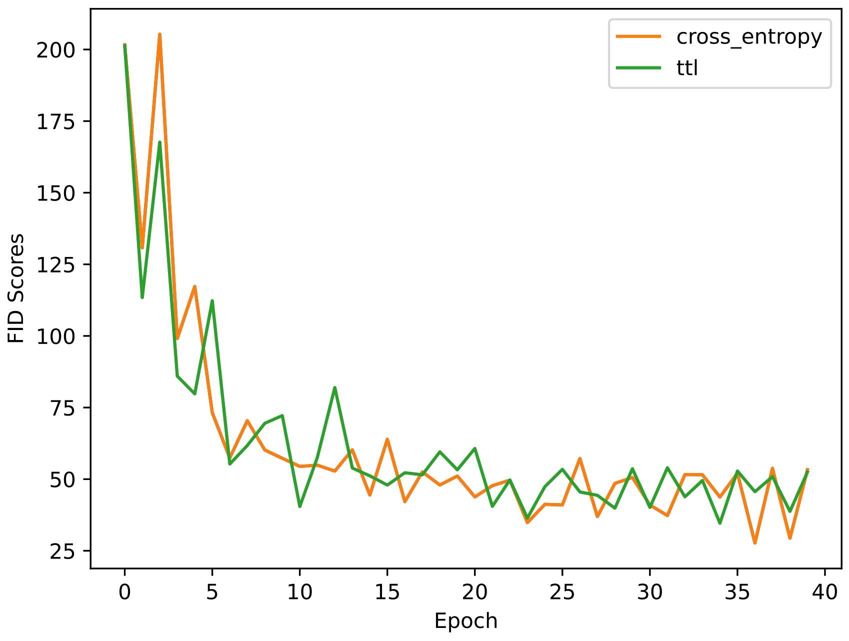
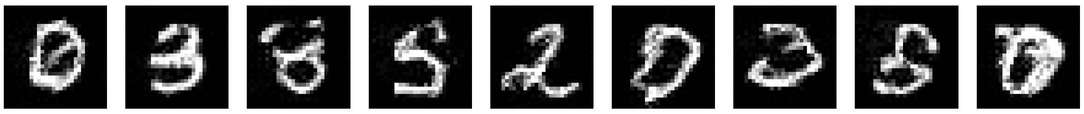
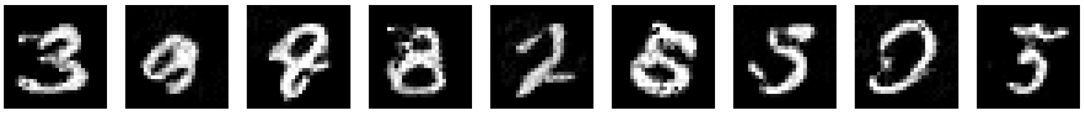
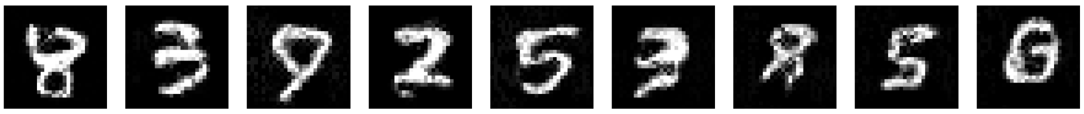


Figure 4.13: Overview of the FID scores from two new discriminators where we add individual modifications of total variation loss and cross-entropy loss on a gradient norm restricted version.


(a) Baseline


(b) Total variation loss


(c) Cross Entropy loss

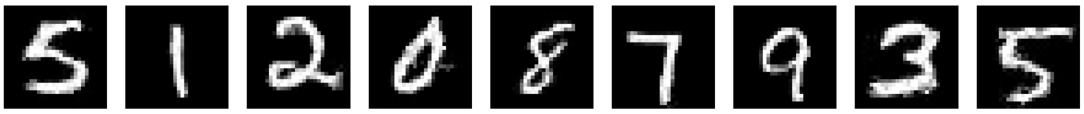

(d) Gradient Norm

Figure 4.14: Qualitative results from effects of different additions to the baseline method. a) Baseline version. FID Score : 57.16 b) Uses total variation loss in generation objective function. FID Score : 64.01 c) Uses cross-entropy loss on finetuning objective function. FID Score : 60.28 d) Enforces gradient norms on real images. FID Score : 30.96.

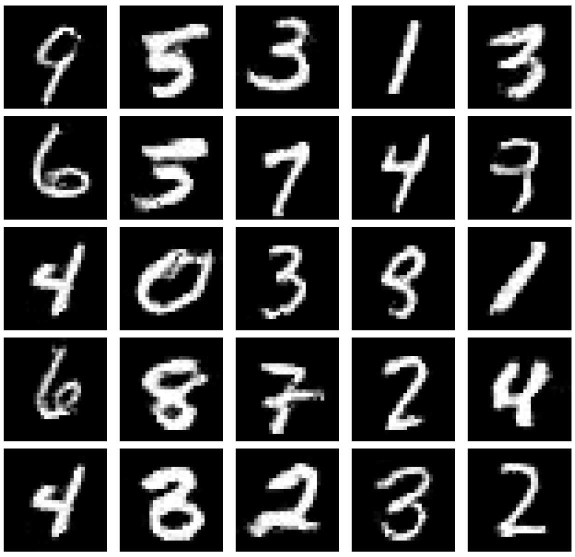


Figure 4.15: Qualitative results from our combined method. Images are chosen from 1 batch of generated data. The discriminator is trained using the MNIST dataset. FID Score : 25.26.

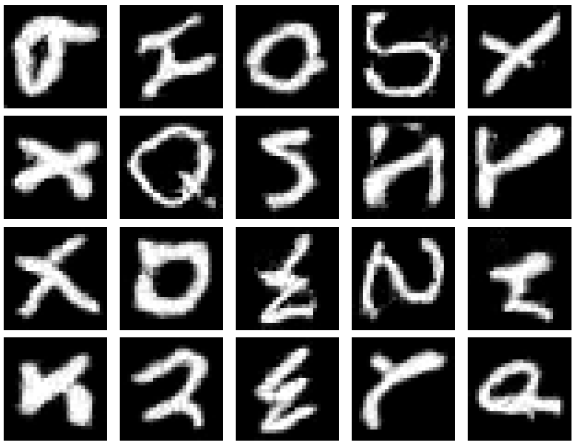


Figure 4.16: Qualitative results from our combined method. Images are chosen from 1 batch of generated data. The discriminator is trained using the EMNIST dataset. FID Score : 35.37

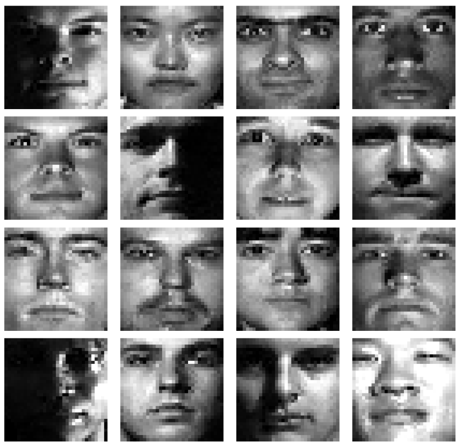


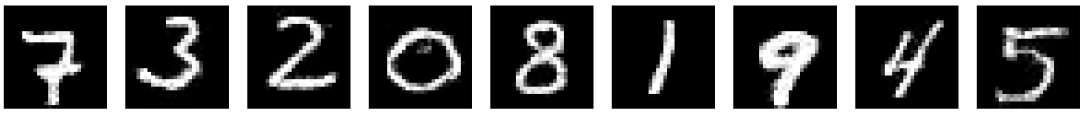
Figure 4.17: Qualitative results from our combined method. Images are chosen from 1 batch of generated data. The discriminator is trained using the Yale Face dataset. FID Score : 45.85.

4.2.5.1 Generated Image Diversity

We test our discriminator's generation ability in regard to overfitting. In Figure 4.18 we present generated images from our discriminator and the nearest example from the dataset to see whether our model memorizes the images rather than generating new examples. Euclidean distance is used to find the nearest samples from the dataset. The first row of Figure 4.18 presents sampled images from one batch of images created via the most successful epoch of our combined model presented in Table 4.1. In the second row, we show the most similar image from the dataset. By comparing these two sets, we show that even though our model generates similar data to the dataset, there are subtle differences. We conclude that there is no memorization of images, and we are able to generate new MNIST data.

4.2.5.2 Uncertainty Estimation

In Table 4.2, we present the AUROC scores of our discriminator. We compare our scores with a state-of-art deep neural network [61], used for uncertainty estimation. We save the most successful model state on epoch 23 from our combined method listed in Table 4.1. The discriminator is trained on MNIST. This version is then loaded to a new script and run on FashionMNIST, KMNIST, and EMNIST datasets


(a) Generated images

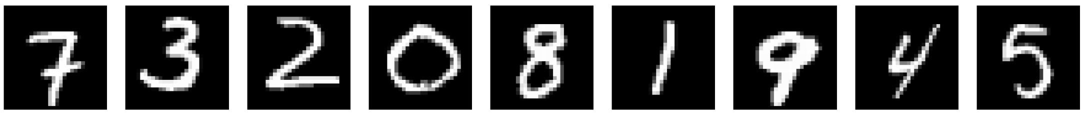

(b) Nearest image from dataset

Figure 4.18: Qualitative results for checking the diversity of our generated images against memorization of dataset images. a) Sampled images from 1 batch of generated data from our discriminator. b) Nearest dataset sample of the generated image from the aligned column.

to see its discrimination abilities with respect to the MNIST dataset. Our model has a high capability of distinguishing data points that do not belong to the dataset it was trained on, which is MNIST for our problem scope. From the FashionMNIST (FM) column on Table 4.2, it can be seen that our model has suppressed the state-of-art uncertainty estimator DUQ [61], for separating MNIST and FashionMNIST datasets. We also present the results of our discriminator on the EMNIST and KMNIST datasets. To sanity check our AUROC scores, we collected the histogram of the output scores from our saved model. Figure 4.19 shows that our model has a tendency to score MNIST data between $[-1.10, -0.95]$ while scoring FashionMNIST data around $[-1.20, -0.99]$. Figure 4.20 shows our discriminator scores KMNIST data around $[-1.20, -0.99]$ therefore increasing the AUROC score. Figure 4.21 shows our discriminator scores EMNIST data around $[-1.07, -0.97]$. This clear distinction among scores allows our model to detect out-of-distribution points successfully and histograms are in line with the high scores we get from the area under the curve calculations.

4.2.6 Compared Models

Table 4.3 compares FID scores based on generated images from different models trained on the MNIST dataset and the learnable parameter counts of respective models. It can be seen that our method's FID score is on-par with different generative models. Respected models such as DCGAN [62], PresGAN [63] and Duelgan [64] has multiple networks similar to vanilla GANs. The most comparable work to ours is Taplı [1] and only has a discriminator network similar to ours. We show that our parameter count is much smaller than the most similar work of Taplı and the model with the best FID score Duelgan. Table 4.4 compares our method's FID scores to the previous work presented by Taplı [1]. We improve the scores on EMNIST and Yale Face datasets significantly.

	FM	KMNIST	EMNIST
DUQ [61]	0.955	-	-
Our method	0.998	0.999	0.938

Table 4.2: Comparison of the AUROC scores gathered from our method and state-of-the-art uncertainty estimator. FM stands for FashionMNIST. Both scores express the capability of models distinguishing datasets from MNIST.

Model	FID Score	Parameter Count
DCGAN [62]	10.85	1,148,098
PresGAN [63]	42.01	6,338,176
Duelgan [64]	7.87	69,154,696
Taplı [1]	28.76	98,902,997
Our method	25.26	14,995,477

Table 4.3: Comparison of the FID scores and learnable parameter counts, including state-of-art generative models, the base model we improved on, and our proposed method.

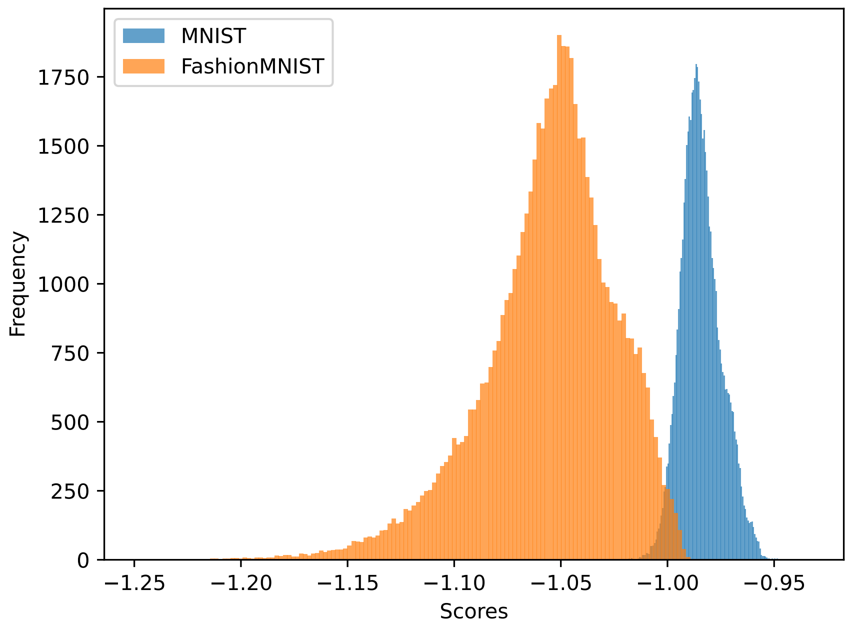


Figure 4.19: Histogram of the output values collected by inputting FashionMNIST and MNIST into our discriminator.

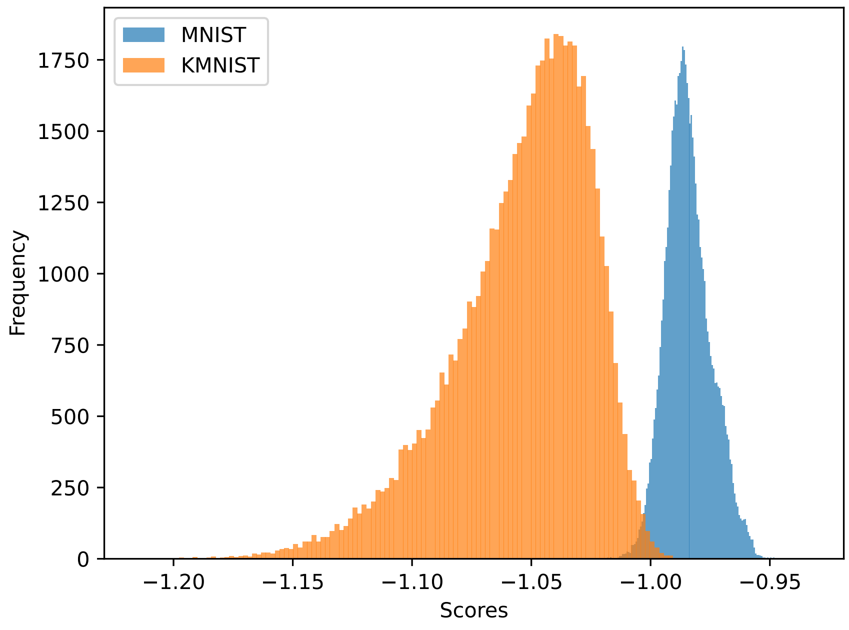


Figure 4.20: Histogram of the output values collected by inputting KMNIST and MNIST into our discriminator.

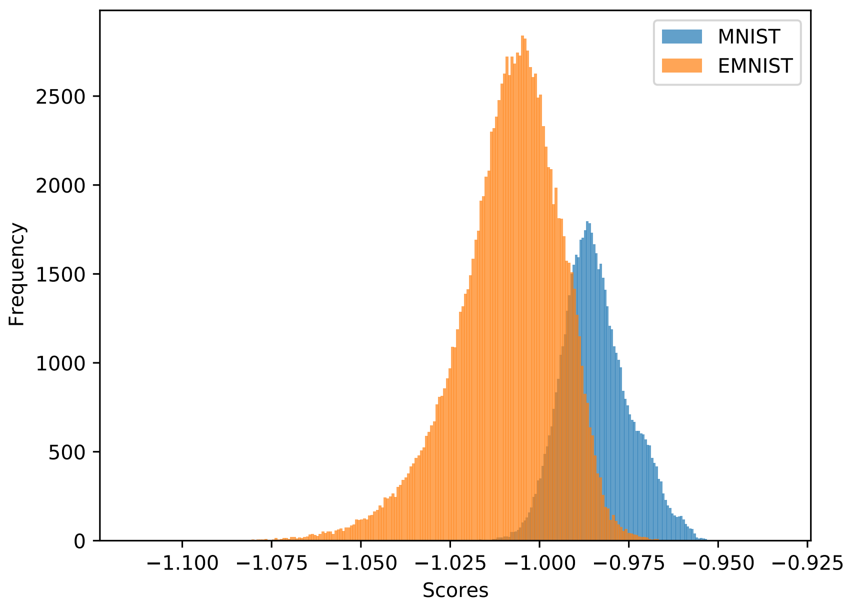


Figure 4.21: Histogram of the output values collected by inputting EMNIST and MNIST into our discriminator.

	Taplı [1]	Our Method
EMNIST	115.68	35.37
Yale Face	70.82	45.85

Table 4.4: Comparison of the FID scores gathered from our method and previous work by Taplı [1] on two different datasets; EMNIST and Yale Face.

4.3 Summary

In this chapter, we presented the experimental results of our method. To substantiate the proposed method, we first simulated on 2D plane, the idea of applying gradient descent on data and showed that by this action, we move randomly generated data into real data boundaries. We then implemented the idea of calculating gradient norms of real images to this simulation and showed that the loss function of the classifier converges faster and reduces to numerically smaller values during training.

We trained the main discriminator on MNIST, EMNIST and Yale Face datasets. Our experiments showed that by applying gradient norm restrictions, cross-entropy loss and total variation loss as regularized terms significantly reduce the FID score, improving the generation abilities. Even though our network uses a much smaller architecture than the base method, it can generate on-par images to the base method and state-of-the-art generative models. We also explore the idea of using our discriminator as an uncertainty estimator and show that our model has a high capability of rejecting out-of-distribution data points belonging to FashionMNIST, EMNIST, and KMNIST datasets.

CHAPTER 5

CONCLUSION

In this thesis, we explored the idea of generating images using only a discriminator network proposed by Taplı [1]. We extended the generation process by employing total variation loss as a measure of the spatial pixel complexity, N-way classification as a measure of the divergence from semantic labels, and gradient norm penalty on real examples. We present the effects of individual modifications previously mentioned. Our experiments show that the additions of the total variation and N-way classification losses do not significantly enhance the generative performance of the discriminator. Applying the gradient norm penalty on real images results in better generative examples. Combining all three modifications result in the best model with a more stable generation process and better FID scores. We showed that our mini convolutional discriminator improves the baseline method's results and can generate realistic-looking images on the MNIST, EMNIST and Yale Face datasets. We reduce the FID scores of generation results to 25.26 on the MNIST dataset. With a discriminator trained on the EMNIST dataset, we get the FID score of 35.37 and on the Yale Face dataset, we reach the FID score of 45.85. We also explored the idea of using our discriminator as an uncertainty estimator. Experiments show that our discriminator trained on the MNIST dataset outperforms state-of-art uncertainty estimator DUQ [61] and has an AUROC score of 0.998 on the FashionMNIST dataset. We also presented the results on KMNIST with a score of 0.999 and EMNIST with a score of 0.938.

5.1 Limitations

One of the drawbacks of our method is having reduced control over the input's generation process and latent space representation. We also need to conduct more experiments to see the effects of our generation method on more complicated datasets with higher dimensionality of the data space, like CIFAR and NORB datasets. Another disadvantage of our model is the reduced classification ability of our model on N-way classifications.

REFERENCES

[1] M. Taplı, “Image generation by back-propagation on input using a discriminator network,” Master’s thesis, Middle East Technical University, 2021.

[2] P. J. Diggle and R. J. Gratton, “Monte carlo methods of inference for implicit statistical models,” *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 46, no. 2, pp. 193–212, 1984.

[3] S. Mohamed and B. Lakshminarayanan, “Learning in implicit generative models,” *arXiv preprint arXiv:1610.03483*, 2016.

[4] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” *Advances in neural information processing systems*, vol. 27, 2014.

[5] A. Brock, J. Donahue, and K. Simonyan, “Large scale gan training for high fidelity natural image synthesis,” *arXiv preprint arXiv:1809.11096*, 2018.

[6] T. Karras, S. Laine, and T. Aila, “A style-based generator architecture for generative adversarial networks,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4401–4410, 2019.

[7] T. Karras, T. Aila, S. Laine, and J. Lehtinen, “Progressive growing of gans for improved quality, stability, and variation,” *arXiv preprint arXiv:1710.10196*, 2017.

[8] A. Gonzalez-Garcia, J. Van De Weijer, and Y. Bengio, “Image-to-image translation for cross-domain disentanglement,” *Advances in neural information processing systems*, vol. 31, 2018.

[9] Y. Choi, Y. Uh, J. Yoo, and J.-W. Ha, “Stargan v2: Diverse image synthesis for multiple domains,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8188–8197, 2020.

[10] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee, “Generative adversarial text to image synthesis,” in *International conference on machine learning*, pp. 1060–1069, PMLR, 2016.

[11] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. N. Metaxas, “Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks,” in *Proceedings of the IEEE international conference on computer vision*, pp. 5907–5915, 2017.

[12] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum, “Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling,” *Advances in neural information processing systems*, vol. 29, 2016.

[13] Y. Jin, J. Zhang, M. Li, Y. Tian, H. Zhu, and Z. Fang, “Towards the automatic anime characters creation with generative adversarial networks,” 08 2017.

[14] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” *arXiv preprint arXiv:1312.6114*, 2013.

[15] D. J. Rezende, S. Mohamed, and D. Wierstra, “Stochastic backpropagation and variational inference in deep latent gaussian models,” in *International conference on machine learning*, vol. 2, p. 2, Citeseer, 2014.

[16] Y. Burda, R. Grosse, and R. Salakhutdinov, “Importance weighted autoencoders,” *arXiv preprint arXiv:1509.00519*, 2015.

[17] A. Van Oord, N. Kalchbrenner, and K. Kavukcuoglu, “Pixel recurrent neural networks,” in *International conference on machine learning*, pp. 1747–1756, PMLR, 2016.

[18] H. Larochelle and I. Murray, “The neural autoregressive distribution estimator,” in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 29–37, JMLR Workshop and Conference Proceedings, 2011.

[19] D. E. Rumelhart, J. L. McClelland, and C. PDP Research Group, eds., *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*. Cambridge, MA, USA: MIT Press, 1986.

[20] Y. Wang, H. Yao, and S. Zhao, “Auto-encoder based dimensionality reduction,” *Neurocomputing*, vol. 184, pp. 232–242, 2016.

[21] L. Gondara, “Medical image denoising using convolutional denoising autoencoders,” in *2016 IEEE 16th international conference on data mining workshops (ICDMW)*, pp. 241–246, IEEE, 2016.

[22] M. Hasan, J. Choi, J. Neumann, A. K. Roy-Chowdhury, and L. S. Davis, “Learning temporal regularity in video sequences,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 733–742, 2016.

[23] D. Gong, L. Liu, V. Le, B. Saha, M. R. Mansour, S. Venkatesh, and A. v. d. Hengel, “Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1705–1714, 2019.

[24] C. Song, F. Liu, Y. Huang, L. Wang, and T. Tan, “Auto-encoder based data clustering,” in *Iberoamerican congress on pattern recognition*, pp. 117–124, Springer, 2013.

[25] P. Baldi, “Autoencoders, unsupervised learning, and deep architectures,” in *Proceedings of ICML Workshop on Unsupervised and Transfer Learning* (I. Guyon, G. Dror, V. Lemaire, G. Taylor, and D. Silver, eds.), vol. 27 of *Proceedings of Machine Learning Research*, (Bellevue, Washington, USA), pp. 37–49, PMLR, 02 Jul 2012.

[26] E. Plaut, “From principal subspaces to principal components with linear autoencoders,” *arXiv preprint arXiv:1804.10253*, 2018.

[27] J. M. Tomczak and M. Welling, “Vae with a vampprior,” 2017.

[28] K. Gregor, F. Besse, D. Jimenez Rezende, I. Danihelka, and D. Wierstra, “Towards conceptual compression,” *Advances In Neural Information Processing Systems*, vol. 29, 2016.

[29] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, “Context encoders: Feature learning by inpainting,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2536–2544, 2016.

[30] U. Demir and G. Unal, "Patch-based image inpainting with generative adversarial networks," *arXiv preprint arXiv:1803.07422*, 2018.

[31] X. Huang, M.-Y. Liu, S. Belongie, and J. Kautz, "Multimodal unsupervised image-to-image translation," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 172–189, 2018.

[32] A. Royer, K. Bousmalis, S. Gouws, F. Bertsch, I. Mosseri, F. Cole, and K. Murphy, "Xgan: Unsupervised image-to-image translation for many-to-many mappings," 2017.

[33] Z. Li, M. R. Min, K. Li, and C. Xu, "Stylet2i: Toward compositional and high-fidelity text-to-image synthesis," 2022.

[34] J. Sun, Q. Deng, Q. Li, M. Sun, M. Ren, and Z. Sun, "Anyface: Free-style text-to-face synthesis and manipulation," 2022.

[35] K. Xu, C. Li, J. Zhu, and B. Zhang, "Understanding and stabilizing gans' training dynamics with control theory," *arXiv preprint arXiv:1909.13188*, 2019.

[36] Z. Li, X. Wu, M. Usman, R. Tao, P. Xia, H. Chen, and B. Li, "A systematic survey of regularization and normalization in gans," *arXiv preprint arXiv:2008.08930*, 2020.

[37] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," 2017.

[38] C. Chu, K. Minami, and K. Fukumizu, "Smoothness and stability in gans," *arXiv preprint arXiv:2002.04185*, 2020.

[39] W. Fedus, M. Rosca, B. Lakshminarayanan, A. M. Dai, S. Mohamed, and I. Goodfellow, "Many paths to equilibrium: Gans do not need to decrease a divergence at every step," 2017.

[40] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *International Conference on Machine Learning*, pp. 2256–2265, PMLR, 2015.

[41] A. Hyvärinen and P. Dayan, "Estimation of non-normalized statistical models by score matching.," *Journal of Machine Learning Research*, vol. 6, no. 4, 2005.

[42] C. M. Stein, "A bound for the error in the normal approximation to the distribution of a sum of dependent random variables," 1972.

[43] Y. Song and S. Ermon, "Generative modeling by estimating gradients of the data distribution," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[44] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in Neural Information Processing Systems*, vol. 33, pp. 6840–6851, 2020.

[45] Z. Kong, W. Ping, J. Huang, K. Zhao, and B. Catanzaro, "Diffwave: A versatile diffusion model for audio synthesis," *arXiv preprint arXiv:2009.09761*, 2020.

[46] G. Parisi, "Correlation functions and computer simulations," *Nuclear Physics B*, vol. 180, no. 3, pp. 378–384, 1981.

[47] Y. Song and S. Ermon, "Improved techniques for training score-based generative models," *Advances in neural information processing systems*, vol. 33, pp. 12438–12448, 2020.

[48] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," 2020.

[49] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," *Advances in neural information processing systems*, vol. 30, 2017.

[50] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[51] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[52] G. Cohen, S. Afshar, J. Tapson, and A. Van Schaik, "Emnist: Extending mnist to handwritten letters," in *2017 international joint conference on neural networks (IJCNN)*, pp. 2921–2926, IEEE, 2017.

[53] A. Georghiades, P. Belhumeur, and D. Kriegman, "From few to many: Illumination cone models for face recognition under variable lighting and pose," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, pp. 643–660, 06 2001.

[54] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," 2017.

[55] T. Clanuwat, M. Bober-Irizar, A. Kitamoto, A. Lamb, K. Yamamoto, and D. Ha, "Deep learning for classical japanese literature," *CoRR*, vol. abs/1812.01718, 2018.

[56] M. Lucic, K. Kurach, M. Michalski, S. Gelly, and O. Bousquet, "Are gans created equal? a large-scale study," *Advances in neural information processing systems*, vol. 31, 2018.

[57] M. Seitzer, "pytorch-fid: FID Score for PyTorch." `https://github.com/mseitzer/pytorch-fid`, August 2020. Version 0.2.1.

[58] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, pp. 8024–8035, Curran Associates, Inc., 2019.

[59] S. Marcel and Y. Rodriguez, "Torchvision the machine-vision package of torch," pp. 1485–1488, 10 2010.

[60] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[61] J. van Amersfoort, L. Smith, Y. W. Teh, and Y. Gal, "Uncertainty estimation using a single deep deterministic neural network," 2020.

[62] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[63] A. B. Dieng, F. J. Ruiz, D. M. Blei, and M. K. Titsias, "Prescribed generative adversarial networks," *arXiv preprint arXiv:1910.04302*, 2019.

[64] J. Wei, M. Liu, J. Luo, A. Zhu, J. Davis, and Y. Liu, "Duelgan: A duel between two discriminators stabilizes the gan training," 2021.