Triple Generative Adversarial Nets

Project Report Submitted by

M Rupesh kumar Yadav (420207)

G Sirisha Ramya (420136)

B Sravani Sandhya (420116)

Under the supervision of

Dr. Nagesh Bhattu Sristy

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY ANDHRA PRADESH

TADEPALLIGUDEM-534101,INDIA

2023

Triple Generative Adversarial Nets

Submitted in the partial fulfillment of the requirements

of the degree of

Bachelor of Technology

By

M Rupesh kumar Yadav (420207)

G Sirisha Ramya (420136)

B Sravani Sandhya (420116)

Under the supervision of

Dr. Nagesh Bhattu Sristy

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY ANDHRA PRADESH

TADEPALLIGUDEM-534101,INDIA

2023

Declaration

Department of Computer Science and Engineering

NATIONAL INSTITUTE OF TECHNOLOGY ANDHRA PRADESH

Certificate

It is certified that the work contained in the thesis titled “Triple Generative Adversarial Nets ” by “Mediboyina Rupesh Kumar Yadav, bearing Roll No: 420207” , “G Sirisha Ramya, bearing Roll No: 420136” and “B Sravani Sandhya, bearing Roll No: 420116” has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

Dr. Nagesh Bhattu Sristy

Computer Science and Engineering

N.I.T.Andhra Pradesh

May 2023

Abstract

Deep Generative models(DGM) are neural networks which tries to estimate the likelihood of each observation and to create new samples from the underlying distribution of data, Synthetic data generation is a specific application of deep generative models where the goal is to create realistic and novel data samples. GANs, in particular, consist of a generator network that produces new samples and a discriminator network that distinguishes between real and generated samples. Through an adversarial training process, both networks improve iteratively, with the generator learning to produce more realistic data.

However, challenges exist. Ensuring diversity and realism in generated samples is an ongoing research focus. Issues like training stability, mode collapse, and addressing biases in generated data are important considerations. Ethical concerns regarding the potential misuse of synthetic data and privacy implications must be taken into account. The Triple GAN focuses on classification and class-conditional generation of data samples and also a better framework to work with as it requires less number of epochs to train, can perform well even without data argumentation and can perform semi-supervised learning(SSL) tasks better.

Contents

List of Figures

Chapter 1

Introduction

Machine learning models can be classified into Discriminative and Generative models. A Discriminative model makes predictions on unseen data based on conditional probability and can be used either for classification or regression problem statements. On the contrary, a generative model focuses on the distribution of a dataset and tries to mimic the distribution. Simply, A Discriminative Model aims to predict the output label Y, hemce it models the posterior distribution P(Y/X).Generative models instead models the distribution P(X) defined over the data points X. Learning Representations, due to high dimensionality of the raw data of real-world tasks, a crucial step for efficiency (memory and computation) is learning a semantically meaningful subspace is important.

Machine learning projects often require large datasets with accurately labeled real-world data, which can be difficult and time-consuming to collect. Synthetic data, however, offers a growing alternative by using various techniques to generate data with similar properties to real data. Synthetic data has several advantages, including reduced cost, high accuracy in labeling, scalability, and variety. It can be used to create data samples for edge cases that may not frequently occur in the real world, leading to improved model performance. This approach offers significant benefits without the need for collecting and labeling large datasets.

In this, we explore the generation of synthetic data using Deep Generative Models and dealing with its shortcomings, especially Generative Adversial Networks(GAN) under extremely low regime Data and semi-supervised Learning(SSL).

Chapter 2

Literature Survey

2.1 Deep Generative Models

Deep Generative Models are a class of machine learning models that learn to generate new data that is similar to the input data. These models use deep neural networks to learn the underlying distribution of the input data, which can then be used to generate new data points. Examples of deep generative models include Variational Autoencoders (VAEs), Generative Adversarial Networks (GANs), and Autoregressive Models. These models have numerous applications in fields such as image and video synthesis, speech and natural language generation, and drug discovery.

2.1.1 Autoregressive models

Autoregressive models are a class of generative models that model the conditional distribution of each element in a sequence given the previous elements in the sequence. In other words, the probability distribution of each element in the sequence depends on the values of the previous elements.autoregressive models for image generation. In this case, the autoregressive model would be trained on a large dataset of images and would learn the conditional distribution of each pixel given the previous pixels. The model would generate the image one pixel at a time, conditioning on the previous pixels. Once the model is trained, it can be used to generate new images that are similar to the original data. One advantage of using autoregressive models for synthetic data generation is that they can generate high-quality samples that are similar to the original data. In addition, autoregressive models can be trained on a wide range of data types, including text, images, and time series data.

However, autoregressive models can be computationally expensive to train, especially for large datasets or long sequences. In addition, autoregressive models may not capture all the important features of the original data, which can result in generated samples that are not representative of the true distribution. Finally, autoregressive models may suffer from the problem of exposure bias, where the model is trained on ground-truth inputs but at inference time, it needs to generate the next element in the sequence based on its own predictions. This can result in a mismatch between training and inference, which can lead to poor performance.

2.1.2 Variational autoencoders (VAEs)

VAEs learn a low-dimensional representation of the data and generate new data by sampling from the learned latent space. The encoder network maps the input data to a latent space, while the decoder network maps a point in the latent space back to the original data space. During training, VAEs learn to maximize a lower bound on the log-likelihood of the data. This encourages the model to learn a low-dimensional representation of the data that captures the underlying structure of the data[2].

VAEs generate synthetic images by taking input vectors representing real-world images and producing output vectors that resemble the input images with some degree of variation. This variation is introduced using a layer of means and standard deviations, which ensures that the generated images are similar to the source images but not identical.

The main drawback of VAEs for synthetic image generation is that they generate blurry outputs, which also tend to be unrealistic. Recent research has suggested modifications to the original VAE model that can improve output quality.

2.1.3 Generative adversarial networks (GANs)

Generative Adversarial Networks (GANs) are a type of deep learning model that can generate synthetic data, such as images, that are similar to real-world data. GANs work by training two neural networks simultaneously: a generator and a discriminator. The generator takes in random noise as input and generates a synthetic image, while the discriminator takes in both the synthetic image and a real image and tries to distinguish between them.

The generator creates new data samples that resemble the training data, while the discriminator evaluates the authenticity of these generated samples. The two networks are trained simultaneously, known as the two-player game framework [5] (Ian Goodfellow,nips2014) with the generator trying to produce samples that can fool the discriminator, and the discriminator learning to distinguish between the generated samples and real ones.

GANs are implicit generative models,[7] (implicit models) which means that they do not explicitly model the likelihood function nor provide a means for finding the latent variable corresponding to a given sample

One advantage of GANs is that they can generate highly realistic images that are difficult to distinguish from real images. They are also able to generate a wide range of diverse images, which is useful for tasks such as data augmentation. Additionally, GANs are able to generate images with a high level of detail, making them well-suited for tasks such as image synthesis and style transfer.

However, GANs also have some limitations. One major challenge is that they can be difficult to train and may suffer from instability during training. This is because the generator and discriminator are both being trained simultaneously, which can lead to oscillations or mode collapse, where the generator learns to generate only a few types of images. GANs can also require a large amount of computational resources and can be slow to train, especially for large image datasets.

In summary, GANs are a powerful approach to generating synthetic image data that can produce highly realistic and diverse images. However, they can be difficult to train and require careful tuning of the hyperparameters to achieve good results

2.1.4 Flow-based models

Flow-based generative models are a type of generative model that learn a bijective mapping between a simple base distribution and the target data distribution. The mapping is typically implemented as a series of invertible transformations, such as affine transformations or nonlinear transformations like neural networks. The basic idea behind flow-based models is to transform the simple base distribution (e.g., Gaussian) into the target data distribution by applying a series of transformations.

The main advantage of flow-based models is that they can model the likelihood of the data directly, which makes them well-suited for density estimation tasks. In addition, they have stable training dynamics and are relatively easy to train compared to other generative models like GANs or VAEs. Once the model is trained, it is easy to sample from it, which makes it a good choice for applications where fast sampling is required and making them well-suited for density estimation tasks.It has fast sampling, stable training and no need for speciallized inference, Unlike VAEs and GANs,making them easier to implement and apply in practice.

- Limited modeling power: Flow-based models are limited by their choice of base distribution. If the base distribution is not expressive enough, the model may not be able to capture complex dependencies in the data.
- Large memory requirements: Flow-based models require storing the full Jacobian of the transformation, which can be computationally expensive and require a large amount of memory.
- Lack of diversity: Flow-based models may suffer from lack of diversity in generated samples, especially if the model is not well-optimized or the data distribution is highly complex.

Overall, flow-based generative models are a promising approach to synthetic data generation. However, they may not be the best choice for all applications, and the choice of model should be carefully considered based on the specific task at hand.

Chapter 3

Problem Statement

Understanding the Deep Generative Models for Synthetic Image generation and discussing the problem in the GAN formulation and ways to overcome it. The problem statement of the research paper is that existing Generative Adversarial Nets (GANs) used in semi-supervised learning have two issues: (1) the generator and discriminator may not be optimal at the same time, and (2) the generator cannot control the semantics of the generated samples. These problems arise from the single discriminator that identifies fake samples and predicts labels, leading to incompatible roles. The paper proposes Triple-GAN, a model with three players (generator, discriminator, and classifier) to address these issues and achieve state-of-the-art classification results while disentangling classes and styles of input.

Chapter 4

Methodology

The architecture of our proposed model is depicted in Figure 4.2

4.1 GAN

GAN is formulated as a two-player game. The Generator G takes a random noise z as input and produces a sample G(z) in the data space. The discriminator D identifies whether a certain sample comes from the true data distribution p(x) or the generator. Both of them are parameterized as deep neural networks, We train D to maximize the probability of assigning the correct label to both training examples and samples from G. We simultaneously train G to minimize log(1 - D(G(z))).

$$\min_G \max_D U(D,G) = E_{p(x,y)}[\log D(x)] + E_{z\sim p(z)}[\log(1 - D(G(z)))] \tag{4.1}$$

In this two-player game formulation the generator and discriminator compete with eachother to reach a stable point, and thus called adversial game play. A known dataset serves as the initial training data for the discriminator. Training involves presenting it with samples from the training dataset until it achieves acceptable accuracy. The generator is trained based on whether it succeeds in fooling the discriminator. Typically, the generator is seeded with randomized input that is sampled from a predefined latent space (e.g. a multivariate normal distribution). Thereafter, candidates synthesized by the generator are evaluated by the discriminator. Independent backpropagation procedures are applied to both networks so that the generator produces better samples, while the discriminator becomes more skilled at flagging synthetic samples.

Existing GANs in SSL have two problems: (1) the generator and the discriminator (i.e., the classifier) may not be optimal at the same time (2) the generator

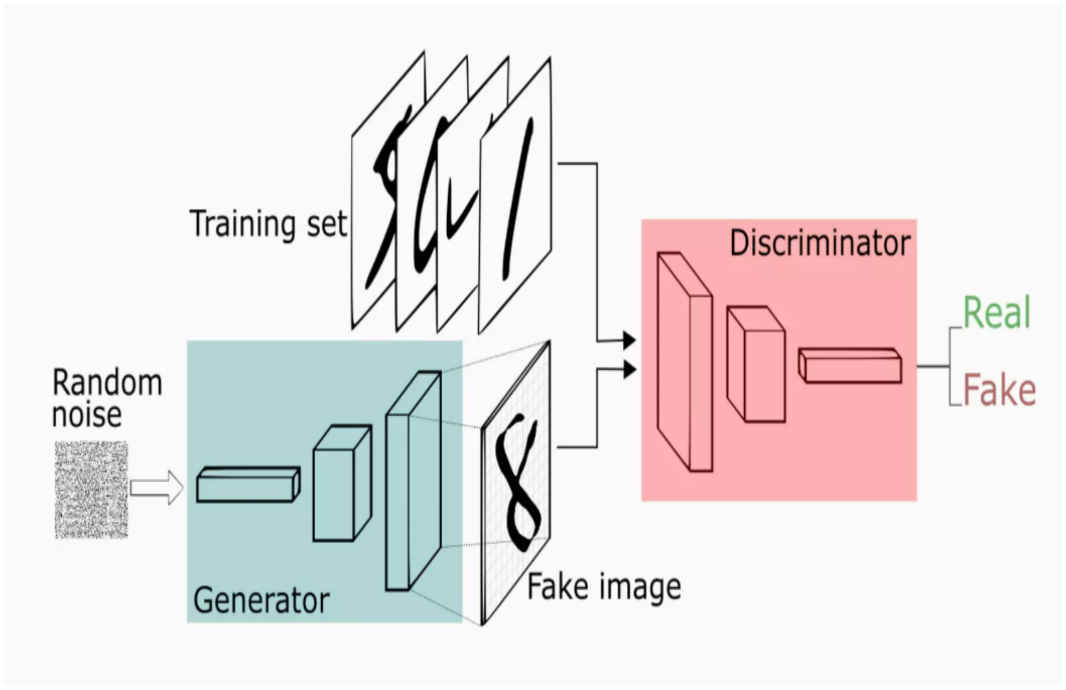


Figure 4.1: GAN Architecture [1]

cannot control the semantics of the generated samples. The problems essentially arise from the two-player formulation, where a single discriminator shares incompatible roles of identifying fake samples and predicting labels and it only estimates the data without considering the labels

For the first problem, as an instance,Salimans et al . propose two alternative training objectives that work well for either classification or image generation in SSL, but not both. They are :
(1) Feature matching works well in classification but fails to generate indistinguishable samples
(2) Minibatch discrimination is good at realistic image generation but cannot predict labels accurately

A single discriminator network which has the sole role of distinguishing whether a data-label pair is from the real labeled dataset or not is assumed as the core reason, and taken into consideration. As the discriminators in this method estimate a single data instead of a data-label pair and information is totally ignored, Thus the generators cannot receive any learning data regarding the label information from the discriminator. Hence the generator cannot control the semantics of the generated samples

4.2 Triple GAN

Triple generative adversarial network (Triple-GAN) is a unified game framework with a three player game formulation to suit for both classification and class-conditional image generation with limited supervision. The Triple GAN architecture contains two conditional networks a Classifier and a Generator to generate fake labels given real data and fake data given real labels, which will perform the classification and class-conditional generation tasks respectively. To jointly justify the quality of the samples from the conditional networks, we define a discriminator network which has the sole role of distinguishing whether a data-label pair is from the real labeled dataset or not. The resulting model is called Triple-GAN where it has three joint distributions, the true data-label distribution and the distributions defined by the two conditional networks

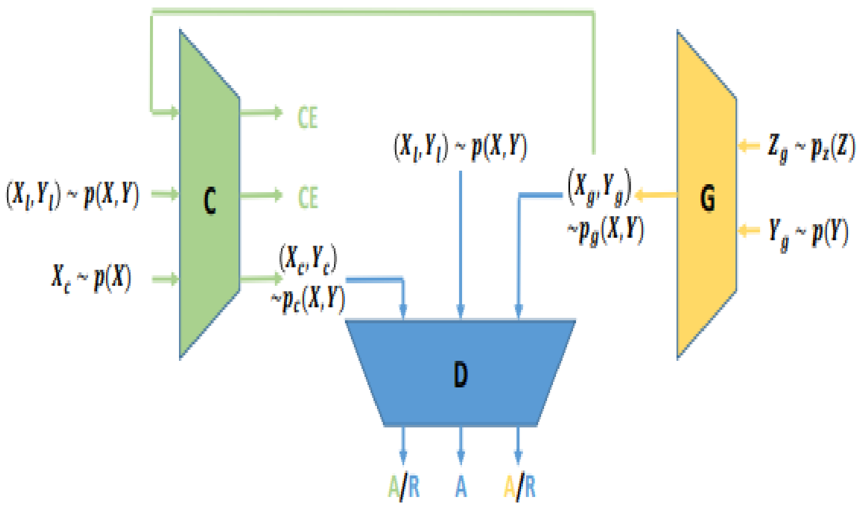


Figure 4.2: Triple GAN [16]

In the semi-supervised setting, we aim to learn DGMs using a partially labeled dataset where input data is denoted as x and the output label is denoted as y. The objective is twofold: to predict labels for unlabeled data and to generate new samples conditioned on y. However, in contrast to the unsupervised setting for pure generation, where the generator only needs to sample data x from a generator to deceive a discriminator, the semi-supervised setting requires the generator to characterize the uncertainty of both x and y. Hence, a joint distribution p(x, y) of input-label pairs is needed. However, applying the two-player GAN is not feasible due to missing values on y. Hence, we extend GANs to Triple-GAN, a three-player game to characterize the process of classification and class-conditional generation in SSL.

4.3 Three Player Formulation

The Triple-GAN consists of three components: (1) a classifier C that (approximately) characterizes the conditional distribution $p_c(y|x) \approx p(y|x)$; (2) a class-conditional generator G that (approximately) characterizes the conditional distribution in the other direction $p_g(y|x) \approx p(x|y)$; and (3) a discriminator D that distinguishes whether a pair of data (x, y) comes from the true distribution $p(x, y)$. All the components are parameterized as neural networks. Our desired equilibrium is that the joint distributions defined by the classifier and the generator both converge to the true data distribution.

In the game, initially a sample x is drawn from $p(x)$, C produces a fake label y given x following the conditional distribution $p_c(y|x)$. The fake input-label pair is a sample from the joint distribution $p_c(y, x) = p(x)p_c(y|x)$. Similarly, a fake input-label pair is sampled from G by first drawing $y \sim p(y)$ and then drawing $x|y \sim p_g(x|y)$, Hence from the joint distribution $p_g(x, y) = p(y)p_g(x|y)$. For $p_g(x|y)$, we assume that x is transformed by the latent style variables z given the label y, namely, x = G(y, z), $z \; p_z(z)$, where $p_z(z)$ is a simple distribution (e.g., uniform or standard normal). The fake input-label pairs (x, y) generated by both C and G are sent to the discriminator D. D can also access the input-label pairs from the true data distribution as positive samples.

$$\begin{aligned} \min_{C,G} \max_{D} \; & E_{p(x,y)}[\log D(x, y)] + \alpha E_{p_c(x,y)}[\log(1 - D(x, y))] \\ & + (1 - \alpha)E_{p_g(x,y)}[\log(1 - D(G(y, z), y))] \end{aligned} \tag{4.2}$$

$\alpha \in (0, 1)$ is a constant that controls the relative importance of classification and generation. For convenience, $\alpha = \frac{1}{2}$.

Our desired equilibrium is that the joint distributions defined by the classifier and the generator both converge to the true data distribution.

4.3.1 Computing Losses

The Triple-GAN the objective functions and distributions are designed such as to reach the desired equilibrium. Let the U(C, G, D) be representing the total loss of the model, and it is an integral over the cross entropy losses of Generator, Classifier and Discriminator. As it cannot be formulated directly, we assume We randomly start with a optimal discriminator and then find the optimal parameters using the derived components

$$\begin{aligned} U(C,G,D) = & \iint p(x,y)\log D(x,y)\,dy\,dx \\ & + (1-\alpha)\iint p(y)p_z(z)\log(1-D(G(z,y),y))\,dy\,dz \\ & + \alpha \iint p(x)p_c(y|x)\log(1-D(x,y))\,dy\,dx \\ = & \iint p(x,y)\log D(x,y)\,dy\,dx + \iint p_\alpha(x,y)\log(1-D(x,y))\,dy\,dx, \end{aligned}$$

Differentiate with respective to D(x,y) and equating it to zero gives the optimal discriminator, which achieves the maximum at $\frac{p(x,y)}{p(x,y)+p_\alpha(x,y)}$.

$D^*_{C,G}$, representing the optimal discriminator, we can reformulate the minimax game with value function U as:

$$V(C,G) = \max_D U(C,G,D)$$

$$V(C,G) = \iint \frac{p(x,y)\log p(x,y)}{p(x,y)+p_\alpha(x,y)}\,dy\,dx + \iint \frac{p_\alpha(x,y)\log p_\alpha(x,y)}{p(x,y)+p_\alpha(x,y)}\,dy\,dx.$$

Following the proof in GAN [5], the $V(C,G)$ can be rewritten as

$$V(C,G) = -\log 4 + 2D_{JS}(p(x,y)||p_\alpha(x,y)),$$

where D_{JS} is the Jensen-Shannon divergence, which is always non-negative, and the unique optimum is achieved if and only if $p(x,y) = p_\alpha(x,y) = (1-\alpha)p_g(x,y) + \alpha p_c(x,y)$.

Assuming that true distribution is mimicked (approximately) by the conditional networks $p(x,y) = p_\alpha(x,y)$, the marginal distributions of $p(x,y)$, $p_c(x,y)$, and $p_g(x,y)$ are the same, i.e., $p(x) = p_g(x) = p_c(x)$ and $p(y) = p_g(y) = p_c(y)$. Taking the integral with respect to x on both sides of $p(x,y) = p_\alpha(x,y)$, we get

$$\int p(x,y)\,dx = (1-\alpha)\int p_g(x,y)\,dx + \alpha \int p_c(x,y)\,dx,$$

which indicates that $p(y) = (1 - \alpha)p(y) + \alpha p_c(y)$, i.e., $p_c(y) = p(y) = p_g(y)$. Similarly, it can be shown that $p_g(x) = p(x) = p_c(x)$ by taking the integral with respect to y.

This shows that there exists a global equilibrium point but which is not unique. Our objective is to arrive at a global equilibrium point such that we reach the desired equilibrium.

$$\begin{aligned}\min_{C,G}\max_{D}\tilde{U}(C,G,D) = E_{(x,y)\sim p(x,y)}[\log D(x,y)] \\ + \alpha E_{(x,y)\sim p_c(x,y)}[\log(1 - D(x,y))] \\ + (1-\alpha)E_{(x,y)\sim p_g(x,y)}[\log(1 - D(G(y,z),y))] + R_L + \alpha_p R_p\end{aligned} \tag{4.3}$$

According to the definition, $\tilde{U}(C,G,D) = U(C,G,D) + R_L$, where $R_L = E_p[-\log pc(y|x)]$, which can be rewritten as: $D_{KL}(p(x,y)||p_c(x,y)) + H(p(y|x))$. Namely, minimizing RL is equivalent to minimizing $D_{KL}(p(x,y)||p_c(x,y))$, which is always non-negative and zero if and only if $p(x,y) = pc(x,y)$.

$$\begin{aligned}R_L &= \iint E_{p(x,y)}[-\log pc(y|x)] \\ &= \iint E_{p(x,y)}[\log\frac{p(x,y)}{pc(x,y)}] - E_{p(x,y)}[\log p(y|x)] \\ &= \iint D(p(x,y)||pc(x,y)) + E_{p(x)}[H[p(y|x)]\end{aligned}$$

Adding any divergence (e.g. the KL divergence) between any two of the joint distributions or the conditional distributions or the marginal distributions, to $\tilde{U}$ as the additional regularization to be minimized, will not change the global equilibrium of $\tilde{U}$.

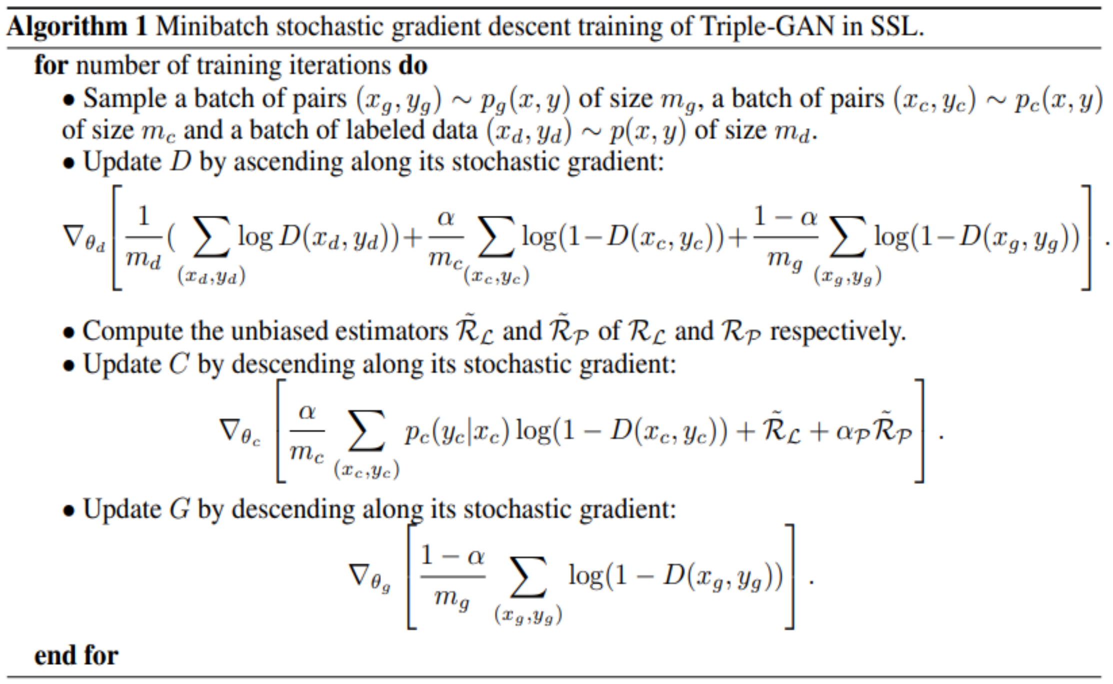
Because label information is extremely insufficient in SSL, we propose pseudo discriminative loss $R_P = E_{p_g}[-\log p_c(y|x)]$, which optimizes C on the samples generated by G in the supervised manner. Intuitively, a good G can provide meaningful labeled data beyond the training set as extra side information for C, which will boost the predictive performance. Indeed, minimizing pseudo discriminative loss with respect to C is equivalent to minimizing $D_{KL}(p_g(x,y)||p_c(x,y))$ and hence the global equilibrium remains same.

The equivalence of the pseudo discriminative loss in the main text and KL-divergence $D_{KL}(p_g(x,y)||p_c(x,y))$ is as follows:

$$
\begin{aligned}
&D_{KL}(p_g(x,y)||p_c(x,y)) + H[p_g(y|x)] - D_{KL}(p_g(x)||p(x)) \\
&= \int\int p_g(x,y)\log\frac{p_g(x,y)}{p_c(x,y)} + p_g(x,y)\log\frac{1}{pg(y|x)}\,dxdy - \int p_g(x)\log\frac{p_g(x)}{p(x)}\,dx \\
&= \iint p_g(x,y)\log\frac{p_g(x,y)}{p_c(x,y)p_g(y|x)}\,dxdy - \iint p_g(x,y)\log\frac{p_g(x)}{p(x)}\,dxdy \\
&= \iint p_g(x,y)\log\frac{p_g(x,y)p(x)}{p_c(x,y)p_g(y|x)}\,dxdy \\
&= \int\int p_g(x,y)\log\frac{p_g(x,y)p(x)}{p_c(x,y)p_g(y|x)}\,dxdy \\
&= E_{p_g}[-\log p_c(y|x)]
\end{aligned}
$$

Also note that directly minimizing $D_{KL}(p_g(x,y)||p_c(x,y))$ is infeasible since its computation involves the unknown likelihood ratio $p_g(x,y)/p_c(x,y)$. The pseudo discriminative loss is weighted by a hyperparameter α_P. See Algorithm 1 for the whole training procedure, where θ_c, θ_d and θ_g are trainable parameters in C, D and G respectively.

The conclusion essentially motivates our design of Triple-GAN, as we can ensure that both C and G will converge to the true data distribution if the model has been trained to achieve the optimum.



Algorithm 1 Minibatch stochastic gradient descent training of Triple-GAN in SSL.

```
for number of training iterations do
    • Sample a batch of pairs $(x_g, y_g) \sim p_g(x,y)$ of size $m_g$, a batch of pairs $(x_c, y_c) \sim p_c(x,y)$ of size $m_c$ and a batch of labeled data $(x_d, y_d) \sim p(x,y)$ of size $m_d$.
    • Update $D$ by ascending along its stochastic gradient:
        $\nabla_{\theta_d}\left[\frac{1}{m_d}(\sum_{(x_d,y_d)}\log D(x_d,y_d)) + \frac{\alpha}{m_c}\sum_{(x_c,y_c)}\log(1-D(x_c,y_c)) + \frac{1-\alpha}{m_g}\sum_{(x_g,y_g)}\log(1-D(x_g,y_g))\right].$
    • Compute the unbiased estimators $\tilde{\mathcal{R}}_{\mathcal{L}}$ and $\tilde{\mathcal{R}}_{\mathcal{P}}$ of $\mathcal{R}_{\mathcal{L}}$ and $\mathcal{R}_{\mathcal{P}}$ respectively.
    • Update $C$ by descending along its stochastic gradient:
        $\nabla_{\theta_c}\left[\frac{\alpha}{m_c}\sum_{(x_c,y_c)} p_c(y_c|x_c)\log(1-D(x_c,y_c)) + \tilde{\mathcal{R}}_{\mathcal{L}} + \alpha_{\mathcal{P}}\tilde{\mathcal{R}}_{\mathcal{P}}\right].$
    • Update $G$ by descending along its stochastic gradient:
        $\nabla_{\theta_g}\left[\frac{1-\alpha}{m_g}\sum_{(x_g,y_g)}\log(1-D(x_g,y_g))\right].$
end for
```

Figure 4.3: Backpropagation in Triple GAN [8]

4.3.2 Metrics

Fréchet Inception Distance (FID)

Fréchet Inception Distance (FID) is an evaluation metric used to compare the quality of generated images with real images. It was proposed by Martin Heusel, et al. in their 2017 paper "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium". The FID score measures the distance between the distributions of real images and generated images in a feature space, which is learned by a pre-trained Inception-v3 network. The FID score is calculated using the Fréchet distance, which is a measure of the similarity between two multivariate Gaussian distributions. calculate the FID score, the activations of the last pooling layer of the Inception v3 network are extracted for both the real and generated images. The mean and covariance of the activations are then calculated for both distributions, and the Fréchet distance between them is computed. A lower FID score indicates that the generated images are more similar to the real images. The FID score has become a popular metric for evaluating the quality of generated images in the GAN literature, and it has been shown to correlate well with human perception of image quality. However, it is important to note that the FID score only evaluates the similarity between the distributions of real and generated images in a feature space, and it does not provide information about other aspects of image quality, such as visual diversity or semantic correctness.

Perceptual Evaluation of Quality (PEQ)

Perceptual Evaluation of Quality (PEQ) is an evaluation metric used to measure the quality of generated images in terms of their perceptual similarity to real images. It was proposed by Richard Zhang, et al. in their 2018 paper "The Unreasonable Effectiveness of Deep Features as a Perceptual Metric". The PEQ metric is based on a pre-trained deep neural network, which is used as a feature extractor to measure the perceptual similarity between real and generated images. Specifically, the deep neural network is used to extract features from both the real and generated images, and the difference between the features is used to calculate a perceptual distance. To calculate the PEQ score, a set of reference images is chosen, and the perceptual distance between each reference image and its corresponding generated image is computed. The average perceptual distance over all reference images is then used as the final score. A lower PEQ score indicates that the generated images are more similar to the real images. The PEQ metric has been shown to correlate well with human perception of image quality, and it has become a popular metric for evaluating the quality of generated images in the GAN literature. However, it is important to note that the PEQ metric only evaluates the perceptual similarity between real and generated images, and it does not provide information about other aspects of image quality, such as visual diversity or semantic correctness.

Chapter 5

Experimental Setup & Practical Techniques

5.1 Practical Techniques: SSL and Regularization in Triple-GAN

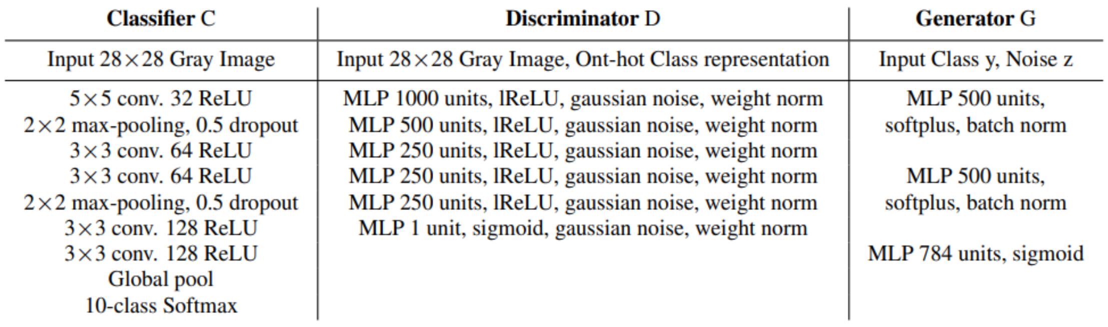
One crucial problem of SSL is the small size of the labeled data. In Triple-GAN, D may memorize the empirical distribution of the labeled data, and reject other types of samples from the true data distribution. Consequently, G may collapse to these modes. To this end, we generate pseudo labels through C for some unlabeled data and use these pairs as positive samples of D. The cost is on introducing some bias to the target distribution of D, which is a mixture of p_c and p instead of the pure p. However, this is acceptable as C converges quickly and p_c and p are close.

Since properly leveraging the unlabeled data is key to success in SSL, it is necessary to regularize C heuristically as in many existing methods to make more accurate predictions. We consider two alternative losses on the unlabeled data. The confidence loss minimizes the conditional entropy of $p_c(y|x)$ and the cross entropy between $p(y)$ and $p_c(y)$, weighted by a hyperparameter α_B, as $R_U = H_{p_c}(y|x) + \alpha_B E_{p(y)}[-\log p_c(y)]$, which encourages C to make predictions confidently and be balanced on the unlabeled data. The consistency loss penalizes the network if it predicts the same unlabeled data inconsistently given different noise ϵ, e.g., dropout masks, as $R_U = E_{x\sim p(x)}||p_c(y|x,\epsilon) - p_c(y|x,\epsilon_0)||^2$, where $||\cdot||^2$ is the square of the l2-norm.

Another consideration is to compute the gradients of $E_{x\sim p(x),y\sim p_c(y|x)}[\log(1-D(x,y))]$ with respect to the parameters θ_c in C, which involves summation over the discrete random variable y, i.e. the class label. On one hand, integrating out the class label is time consuming. On the other hand, directly sampling one label to approximate the expectation via the Monte Carlo method makes the feedback of the discriminator not differentiable with respect to θ_c. As the REINFORCE algorithm can deal with such cases with discrete variables, we use a variant of it for the end-to-end training of our classifier. The gradients in the original REINFORCE algorithm should be $E_{x \sim p(x), y \sim p_c(y|x)}[\nabla_{\theta_c} \log p_c(y|x) \log(1 - D(x, y))]$.

5.2 Experimental Setup

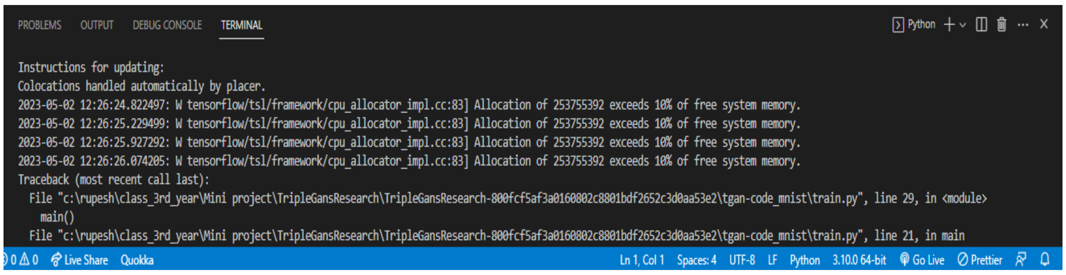
We evaluate Triple-GAN on the widely adopted MNIST [66]. MNIST consists of 50,000 training samples, 10,000 validation samples and 10,000 testing samples of handwritten digits of size 28 $\times$ 28.



Classifier C	**Discriminator D**	**Generator G**
Input 28×28 Gray Image	Input 28×28 Gray Image, Ont-hot Class representation	Input Class y, Noise z
5×5 conv. 32 ReLU	MLP 1000 units, lReLU, gaussian noise, weight norm	MLP 500 units, softplus, batch norm
2×2 max-pooling, 0.5 dropout	MLP 500 units, lReLU, gaussian noise, weight norm	
3×3 conv. 64 ReLU	MLP 250 units, lReLU, gaussian noise, weight norm	
3×3 conv. 64 ReLU	MLP 250 units, lReLU, gaussian noise, weight norm	MLP 500 units, softplus, batch norm
2×2 max-pooling, 0.5 dropout	MLP 250 units, lReLU, gaussian noise, weight norm	
3×3 conv. 128 ReLU	MLP 1 unit, sigmoid, gaussian noise, weight norm	
3×3 conv. 128 ReLU		MLP 784 units, sigmoid
Global pool		
10-class Softmax		

Figure 5.1: Triple GAN Model architecture

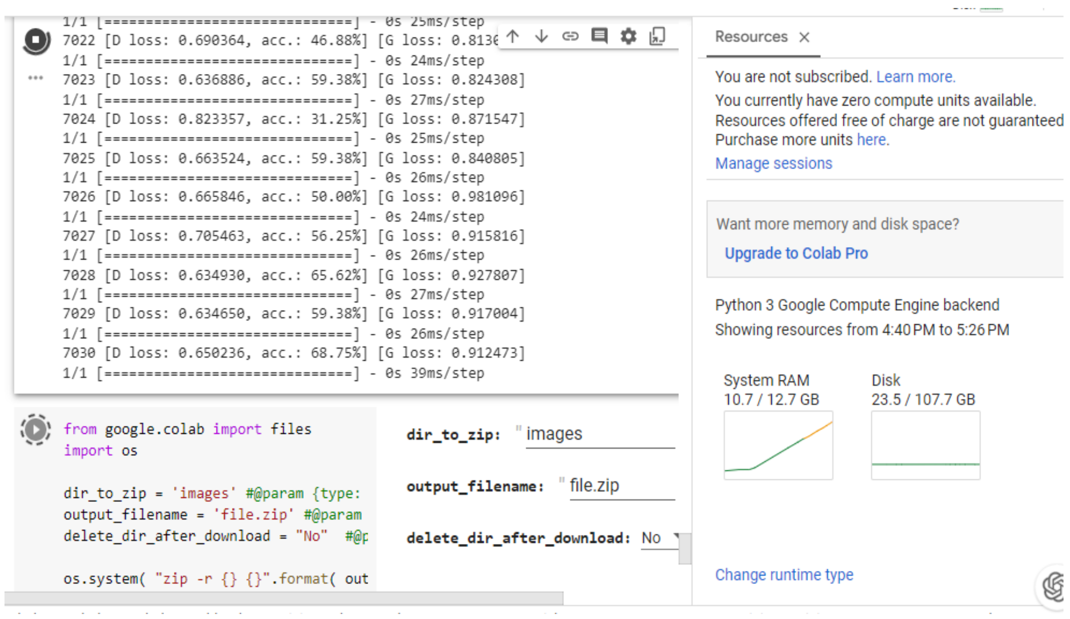
We evaluate our method with 20, 50 and 200 labeled samples on MNIST for a systematical comparison with our main baseline Improved-GAN. Triple-GAN consistently outperforms Improved-GAN with a substantial margin, which again demonstrates the benefit of Triple-GAN. Besides, we can see that Triple-GAN achieves more significant improvement as the number of labeled data decreases, suggesting the effectiveness of the pseudo discriminative loss.



```
Instructions for updating:
Colocations handled automatically by placer.
2023-05-02 12:26:24.822497: W tensorflow/tsl/framework/cpu_allocator_impl.cc:83] Allocation of 253755392 exceeds 10% of free system memory.
2023-05-02 12:26:25.229499: W tensorflow/tsl/framework/cpu_allocator_impl.cc:83] Allocation of 253755392 exceeds 10% of free system memory.
2023-05-02 12:26:25.927292: W tensorflow/tsl/framework/cpu_allocator_impl.cc:83] Allocation of 253755392 exceeds 10% of free system memory.
2023-05-02 12:26:26.074205: W tensorflow/tsl/framework/cpu_allocator_impl.cc:83] Allocation of 253755392 exceeds 10% of free system memory.
Traceback (most recent call last):
  File "c:\rupesh\class_3rd_year\Mini project\TripleGansResearch\TripleGansResearch-800fcf5af3a0160802c8801bdf2652c3d0aa53e2\tgan-code_mnist\train.py", line 29, in <module>
    main()
  File "c:\rupesh\class_3rd_year\Mini project\TripleGansResearch\TripleGansResearch-800fcf5af3a0160802c8801bdf2652c3d0aa53e2\tgan-code_mnist\train.py", line 21, in main
```

Figure 5.2: Local Machine Incompatibility

Practical Implementations issues like incompatible hardware, computational power etc were faced during the initial stages of execution of the official repository[17] [18], Thus we preferred to code the Triple GAN toy model in Keras & TensorFlow.



```
1/1 [==============================] - 0s 25ms/step
7022 [D loss: 0.690364, acc.: 46.88%] [G loss: 0.8136
1/1 [==============================] - 0s 24ms/step
7023 [D loss: 0.636886, acc.: 59.38%] [G loss: 0.824308]
1/1 [==============================] - 0s 27ms/step
7024 [D loss: 0.823357, acc.: 31.25%] [G loss: 0.871547]
1/1 [==============================] - 0s 25ms/step
7025 [D loss: 0.663524, acc.: 59.38%] [G loss: 0.840805]
1/1 [==============================] - 0s 26ms/step
7026 [D loss: 0.665846, acc.: 50.00%] [G loss: 0.981096]
1/1 [==============================] - 0s 24ms/step
7027 [D loss: 0.705463, acc.: 56.25%] [G loss: 0.915816]
1/1 [==============================] - 0s 26ms/step
7028 [D loss: 0.634930, acc.: 65.62%] [G loss: 0.927807]
1/1 [==============================] - 0s 27ms/step
7029 [D loss: 0.634650, acc.: 59.38%] [G loss: 0.917004]
1/1 [==============================] - 0s 26ms/step
7030 [D loss: 0.650236, acc.: 68.75%] [G loss: 0.912473]
1/1 [==============================] - 0s 39ms/step
```

```python
from google.colab import files
import os

dir_to_zip = 'images' #@param {type:
output_filename = 'file.zip' #@param
delete_dir_after_download = "No"  #@p

os.system( "zip -r {} {}".format( out
```

dir_to_zip: images

output_filename: file.zip

delete_dir_after_download: No

Resources

You are not subscribed. Learn more.
You currently have zero compute units available.
Resources offered free of charge are not guaranteed
Purchase more units here.
Manage sessions

Want more memory and disk space?
Upgrade to Colab Pro

Python 3 Google Compute Engine backend
Showing resources from 4:40 PM to 5:26 PM

System RAM
10.7 / 12.7 GB

Disk
23.5 / 107.7 GB

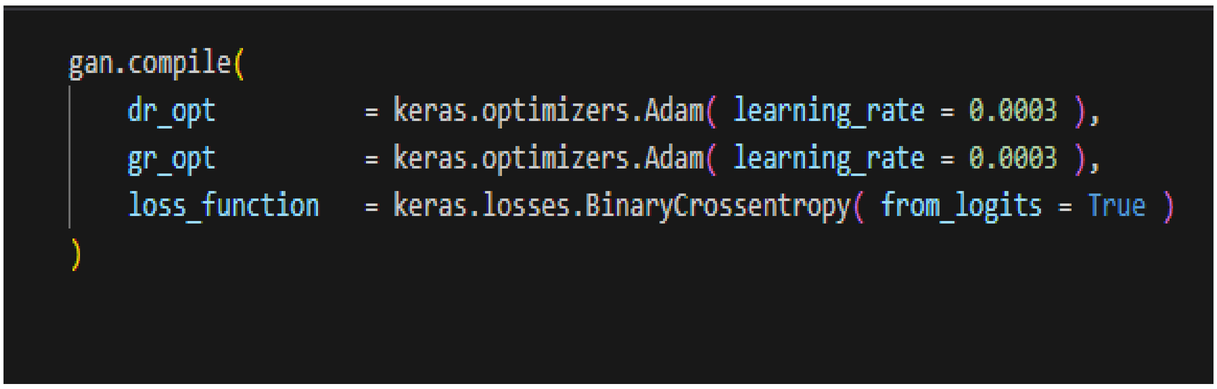
Change runtime type

Figure 5.3: Using Google Colab

Chapter 6

Analysis and Results

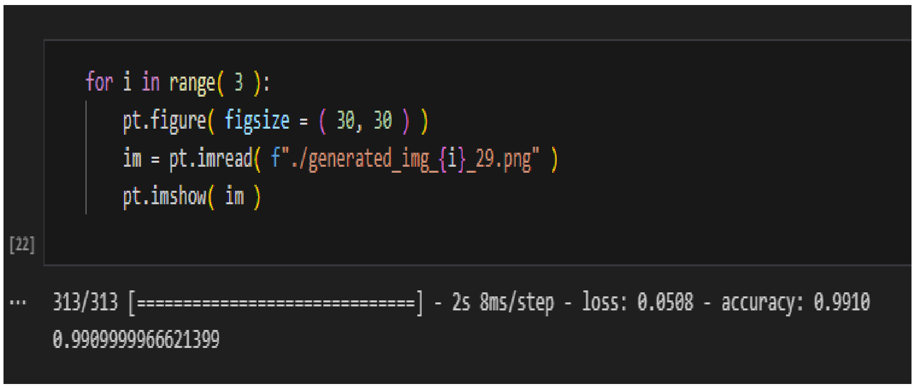
With carefully designed objective functions, Triple-GAN leads to a unique desirable equilibrium under a nonparametric assumption [5]. Using a commonly adopted 13-layer CNN classifier, Triple-GAN outperforms extensive semi-supervised learning methods. It is also observed that given partially labeled data, Triple-GAN is able to disentangle category from style features and get samples of comparable quality to the strong baseline with full labels. And unlike the two player formulation where the generator and discriminator compete against each other, here the classifier and generator distributions converge and helpp each other to mimic the real data distribution.



```
gan.compile(
    dr_opt          = keras.optimizers.Adam( learning_rate = 0.0003 ),
    gr_opt          = keras.optimizers.Adam( learning_rate = 0.0003 ),
    loss_function   = keras.losses.BinaryCrossentropy( from_logits = True )
)
```

Figure 6.1: Model Optimizers

We observed that the training techniques for the original two-player GANs [5]are sufficient to stabilize the optimization of Triple-GAN. The pseudo discriminative loss is not applied until the number of epochs reaches a threshold that the generator could generate meaningful data. We search the threshold in 200, 300, P in 0.1, 0.03 and the global learning rate in 0.0003, 0.001 based on the validation performance on each dataset. All of the other hyperparameters including relative weights and parameters in Adam [16] are fixed according to [20] across all of the experiment.



```
for i in range( 3 ):
    pt.figure( figsize = ( 30, 30 ) )
    im = pt.imread( f"./generated_img_{i}_29.png" )
    pt.imshow( im )
[22]
...  313/313 [==============================] - 2s 8ms/step - loss: 0.0508 - accuracy: 0.9910
     0.9909999966621399
```

Figure 6.2: Evaluation

You can Observe the results generated by the model and the evaluation results from the Figure 6.3 , Figure 6.4 and Figure 6.2 respectively. The Figure 6.3 shows the initial Gaussian noise at epoch 0 and the Figure 6.4 shows the generated data at epoch 8500(approximately).

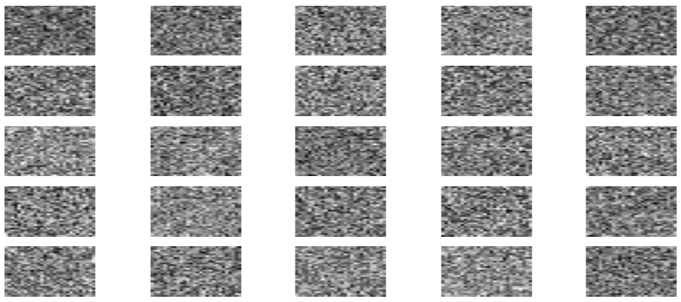


Figure 6.3: Epoch 0

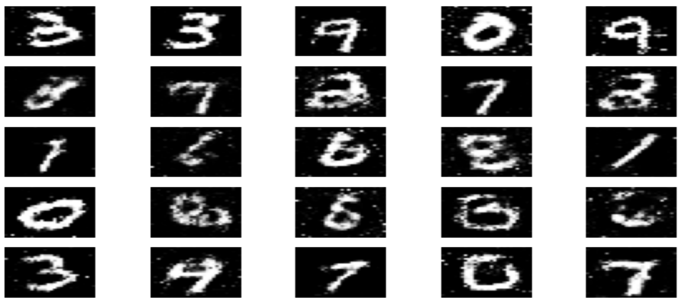


Figure 6.4: Epoch 8490

Chapter 7

Future Scope

The synthetic data generation of images can be further expanded with including various datasets and domain data into it, an example is in the field of microbiology studies to enhance the images produced at nano-level and further lower, incorporating live data to the network and update the current pix2pix GAN, building VAE's on top of existing GAN's etc.

One potential area of research is to investigate the performance of TRIPLE GAN on more complex image datasets, such as natural images. And to explore variations on the TRIPLE GAN architecture, such as adding more sub-generators and sub-discriminators or modifying the training process to incorporate feedback loops or other techniques.

Ensuring diversity and realism in generated samples is an ongoing research focus. Issues like training stability, mode collapse, and addressing biases in generated data are important considerations. Ethical concerns regarding the potential misuse of synthetic data and privacy implications must be taken into account.

Chapter 8

Conclusion

Semi- supervised approach for synthetic data generation to prepare high-quality images for requirements of various Machine Learning algorithms and tasks. The cost incurred in producing training data for such applications is generally high and we have very limited amount of labelled data available for training, testing and validation, and it is also not suggestable to manually get the labelled data for such huge tasks, as generative networks, unsupervised Models, SSL models require large amounts of data for training to understand the hidden latent space configurations.

Deep generative models are powerful tools for synthetic data/image generation. They leverage deep neural networks and techniques like GANs to create new data samples. These models find applications in computer vision and creative fields. In Triple-GAN instead of competing as stated in the two player formulation in GAN, a good classifier will result in a good generator and vice versa, and also the discriminator can access the label information of the unlabeled data from the classifier and then force the generator to generate correct image-label pairs thus it can disentangle the classes and styles of the input and transfer smoothly in the data space via interpolation in the latent space class-conditionally.Deep generative models and synthetic data generation have practical applications in computer vision and creative domains. In computer vision, they can augment training data, enhancing model performance. In creative fields, they can generate artwork, virtual environments, and realistic images for games and movies.

References

1. Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014

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

3. Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semisupervised learning with deep generative models. In NIPS, 2014

4. Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. arXiv preprint arXiv:1610.02242, 2016.

5. Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In NIPS, 2014.

6. Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen.Improved techniques for training GANs. In NIPS, 2016.

7. Mohamed, Shakir; Lakshminarayanan, Balaji (2016). "Learning in Implicit Generative Models". arXiv:1610.03483

8. Triple Generative Adversarial Nets Chongxuan Li, Kun Xu, Jun Zhu , Bo Zhang Dept. of Comp. Sci. Tech., TNList Lab, State Key Lab of Intell. Tech. Sys.,Center for Bio-Inspired Computing Research, Tsinghua University, Beijing, 100084, China licx14, xu-k16@mails.tsinghua.edu.cn, dcszj, dcszb@mail.tsinghua.edu.cn nips2017.

9. V. Dumoulin, I. Belghazi, B. Poole, A. Lamb, M. Arjovsky, O. Mas tropietro, and A. Courville, “Adversarially learned inference,” arXiv preprint arXiv:1606.00704,

2016

10. D. Jeff and K. Simonyan, “Large scale adversarial representation learning,” arXiv preprint arXiv:1907.02544, 2019.

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, 2017, pp. 5907–5915.

12. J. T. Springenberg, “Unsupervised and semi-supervised learning with categorical generative adversarial networks,” arXiv preprint arXiv:1511.06390, 2015

13. X. Zhang, Z. Wang, D. Liu, and Q. Ling, “Dada: Deep adversarial data augmentation for extremely low data regime classification,” arXiv preprint arXiv:1809.00981, 2018.

14. T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, “Spectral normalization for generative adversarial networks,” arXiv preprint arXiv:1802.05957, 2018

15. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen, and T. Aila, “Training generative adversarial networks with limited data,” arXiv preprint arXiv:2006.06676, 2020.

16. D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” arXiv preprint arXiv:1412.6980, 2014

17. Z. Zhao, S. Singh, H. Lee, Z. Zhang, A. Odena, and H. Zhang, “Improved consistency regularization for gans,” arXiv preprint arXiv:2002.04724, 2020.

18. Theano Development Team, “Theano: A Python framework for fast computation of mathematical expressions,” arXiv e-prints, vol. abs/1605.02688, May 2016. [Online]. Available: http://arxiv.org/abs/1605.02688

19. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga et al., “Pytorch: An imperative style, high-performance deep learning library,” in Advances in neural information processing systems, 2019, pp. 8026– 8037.

20. T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training GANs," in NIPS, 2016.