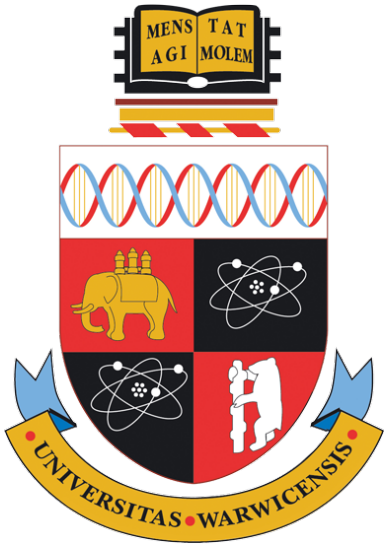
WARWICK
THE UNIVERSITY OF WARWICK



Optimizing GAN for Generating High Quality Samples

by

Hao Wu

A thesis submitted in partial fulfilment of the requirements for the

degree of

Doctor of Philosophy in Computer Science

University of Warwick, Department of Computer Science

January 2022

Contents

List of Tables

List of Figures

Acknowledgments

First, I would like to express my sincere gratitude to my supervisor Dr. Ligang He, whose guidance, encouragement and support have been invaluable to me during my time at the Department of Computer Science at the University of Warwick. I benefited greatly from his insightful suggestions and comments in solving research problems and my life issues.

I would like to thank Yuelu Pan whose patience and encouragement has been the significant ingredient in my life. I can not get through the most difficult time without her support. I thank my parents for their unreserved support for my living in aboard.

I would like to thank all my friends, especially Yuze Yang, Yitian Wu, Qianwen Qing, Jiayao Ding, Oz, Ruiguang Xu, Liang Huang and XiaoXiao.

I thank the staff in our department, particularly Roger Packwood, Ruth Cooper, Sharon Howard and Maria Ferreiro, for their help and warm supports.

In the end, I would like to give my special thanks to my lab-mates, particularly Wentai Wu, Junyu Li, Zhiyan Chen, Shenyuan Ren, Yijun Quan, Chao Chen, Zhuoer Gu, Peng Jiang, Shenyuan Ren, Qingzhi Ma, Haoyi Wang and Yujue Zhou.

Declarations

This thesis is submitted to the University of Warwick in support of the author's application for the degree of Doctor. It has been composed by the author and has not been submitted in any previous application for any degree. The work presented was carried out by the author except where acknowledged.

Publications

Parts of this thesis have been previously accepted by the author in the following:

- Hao Wu, Ligang He, Chang-Tsun Li, Junyu Li, Wentai Wu, Carsten Maple. MGGAN: Improving Sample Generations of Generative Adversarial Networks. In 2021 IEEE 23rd International Conference on High Performance Computing and Communications.

In addition, the following works are in progress of being submitted:

- Hao Wu, Ligang He and Chang-Tsun Li, Block Paralleling GAN: Decreasing the Time Cost via a novel Model Parallelism Method based on Distributed Computing, submitted to IEEE Transactions on Parallel and Distributed Systems.

- Hao Wu, Ligang He, Chang-Tsun Li, Privacy GAN: A Teacher-Student Framework based on GAN to Simulate Limited Data in Clients, submitted to IEEE Transactions on Image Processing.

Abstract

Generating high-quality and various image samples is a significant research goal in computer vision and artificial intelligence. The Generative Adversarial Networks (GAN) and Variational Autoencoder (VAE) are widely used to capture the distributions of actual distributions samples. In this thesis, we pay attention to the GANs, a prominent unsupervised learning method that can automatically capture the patterns in the training data. However, the training of GANs has simple memory imitation and non-convergence issues. The memory imitation issue means generators produce same samples lazily to fool discriminators. To generate various image samples and improve the GANs' performance efficiently, we develop the GAN structure from the following three aspects: 1) The training procedure is not stable enough, which incurs the mode collapse issues. The mode collapse means the GAN will generate samples with single diversity; 2) The training process requires enormous time to capture the pattern from the training data. The complexity of GAN structure and the amount of training data influence the total training expense; 3) GAN demands enough training data to ensure the accuracy and stability of the model. Lack of comprehensive training data usually causes deterioration of the performance of the network. Thus, we investigate training techniques and propose the framework to develop the GANs' performance and ability. First, we present the Multi-group GAN (MGGAN), a light framework to solve the mode collapse while increasing the diversity of generated samples. Next, we present the Block Paralleling GAN (BPGAN) to decrease the total training time. It uses a novel model parallelism to reduce the transmission cost. We provide theoretical analysis to prove the benefit of our method. Finally, we present Privacy-aware GAN (PrivacyGAN), a teacher-student framework based on a generative adversarial network, to generate similar sensitive personal data

from private clients. The experimental results and theoretical analysis demonstrate that the techniques proposed in this thesis is effective.

Acronyms

Adam A Method for Stochastic Optimization.

BigGAN Large Scale GAN Training for High Fidelity Natural Image Synthesis.

BN Batch Normalization.

cGAN contitional GAN.

CoGAN Coupled Generative Adversarial Networks.

CycleGAN Cycle-Consistent Adversarial Network.

DA Data Augmentation.

DCGAN Unsupervised representation learning with deep convolutional GAN.

DD Decentralized Deployment.

DPGAN distribution propagation graph network.

EM Earth Move.

FCGAN High-Quality Face Image SR Using Conditional Generative Adversarial Networks.

FID Frechet Inception Distance.

FitNets Hints for Thin Deep Nets.

FLOPs Floating-point Operations per Second.

FSL Few-Shoot Learning.

GAN Generative Adversarial Network.

GDPP Learning Diverse Generations Using Determinantal Point Process.

GSN Generative Stochastic Network.

ICA Independent Components Analysis.

IM Inception Module.

InfoGAN Information Maximizing GANs.

INGAN Capturing and Remapping the "DNA" of a Natural Image.

IS Inception Score.

JRLF Joint Restraint Learning Function.

KD knowledge Distillation.

KL Kullback Leibler.

LSUN Large-scale Scene UNderstanding Challenge.

MAD-GAN Multivariate Anomaly Detection for Time Series Data with Generative Adversarial Networks.

MADE Masked Auto-encoder for Distribution Estimation.

MobileNet Efficient Convolutional Neural Networks for Mobile Vision Applications.

MS Mode Score.

MSE Mean Squared Error.

NADE Neural Auto-regressive Distribution Estimation.

NIN Network in Network.

OBD Optimal Brain Damage.

OBSP Optimal Brain Surgeon Paper.

PSNR Peak Signal-to-noise Ratio.

RMSE Root Mean Square Error.

RMSProp Root Mean Squared Propagation.

SA Simply Accepting.

SAGAN Self-Attention GAN.

SGD Stochastic Gradient Descent.

SinGAN Learning a Generative Model from a Single Natural Image.

SRGAN Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network.

SSIM Structural Similarity.

stackGAN Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks.

StyleGAN A Style-Based Generator Architecture for GAN.

TPU Tensor Processing Unit.

TTUR Two Time-Scale Update Rules.

UnrolledGAN Unrolled Generative Adversarial Networks.

VAE Variational Autoencoder.

VEEGAN Reducing Mode Collapse in GANs using Implicit Variational Learning.

VGG Very Deep Convolutional Networks for Large-Scale Image Recognition.

VRL Variation Ratio of the Loss.

WGAN Wasserstein GAN.

WGAN-GP Improved Training of Wasserstein GANs.

Chapter 1

Introduction

This chapter presents the motivation, the introduction to Generative Adversarial Networks (GANs), the research contributions and the thesis organisation. We start with the motivation for our research ideas, the significance, and the process of our research. We then review the principle behind the GANs, the variations and their applications in detail. Next, we introduce the evaluation metrics of GANs. Finally, we present our research contributions and the thesis organisation to explain our ideas.

1.1 Motivation and Objectives

With the development of artificial intelligence and deep learning, data has been recognised as an essential resource. People have increasingly higher requirements for high quality data in order to obtain a better trained network. The work in [11][14][59][73][88][105][150] emphasises the significant role of data in deep learning. This thesis mainly focuses on GANs, a unique unsupervised learning approach to generate target samples. GANs have been successfully applied in many aspects such as image-to-image translation [111], image super-resolution [228], image manipulation [46], data augmentation [136][178][209] and text-to-image [47][157][218]. However, GANs still suffer from the issues like mode collapse and instability. Therefore, our research goal is to optimise the GANs training from different aspects to improve the generated samples' quality and reduce training cost and complexity. Moreover, we also design the frameworks to extend the application scenarios of GANs.

Our first idea is to design a framework with better generality to solve the mode collapse and improve the diversity of the generated samples. The "generality" means that the framework can be applied to GAN variations while keeping the training stable. However, the experimental results demonstrate that our framework, unfortunately, increases the time expenses. Because the training of GANs requires enormous computational resources and time, our second idea

is to reduce the training time by designing a novel model parallelism method based on modifying the training workflow of the GAN model. In model parallelism, all the training data is stored in the server. As artificial intelligence becomes very popular nowadays, the data are collected everywhere by such as mobile devices, individual organizations. The data can be used to train the AI models. However, it is not feasible to upload these distributed data to a central repository for model training, partly because it will consume enormous communication bandwidth and storage facilities and partly because it may violate the data privacy. Therefore, it is a big challenge to develop an efficient and effective method to train the GAN models over distributed training data. This is the motivation of the third and final work presented in this thesis. It develops a novel GAN framework based on the teacher-student mechanism to synthesize the clients' private data. The proposed method is able to generate the targeted private information using limited training data.

1.2 Generative Adversarial Networks

1.2.1 The Concepts of GANs

GANs [54] have become a popular research topic nowadays. There are an enormous number of papers related to GANs, according to Google scholar. GANs belong to generative algorithms, which means that the algorithm is based on a probabilistic model of the training data. Generative algorithms can be classified into explicit density models and implicit density models. Fig.1.1 shows the taxonomy of the generative models.

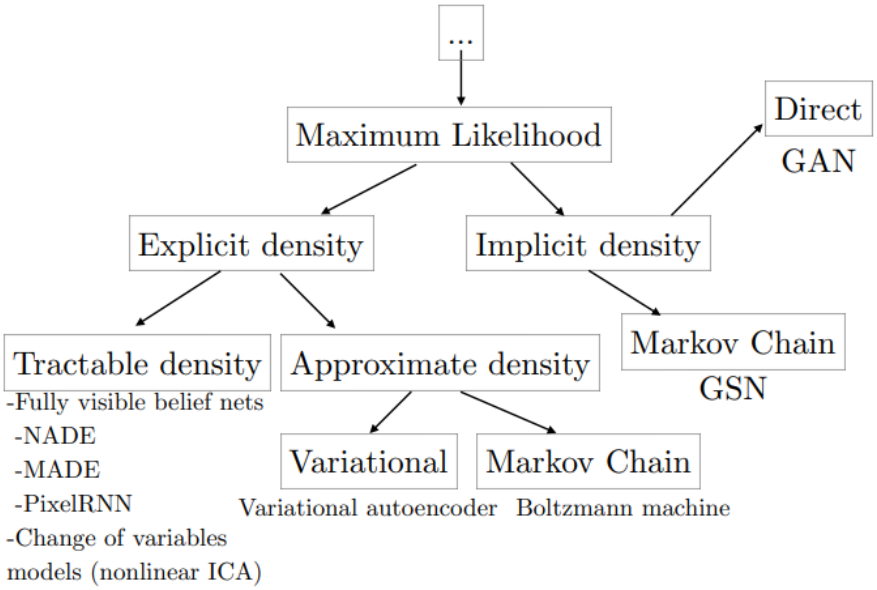


Figure 1.1: The Taxonomy of Generative Models

Explicit Density Model

The explicit density models simulate the distribution from the real datasets to train the model to fit the training data. The explicit density models include Neural Auto-regressive Distribution Estimation (NADE) [185], Masked Auto-encoder for Distribution Estimation (MADE) [51], PixelRNN [187], nonlinear Independent Components Analysis (ICA) [34], Variational Autoencoder (VAE) [103] and Markov Chain [68][2][69].

Implicit Density Model

The implicit density models produce data instances from the distribution instead of directly fitting the data distribution. The implicit density models include Generative Stochastic Network (GSN) [3], Markov Chain and GANs. These generative algorithm has their own disadvantages: PixelRNN has to generate the pixels one by one, which can not be parallelized; VAE can only generate samples with limited quality because it does not have a proper likelihood function; NADE and MADE both have to depend on the previous output, which requires enormous training time. However, GAN can overcome the disadvantages of other generative algorithms. Moreover, GAN is an exceptionally designed deep learning network to produce better samples. Therefore, in this thesis, we mainly introduce the related knowledge of GANs.

1.2.2 Original GANs

GAN aims to model high-dimensional distributions from real data. The original GAN framework defines a game of two players: a G (Generator) and a D (Discriminator).

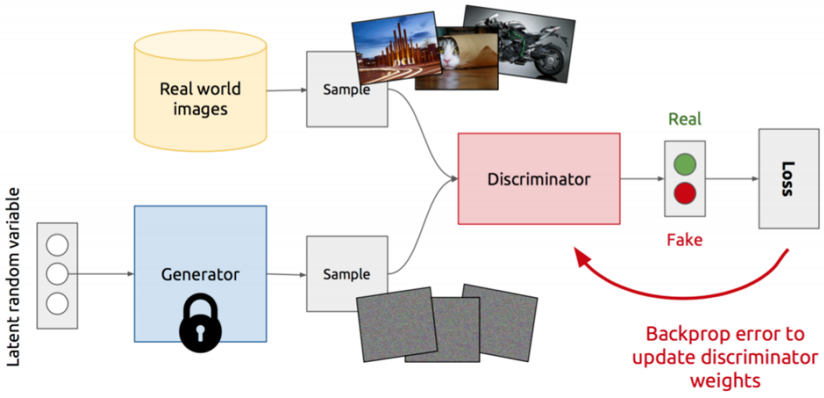


Figure 1.2: The training of discriminator.

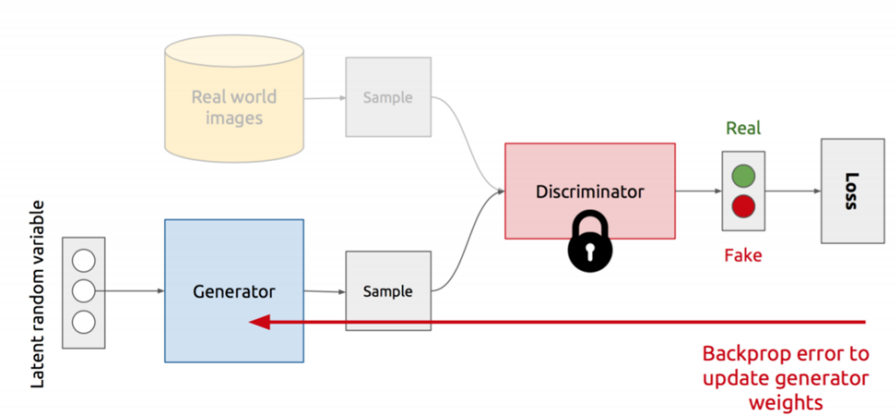


Figure 1.3: The training of generator (The faded diagram means the training of generator does not need real world images).

Fig.1.2 and Fig.1.3 show the training schema of the original GAN framework. It consists of a generator, which takes the noise vector as input and exports the generated samples, and a discriminator, which distinguishes the samples from natural images. The generator and discriminator are trained adversarially in a minimax game. A generator tries to map a vector z from the noise space Z to the realistic data space, generating a sample $G(z)$ to simulate the real data and fooling the discriminator. The discriminator is trained to maximise the probability of the labels correctly given to the real data and the fake samples generated by the generator. The generator is trained simultaneously to minimise $log(1-D(G(z)))$. $p_{data}(x)$ means the distribution of training data and $p_g(x)$ is the distribution of generated samples. In other words, generator and discriminator play the following minimax game with the objective function $V(G,D)$:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))] \tag{1.1}$$

When the generator is fixed, the optimal discriminator is given by [54]:

$$D^*_G(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)} \tag{1.2}$$

The minimax game in Eq.1.1 can now be reformulated as:

$$\begin{aligned} V(G) &= \max_D V(G,D) \\ &= \mathbb{E}_{x\sim p_{\text{data}}\ (x)}\left[\log D_G^*(x)\right] + \mathbb{E}_{z\sim p_z(x)}\left[\log\left(1-D_G^*(G(z))\right)\right] \\ &= \mathbb{E}_{x\sim p_{\text{data}}\ (x)}\left[\log D_G^*(x)\right] + \mathbb{E}_{x\sim p_g(x)}\left[\log\left(1-D_G^*(x)\right)\right] \\ &= \mathbb{E}_{x\sim p_{\text{data}}(x)}\left[\log \frac{p_{\text{data}}\ (x)}{p_{\text{data}}\ (x)+p_g(x)}\right] + \mathbb{E}_{x\sim p_g(x)}\left[\log \frac{p_g(x)}{p_{\text{data}}\ (x)+p_g(x)}\right] \end{aligned} \tag{1.3}$$

The Kullback-Leibler (KL) divergence and Jensen-Shanon (JS) divergence between two model's distribution $p(x)$ and $q(x)$ is defined as:

$$\begin{aligned} KL(p\|q) &= \int p(x)\log\frac{p(x)}{q(x)}dx \\ JS(p\|q) &= \frac{1}{2}KL\left(p(x)\|\frac{p(x)+q(x)}{2}\right) + \frac{1}{2}KL\left(q(x)\|\frac{p(x)+q(x)}{2}\right) \end{aligned} \tag{1.4}$$

Combining Eq.1.3 and Eq.1.4, we obtain:

$$\begin{aligned} V(G) &= KL\left(p_{\text{data}}\ (x)\|\frac{p_{\text{data}}\ (x)+p_g(x)}{2}\right) + KL\left(p_g(x)\|\frac{p_{\text{data}}\ (x)+p_g(x)}{2}\right) \\ &= 2JS\left(p_{\text{data}\ (x)}\|p_g(x)\right) - 2\log 2 \end{aligned} \tag{1.5}$$

Since the Jensen-Shanno divergence between two distributions is always positive or zero when they are equal, V^* =-2log2 is the global minimum of $V(G)$, and the only solution is $p_g(x) = p_{data}(x)$. Thus, the objective function of GANs is related to both KL divergence and JS divergence. G is generally poor at the early stage of the training, and the samples differ significantly from the training data. As a result, D will confidently reject the generated samples with high confidence. Therefore, we train G to maximize $log(D(G(z)))$ instead of minimizing $log(1-D(G(z)))$. It has been theoretically shown that the minimax game will reach a Nash Equilibrium. At the Nash Equilibrium, the distribution of the generated samples is similar to the real data distributions, $p_{data}(x) = p_g(x)$. Furthermore, the discriminators get $D(x) = 0.5$ for all x because they can not determine whether the data is real or fake.

A significant advantage of GAN is that the network can be flexibly applied to different tasks and datasets, which enables GAN variants to solve the problems in other challenging fields such as image generation and style transferring.

1.2.3 GAN Varitations

Many papers related to GANs have been proposed for generating the images of higher resolution, higher diversity, or subject to conditional constraints. In

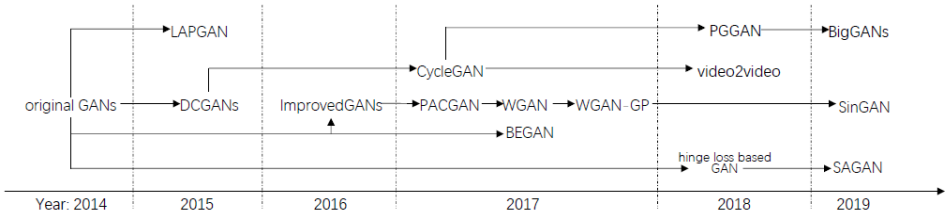


Figure 1.4: The road map of GANs from [56].

this subsection, we will introduce several representative variants of GANs that have been shown to have high performance. The Fig.1.4 shows the brief history of GANs from 2014.

DCGAN Unsupervised representation learning with deep convolutional GAN (DCGAN) [155] is the first GAN architecture in which both the generator and discriminator are designed by deep convolution neural networks. The generator is based on transpose convolution structures, while the discriminator is based on regular convolution structures. Fig.1.5 shows the structure of the generator applied in DCGAN.

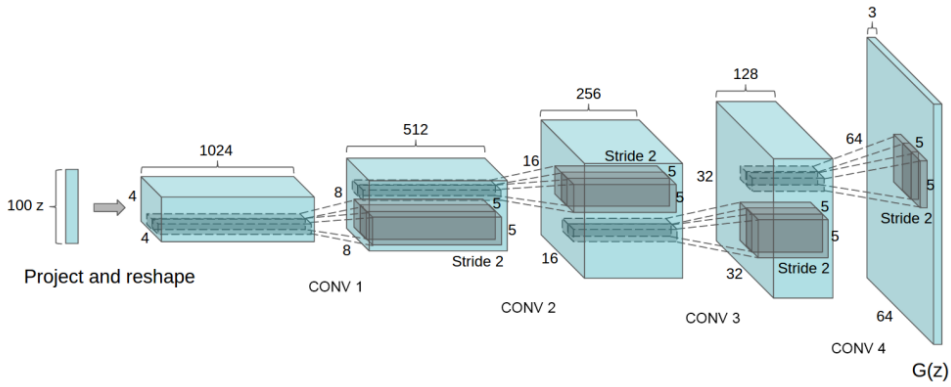


Figure 1.5: The structure of the generator used in DCGAN from [155].

In DCGAN, the authors suggest the following several essential tricks and constraints:

1.Avoid the pooling layers and replace them with strided-convolutions. Use the convolutional layer with stride of 2 in the discriminator and the transposed convolutional (fractional-strided convolutions) layer with zero-padding in the generator to export better gradients.

2. Use the Batch Normalization (BN) [91] in both generator and discriminator. The last layer of G and the first layer of D are not batch normalized.

3. Remove the hidden fully-connected layers for deeper networks.

4. Use the ReLU activation in the generator and the Leaky-ReLU activation in the discriminator.

5. Use the Adam [101] optimizer instead of SGD with momentum.

These techniques and tricks efficiently stabilize the training process of the GANs, which benefit the generation of high-quality images. Moreover, DCGAN explores the question of latent space: how does the generator map a linear combination of two vectors from the latent space if each of them can be mapped to a different image?

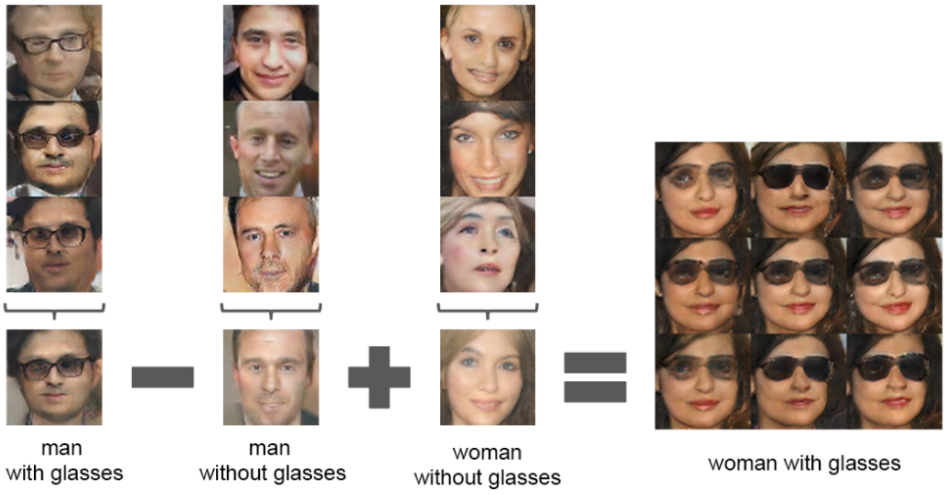


Figure 1.6: Vector arithmetic for visual concepts. For each column, the Z vectors (Z means the noise space) of samples are averaged. Arithmetic was then performed on the mean vectors to create a new vector Y (Y means another new noise space). The centre sample on the right hand side is produced by feeding Y as the input to the generator.

Fig.1.6 from the original DCGAN paper demonstrates the research on applying arithmetic in the input space. The different underlying features, such as male/female and with/without glasses, can be automatically learned by DCGAN from the distinct direction of the vector in the latent space.

Wasserstein GAN The Wasserstein GAN (WGAN) [5] is proposed to solve the issues of converge and mode collapse during the training procedure of GAN from the perspective of cost functions. It applies the Wasserstein Distance to measure the distance between two probability distributions. The Wasserstein Distance, also called Earth Mover's Distance (EM distance), can be explained as the minimum energy cost of moving the dirt in the shape of one distribution to the form of another distribution. The definition of the EM distance is:

$$W(p_r, p_g) = \inf_{\gamma \sim \Pi(p_r, p_g)} \mathbb{E}_{(x,y)\sim\gamma}[\|x-y\|] \tag{1.6}$$

The $\Pi(p_{data}, p_g)$ is the set of all possible joint probability distribution between p_{data} and p_g. $(x,y) \sim \gamma$ means the percentage of dirt that should be moved from point x to y. The EM distance is calculated by the following

expression, as defined in [5]:

$$\max_{w \in \mathscr{W}} E_{x \sim p_{\text{data}}(x)}[f_w(x)] - E_{z \sim p_z(z)}[f_w(G(z))] \tag{1.7}$$

$\{f_w\}_{w \in \mathscr{W}}$ is a Lipschiz function determined by the discriminator D. When D is fixed, Eq.1.7 calculates the EM distance. Then G aims to minimize Eq.1.7 to optimise the generated samples as close to the real data as possible. By comparing Eq.1.1 and Eq.1.7, we find the following differences between the original GAN and WGAN:

1) The D applied in WGAN approximates the EM distance, which is a regression task while the label from the original GAN is binary. Therefore, the D from WGAN does not perform the sigmoid function in the last layer.

2) The D in WGAN has to be a K-Lipschitz, which means the weights of the discriminator in WGAN must be within a specific range. Therefore, WGAN applies a simple clipping to restrict the weight. The network design of WGAN is similar to that of the original GAN, except that WGAN does not have a sigmoid output function.

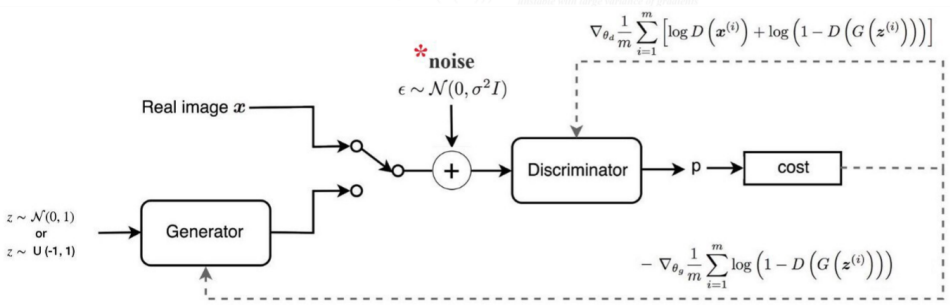


Figure 1.7: The WGAN added noise to generated images to imporve the stability. The p between discriminator and cost is the penalty.

3) Empirically, the authors suggested the RMSprop optimizer instead of the momentum optimizer such as Adam. Wasserstein provides a smooth measure method even when the training data distribution and generated sample distribution are far from each other. This distance stabilizes the learning procedure, alleviating the mode collapse issues and improving the learning ability. Though WGAN has many advantages, the authors mentioned that the WGAN still suffers from the issues of unstable training, slow convergence and vanishing gradients because of the weight clipping methods. Therefore, the improvement, using the gradient penalty to replace the weight clipping, has been proposed by [57]. Moreover, [104][201][151] also put forward different approaches to improve the WGAN. The Fig.1.7 shows the trick to smoothen the data distribution of the probability mass the by adding noise from [5].

Conditional GAN The original GAN can be extended to the conditional GANs (cGANs) [142] if both the discriminator D and the generator G are conditioned by extra constraints y. The objective function of a two-player minimax game has the following form:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}\ (x)}[\log D(x \mid y)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z \mid y)))] \quad (1.8)$$

The cGANs are allowed to limit the type of generated samples according to the extra constraint information. The conditioning is performed by feeding y into both the discriminator and the generator as an additional input layer. The generator receive the joint hidden representation that is combined by noise and y as input. In the discriminator x and y are presented as inputs and to a discriminative function (embodied again by a multilayer perceptron in cGAN). The Fig.1.8 shows the structure of the simple cGAN.

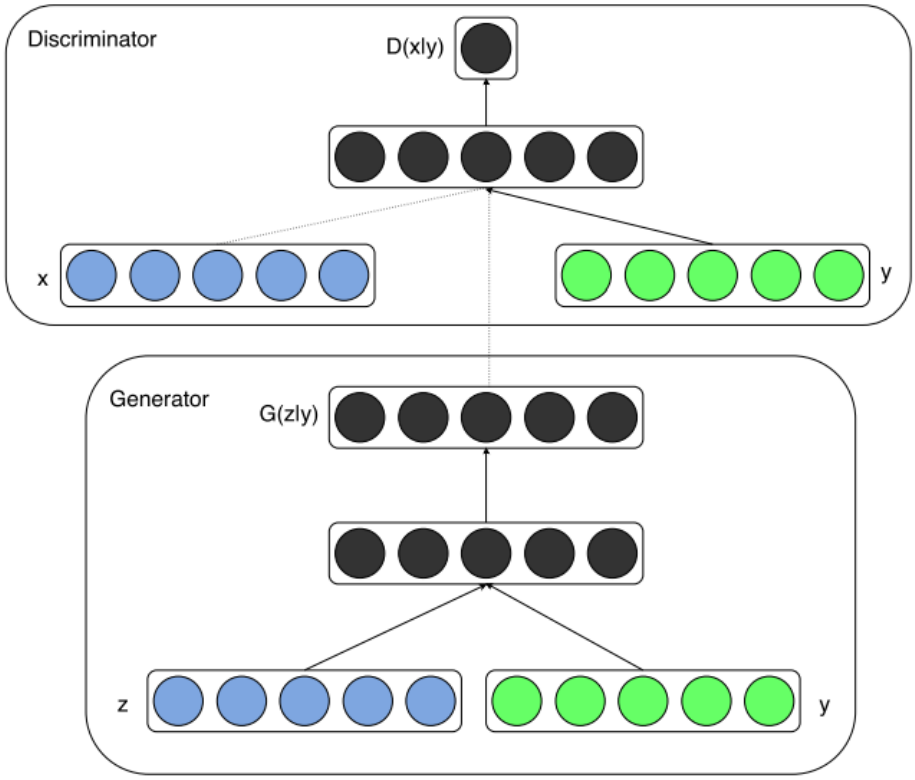


Figure 1.8: The structure of a simple conditional adversarial net (The dotted line and solid line means the y and x are fed into the network separately).

y could be any kind of auxiliary information, such as class labels or data from other modalities. Such limitation allows the cGAN to generate the samples on class labels [149][147], text [218][157]. cGAN has been applied in face generation [50], image translation [179] and image description [29].

Moreover, [92] used cGANs for image-to-image translation, which is called pix2pix. The generator from pix2pix learns a mapping from an image y to the output image $G(y)$.

CycleGAN The Unpaired image-to-image Translation using Cycle-Consistent adversarial networks (cycleGAN) [229] is designed to translate an image from a source domain X to a target domain Y in the absence of paired examples. Fig.1.9 illustrates the architecture of cycleGAN. It consists of two generators ($G : X \to Y$ and $F : Y \to X$) and two discriminators (D_x and D_y) with an adversarial loss $\mathscr{L}_a$ and a cycle-consistency loss $\mathscr{L}_c$. D_y encourages G to transform X into the output that is indistinguishable from domain Y, and vice versa for D_x and F. To further regularize the mapping, cycleGAN proposed two cycle consistency losses that capture the intuition.

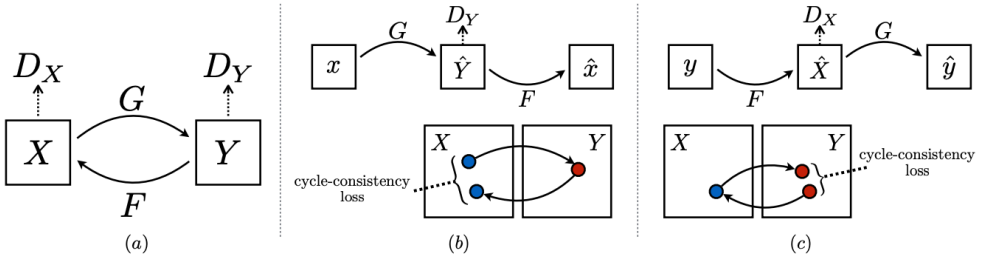


Figure 1.9: The model structure of the cycleGAN.

The adversarial loss $\mathscr{L}_{\text{GAN}}$ for the mapping function $G : X \to Y$ and its discriminator D_Y has the following equation:

$$\begin{aligned}\mathscr{L}_{\text{GAN}}(G, D_Y, X, Y) = &\mathbb{E}_{y \sim p_{\text{data}}(y)}[\log D_Y(y)] \\ &+ \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log(1 - D_Y(G(x))]\end{aligned} \tag{1.9}$$

The G is applied to generate the images $G(x)$ that has a similar distribution to the images from domain Y, while D_Y aims to identify $G(x)$ and the real samples y: $\mathscr{L}_{\text{GAN}}(G, D_Y, X, Y)$. The similar adversarial loss is used for the mapping function: $F : Y \to X$ and its discriminator D_X: $\mathscr{L}_{\text{GAN}}(F, D_X, Y, X)$. Moreover, the cycle consistency loss $\mathscr{L}_{\text{cyc}}$, which is used to reduce the possible space of the generative functions and ensure the generated images to resemble the input images, has the following form:

$$\begin{aligned}\mathscr{L}_{\text{cyc}}(G, F) = &\mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1] \\ &+ \mathbb{E}_{y \sim p_{\text{data}}(y)}[\|G(F(y)) - y\|_1]\end{aligned} \tag{1.10}$$

The cycle consistency loss is implemented by minimizing the $L1$ distance between the input images and the reconstructed images. Therefore, the full objective is:

$$\begin{aligned}\mathscr{L}(G,F,D_X,D_Y) &= \mathscr{L}_{\text{GAN}}(G,D_Y,X,Y) \\ &+ \mathscr{L}_{\text{GAN}}(F,D_X,Y,X) \\ &+ \lambda \mathscr{L}_{\text{cyc}}(G,F)\end{aligned} \tag{1.11}$$

where λ limits the relative weight of the two objectives. Eq.1.11 aims to solve the following equation:

$$G^*,F^* = \arg\min_{G,F}\max_{D_x,D_Y} \mathscr{L}(G,F,D_X,D_Y) \tag{1.12}$$

In conclusion, cycleGAN is an effective unsupervised image translation model using the unpaired images from two different domains.

InfoGAN Information Maximizing GANs (InfoGANs) [19] is a generative adversarial network that modifies the objective of GANs to learn meaningful representations by maximizing the mutual information between a fixed small subset of GAN's noise variables and observations, which turns out to be relatively straightforward. The work in [19] decomposes the input noise vector into two parts: 1) z, which is treated as the source of incompressible noise; 2) c, which is called the latent code and targets the salient structured semantic features of the distributions of the training data. In the information theory, the mutual information between X and Y measures the amount of information learned from the knowledge of the variable X about the other variable Y. The mutual information can be expressed as the difference of two entropy terms:

$$I(X;Y) = H(X) - H(X \mid Y) = H(Y) - H(Y \mid X) \tag{1.13}$$

If X and Y are independent, $I(X;Y) = 0$, because knowing one variable reveals nothing about the other. Given any x from $P_G(x)$, we want $P_G(c|x)$ to have a small entropy. In other words, the information in the latent code c should not be lost in the generation process. [19] proposed the following information-regularized minimax game (The $V_I(D,G)$ means the loss function based on the mutual information I):

$$\min_G \max_D V_I(D,G) = V(D,G) - \lambda I(c;G(z,c)) \tag{1.14}$$

In practice, we parameterise the auxiliary distribution Q as a neural network. In most experiments, the auxiliary distribution Q and the discriminator D share all convolutional layers. There is an extra final fully connected layer to output the parameters for conditional distribution $Q(c|x)$, which means InfoGAN only adds an ignorable computation cost to GAN. Moreover, InfoGAN adopted the techniques from DCGAN [155] to stabilize the training process. The experimental results (Fig.1.10 [19]) show that InfoGAN can learn the inter-

pretable representations that are competitive with the representations known by the existing supervised methods.

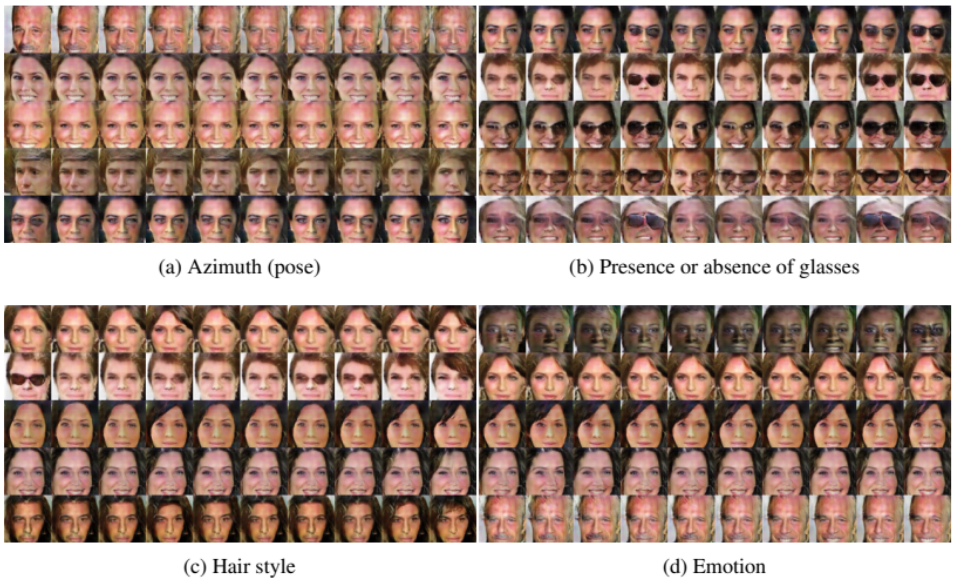


Figure 1.10: Manipulating the latent codes on CelebA: (a) shows that a categorical code can capture the azimuth of face by discretizing this variation of continuous nature; in (b) a subset of the categorical code is devoted to signalling the presence of glasses; (c) shows the variation in hair style, roughly ordered from less hair to more hair; (d) shows the change in emotion, roughly ordered from stern to happy

Progressive GAN The Progressive GAN [95] proposed a new training methodology for GAN to train both generator and discriminator progressively: starting from a low resolution, they added new layers that model increasingly fine details as the training progresses. This incremental mechanism allows the model to first explore the large-scale structure of the image distributions and then shift to the increasingly finer scale detail. Fig.1.11 (from [95]) shows the training procedure of the Progressive GAN.

Fig.1.11 shows that the Progressive GAN uses a generator and discriminator model from the original GAN structure and starts with a very small image. The training process adds new blocks of convolutional layers to both the generator and discriminator to allow the network to learn the coarse-level details effectively and later acquire the fine-level details. The Progressive GAN has the following benefits: 1) The generation of 4x4 pixel images is substantially stable because fewer categories and modes exist; 2) It is much simpler to increase the resolution step by step compared to discovering the fine-level distribution from 1024x1024 images; 3) In practice, it stabilizes the training sufficiently; 4) The training of the Progressive GAN needs less time because most of the iterations are achieved at lower resolutions.

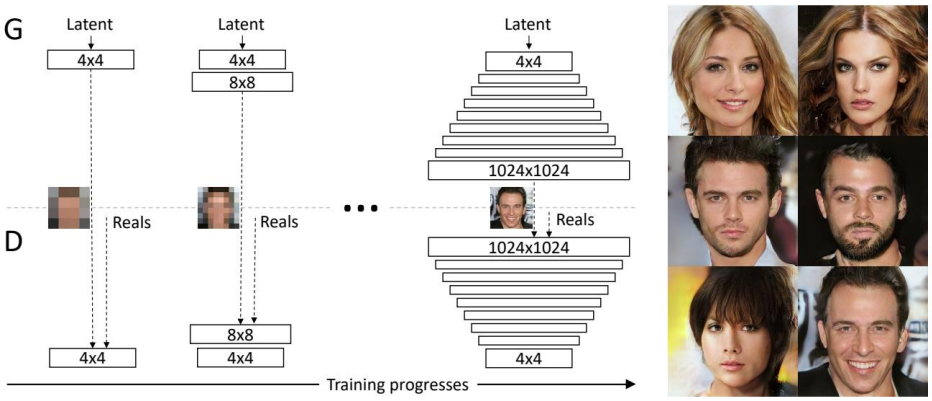


Figure 1.11: The training starts with both the generator and the discriminator having a low spatial resolution of 4x4 pixel. As the training advances, the layers are added incrementally to the generator and the discriminator to increase the spatial resolution of the generated images.

StackGAN The StackGAN [218] is proposed to generate photo-realistic images conditioned on text descriptions. Samples generated by StackGAN are more plausible than those generated by existing approaches.

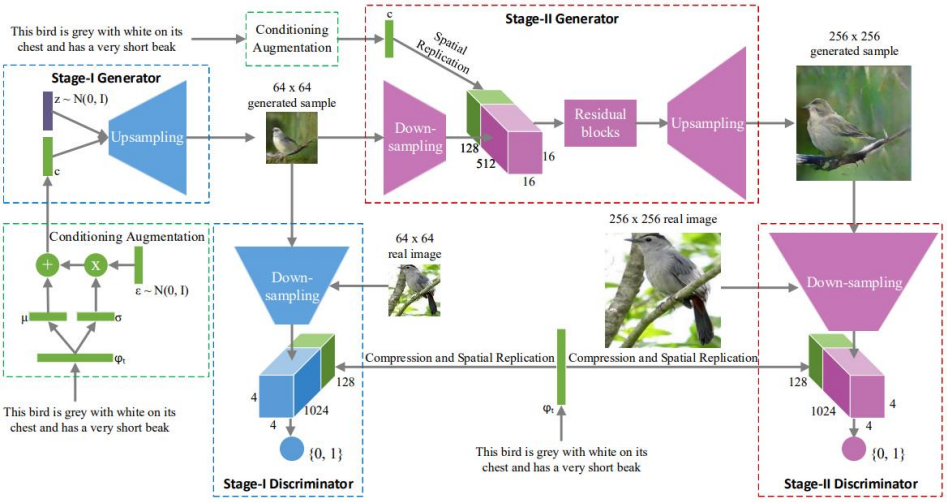


Figure 1.12: The architecture of the StackGAN. *Stage* $-$ *I* draws rough shape and basic colour of the object from the given text with low resolution. *Stage* $-$ *II* generates a high-resolution image with the photo-realistic details by conditioning on both the text and the results of *Stage* $-$ *I*.

The StackGAN model (Fig.1.12) consists of Stage-*I*-GAN and Stage-*II*-GAN. Instead of directly generating a high-resolution image conditioned on the text description, it simplifies by first generating a low-resolution image. The Stage-*I*-GAN generates the low-resolution images and mainly concentrates on drawing a rough shape and correct colours for the objects. The Stage-*II*-GAN receives the results from the Stage-*I*-GAN to generate the photo-realistic high-

resolution images. It conditions on the low-resolution images generated by the previous stage and the text embedding again to correct the defects in the results of Stage-*I*-GAN, and encourages the model to extract the previously ignored information in the text to generate more photo-realistic details. Importantly, StackGAN for the first time generates the realistic 256x256 images conditioned on only text descriptions, while the state-of-the-art methods can generate at most 128x128 images.

UnrolledGAN The unrolledGAN [141] proposed a method to stabilize the Generative Adversarial Networks by the generator's objective concerning an unrolled optimization of the discriminator. It lowers the probability that the generator is overfitted for a specific discriminator, which can lessen mode collapse and improve stability.

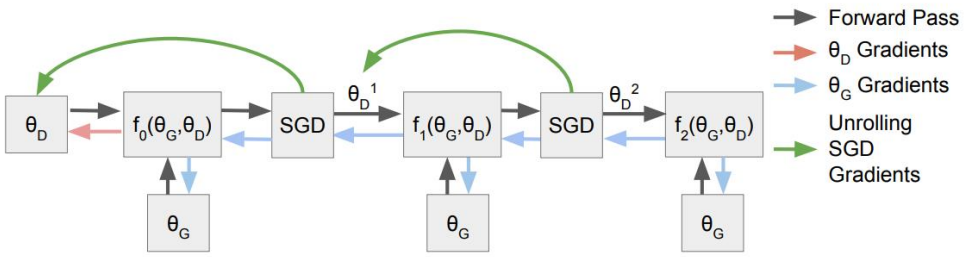


Figure 1.13: An illustration of the computation graph for an unrolled GAN with 3 unrolling steps (From [141]). The update of the generator involves backpropagating the generator's gradient (blue arrows) through the unrolled optimization. Each step k in the unrolled optimization is indicated by the green arrows. The discriminator's update does not require the unrolled optimization (red arrow).

UnrollGAN (Fig.1.13) performs K steps to figure out how the discriminator might optimise itself for a given generator. It demonstrates that 5 to 10 unrolled steps (From the experimental results) attain a rather good performance of the model. Each step applies the gradient descent to optimise a new model for the discriminator. However, the generator used the unrolling to forecast the motions. But it is not employed in the discriminator optimization, which might overfit the discriminator for a specific generator. In conclusion, this technique addresses the problem of mode collapse, stabilizes the GAN training with the complicated recurrent generators, and improves the variety and the coverage of the data distributions of the generated samples.

SAGAN [220] proposed the Self-Attention Generative Adversarial Network (SAGAN) that allows the attention-driven, long-range dependency modelling for image generation tasks. The goal of SAGAN is for the network to recognise

the geometric and structural patterns that frequently appear in certain classes (For example, generated images of dogs do not have defined paws, but have realistic fur). In SAGAN, the details can be generated using the cues from all feature locations, and the discriminator can check that the highly detailed features in distant portions of the image are consistent with each other.

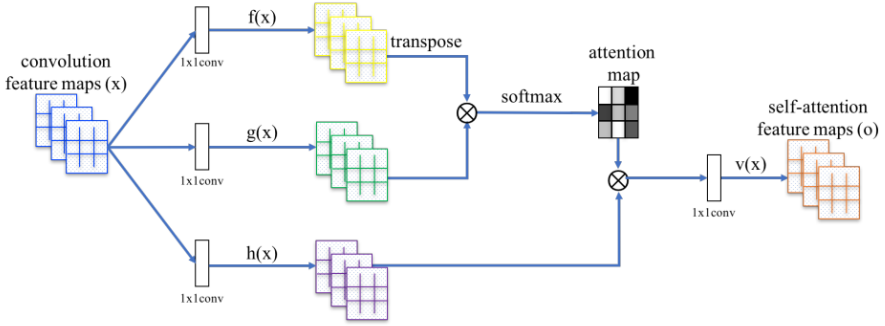


Figure 1.14: The proposed self-attention module for the SAGAN. ⊗ denotes the matrix multiplication. The softmax operation is performed on each row.

Armed with self-attention (The Fig.1.14), the generator may draw the graphics in which the fine details at every location are meticulously coordinated with the fine details in distant regions of the image. Moreover, the discriminator can apply the complex geometric constraints more precisely to the overall image structure. The authors also put forward two techniques to stabilize the training of GANs. First, they use the spectral normalization [143] in both the generator and the discriminator. Second, they use two time-scale update rules (TTUR) [67] specifically to address the slow learning in regularized discriminators. SAGAN achieves the state-of-the-art performance on the class-conditional image generation on ImageNet.

StyleGAN & BigGAN A Style-Based Generator Architecture for Generative Adversarial Networks (StyleGAN) [96] is proposed to generate impressively photo-realistic high-quality images of faces with the control style. StyleGAN uses the mapping network to map the points in latent space to an intermediate latent space. It allows separating different types such as hair, age, and sex to control the image's appearance. Fig.1.15 shows the model structure of StyleGAN.

StyleGAN has six significant aspects : a) Using the basic structure of Progressive GAN [95], b) Tuning the parameters of the model, c) Adding mapping and styles to the model, d) Removing the traditional input, e) Adding noise inputs, f) Adding mixing regularization. The input is first mapped to an intermediate latent space W, which controls the generator through the adaptive instance normalization (AdaIN) [86] at each convolution layer. Gaussian noise

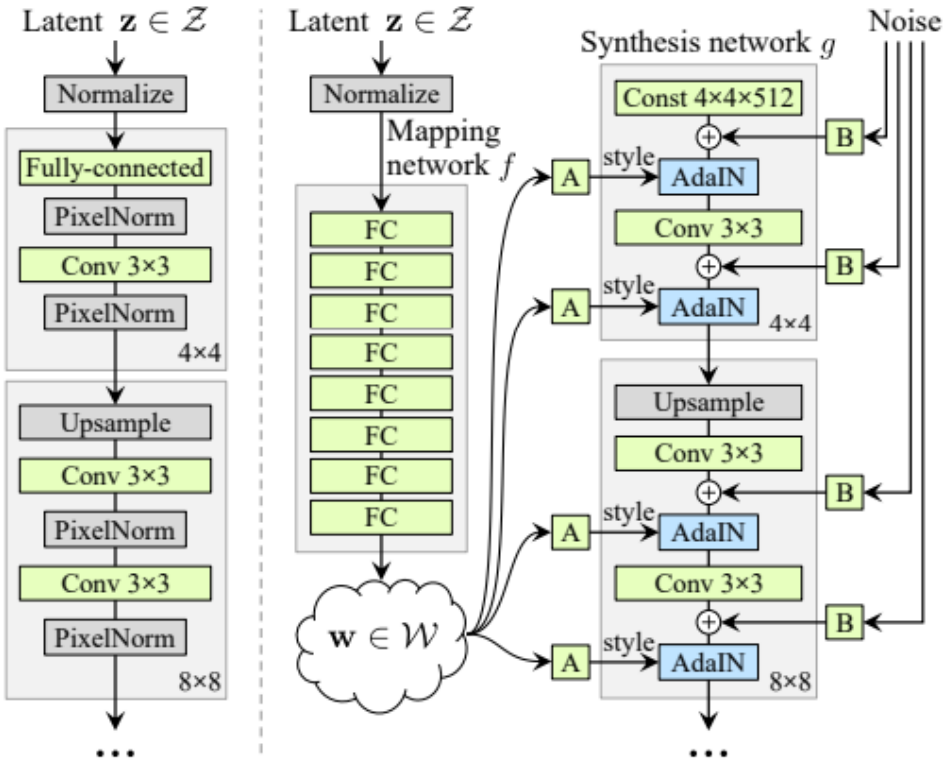


Figure 1.15: The model structure of the StyleGAN .

is added after each convolutional operation before evaluating the non-linearity. Here "A" learned a affine transform from the latent code, and "B" applies learned per-channel scaling factors to the noise input. The mapping network f consists of 8 layers and the synthesis network g consists of 18 layers - two for each resolution. The output of the last layer is converted to RGB using a separate 1x1 convolution. In Fig.1.15, styleGAN can generate high-quality images and also allows controlling the style of the generated images.

Large-scale GAN training for high fidelity natural image synthesis, also called BigGAN [15], is proposed to generate high-quality samples by scaling up the architecture. BigGAN is a large scale Tensor Processing Unit (TPU) implementation of GANs by simply increasing the batch size and scaling the model. BigGAN efficiently generates the images with the high resolution up to 512x512 pixels. However, although BigGAN has the impressive performance, it demands enormous training data and computation resources.

SinGAN Learning a Generative Model from a Single Natural Image, which is called SinGAN [167], is a variant of GAN that the network learns a generative model from a single natural image. SinGAN is an unconditional generative model that can capture internal features within a single image. SinGAN has a pyramid of fully convolutional GAN (illustrated in Fig.1.16), which consists of

a multi-scale pipeline. Each of them learns the distribution at a different scale

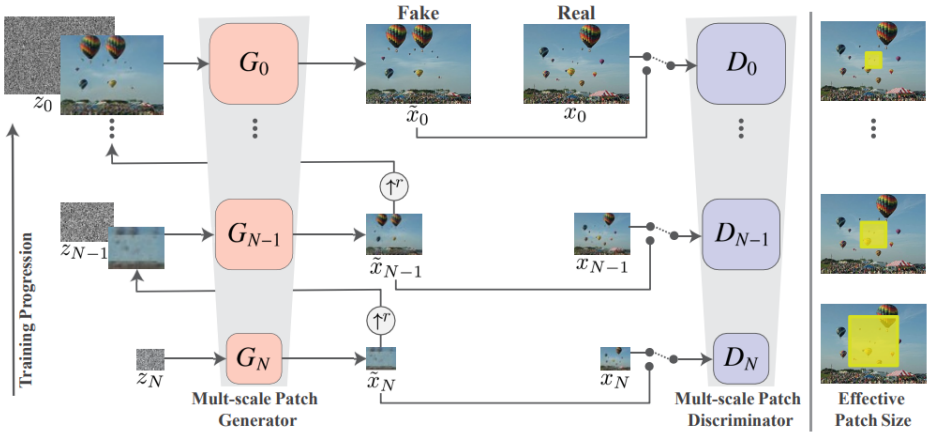


Figure 1.16: SinGAN's multi-scale pipeline.

of the image. The generator and the discriminator at the lowest scale capture the coarse features like the background and the style, whereas at higher scales they are allowed to capture the fine features like edges and corners. SinGAN is trained sequentially starting from the lowest level and rising to the highest level, which includes a combination of two-loss functions:

1) Adversarial Loss: Adversarial loss is applied to measure the distribution difference between the real data and the generated samples. The original implementations in [167] used Improved Training of WGAN (WGAN-GP) [57] to keep the stability of the training procedure.

2) Reconstruction Loss: Reconstruction loss is used to penalize the network for generating the samples that resemble the real training data. The original implementation used the Root Mean Squared Error (RMSE) loss. Despite using only one image as input, the generated samples show high diversity and all the samples are distinguished from each other. Moreover, the internal GAN (INGAN) [168] also trains the network through a single natural image.

1.2.4 The Training of GAN

In practice, training the original GANs model suffers the following common failures:

1) Non-convergence: GAN is based on the zero-sum non-cooperative game. In short, if one wins the other loses. Your opponent wants to maximize its loss while your actions are to minimize them. In the game theory, the GAN model converges when the discriminator and the generator reach a nash equilibrium. When the generator and the discriminator fail to get a nash equilibrium, the models do not converge and maybe become unstable.

2) Mode Collapse: Mode collapse is one of the most challenging issues in GAN. The generator can only produce similar samples with limited diversity.

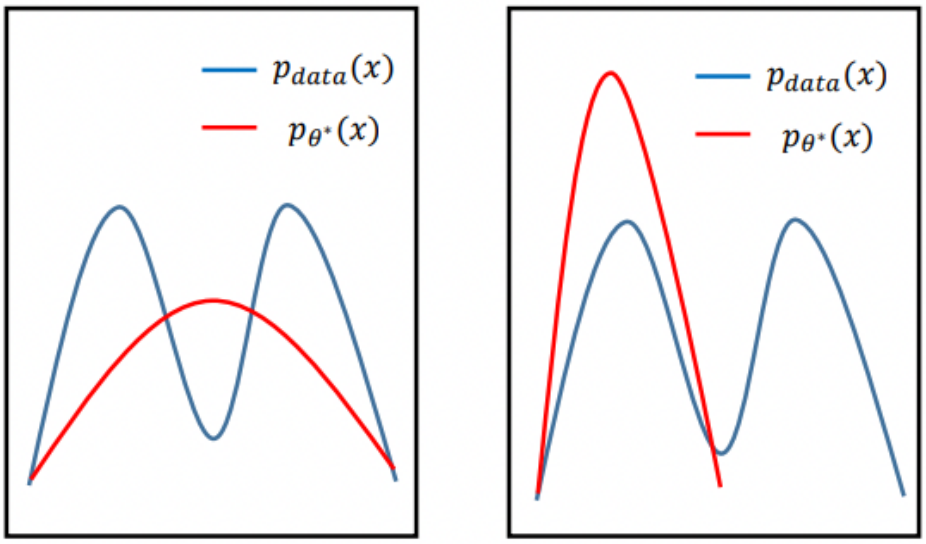


Figure 1.17: a) $KL(p_{data}||p_g)$, and b) $KL(p_g||p_{data})$. Different behavior of asymmetric KL divergence [54].

Maximizing likelihood is equivalent to minimizing $KL(p_{data}||p_g)$. Because the KL divergence is not symmetrical, minimizing $KL(p_g||p_{data})$ leads to a different result. Figure.1.17 from [54] shows the details of different outputs of the asymmetric KL divergence. In Figure.1.17a, the points where $p_{data} \neq 0$ contribute most to the value of KL divergence while other points has small influence on the KL divergence. Thus, p_g is not zero when p_{data} is not zero. Therefore, $p_\theta * (x)$ in Figure.1.17a covers all modes of $p_{data}(x)$, which means $KL(p_{data}||p_g)$ focuses on covering all modes. As for $KL(p_g||p_{data})$, the points where $p_{data} = 0$ but $p_g \neq 0$ contribute to a higher cost. This is why $p_\theta * (x)$ in Figure.1.17b focus on generating the samples that are likely to come from $p_{data}(x)$.

3) Diminished gradient: The discriminator is too powerful to provide a useful gradient. Therefore, the generator cannot be updated, and face the problem of gradient vanishing.

4) Unbalance between the training weights between generators and discriminators.

5) Highly sensitive to the hyper-parameter set during the training process.

The unstable GAN training can be improved by the following techniques that are heuristically motivated to encourage the convergence:

Feature Matching [160] The generator tries to find the best image to fool the discriminator. However, the optimization might turn too greedy, which leads to the model does not converge. The feature matching solves the instability of GANs by specifying a new cost function for the generator that prevents it from over-training the current discriminator. The new objective requires

the generator to produce the data that matches the statistics of the real data, where the discriminator is only applied to specify the statistics that we think are worth matching. Usually, the L2 distance is used as the measurement to define the new objective function:

$$\left\|\mathbb{E}_{x\sim p_{\text{data}}}\,\mathbf{f}(x)-\mathbb{E}_{z\sim p_z(z)}\mathbf{f}(G(z))\right\|_2^2 \tag{1.15}$$

The discriminator and hence $f(x)$, are trained in the usual way.

Minibatch Discrimination [160] When the collapse to a single-mode is imminent, the gradient of the discriminator may point at the similar directions for many similar points. The discriminator will receive an extra similarity between the images in the same mini-batch that is computed by a transformation matrix T. The similarity $c(x_i,x_j)$ between image i and j using the L1-norm and the following equation:

$$c_b\left(x_i,x_j\right)=\exp\left(-\left\|M_{i,b}-M_{j,b}\right\|_1\right)\in\mathbb{R} \tag{1.16}$$

The similarity $o(x_i)$ between image x_i and the rest of images in the batch is:

$$\begin{aligned} o\left(x_i\right)_b &= \sum_{j=1}^{n} c_b\left(x_i,x_j\right)\in\mathbb{R} \\ o\left(x_i\right) &= \left[o\left(x_i\right)_1, o\left(x_i\right)_2,\ldots,o\left(x_i\right)_B\right]\in\mathbb{R}^B \end{aligned} \tag{1.17}$$

The discriminator can easily identify the generated samples from a real one, which helps alleviate the mode collapse issue.

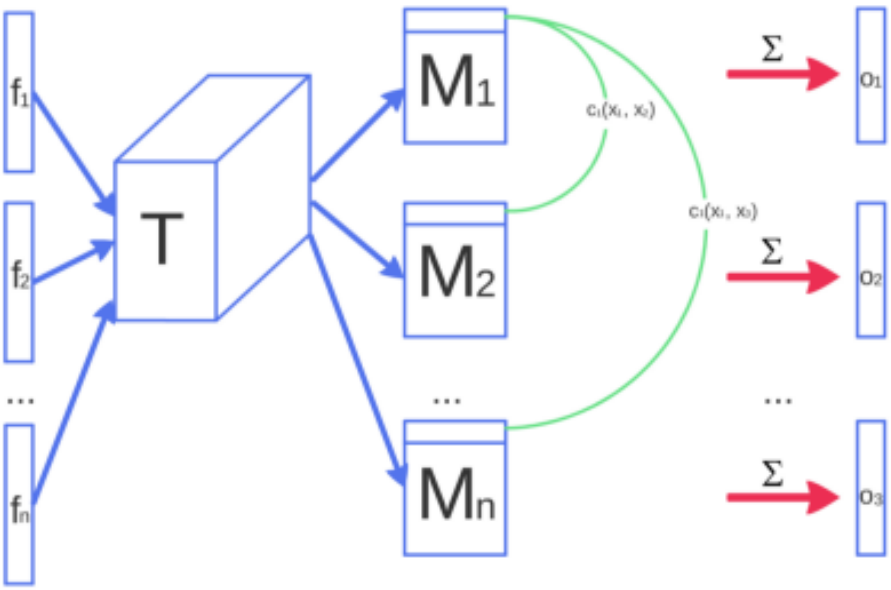


Figure 1.18: Figure sketches how minibatch discrimination works.

Historical Averaging [160] The authors add an $L2$ cost to the cost function to penalize the models from the historical average (Eq.1.18). This trick (We keep the track of the model parameters for the last t models) helps address the non-convergence problem.

$$\left\| \theta - \frac{1}{t} \sum_{i=1}^{t} \theta[i] \right\|_2 \tag{1.18}$$

One-sided Label Smoothing [160] Label smoothing replaces the 0 and 1 targets for a classifier with the smoothed values like 0.1 or 0.9. The work in [1] shows that the label smoothing technique can reduce the vulnerability of the neural networks to the adversarial examples. [174] also proposes a similar idea to add the noise to both real and fake samples.

More effective approaches for improving the GAN training can be found in [22][160][140].

1.2.5 GANs Application

The most impressive applications of GANs are mainly in the areas of image processing and computer vision, such as image super-resolution, image manipulation, data augmentation and image restoration.

Image Super-Resolution The work presented in [111], which is called Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network (SRGAN), is the first framework capable of inferring the photo-realistic natural images for upscaling factors. The authors proposed a super-resolution generative adversarial network, for which they employ a Deep Residual Network (ResNet) [64] with the skip-connection and diverge from Mean Square Error (MSE) as the sole optimization target. They also defined a novel perceptual loss using the high-level feature maps of the Very Deep Convolutional Networks for Large-Scale Image Recognition (VGG) network [171][94][16] combined with a discriminator. Fig.1.19 shows an example of the photo-realistic images that were super-resolved with the upscaling factor.

Moreover, [193] studied the key components of SRGAN and improved the performance. The work in [12] uses GAN for creating the versions of photographs of human faces. The training and testing phases from Face Convolutional GAN (FCGAN) [52] are the end-to-end pipeline with little pre/post-processing. To enhance the convergence speed and strengthen the feature propagation, the skip-layer connection is employed in generative and discriminative networks. The Fig.1.20 shows the experimental results from FCGAN. The work in [95][95] successfully generates the plausible, realistic photographs

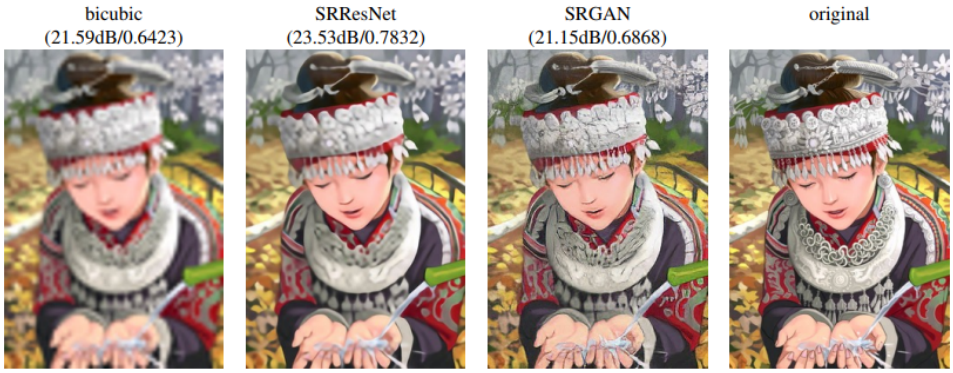


Figure 1.19: The samples from the SRGAN. From left to right: bicubic interpolation, deep residual network optimised for MSE, deep residual generative adversarial network optimised for a loss more sensitive to human perception, original HR image. Corresponding Peak Singal-to-Noise Ratio (PSNR) and Structural Similarity (SSIM) are shown in brackets.

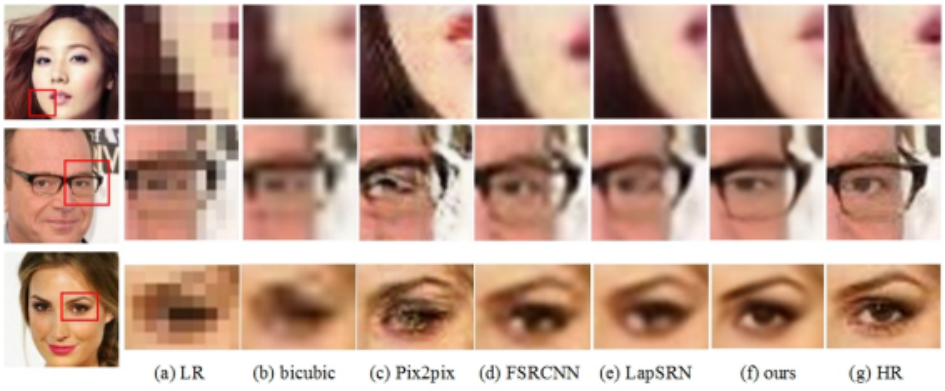


Figure 1.20: Visual comparisons on local details [52].

of human faces by adding new layers (starting from a low resolution) that model increasingly fine details as the training progresses. It also proposed a simple way to increase the variation in the generated images (See Fig.1.21).

Figure 1.21: 256 x 256 images generated [95] from different Large-scale Scene Understanding Challenge (LSUN) categories.

The Deep Tensor Generative Adversarial Networks (TGAN) [33] puts forward the idea of generating high-quality images by improving the tensor structures. Moreover, BigGAN [15] is proposed to generate realistic samples by scaling up the system of GAN (See Fig.1.22).

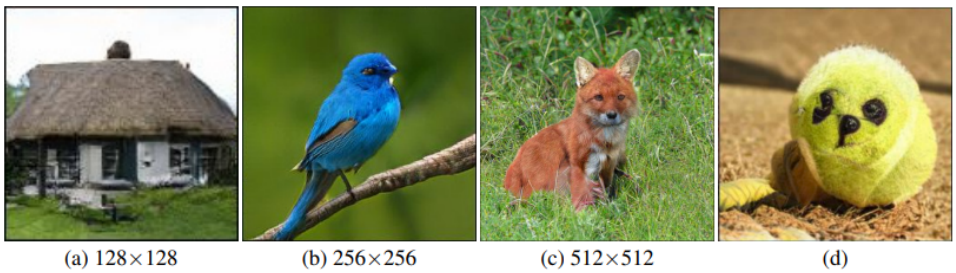


Figure 1.22: The samples from the BigGAN [15] model with the truncation threshold (a-c) and an example of class leakage in a partially trained model (d).

The work in [219] proposed the SAGAN that allows the attention-driven, long-range dependency modelling for image generation tasks. It is different from the traditional convolutional GANs that generate the high-resolution

details as a function of only spatially local points in the lower-resolution feature maps. There also exist other methods [216][227][85] for the generation of high resolution images.

Image-to-image translation The image-to-image translation is a task of taking the images from one domain and transforming them so that they have the style (or characteristics) of the images from another domain. Pix2Pix [92] is a simple and efficient conditional GAN framework for a supervised image-to-image generation. It can supplement the training dataset with generated data to reduce the cost of data collection. Fig.1.23 shows the samples generated by the Pix2Pix.The networks not only learn the mapping from the input images to the output images, but also learn a loss function to train this mapping. The authors also demonstrated that the method was good at generating the photos from the label maps.

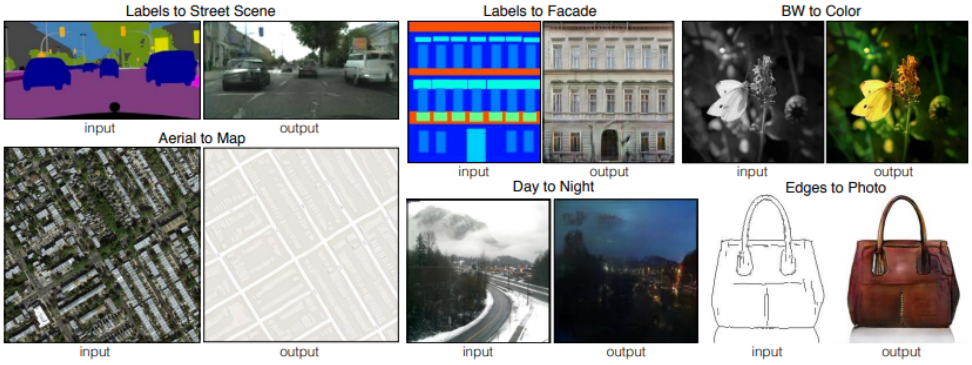


Figure 1.23: Conditional adversarial nets are a general-purpose solution that works well on a wide variety of problems. Here we show the results of the method on several applications [92]. In each case the same architecture and objective are used for the training on different data.

The work in [100][25][229] is proposed to transform the images from one domain to another domain, such as converting the style of the handbag to shoes (See Fig.1.24), transforming the happy face to the sad one [25] and transforming zebras to horses [229].

Text-to-image translation(text-to-image) A model of text to image generation aims to generate the photo-realistic images which are semantically consistent with the text descriptions. The work in [218][157] generates multiple images fitting the description of sentences. StackGAN [218] breaks down the problem into more manageable sub-problems through a sketch-refinement process. Stage-*I*-GAN captures the primitive shape and colors of the object based on low-resolution images. The Stage-*II*-GAN receives the output from Stage-*I*-GAN and text descriptions as the inputs and then exports the high-resolution

Figure 1.24: The samples from GAN that Learns to Discover Relations Between Different Domains (DiscoGAN) [100]. a) The colored images of handbags are generated from the sketches of handbags; b) The colored images of shoes are generated from the sketches of shoes; c) the sketches of handbags are generated from the colored images of handbags

images with the photo-realistic details. Samples generated by StackGAN are more plausible than those generated by the existing approaches (See Fig.1.25).

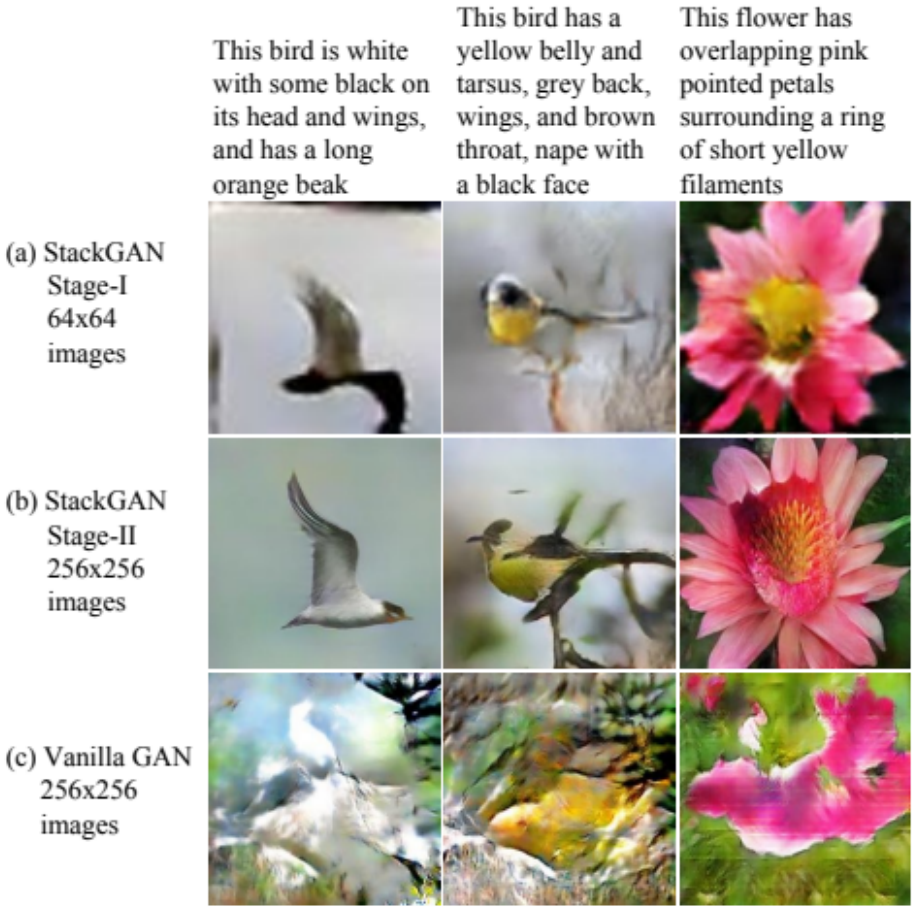


Figure 1.25: Comparison of the propose StackGAN and a vanilla one-stage GAN for generating 256×256 images. a) Given the text descriptions, Stage-*I*-GAN sketches the rough shapes and the basic colors of the objects, yielding low-resolution images. b) Stage-*II*-GAN takes as inputs Stage-*I*-GAN results and text descriptions, and generates the high-resolution images with the photo-realistic details. c) Results of a vanilla 256×256 GAN, which simply adds more upsampling layers to the state-of-the-art Generative Adversarial Text to Image Synthesis (GAN-INT-CLS) [157]. It is unable to generate any plausible images of 256×256 resolution.

The work in [112] proposed a method called as MGANs, precomputing a feed-forward, strided convolutional network that captures the feature statistics of the Markovian patches and can generate the outputs of arbitrary dimensions directly. With the adversarial training, the work in [125] obtains the sample quality comparable to the recent methods of neural texture synthesis (shown in Fig.1.26).

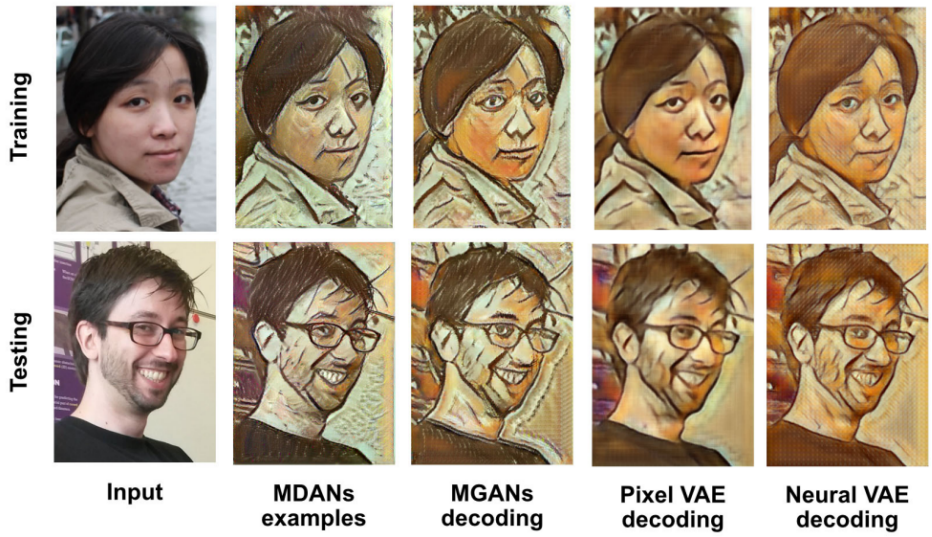


Figure 1.26: MGANs learn a mapping from VGG 19 encoding of the input photo to the stylized example (MDANs). They compare the results of MGANs to Pixel VAE and Neural VAE with both training and testing data.

1.2.6 The GAN Evaluation

Evaluating the generative models is a relatively complicated job. Although human evaluation can sometimes acquire the correlation between samples, people usually tend to appraise the images according to the visual quality and ignore the global distributions, which are the significant features for the GAN training. Fortunately, some effective metrics have been proposed to replace human evaluation. In this section, we show the evaluation metrics used for GAN.

Inception Score Inception Score (IS) is proposed in [160], which uses the inception model [177] for every generated image to obtain the difference between two distributions. IS is the most widely adopted metric, which is computed as [160]:

$$\mathrm{IS}(G) = \exp\left(\mathbb{E}_{\mathbf{x}\sim p_g} D_{KL}(p(y \mid \mathbf{x}) \| p(y))\right) \tag{1.19}$$

where $\mathbf{x} \sim p_g$ indicates that $\mathbf{x}$ is an image sampled from p_g, $D_{KL}(p||q)$ is the KL-divergence between the distributions p and q, $p(y \mid \mathbf{x})$ is the conditional class distribution. where $p(y \mid \mathbf{x})$ is the conditional label distribution for the image $\mathbf{x}$.

A high IS is obtained when the generated samples belong to a specific ImageNet category. It indicates the generative network's ability to produce the data with high quality and variety. Also, the IS is proved to be correlated well with human judgement [160]. However, one disadvantage is that it might make the wrong decision if the generators only produce one image per class.

Frechet Inception Distance (FID) Heusel et al. [67] propose a metric called FID, which measures the distance between the generated data and the real data. The FID between the authentic images x and generated samples g is computed as [67]:

$$\mathrm{FID}(x,g) = \|\mu_x - \mu_g\|_2^2 + \mathrm{Tr}\left(\Sigma_x + \Sigma_g - 2\left(\Sigma_x\Sigma_g\right)^{\frac{1}{2}}\right) \tag{1.20}$$

where (μ_r, Σ_r) and (μ_g, Σ_g) are the mean and covariance of the real data and model distributions,respectively. The trace of a matrix A, designated by $Tr(A)$, is the sum of the elements on the main diagonal. FID is sensitive to mode collapse. So FID is a better assessment of the image diversity. FID is more robust than IS. A lower FID value indicates the better image quality and variety.

Mode Score The Mode Score [18][204] is an improved version of IS, which can calculate the dissimilarity between the real distributions and the generated distributions.
However, selecting an appropriate evaluation metric for GANs is still a disputable subject [182]. Evaluation metrics discussed above still cannot measure the performance of GAN exactly. Moreover, we cannot evaluate the accuracy or error rate as we do in supervised learning because GAN is an unsupervised learning method. Therefore, there still exist the room to improve the evaluation for GAN. We decide to use IS and FID as the evaluation metrics because many researchers used these metrics in their work [160][67][195].

1.3 Research Contributions

This thesis makes the following contributions:

1) We propose a framework called Multi-Group Generative Adversarial Networks (MGGAN) to effectively solve the mode collapse problem and increase the diversity of the generated samples, even for the complicated dataset such as face images. MGGAN trains a set of generative groups simultaneously, each being constituted by a generator and a discriminator. During the training, the generators and the discriminators will be regrouped. A generator will be paired with a discriminator from another randomly selected generative group. Further, we propose the learning rate adjustment strategies that allow the network to jump out of the local minimums swiftly. We conduct the extensive experiments with MGGAN on a synthetic dataset and three realistic datasets (MNIST, CIFAR-10, Lfw). The experimental results show that MGGAN can generate high-quality samples while reducing the training time. Moreover, MGGAN can be applied to other GAN variants. For example, we

can use components from DCGAN or WGAN to compose generative groups.

2) We introduce a simple, communication-cheap model parallelism method without modifying the internal structure of the training network to accelerate the training of GANs. The proposed parallelization method can be applied to any GAN extension. After analyzing the workflow of the GAN training, we propose a novel training mechanism to reduce the total transmission expense during the training procedure. Our method transforms part of transmission time into the execution time within the working nodes since the executions of the training operations in the nodes are more productive than the data transmission. Next, we provide the theoretical analysis and the pseudo-code to compare the proposed method and the traditional data parallelism method to prove that our method can effectively benefit the training procedure. In other words, we investigate both the training procedures of the proposed method and the data parallelism method and construct the mathematical formulae to measure the total training cost. The theoretical modeling clearly shows the superiority of our approach. At last, we conduct the experiments on the synthesis datasets, CIFAR-10, Lfw and LSUN. The results show that our method can decrease the training cost while retaining the quality of the generated samples.

3) We propose a GAN model to generate the unique samples for each of a collection of mobile devices. We call our model PrivacyGAN as it can generate the high quality samples without directly accessing the data in the mobile devices (clients). In this work, we consider a scenario where we need to make use of the collective data in a network of mobile devices as in the distributed training paradigms such as Federated Learning. However, the mobile devices may not have enough data to conduct the local training (a type of few shot learning). PrivacyGAN consist of a teacher GAN and multiple student GANs with the teacher residing in the server while each student GAN residing in a client. The teacher shares the knowledge with the students to improve the generalization and robustness of the generated samples. Our proposed model enables us to a) improve the initial training condition and b) generate the samples without the problem of over-fitting. We do not use the data augmentation techniques to generate the training samples because many data augmentation methods are specifically designed for images, such as translation, flip and rotation, while a GAN model can be applied to generate any type of data and hence be used for a wider range of applications. Importantly, since we consider a network of mobile devices in this work, which are typically resource-constrained, the data augmentation methods [208][97], which are typically computation-intensive, are not allowed. We claim that our PrivacyGAN is lightweight because of the decentralized deployment method proposed in this work. In addition, the decentralized deployment method is

also a key for meeting the requirement of generating the samples for the clients without accessing the clients' data directly. We transform the sensitive data into fake samples and label them using a local discriminator residing in the client. Considering the resource capability (computation and transmission cost) and model performance, we do not use the encryption algorithms or compression approaches to achieve data privacy of the clients. At last, we propose a Joint Restraint Learning Function (JRLF) to limit the information that the student learns from the teacher. The features from the teacher model may not be reliable, sometimes even harmful. Therefore, the strategy is proposed to control the student model's dependency on the teacher model. We also conduct the experiments and the theoretical analysis to demonstrate the efficiency of our method.

1.4 Thesis Organisation

In section 1.1, we demonstrate the motivations of the research presented in this thesis. In section 1.2, we introduce the GAN, including the concepts, variations, the challenges during the training of the GAN, the evaluation and the typical applications of GAN. In section 1.3 we outline the main research contributions.

In section 2.1, 2.2 and 2.3, we present the related work to optimise the GAN training procedure and improve the quality of generated samples by addressing for example the mode collapse issue. In section 2.4, we present the methods to tackle the few-shot learning given our PrivacyGAN is proposed in a few-shot learning scenario. In section 2.5, we discuss the teacher-student model given our PrivacyGAN adopts the teacher-student model.

In chapter 3, we propose MGGAN to solve the mode collapse issue in GAN training to improve the diversity and quality of the generated samples.

In chapter 4, we propose a novel framework called BPGAN to accelerate the GAN training without sacrificing the quality of the samples generated by GAN.

In chapter 5, a lightweight GAN framework called privacyGAN is proposed. It can generate the samples for the clients without accessing their real data and hence protect the privacy of the client's sensitive data.

Finally, chapter 6 concludes the thesis and chapter 7 discusses the future work.

Chapter 2

Literature Review

2.1 The Optimization of GAN Training

GANs training is difficult and insecure for a variety of reasons [160][155][4]. Many methods are proposed to address the problems that occur during the training procedure.

The work in [4] presented the theoretical analysis to help explain the training of GANs. It analyzed why GANs are challenging to train and provided rigorous studies to show the problem during the training, such as saturation and instability. Moreover, the authors examined a practical and theoretically grounded direction to mitigate the issues. The work in [67] proposed a TTUR for training GANs with stochastic gradient descent on arbitrary GAN loss functions. It is proven that TTUR for both generator and discriminator can assist the training in converging to a local nash equilibrium.

The work in [206] proposed a simple modification of stochastic gradient descent, called a prediction step, which stabilizes the adversarial network. The authors presented the theoretical results that demonstrate that the prediction step is stable for solving the saddle point problems. Liang et al. [120] firstly regard the GAN training as a continual learning problem. Next, the authors proposed to utilize the continual learning methods to improve the discriminator in preserving the learning ability to recognize the generator's samples.

2.2 The Mode Collapse in GAN

GAN has shown impressive success in generating realistic, high-quality samples when being trained on the class-specific datasets (e.g. Faces [130]). A few successful applications of GANs are conditional GANs [92][157], image-to-image translation [229], image super-resolution [111], image manipulation [228]; data augmentation [46][136], 3D data generation [133]; text to image [218] and images restoration [215]. Furthermore, adversarial attacking [161][80][128] is a

popular research topic because of its development in face recognition systems. Both adversarial attacking models and face recognition systems require a substantial number of human face images with high quality and diversity.

However, GANs still suffer from the mode collapse problem [18][160][175][9], in which the generator only produces the samples from limited data modes and ignores other modes. The worst outcome is that the generator might simply generate a single sample, which results in complete collapse [4][6]. Many efforts have been made to solve this problem from the following different perspectives.

2.2.1 Covering diverse modes

Many methods were proposed to solve mode collapse by improving the diversity of the generated data [175][40]. For example, Reducing Mode Collapse in GANs using Implicit Variational Learning (VEEGAN) adds a reconstruction network by introducing an implicit variational encoder to map the distributions from training data to noise [175]. Learning Diverse Generations Using Determinantal Point Process (GDPP) uses the determinantal point process to enforce the generated data to have a distribution similar as the real data [40]. StyleGAN [96] redesigns the structure of the network to solve mode collapse by adjusting the latent code. Moreover, BigGAN [15] enlarges the size of the GAN model to improve the performance.

2.2.2 Enhancing the training process of the network

The UnrolledGAN proposed a novel method to tackle the instability by defining an unrolled optimization of the discriminator [140], which shows the reduction of mode collapse. UnrolledGAN differs from the standard GAN in that it updates the generator based on a k-step updated discriminator given the update of the current generator, which aims to review how the discriminator responds to the current generator's state. When the generator is updated, it unrolls the discriminator's training step to consider the discriminator's state in the future k steps with respect to the generator's current update. At the same time, the discriminator is updated in the same manner as the standard GAN. WGAN presents a new cost function based on the Wasserstein distance that has a smoother gradient [5], which can improve the generation's convergence. The Deep regret analytic GAN (DRAGAN) [104] found that the undesirable local equilibria caused the mode collapse phenomenon in this non-convex game. DRGAN adds a gradient penalty scheme that biases the discriminator to avoid the local equilibria.

2.2.3 Using multiple generators and discriminators

Another way to reduce mode collapse is adopting more than one generator or discriminator to capture various modes. Multivariate Anomaly Detection for Time Series Data with Generative Adversarial Networks (MAD-GAN) [113] combines multiple generators with one discriminator. The system encourages each generator to capture its own mode. This technique is inspired by the fact that when the images from two different generators become similar, a higher similarity value is produced. Therefore, it can potentially solve the mode collapse problem by making each generator move toward different modes. Coupled Generative Adversarial Networks (CoGAN) [125] proposed an extension to model the pairs of corresponding images in two different domains. The model combines two GANs and shares the weights of the higher layers of both generator and discriminator. A trained CoGAN (See Fig.2.1) can be used to synthesize pairs of corresponding image that share the same high-level abstraction but have different low-level realizations.

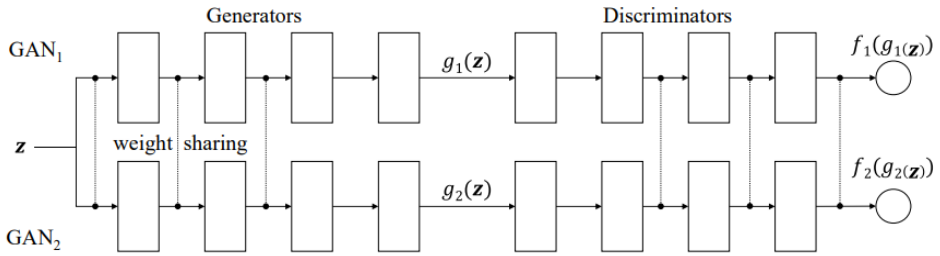


Figure 2.1: CoGAN consists of a pair of GANs: GAN_1 and GAN_2. Each has a generative model for synthesizing realistic images in one domain and a discriminative model for classifying whether an image is real or synthesized.

Dual Discriminator Generative Adversarial Nets (D2GAN) [148] proposed a minimax game that couples two discriminators with one generator, where one discriminator provides high scores for generated samples while the other discriminator identifies the data from the true distribution. The generator produces the data to fool both discriminators. The work in [148] also developed the theoretical analysis to show D2GAN can reduce the mode collapse by minimizing both the KL and the reverse KL divergences between the true distribution and the distribution of the generated data. [38] extends GANs to multiple discriminators and allows the generator to automatically tune its learning schedule, which outperformed GANs with a single discriminator. This framework achieved the faster convergence to higher quality and stability on different tasks. The work in [90] proposed a framework to parallelize many networks and pick the best one to cover diverse modes.

The work in [71] proposed the method called Mixture Generative Adversarial Nets (MGAN). They present the theoretical analysis to demonstrate

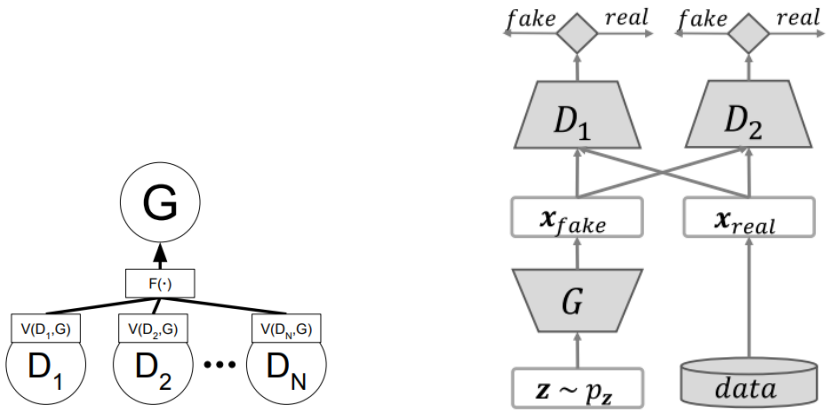


Figure 2.2: The MGAN (left) and D2GAN (right)

that the JS divergence between the mixture of generators' distributions and the empirical data distribution is minimal at the equilibrium point (See Fig.2.3). Fig.2.2 (left) illustrates the general architecture of our proposed MGAN, where all components are parameterized by neural networks. $G_k(s)$ tie their parameters together except the input layer, whilst C and D share parameters except the output layer. This parameter sharing scheme enables the networks to leverage their common information such as features at low-level layers that are close to the data layer, hence helps to train model effectively. It can avoid the mode collapse issue because at this point the JS divergence among the generators' distribution is maximal. Moreover, the authors used the parameter sharing technique, which adds only minimal computational cost to the standard GAN.

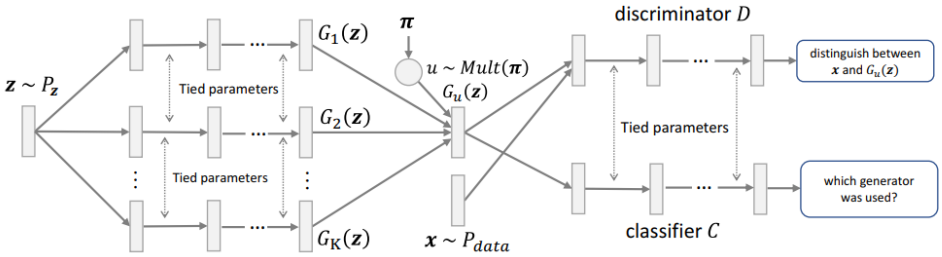


Figure 2.3: The structure of MGAN with K generators, a binary discriminator, a multi-class classifier. Each generator G_k maps z to $x = G_k(z)$, thus inducing a single distribution P_{Gk}; and K generators altogether induce a mixture over K distributions, namely P_{model} in the data space. An index u is drawn from a multinomial distribution Mult(π) where $\pi = [\pi_1, \pi_2, ..., \pi_K]$ is the coefficients of the mixture; and then the sample $G_u(z)$ is used as the output.

However, these methods have some disadvantages, which might result in the following problems. i) The methods make the discriminators too strong, which leads to the problem of vanishing gradient [90][113]. The generators cannot receive enough gradient to make progress from an optimal discriminator. A powerful discriminator does not provide enough information for parameter

updating. ii) Some methods increase the complexity of the network structure and loss function. It might increase the computational time or make the model even harder to converge [125]. iii) Some approaches [140] and [5] have limited ability of generalization. The methods are based on particular training models, which makes it difficult to apply them to other training networks.

2.3 Speeding up the Training of GAN

Many researchers put forward the approaches to reduce the training and learning time from two points of views: 1) Decreasing computation expense; 2) Accelerating the training speed. Since the GAN belongs to the deep neural networks, some methods used to accelerate the deep neural networks can also be applied in the GANs.

2.3.1 Decreasing Computation Expense

Optimize the Model Internal Structure Several recent works were proposed to reduce the computational cost by optimizing the internal structure of the deep neural networks. The tensor decomposition methods belong to the scope of designing optimal architectures. One of the most important attempts at building the architecture is the Network-In-Network (NIN) [122]. NIN builds the micro neural networks with more complex structures to abstract the data within the receptive field. Multilayer (two-layer) perceptron, which is regarded as a universal approximator, is chosen as the replacement.

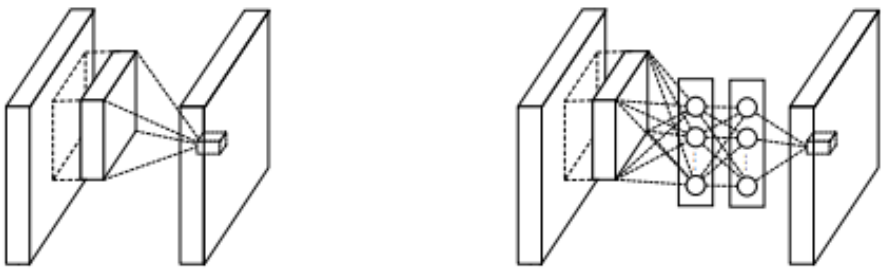


Figure 2.4: Comparison of linear convolution layer and mlpconv layer. The linear convolution layer includes a linear filter while the mlpconv layer includes a micro network

SqueezeNet [89] proposed a compact network that achieves AlexNet-Level accuracy with 50x fewer parameters. There are three main strategies when designing the architecture:
1) Replace 3x3 filters with 1x1 filters whenever possible. Smaller filters have

fewer parameters.
2) Decrease the number of input channels to 3x3 filters. To get a small total number of parameters in a CNN, it is essential to also decrease the number of input channels to the 3x3 filters. The authors decrease the number of input channels to 3x3 filters using the squeeze layers.
3) Downsample late in the Network so that the convolution layers have large activation maps. The authors present their intuition that large activation maps (due to delayed downsampling) can lead to higher classification accuracy, with all else being held equal.

Strategies 1 and 2 judiciously ensure that the model is both small and fast, while the third principle is about maximizing the accuracy on a limited budget of parameters. Moreover, [89] also proposed a building block for the CNN architectures called Fire Module, which allows strategies 1,2 and 3 to be successfully employed.

Efficient Convolutional Neural Networks for Mobile Vision Applications (MobileNet) [78] is based on a streamlined architecture that uses the depth-wise separable convolutions [203] to build the light-weight deep neural networks for mobile and embedded vision applications. The authors used an efficient network and two hyper-parameters to construct a tiny and low latency model that could easily satisfy the design requirements for mobile and embedded vision applications. MobileNet employs the simple and effective tricks to control the model performance: the network width is adjusted by the width multiplier α while the input resolution is adjusted by the resolution multiplier ρ.

Many researchers target at designing a light module with the similar functions to replace the heavy components as used in the deep learning model. The work in [171] uses the microscopic receptive fields (3x3 with a stride of 1) instead of large receptive fields like AlexNet [107] (11x11 with a stride of 4). The small-sized convolution filters allow VGG [171] to have a large number of weight layers and fewer parameters. GoogleNet [159] adopts different convolution layers to create an Inception Module (IM) and capture different information. The improved IM with a 1x1 convolution filter enhances the network's performance while reducing the number of parameters. The work in [122][78] also optimizes the convolution layers to lower the number of parameters. However, the module optimization techniques usually depend on the specific module and service, which requires the customized design. Finally, it is noted that the efficient architecture design is not limited to specific tasks, such as the classification task.

Model Compression [21] is proposed to accelerate the model training by dropping the redundant weights and parameters. A model can be compressed from

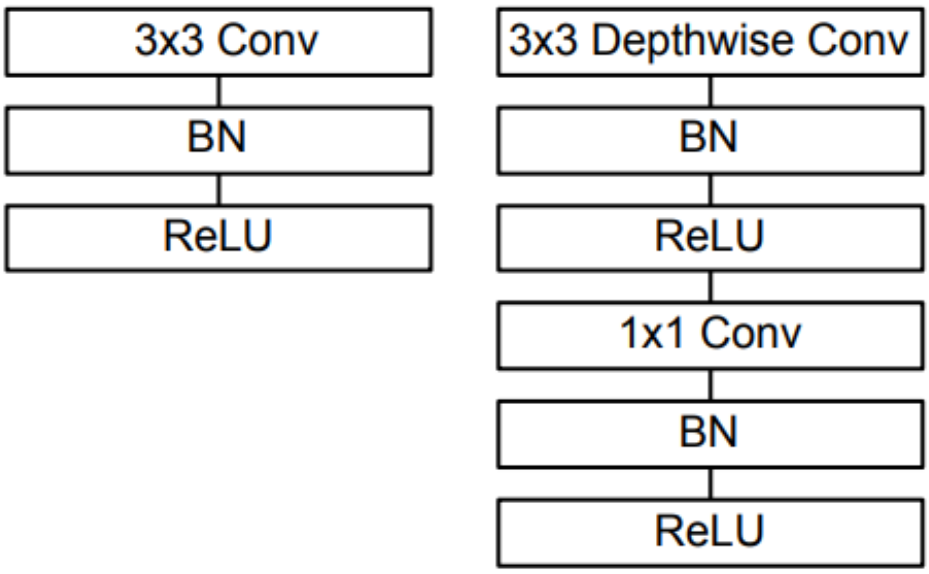


Figure 2.5: Standard convolutional layer with batchnorm and ReLU (left) and Depthwise Separable convolutions with Depthwise and Pointwise layers followed by batchnorm and ReLU (right).

the following three aspects: quantization, network pruning and designing of structural matrices.

A) Quantization

Almost all the weights and activations parameters of a typical network are the values of the 32-bit floating-point format. Quantization is used to compress the original network by reducing the number of bits required to represent each weight. The methods can be further categorized into the following two aspects: weight quantization to lower the model size, and activation quantization [93][188] to lower the inference latency.

i) Weight Quantization: The work in [74] provided a theoretical analysis of the errors caused by the low-bit quantization to determine the bandwidth for a multiplayer perceptrons. These early works focused on simple multiplayer perceptrons. The work in [24] shows that Hessian-weighted k-means can be used to measure the network parameters, which is proposed for clustering and quantizing the network parameters. The work in [79] proposed a novel method called BWNH to train the Binary Weight Networks via hashing. The authors revealed the connection between inner-product preserving hashing and binary weigh network and shows that training binary weights networks is equal to a hashing problem. The work in [192] introduced a novel Fixed-point Factorized Networks (FFN) for pre-trained models to reduce the computational complexity as well as the storage requirement of the networks. The networks have only the weights of -1, 0 and 1 , which efficiently eliminates the most

resource-consuming multiply-accumulate operations (MACs). Moreover, the work in [53][200] used k-means to quantize the weights and the work in [58] introduced a 16-bit representation in the stochastic CNN training to reduce the memory usage.

ii) Activation Qunatization: In order to achieve the latency improvement with the quantized networks, the math operations have to be executed in the fixed-point representations. Binarized Neural Network (BNN) [99] is suitable for resource-constrained environments because they replaces either floating- or fixed-point arithmetic with more efficient bitwise operation. The benefit is that BNN demands less spatial complexity, less memory bandwidth and less power consumption in hardware. The authors proposed several training schemes such as weight compression and noisy backpropagation, which results in a bitwise network that behaves almost as well as the counterpart network with real values. In addition, the work in [180] extends BNN to the ImageNet classification task and receives the higher accuracy. The work in [197] used a sparsity regularizer on each layer to reduce the number of neural parameters and channels. The work in [188] achieved a 3x speedup for inference by using a purely fixed-point model on an x86 CPU without losing the accuracy compared to a floating-point model on the same CPU.

B) Network Pruning

Pruning (See Fig.2.6) is a technique for removing the unimportant parameters and expanding the sparsity of the parameters while ensuring the model performance remains above the desired threshold. Some classical works are

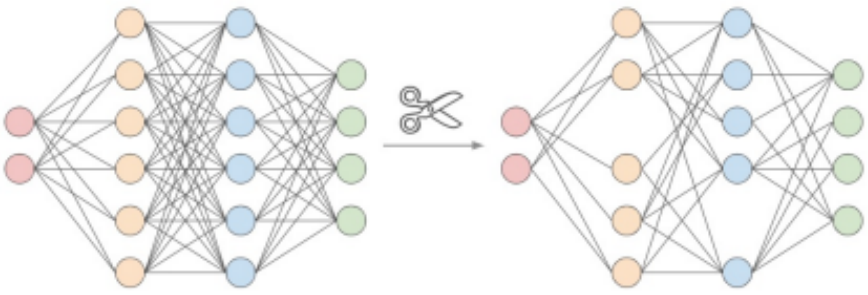


Figure 2.6: A simplified illustration of pruning weights (connections) and neurons (nodes) in a neural network [138]

Optimal Brain Damage(OBD) [110] and Optimal Brain Surgeon Paper(OBSP) [63]. These methods start with a network that has been pre-trained to a decent level of quality to reduce the influence on validation loss. After each pruning, the network is updated with the rest parameters. This iteration is repeated several times until the required number of parameters have been removed. Their work demonstrated that such pruning methods have higher

accuracy than magnitude-based pruning, such as the weight decay method. The work in [61] propose to reduce the total number of parameters and computation in the networks by filtering the important layers. In [184], a simple regularization method based on soft weight-sharing was proposed, which used both quantization and pruning in each training procedure. The work in [197] applied a structured sparsity regularizer on each layer to decrease the irrelevant filters, parameters or layers. Moreover, the work in [114] employs the L1-norm to select and prune unimportant filters. The work in [197] used a sparsity regularizer on each layer to reduce neural parameters and channels. The work in [60] proposed a three-stage compression method including pruning, quantization and Huffman coding. It prunes the small-weight connections after regular network training. Finally, the sparsely connected networks are re-trained to learn the final weights.

C) Design Structural Matrix

A structural matrix can use much fewer parameters to describe an $m \times n$ matrix. Significantly, the structure can reduce the memory usage and accelerate the training and gradient computations. The work in [211] introduces a novel Adaptive Fastfood Transform to re-parameterize the fully connected layers. Moreover, the work in [172][26][146] showed the effectiveness of using the structured matrices. However, the traditional compression methods are not appropriate for GAN because of the training difficulty and the differences in architecture. Many researchers propose specific methods to compress the model. The work in [117] proposed a compression framework to reduce the inference time and the model size in cGANs [142]. The work in [169] developed a novel co-evolutionary approach to reduce their memory usage and Floating-Point Operations Per Second (FLOPs) simultaneously by pruning the network.

Although the above three compression methods bring many benefits, they have the following drawbacks in addition to the difficulty of implementation: 1) Quantization might lower the accuracy of the networks when it is applied in large CNNs such as GoogleNet [159]. Another drawback of such binary nets is that the simple matrix approximations ignore the effect of binarization on the accuracy loss. 2) As for the network pruning technique, first it usually requires superfluous training to find the redundant weights. Second, pruning with L1 and L2 regularization requires more iterations to converge than the general methods. Finally, network pruning will not improve the efficiency although it can reduce the model size. 3) When designing the structural matrices, the structural constraint may hurt the performance because the constraint increases the deviation of the model. On the other hand, there is no theoretical way to find an appropriate matrix.

2.3.2 Accelerating Computation Speed

Data Parallelism is a widely used strategy that divides the dataset into several independent splits and loads them to multiple nodes for parallel training. It accomplishes the target of parallelization by processing more data in a time unit. Data parallelism means that each node trains the same model on a different subset of data. In data parallelism, the forward computing in the nodes does not need the synchronization because each node has a copy of the entire network, including the structure and the parameters. However, the gradients from different nodes must be synchronized in backpropagation. In addition, data parallelism can still accelerate the training even when the computation is performed on a single device, which is achieved by loading a mini-batch of samples to the device.

Model Parallelism refers to the approaches that divide the network structure into different parts. It indicates that each node is responsible for training one piece of model on the same data samples. There are three typical scenarios where model parallelism is applied: 1) When training a deep network, such as CNN, the forward or backward computation of different layers can be dispatched on different nodes. Each working node can concentrate on training the specific layers. 2) When the model is too big to fit in the memory of one single device, one can partition the model into multiple parts. 3) Some deep learning programs, such as matrix factorization [144], only update a small part of parameters per step. In this case, model parallelism can be used to parallelize the work and update them simultaneously. The performance of model parallelism is usually worse than data parallelism because it has much communication cost. In parallel mechanisms, many researchers focus on reducing the bottleneck caused by communications. For example, the work in [115][116] introduced the Parameter Server and the work in [52] is used to reduce the communication cost among working nodes. Moreover, the gradient compression [226][166][123], gradient accumulation [66][191] and gradient compensation [225] are common approaches that target at the gradients. However, parallel processing manifests the defect that the extra cost will be incurred to exceed the profit of parallelization when too many nodes are used.

2.4 Generating Samples with Few Training Data

2.4.1 Few-Shot Learning

The FSL problem is also defined as an N-way-K-Shot classification problem. The training datasets of Few-Shoot Learning includes N class labels and K labelled images for each class. Now we want query Q images among the N classes. FSL [194][173] is a type of machine learning problem that the model

only contains a limited quantity of samples with supervised information. The existing problems in FSL mainly belong to the problems in supervised learning, such as few-shot classification [198][76][39], few-shot regression [181][82] and few-shot reinforcement learning [30][36]. The current FSL works can be classified from the following three aspects.

A) Data

The key problem in FSL is that the network cannot export a robust model with limited training data. Therefore, data augmentation is proposed to solve the issue. Except for simple techniques such as translation, rotation, and reverse, there are specific methods to enlarge the training dataset. The work in [165] proposed a technique named Delta-encoder to augment additional training examples and solve the data insufficiency. This method employs a non-linear function to learn the transformations. The work in [132] considers the information at the instance level and combines the attention mechanism to generate various image data. The work in [28] uses two components: a search algorithm and a search space to auto-generate valuable samples. The search algorithm has two parts: a controller, which is a recurrent neural network and a training algorithm. The search space consists of 5 sub-policies, and each sub-policy has two image operations. Moreover, the work in [20] combines a meta-learner with an image deformation to generate additional training samples. They augment and diversify the one-shot training images by using the deformation network.

In [153], a simple and effective solution is presented to many different target domains: self-training a source domain representation on the unlabelled data from the target domain, which is called Self Training to Adapt Representations To Unseen Problems (STARTUP). The goal of STARTUP is to build the learners for new domains that can be quickly trained to identify new classes when being trained with very few labelled data points. The proposed method generates a feature representation that: a) is adapted to the target domain and, b) receives prior knowledge from the source task to the extent that it is relevant.

The authors from TIM [13] proposed a method to maximize the mutual information between the query features and their label predictions for a few-shot task at the inference while minimizing the cross-entropy loss on the support set.

B) Model

Solving the FSL problem from the model perspective, one needs to generalize a robust model with the auxiliary data. The issues can tackled from three aspects: i) Multitask Learning, ii) Embedding Learning, and iii) Learning with External Memory.

i) Multitask Learning

The work in [223][81] uses the relevance of multitasking and shares the underlying features to learn the specific elements from each task. These approaches can share the parameters during the training process.

In [10], both the original and generated samples are first mapped to a task-specific space by learning separate embedding functions for the source and target tasks and then embedded by a shared variational auto-encoder.

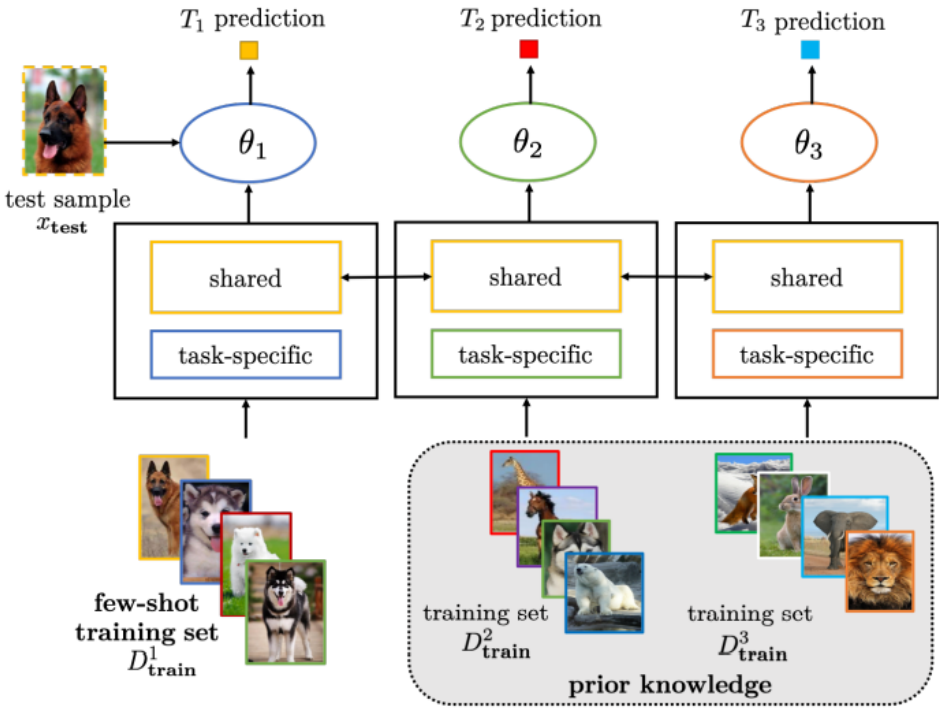


Figure 2.7: Solving the FSL problem by multi-task learning with parameter sharing [194].

ii) Embedding Learning The embedding learning [42][173] depends on the additional tasks to learn a metric space, which can be applied to new jobs without updating the parameters. Moreover, it can avoid over-fitting during the training.

In [210], a novel method named distribution propagation graph network (DPGN) is proposed for few-shot learning. It transmits both the distribution-level relations and instance-level relations in each few-shot learning task. The authors designed a dual complete graph network, which consists of a point graph and a distribution graph with each node representing an occasion of combining the distribution-level relations and instance-level relations. In [224], the authors proposed a Dynamic Convolutional Network (DCN) to deal with conditional few-shot learning. DCN consists of two subnets: DyConvNet contains a dynamic convolutional layer with a bank of basis filters. CondiNet predicts a set of adaptive weights from conditional inputs to linearly combine the basis filters. This greatly improves the parameter learning across different

conditions when there are only limited data.

iii) Learning with External Memory

The work in [156][163] applies the external memory to store the features. Then the learning of the new task matches the elements to generalize the model. The work in [134] presented an inverted pyramid network (IPN) that demonstrates the human's coarse-to-fine cognition paradigm. IPN consists of the global stage and local stage. At the global stage, a class-specific contextual model with a memory mechanism (CCMNet) is proposed to learn the discriminative global support-query relation embeddings. Then at the local stage, a fine-grained calibration is further appended to complement the coarse relation embeddings, targeting more precise query-to-class similarity evaluation.

Meta-Learning [199][75] aims to train a mapping from a few training samples to the hidden parameters that benefit the optimization process. For example, Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks (MAML) [43][44][45] is the most popular method to initialize the parameters of a neural network. The parameter-centric methods depend on current optimizer such as SGD or ADAM [101] to extract the initialization. The approaches [77][176] learn the inner optimizer and generate optimization steps for each training iteration.

Parameters Fine-tuning: The key to optimizing the few-shot learning is to design a proper algorithm to avoid over-fitting [17][72]. However, these methods have their limitations. In practice, the typical data augmentation methods, such as rotation and flip, do not apply to the texts because only the images are invariant to rotation. Other complex data augmentation methods [208][97] improve the model performance at the expense of computation and stability. Therefore, in this thesis, we dispense these approaches to enlarge the dataset. Moreover, in Chapter 5, we assume that the clients' personal information are unlabeled and therefore its learning is unsupervised learning, which makes these approaches not suitable for modelling and algorithm optimization in FSL.

2.5 Teacher-Student Model

The teacher-student model is a concept from Knowledge Distillation (KD), which was previously proposed to compress the deep neural networks [70]. KD refers to the method that improves the training process of a smaller student network under the guidance of a more extensive teacher network. Unlike other compression approaches, KD can downsize the parameters from a network regardless of the structural difference between the teacher and the student network.

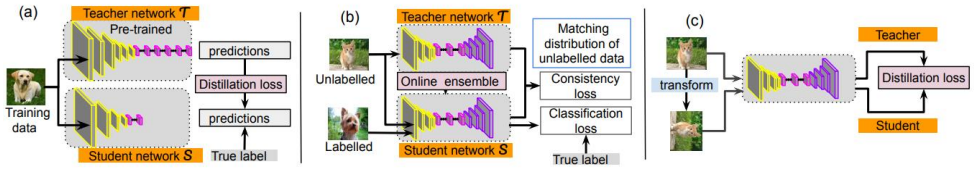


Figure 2.8: Illustrations of the KD methods with the teacher-student frameworks (from [190]). (a) for model compression and for knowledge transfer, (b) semi-supervised learning, and (c) self-supervised learning

2.5.1 KD on Teacher-Student Model

There are the following aspects of KD with the teacher-student model: a) KD based on the number of teachers, b) KD based on data format and c) KD based on teacher-free model, which are described in the following subsections.

a) KD based on the number of Teachers

Transmitting the knowledge from the teacher network to the student network can be performed by logits or internal features from the teacher model.

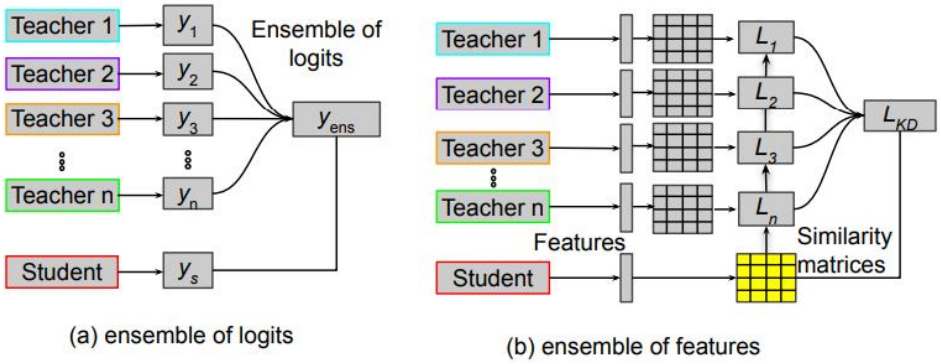


Figure 2.9: Illustration of KD with multiple teachers. The KD methods can be categorized into six types: (a) KD from the ensemble of logits, (b) KD from the ensemble of feature representations via some similarity matrices.

i) KD from Logits

Soft Labels: The work in [7] proposed to transfer the knowledge from the teacher network to the student network by learning the distributions from soft labels. The output of the softmax function on the teacher model's logits has the correct class with a very high probability, while the probabilities of other class are close to zero. It does not provide enough information beyond the ground truth labels. Therefore, the work in [32] puts forward the residual label and the residual loss to allow the student model to use the wrong experience during the training phase to prevent over-fitting and improve the performance. Similarly, the work in [183] proposed the teacher's knowledge as the structured information and trained a student model to extract the important mutual information during the contrastive learning. The work in [23] proposed that

the teacher model with an early-stopping technique makes a better student model.

Noisy Labels: The work in [119] assumes a small clean dataset and a large noisy dataset. It enriches the small clean data to learn a better visual representation by the large amount of noisy data. The knowledge is extracted from the small clean dataset in order to improve the model generated from the large noisy dataset. The work in [202] trains a noisy student model by the following three steps: 1) training a teacher model by the labelled data; 2) using the teacher model to generate the fake labels on unlabeled images, and 3) training a student model on a mixture of labelled and fake labelled images while introducing noise into the student model for improved generalization and robustness. The disadvantage of this method is that when the capacity of the student model is insufficient, it is difficult for the student to assimilate the teacher's logits knowledge successfully. Therefore, it is expected to improve the generality and provide a better representation of logits information that can be easily accepted by the student model.

ii) The Knowledge Distillation from Intermediate Layers

The work in [65] adds a batch normalization layer after a 1x1 convolution layer to determine the margin of the margin ReLU transformer proposed by the teacher. In KD, there are certain benefits to applying 1x1 convolution layer. First, it provides a channel-wise pooling without reducing spatial dimensionality. Second, it can be used to create a one-to-one linear projection of the stack of feature maps. Finally, the projection created by the 1x1 convolution can also be used to increase the number of feature maps directly.

There are several drawbacks to use internal features from the layers in KD. First, most research chooses the intermediate features randomly as knowledge and does not explain why they can be the representative information across all layers. Second, the distillation position of the features is manually selected based on the network or the task. Finally, numerous features may not indicate the knowledge superior to a single layer's feature. As the result, the better methods could be investigated for selecting the knowledge from the layers and representing the knowledge.

b) KD based on the Data Format

There are the following two aspects of KD based on the data format : i) KD using the generators, ii) KD with a few training data, which are described in the following subsections.

i) KD Using the Generators

The work in [41] took both the teacher and student as the discriminator to reduce the discrepancy between them, while a generator is trained to generate some samples to adversarially enlarge the discrepancy. In [213], the generator is set to receive two inputs: a sampled class label y, and a noise input z. A

decoder is also applied to reconstruct the noise z' and the class label y' from the fake data generated by the generator from the noise. The generator produces the higher quality samples by minimizing the errors between y and y' and between z and z'.

ii) KD with a Few Training Data

To enable the student model to learn more efficiently with a small amount of training data, the work in [108][118][8] presented the few-sample KD strategies. The work [126] generated the fake labels from a teacher model with semantic information to provide a supervision signal for the student. Moreover, the work in [118] added an extra 1x1 layer after each pruned layer block in the student model, and estimated the least-squared error to align the parameters with the student. However, although the KD methods with a limited amount of training data can be improved by the approaches of data augmentation and layer-wise learning, the performance of the student is heavily dependent on the number of fake labels, which may influence the effectiveness of the model.

c) The KD based on Teacher-free Model

The traditional KD methods [212][98] have several drawbacks to overcome, although the impressive performance has been achieved. First, these methods are inefficient since the student models rarely utilize all of the knowledge from the teacher models. Second, there are still many challenges in designing and training high-capacity teacher models. Finally, offline distillation necessitates a high level of computing and storage resources. Several unique self-distillation frameworks [207][131][145][205] have been proposed recently to address these issues. Self-distillation is a technique for learning a student model by distilling the knowledge without referencing other models. The work in [48] introduced the notion of self-distillation, in which the student models are parameterized identically to their teacher models.

However, there are still numerous obstacles to overcome. First, the theoretical analysis is needed to explain why self-distillation is more effective. Second, the existing methods focus on self-distillation with certain types of group-based network structures. Finally, all present methods are based on the classification-based activities. It is unclear whether the self-distillation works appropriately for other tasks.

2.5.2 KD on Student Model

The complexity of a deep neural network mainly comes from its depth and width. The teacher model usually belongs to an extraordinarily complex network and is trained separately using the whole dataset, demanding heavy computation resources. The student model is usually: 1) a simplified version of the teacher network which contains fewer layers or parameters [129], 2) a smaller

network with proper basic operations [222][83] or 3) the same network as the teacher [48]. Various methods have been proposed [55][124][189] to transfer the information to the student networks. The work in [154] combines model quantization with knowledge distillation. The student model is a quantized version of the teacher model.

The work in [158] proposed Hints for Thin Deep Nets (FitNets) to train a deeper and thinner student neural network. Because the depth of neural networks is more significant than the width, the framework compresses the wide and deep networks into thin and deeper ones by using the intermediate-level hints from the teacher's hidden layers to guide the training process of the student. Moreover, the work in [139][202][217] proposed to improve both accuracies as well as the speed of convergence of the student models across the domains by using the teacher-student mechanism.

However, the traditional teacher-student model has the "Simply Accept" problem: The student simply agrees with all the knowledge from the teacher model. The teacher model targets the global dataset in the server, which is different from the data owned the student. Therefore, it may not produce the best result if the student simply accept all the knowledge shared by the teacher. In chapter 5, we will propose an efficient method to restrict the student's acceptability of the information from the teacher model by applying the JRLF to the training.

Chapter 3

MGGAN: Improving Sample Generations of Generative Adversarial Networks

GANs are powerful generative models that are widely used to produce synthetic data. This chapter proposes a Multi-Group Generative Adversarial Network (MGGAN), a framework that consists of multiple generative groups for addressing the mode collapse problem and creating high-quality samples with less time cost. The idea is intuitive yet effective. The distinguishing characteristic of MGGAN is that a generative group includes a fixed generator but a dynamic discriminator. All the generators need to combine with a random discriminator from other generative groups after a certain number of training iterations, which is called regrouping. The multiple generative groups are trained simultaneously and independently without sharing the parameters. The learning rate and the regrouping interval are adjusted dynamically in the training process. We conduct extensive experiments on the synthetic and real-world datasets. The experimental results show the superior performance of our MGGAN in generating high quality and diverse samples with less training time.

3.1 Introduction

Generative models have been rapidly growing in recent years. The elemental concept behind such models is to extract the distributions of high dimensional features in the data such as images and texts. Training generative models demands huge computation resources as it requires the complex integration in a high-dimensional space. The model training can be implemented by deep neural networks with the back-propagation algorithm. There are two well-known generative models: GAN [54] and Variational Autoencoders [102]. This

work mainly focuses on GAN, which is known to generate persuasive and agile samples. The generative model includes a generator and a discriminator playing a minimax game that targets reaching a nash equilibrium. The objective of the generator is to generate the data in the same format as the real data in the training set. The discriminator is responsible for capturing the difference in feature distribution between real data (from realistic data $P_{data}(x)$) and fake data (from the generated data $P_g(z)$). When the Nash Equilibrium is reached, the generator can produce the data with the similar feature distributions so that they are regarded as being authentic.

Though GANs can generate the samples for data augmentations, it suffers from a problem called mode collapse [4][18][19][140][160], in which the generator collapses and only generates the samples with limited variety. If the discriminator identifies the samples produced by the generator as genuine, the generator will always produce similar distributions. It is considered a major challenge to extract the feature distributions from complicated datasets with multiple object classes (e.g., ImageNet [31]) since it is difficult for GAN to converge and sometimes the Nash equilibrium does not even exist. In this case, mode collapse becomes a prominent problem. The mode collapse problem is even more severe when using GAN to generate face images. This is because the initial layers in the model mostly learn the same fine-grained feature distributions for a specific type of dataset. In this phase, the generator is limited to targeting a particular pattern even though the discriminator offers favourable feedback. There are two widely used approaches to deal with the puzzle: 1) reinforcing the GANs' learning ability [4][140][160], and 2) ensuring the GANs can extract a variety of modes from different data distributions [18][19][125]. This work takes the second approach.

In this work, we propose a framework called Multi-Group Generative Adversarial Nets (MGGAN) to effectively solve the mode collapse problem and increase the diversity of the generated samples, even for the complicated dataset such as face images. MGGAN trains a set of generative groups simultaneously, each of which is constituted by a generator and a discriminator. During the training, the generators and the discriminators will be regrouped. A generator will be paired with a discriminator from another randomly selected generative group. The regrouping gives the system the opportunity of inheriting the network parameters from the previous group, which prevents the system from dropping the distributions that have been learned.

The challenge of introducing multiple generative groups and performing regrouping is as follows. After each regrouping instance, a new network is formed in each generative group. The new network may produce worse results in the initial period of a regrouping instance because the network starts to capture new features, and then recover gradually to generate better samples. This may

generate many local optimums in the loss curve. We made careful observations to the loss trend during the training in this multi-group setting and proposed a learning rate adjustment strategy to allow the network to jump out of the local minimums swiftly. Further, we propose a strategy to determine the regrouping interval dynamically (i.e., the number of training iterations between two consecutive regrouping instances in MGGAN). The learning rate adjustment strategy combined with the regrouping strategy can facilitate MGGAN to learn disparate modes and share the network parameters efficiently among multiple generative groups. We conduct the extensive experiments with MGGAN on a synthetic dataset and three realistic datasets (MNIST, CIFAR-10, LFW). In the experiments, DCGAN [155] is implemented with our MGGAN framework and is evaluated in terms of the metrics of IS (Inception Score) and FID (Frechet Inception Distance). The experimental results show that MGGAN can generate high quality samples while reducing the training time. Moreover, MGGAN can be applied to other GAN variants. Namely, we can train several different types of GANs simultaneously by employing MGGAN in the same training procedure.

In summary, our main contributions are the following. 1) A novel GAN framework is proposed to reconstruct the GAN group and increase the diversity of generated samples and address the mode collapse problem. Moreover, our proposed structure can improve the quality of the generated samples, which can be applied in the field that requires high-quality samples such as adversarial attacking. 2) A dynamic learning rate adjustment strategy is developed to speed up the training procedure and improve the system stability. It can also decrease the stability of loss function as well. 3) Comprehensive experiments have been conducted to evaluate the effectiveness of our framework with the quantitative benchmarks on different types of datasets.

3.2 The GAN Framework

In GANs training, minimizing G can be transformed to minimize the Jensen-Shanon (JS) divergence between the $P_{data}(x)$ and $P_G(x)$ distribution:$D_{js} = (P_{data}(x) \parallel P_G(x))$. At the Nash Equilibrium, the distribution of the generated samples is similar to the real data distributions, $P_{data}(x) = P_G(x)$. Furthermore, the discriminators get $D(x) = 0.5$ for all x because they can not determine whether the data is real or generated. Because the JS divergence is also known as the symmetric form of the KL divergence [182][87], GAN suffers from the mode collapse problem, which produces samples with limited variety. As the consequence, the synthesized data has low diversity [18][140]. Training GANs is usually separated from the optimal situation (Reaching the nash equilibrium) because of the following reasons:

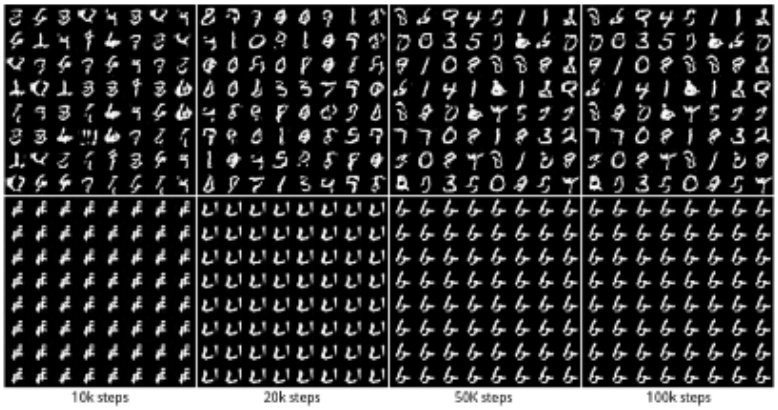
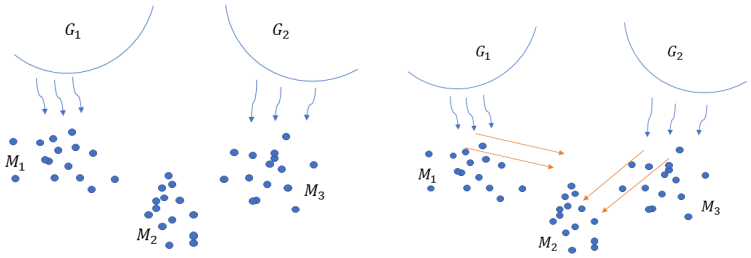


Figure 3.1: Mode Collapse

1) The generator collapses and can only generate limited varieties of samples, which means the system can only capture a few modes from the real distribution. For example, there exist 10 patterns from 0 to 9 in the MNIST dataset. The top row generates all 10 modes but the second row creates only one mode.This is the issue of 'Model Collapse' (Fig.3.1). Namely, only part of the data modes are produced.
2) The model parameters oscillate and never converge. The well-trained discriminators will reject the samples produced by generators, which lead to the gradient vanishing. Some loss functions will not unite with gradient descents, especially for a non-convex game. In other words, the opponent is always trying to confront the behaviour to prompt the models harder to converge.
3) GANs training consumes lots of time. Generators need to transform the one-dimensional noise to the high-dimensional data, in which feature extraction takes long training time.
The first issue is caused by that if the discriminators identify the samples produced by generators as genuine,the generators will always produce similar distributions because it is the easiest way to fool the discriminators. Suppose the discriminators cannot find the best strategy and obtain a plausible output. Then the generators will produce the same data in next iteration, which causes mode collapse. The second issue occurs when the discriminators are powerful enough to distinguish the fake samples produced by the generators effortlessly. The generators cannot get any useful feedback from the training procedure, and the gradient will stop flowing to the generators. However, the convergence of training process is not necessary in accordance with the advance of the data quality generated from GAN. The gradient optimization approaches can only converge to a Nash Equilibrium for convex functions. But the loss curves from GANs are mostly non-convex. The last issue comes from the randomness of the GAN. GAN is an unsupervised learning system and receive random noise as input, which means the output may be different for



(a) (b)

Figure 3.2: a) is the standard GAN working flow. b) is the proposed GAN working flow. In spatial distribution, M_1 and M_2 are the initial targeted of generators G_1 and G_2. In original GAN, G_1 and G_2 target their own distributions, which ignore other distribution. In MGGAN, during the swapping instance, G_1 and G_2 will turn to learn M_2 and M_1. Therefore, the G_1 and G_2 can learn another different distribution M_3.

each training. The generators' samples may not satisfy the requirements even though we have an effective network and a reliable dataset. The targeted data, which belong to high-dimension, have many features compared to the noise samples. Therefore, each training will generate different samples while having the same conditions.

3.2.1 Generative Groups

To tackle the mode collapse problem in the GAN training, we propose a new GAN training framework called MGGAN. The standard GAN consists of a generator and a discriminator, which interact in the same network. In MGGAN, multiple generative groups are generated, each of which consists of a generator and a discriminator. After a certain number of training iterations (called regrouping interval T) in MGGAN, a generator is regrouped with a discriminator randomly selected from a generative group. Between two consecutive regrouping instances, the generative groups are trained simultaneously without sharing the parameters.

Empirically, the generator accepts the data from the noise space, which are of low dimension, to produce the samples, while the real data are usually high dimensional data. The high dimensional data convey much more information than low dimensional data. Thus after a regrouping instance, the samples produced by the generator in a new group will be distinct (new) for the discriminator. The regrouping presents the opportunity to improve the diversity of the generated samples because a generator needs to produce various samples to fool the new discriminators after each regrouping. However, the regrouping also challenges the discriminators' recognition ability because

they have to identify the samples produced by different generators, which may slow down the training process or even make the training difficult to converge. This is the reason why we propose the strategy to adjust the learning rate dynamically.

3.2.2 The Strategy of Adjusting the Learning Rate

Our MGGAN can work with any GAN model based on a single (G, D) group (which we call the base GAN model), such as DCGAN [155]. When we implement MGGAN to work with a base GAN model, multiple (G, D) groups are generated with each group trains with the base GAN model. During the training, regrouping is performed according to our strategy to be presented in this section. Note that different generative groups can also train with different base GAN models.

As we have discussed in previous section, regrouping instances offers the opportunity to increase the diversity of the generated samples. But after each regrouping instance, generators and discriminators will face new partners and need to capture new features, which may cause the loss to increase after regrouping and consequently increase the training time. We conducted the experiments to observe the trend of loss as the training progresses with regrouping being performed from time to time. Fig.3.3 shows the loss trace of a generator in a randomly selected generative group over epochs (up to 200 epochs) when training two generative groups on CIFAR-10 with DCGAN as the base GAN model. After the careful observations and analysis to the loss trace, we found that the training process with regrouping can be divided into three periods, which are labelled as A, B and C in Fig. 3.3.

Period A is the period before the first regrouping instance. In period A, the loss may oscillate in the early stage of the train and then begin to decrease. Period B starts with a dramatic increase in loss (point a in Fig. 3.3). Then the following general pattern occurs throughout period B: the loss decreases fast to some point (point c in Fig. 3.3), increases dramatically again (to point f in Fig. 3.3) and then decreases to a point (point e in Fig. 3.3), which may be lower than the previous low point (i.e., point c). Our experimental records show that the dramatic increases in loss during period B are due to the regrouping in the training. This phenomenon is reasonable, because in the initial period of each regrouping instance, the samples generated by a generator is new and distinct to the discriminator in the new group. If the generator could learn the new features effectively after a regrouping, the loss may decrease to a point lower than the previous low point. In Period C, the loss becomes fluctuating without showing the decreasing pattern seen in period B. Based on the above observations, we propose the following strategy to set the learning rate as the

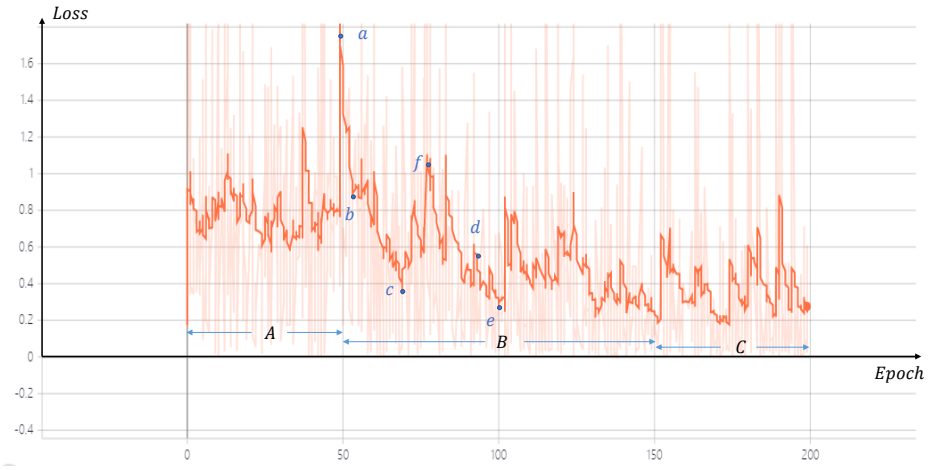


Figure 3.3: The loss trace of a generator with the G-D regrouping. The graph generated by 2 generative group based on the DCGAN and CIFAR-10 with E=200, T=100.

training progresses.

In period A, we use the learning rate set by the base GAN model. When the loss does not show the obvious decrease, we set it at the end of period A and performs the first regrouping. Now the training process enters period B. We adjust the learning rate in each generative group back to the value at the beginning of period A. This is because after regrouping, each generative group needs to learn new features. After the dramatic increase in loss after a regrouping, the loss will typically decrease fast (e.g., from point a to c). This is because the network has learned some features in previous training iterations. A problem of the regrouping scheme is that many deep local optimums (such as point c) appear in period B since the loss decreases fast after last regrouping instance and will increase drastically when the next regrouping instance is performed. Without appropriate handling, the training may be stuck in a local optimum. This is the reason why the loss may not decrease to a point lower than the previous low point after a regrouping. In order to address this situation, we adopt the strategy outlined in Algorithm 1 to adjust the learning rate in period B.

Assume the current regrouping interval is T (i.e., T iterations need to be run between the pervious regrouping instance and the current regrouping because we want to reserve the offset). With T, we can know at which iteration (denoted by i^T) the next regrouping instance will be performed. When the iteration index is in the range $[i^T - \alpha, i^T + \alpha]$ (Line 1), we increase the learning rate by a fraction of δ each time (Line 2 in Algorithm 1), which aims to help the training escape the local optimum. The training may not be stable

Algorithm 1: Adjusting the learning rate in Period B

Input: current iteration (i), the iteration at which the next regrouping is performed (i^T), Learning rate adjusting region (α), upper limit of learning rate ($\overline{L}$), learning rate adjusting ratio (δ), Variation Ratio of the Loss (γ), threshold of Variation Ratio of the Loss (VRL) ($\overline{\gamma}$), current learning rate (L), learning rate at the beginning of period A (L_0)

Output: the adjusted learning rate ($L^{'}$)

```
1  if ( i^T − α ≤ i ≤ i^T + α  and  γ ≤ γ̄) then
2  |   L+ = L ∗ δ;
3  |   L′ = L;
4  |   if L′ ≥L̄ then
5  |   |   L′ = L̄;
6  |   end
7  end
8  if (i = i^T + α + 1) then
9  |   L=L_0;
10 |   L′=L;
11 end
```

if the learning rate is too high. Therefore, a upper limit ($\overline{L}$) is used to cap the learning rate (Lines 3-4). When the training comes out of the range of $[i^T - \alpha, i^T + \alpha]$ (e.g., α is 10 iterations), it means that a new grouping is performed, we adjust the learning rate back to the initial value at the beginning of period A (L_0) to allow the base GAN model to learn new modes. There are the variables γ and $\overline{\gamma}$ in the algorithm. We will present the meaning of them in the next subsection. In this algorithm, α and $\overline{\gamma}$ are the hyper-parameters in the training.

3.2.3 Variation Ratio of the Loss

In Line 1 of Algorithm 1, the if condition includes a term $\gamma \leq \overline{\gamma}$. This is related to a notion we propose in this work: Variation Ratio of the Loss (VRL). γ is the current value of VRL while $\overline{\gamma}$ is the threshold value of VRL. VRL is calculated by Eq.3.1, where $L(n)$ is the loss value at iteration n. The idea of introducing VRL is based on the following considerations and observations to the training process.

$$VRL_n = |L_{n-1} - L_n|/L_n \tag{3.1}$$

First, the reason why we want to increase the learning rate because we want to jump out of the local optimums quickly. But through the observations to the training process, we found that when there is a sufficient change in the loss between two consecutive iterations (i.e., $|L_{n-1} - L_n|$) compared with the current value of the loss, it is very likely that the training can jump out of

the local optimums itself without the need of adjusting the learning rate. It is reasonable because when there are sufficient changes in the loss between iterations, it means that the current learning rate is performing well.

Second, Algorithm 1 takes i^T as input. The value of i^T is directly based on the value of the regrouping interval T (we will present how to determine T in next subsection). The computed value of T may not be accurate. The consequence of the inaccurate T is that the next regrouping and the adjustment of the learning rate are not performed around the local optimum regions (we will discuss the impact of inaccurate T in more detail in next subsection). Introducing VRL can serve as a remedy scheme in the case where the value of T is accurate. Namely, when we attempt to adjust the learning rate, the loss curve is not around a local optimum region. Then it is likely that the current training iterations still produce sufficient changes in loss, which can be detected by checking whether $\gamma \leq \overline{\gamma}$.

3.2.4 The Regrouping Interval

The regrouping interval is the number of iterations performed between two consecutive regrouping instances. We draw an analogy to illustrate the impact of the regrouping interval on the training process. Suppose each generative group is a student who needs to solve a set of mathematical questions and different students are given different sets of questions. Regrouping is to swap the question sets among the students. The regrouping interval T is then the time given to the students to work out their questions before they face new questions. If T is set too big, the generators cannot learn more features from the current samples and waste the training time. If T is too small, the generators do not have enough time to learn the features in the current samples before they are forced to learn other features, which may aggravate the concussion during the training, and even worse the training may never converge. Therefore, if the value of T cannot be accurately predicted anyway, we would rather T to be overestimated than to be underestimated.

Based on the above discussion, we propose the following strategy to determine the regrouping interval T. The initial value of T, denoted by T_0, is set as the length of period A. Given the current T, the next T is calculated by $T = T * \xi$, where ξ is a number greater than one and is a hyper-parameter in the training. This suggests that in our MGGAN the regrouping interval increases as more regrouping instances are performed. The reasons for this are two-fold.

First, as a generator has learned more features (modes), it needs, in theory, to spend longer time in capturing additional new feature. This is because the network needs to adjust more parameters associated with the modes which

have been learned to minimize the divergence in the characteristic distributions between the generator and the discriminator. As the training progresses, more regrouping instances are performed and more modes have been learned. Therefore, it should take longer for a generative group to learn even more modes. Hence the regrouping interval should be increased at subsequent regrouping.

Second, as discussed above, usually the generative group require enough time to capture efficient features after each swapping instance , which means we would rather T to be overestimated than to be underestimated.

We also set an upper limit for T, denoted by $\overline{T}$, which is a hyper-parameter in training. The reason is because when T is too big, the frequency at which the generative groups are fed new samples will be too low. We found this will cause the generative group to regard the new features contained in the new samples as the noise. This claim is also supported by our experimental results (Fig. 3.4 in the experiment section).

3.3 Experiments

In this section, we conduct the experiments to evaluate MGGAN and analyze its behaviour. Both synthetic data and real-world datasets are used. The synthetic data are used to visualize the generated samples and evaluate the learning ability of MGGAN. Meanwhile, we use the real-world datasets to demonstrate the reliability and robustness of MGGAN in tackling the mode collapse in a high-dimensional data space. We train our model on a GPU server equipped with two 1080Ti GPUs. This equipment meets the resource demand for training the GAN.

3.3.1 Experimental Setup

The GAN Structure In the experiments, we adopt DCGAN[155] as the base GAN model . All generators and discriminators are mostly convolutional nets. Only the first layer is the fully connected layer in the generators. The output of previous convolution layer is flattened and fed to the sigmoid function, whose output is then sent to the discriminators. The batch normalization is applied to all the layers except the output layer for the generator and the input layer of the discriminator. In addition, we can use different types of generative models as the base GAN model in MGGAN. For example, we can use both DCGAN and WGAN as the base GAN models. The two GAN models can be trained simultaneously and the network shares the parameters between them.

Datasets We performed the experiments on two classic datasets: synthetic data and realistic datasets. The synthetic data are sampled from a 2D mixture

of 8 Gaussians arranged in a circle with a covariance matrix of 0.02I and means put in the process of zero centroids and a radius of 2.0. The mixture of Gaussian can help investigate the behaviour of our method. Meanwhile, it can evaluate the ability to extract multiple data modes. Though the synthetic data has limited modes and diversity, it can be quickly assessed via visual inspection.

The large-scale datasets includes MNIST[109], CIFAR-10[106] and LFW[84]. MNIST is the dataset of handwritten digits which has a training set of 60000 grayscale images of 28x28 pixels and a test set of 10000 images. All the digits have been size-normalized and centred in a fixed-size image. The training set in CIFAR-10 has 50000 RGB images of 32x32 pixels, while the test set has 10000 images. The images belong to one of the 10 classes. Each class is composed of 6000 images. Label Faces in the Wild (LFW) is a dataset of face photographs designed for studying the problem of unconstrained face recognition. It is composed of 13233 RGB images of 250x250 pixels from 5749 people.

Parameters Configurations For the fair comparison, we use identical experimental parameters for all training networks:
1) Length of a latent vector 100;
2) ReLu activations for all the hidden units;
3) Tanh activation for the output units of the generators;
4) Adam optimizer [101] with dynamic learning rate;
5) The weights are randomly initialized from a normal distribution $N\sim(0,0.02)$.

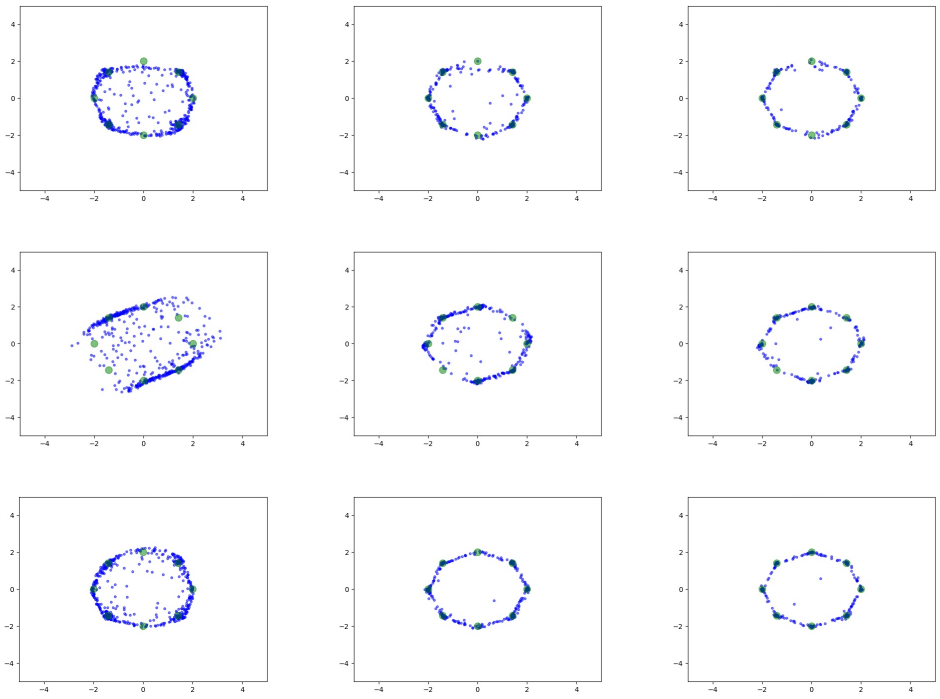
Evaluation Metrics IS [160] and FID [67] are used as the evaluation metrics. A high Inception Score is obtained when the generated samples belong to a specific category from ImageNet. It indicates the generative network's ability to produce the data with high quality and variety. Also, the Inception Score is proved to be well correlated with human judgment[160]. However, one disadvantage is that it might make the wrong decision if the generators only produce one image per class.

Heusel et al.[67] propose a metric called FID, which measures the distance between the generated data and real data. FID is sensitive to mode collapse and is a better assessment to evaluate image diversity. Therefore, FID is more robust than IS. A lower FID value indicates better image quality and variety.

3.3.2 Experimental Results

Synthetic Data

In these experiments, we use the synthetic data to evaluate the ability of MGGAN in learning and capturing multiple data modes and avoid mode collapse. We did not apply our adjustment strategy for the learning rate and the regrouping interval in this training because i) the toy dataset has limited modes and ii) we want to evaluate the impact of different regrouping intervals on the training outcome. Figure.3.4 shows the output of MGGAN and the original GAN. The generated samples are around the authentic modes of the distribution but usually ignore some specific modes. When we apply MGGAN to the training, the generators converge to the real distribution quickly, and the system learns all the modes with the suitable regrouping interval.



(a) E=600 (b) E=1500 (c) E=2700

Figure 3.4: The MGGAN training outputs on a 2D mixture Gaussians dataset. The columns display the data distribution after certain epochs, while the first, second and third row show the training output with the regrouping interval of 0, 25 and 300, respectively. The blue dots are the generated samples, and the green dots are the targeted distributions.

The top row of Fig.3.4 shows that the GAN without regrouping ignores a specific mode. The second and third rows employ MGGAN and converges to the targeted distribution quickly. However, we notice that the data distribution in the second row is more confusing. The distribution spreads out with

disorder due to the different value of regrouping interval T. The learning procedure in the second row has more regrouping instances than that in the third row, which means that the model requires sufficient time to capture explicit features before they learn new samples.

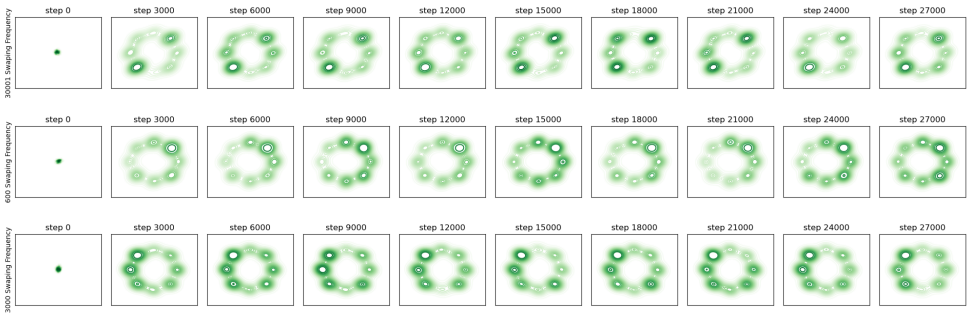
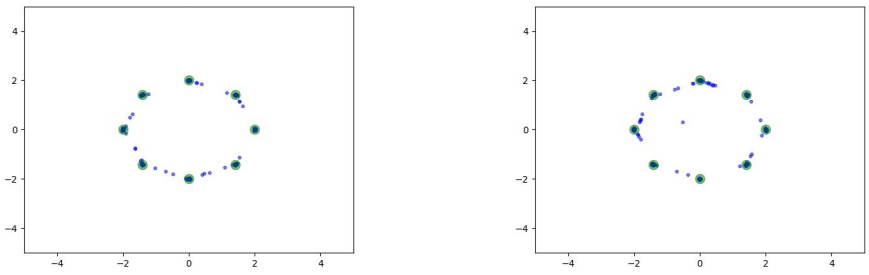


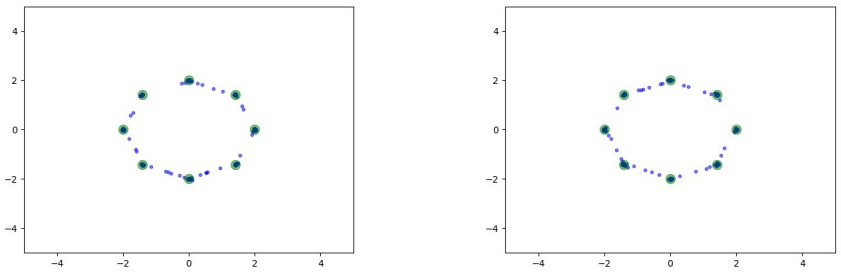
Figure 3.5: The heatmap of the generated distribution at different regrouping frequencies with the increasing number of training epochs

Fig.3.5 displays the heatmap from the mixture Gaussians dataset. The top row displays the output from a standard GAN. The graph shows the generated data disperse instantly and move to the targeted distribution. The top row indicates that even enough training time is given, the model still mislaid some modes from the targeted distribution. The samples generated with proper regrouping interval (The second row of Fig.3.5) moderately spread out to the targeted modes. Two other experiments (The second and third row of Fig.3.5) generate the data that focus on several points near the circle constituted by the modes. For example, the bottom row displays that the generative model mainly focuses on the left part of the circle modes, while the model with the proper parameters from the second row covers all the mode distributions.
Moreover, we conduct experiments to get results from more generative groups. Fig.3.6 shows samples produced by MGGAN with the different number of generative groups (GG) trained for 30k epochs. The model with 2, 3 and 4 generative groups cover all the modes successfully, but the model with more generative groups produces congregated points between contiguous modes. Fig.3.7 shows that a model with more generative groups (such as 6GG) has a better performance than a model with less generative groups (such as 3GG).

Fig.3.8 and Fig.3.7 show that our method has better performance than the original GAN, Adversarially Learned Inference (ALI) [37], and VEEGAN. Moreover, unrolledGAN generates the better results in covering all the modes. Although unrolledGAN has better performance, it requires extra K updating iterations to adjust the training mechanism. Because of this additional computation cost, unrolledGAN is not suitable to be applied in the devices with limited computing power.



(a) 2 GG (b) 3 GG



(c) 4 GG (d) 6 GG

Figure 3.6: Samples generated by MGGAN on synthetic data with 2, 3, 4, 6 generative groups. Generated data is in blue and targeted data is in green.

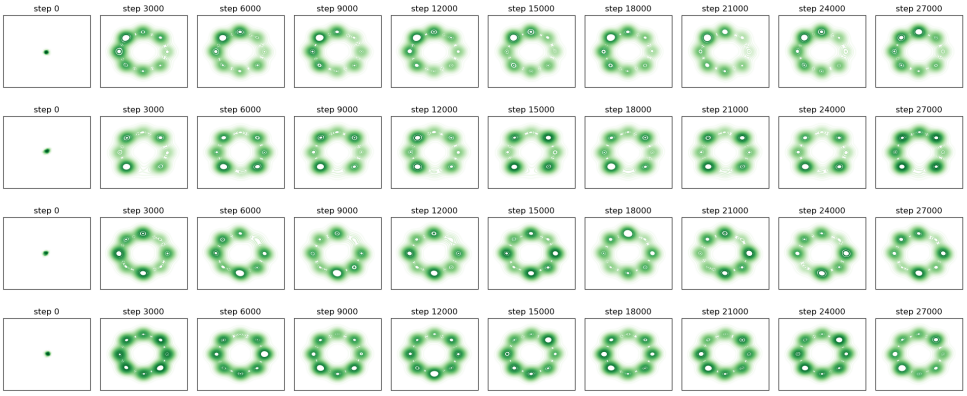
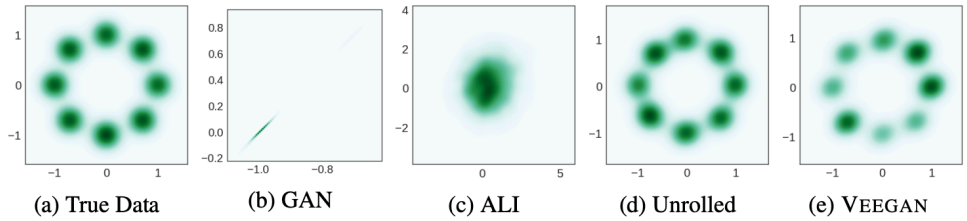


Figure 3.7: The heatmap of the generated distribution from different 2, 3, 4, 6 generative groups

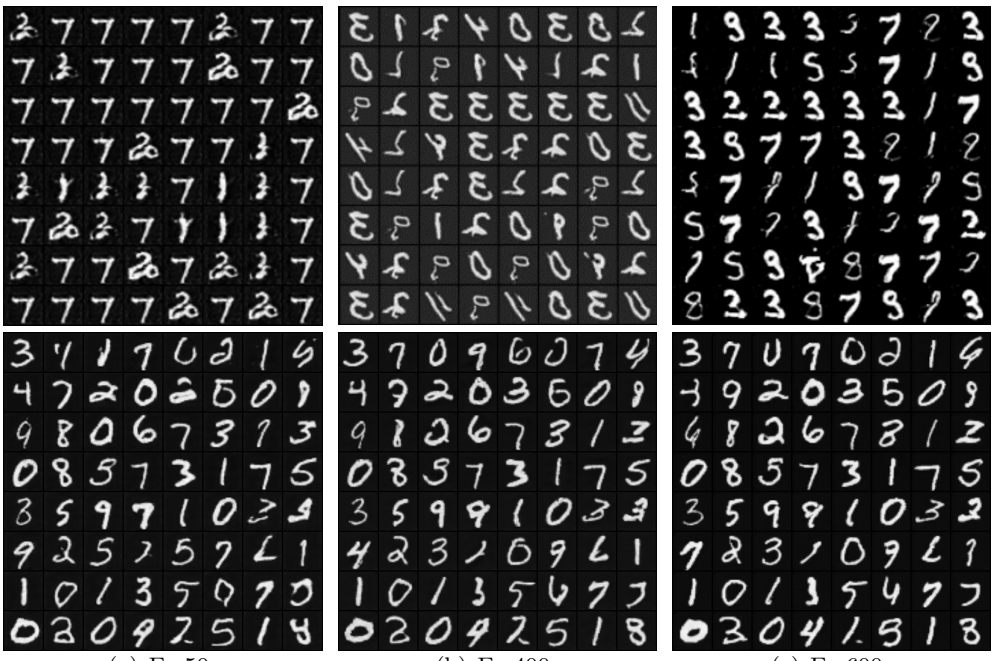


(a) True Data (b) GAN (c) ALI (d) Unrolled (e) VEEGAN

Figure 3.8: Density plots of the true data and the distributions of the samples generated by different GAN methods. The GANs are trained on the mixtures of Gaussians arranged in a ring [175].

MNIST

Through this experiment, the generated samples demonstrate that our method produces various modes of data, which can be seen by inspecting Fig.3.9. The model without regrouping generates the same modes even if it has been trained maturely. The samples in the top row generate same digits, such as 3 and 7, after a long time of training. The second row can produce a variety of modes from the early stage of training. The images in the second row demonstrate that MGGAN increases the quality and diversity of the generated samples.



(a) E=50 (b) E=400 (c) E=600

Figure 3.9: MGGAN avoids mode collapse for a GAN trained on MNIST. The top row was generated by the standard GAN, while the bottom row was generated by MGGAN. The images are generated by the generators after the specified number of training epoch.

CIFAR-10

CIFAR-10 is a real-world dataset with high-dimensional images and is much more complicated compared with the MNIST. Therefore, we test MGGAN on a more effective convolution architecture called DCGAN [155]. In this experiment, we employ our learning rate adjustment strategy, but use different regrouping interval to examine the impact on the quality and the time cost of the samples.

We first show the performance in terms of IS on CIFAR-10 collected from the standard GAN and MGGAN with the same network structure. We set the output from the standard GAN as the benchmark to evaluate the performance.

The positive values in the figures mean that MGGAN returns a better result. Fig.3.10a and Fig.3.10b shows that MGGAN can improve the capability of the generative model in producing the images with high quality and diversity. Fig.3.10c shows the effectiveness of MGGAN in accelerating the training.

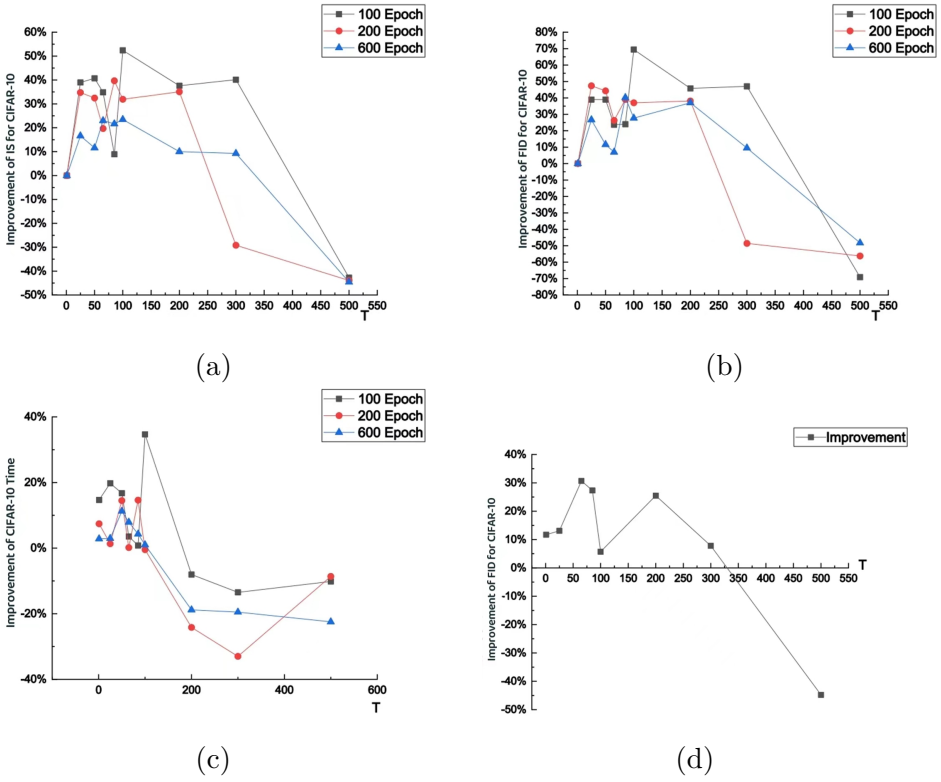


Figure 3.10: The performance improvement over the epochs using MGGAN on CIFAR-10 with T=0, 25, 50, 65, 85, 100, 200, 300, 500. a) Improvement of IS; b) Improvement of FID; c) Improvement of training time; d) Improvement of FID from a model with 4GG to a model with 2GG

Fig.3.10a shows that when $T < 200$, the results are all positive (positive means that MGGAN can benefit the IS value of the generated samples). When $T > 200$, we find that the model performance decreases and is even worse than that of the original GAN. This is because when T is too large, the generative groups are fed with new samples very infrequently. Then the generative groups are likely to treat the new features in the new samples as noise. This is reason why we set a upper limit for T in the training. Meanwhile, from Fig.3.10a, we found that when T is too small, there is not enough time for the model to learn new distributions from the new generative group. The same trend can also be found in Fig.3.10b and Fig.3.10c.

MGGAN can speed up the training and capture the distribution if enough iterations are provided. From Fig.3.10c, we can see that our approach can significantly reduce the training cost with suitable regrouping interval. Fig.3.10c displays that the model gets better improvement in terms of time when T

is small. The optimization idea of our model is based on the swapping instances. More swapping instances tend to generate better results. From the experimental results in Fig.3.10d, we found a model with 4 generative groups produces a better FID value than the model with 2 generative at the training condition while keeping similar IS outputs. Fig.3.10d also demonstrates that more generative groups with appropriate T improves the diversity of generated samples.

Finally, we randomly select several samples generated by MGGAN trained on the CIFAR-10 and present them in Fig.3.11. It shows that MGGAN can produce visually appealing images with convoluted features like cars, trucks, aeroplanes and animals.

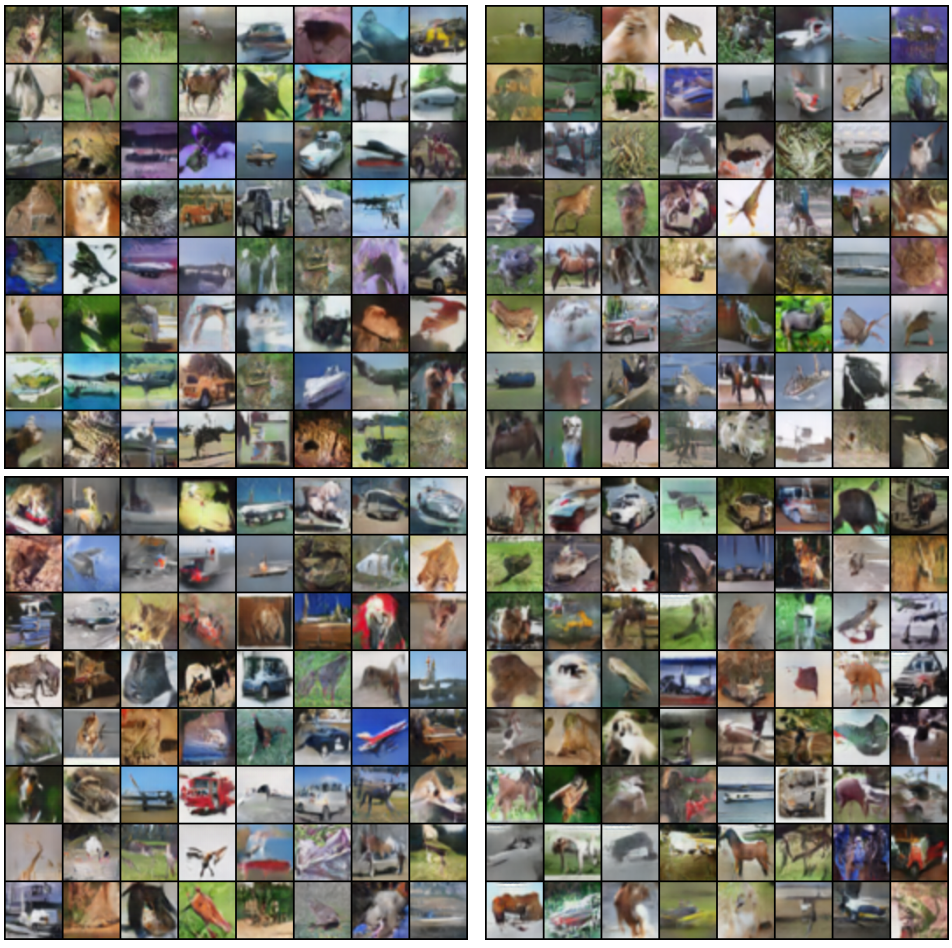
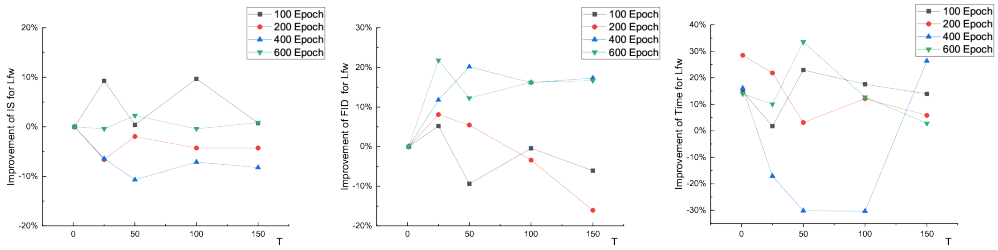


Figure 3.11: Images generated by MGGAN on the CIFAR-10 dataset.

LFW

LFW is a fine-grained human face dataset with sophisticated features such as hair, clothes and facial expressions. These factors make the training more difficult if we do not improve the capability of the generative model. The

standard generators often mislay some particular texture, resulting in mode collapse and low-quality samples. MGGAN can guide the generators to learn the fine-grained features with less training time and increase the learning quality of the base GAN models even for the complex data.



(a) (b) (c)

Figure 3.12: The performance improvement over epochs by using MGGAN on LFW with T=0, 25, 50, 100, 150. a) Improvement of IS for LFW; b) Improvement of FID for LFW; c) Improvement of training time for LFW

Though the data from LFW have much more fine-grained texture than the images from CIFAR-10, MGGAN can still save a lot of training time while keeping the quality and diversity of generated samples. The figures show that MGGAN has better performance in improving IS. Moreover, training process with more epochs can generate the samples with a higher FID value. Fig.3.13 shows some samples produced by MGGAN after training on LFW.

3.4 Discussion

3.4.1 Limitations

However, some limitations should be noted. First, our method increases the computational expense and memory budget compared to the standard GAN. Therefore, we propose a dynamic strategy to reduce the converge time. Second, our method focuses on light models like DCGAN because our method aims to tackle the mode collapse issues without modifying the internal structure. At last, the improvement of our framework is not linearly with the number of generative groups.

3.4.2 Conclusions

In this chapter, we present a novel GAN framework called MGGAN. In MGGAN, we propose to use a set of generative groups. A learning rate adjustment strategy is proposed to help the model accomplish faster convergence and reduce concussion during the training. The regrouping interval is also craftily determined to ensure the model can capture more modes effectively and efficiently. We have conducted the extensive experiments to evaluate the

Figure 3.13: Images generated by MMGAN on the LFW dataset.

robustness and generality of MGGAN by using the Gaussian mixture distribution and the real-world datasets. The experimental results show that MGGAN i) addresses the mode collapse problem well, ii) generates more diverse and higher quality samples with different type of images such as aeroplane and animals, iii) achieves better results when training with fine-grained and complicated datasets such as human faces, and iv) reduces the training cost while maintaining the high quality of the generated samples.

Chapter 4

BPGAN: Acceleration the GAN Training by a Novel Model Parallelism Scheme

With the development of computer vision and artificial intelligence, generative models are widely used to simulate the original data and receive great success in producing images and videos. Although GAN has been widely used to produce synthetic data. It demands enormous training time to obtain a mature GAN model. The training time consists of executing time and transmitting time. In this thesis, we propose the Block Parallelization-based Generative adversarial networks (BPGAN) to reduce the total time cost during the training process. BPGAN is a distributed GAN acceleration framework based on model parallelism. The framework focuses on the operation executions in the model to divide the GAN structure. Different execution components will be deployed in different work nodes (e.g., GPUs) to reduce the training cost. Moreover, BPGAN is an acceleration approach at the framework level, which means that it has excellent compatibility and can be applied to various GAN models. We also provide the theoretical analysis to model the benefit of BPGAN. We conduct extensive experiments on real-world datasets, such as MNIST, CIFAR-10, LFW and LSUN. The experimental results demonstrate the superior performance of our method.

4.1 Introduction

Generating the samples from the complex distribution is one of the fundamental problems in deep learning. GAN has shown impressive success in generating realistic and high-quality samples on specific datasets (e.g., faces [130], bedrooms [170]). Though GAN attains huge success in images [221][127][164] and audio [135][35], it still suffers from the demand for intensive computation

resources and high latency. For example, capturing the distributions of highly diverse datasets usually requires very long training time due to the sophisticated GAN structure and the large training datasets. For example, in [15] 128 TPUs are needed to achieve the training goal. In [27], 48 hours are required to finish each training step.

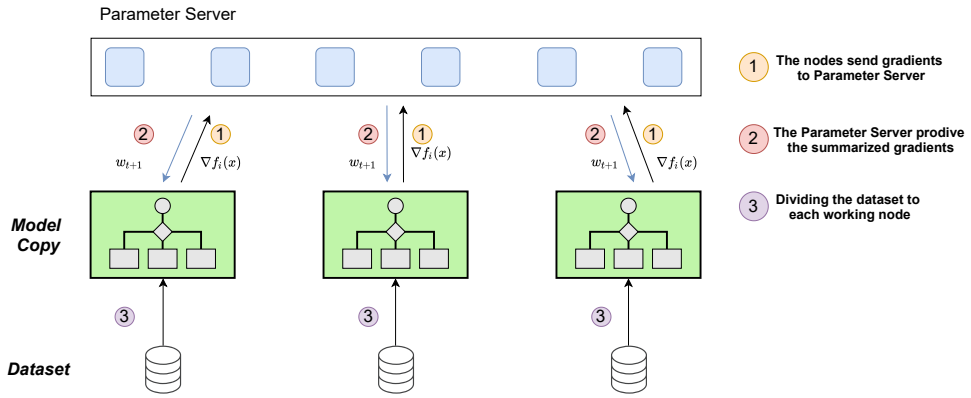


Figure 4.1: The procedure of the data parallelism method. Each work node receives the same batch of data to update the model and sends the gradients to the Parameter Server.

There are mainly two types of parallelism for training the models: data parallelism (DP) and model parallelism (MP). Data parallelism is illustrated in Fig.4.1. In step 3 of this figure, the dataset is divided into N parts (N is also the number of GPUs). Then, a copy of the model is placed on each GPU and trained with the corresponding data parts. After a round of training, the gradients are sent to the parameter server [116][196]. This step is asynchronous because the speed of each work node is different (step 1). Once the parameter server collects all the gradients, it calculates the gradients' average and sends the gradients back to the work nodes, where the model is updated (step 2). Then the data parallelism process moves to the next iteration.

There are two problems with this approach: 1) the entire set of gradients must be transmitted from the parameter sever to the work nodes in order to update the model (step 2). The communication speed will limit the progress of the training iterations. Consequently, this data parallelism approach does not scale well with the number of work nodes; 2) Data parallelism can be further divided into synchronous (Fig.4.2) and asynchronous approaches (Fig.4.3), each of which has its own problems.

The synchronous mechanism means that all the nodes use the same model parameters to conduct the local training. The parameter server waits for the gradients until all the nodes complete their training. This is equivalent to training the model by aggregating small batches on many nodes to form a large cluster. However, the synchronized methods require to balance the com-

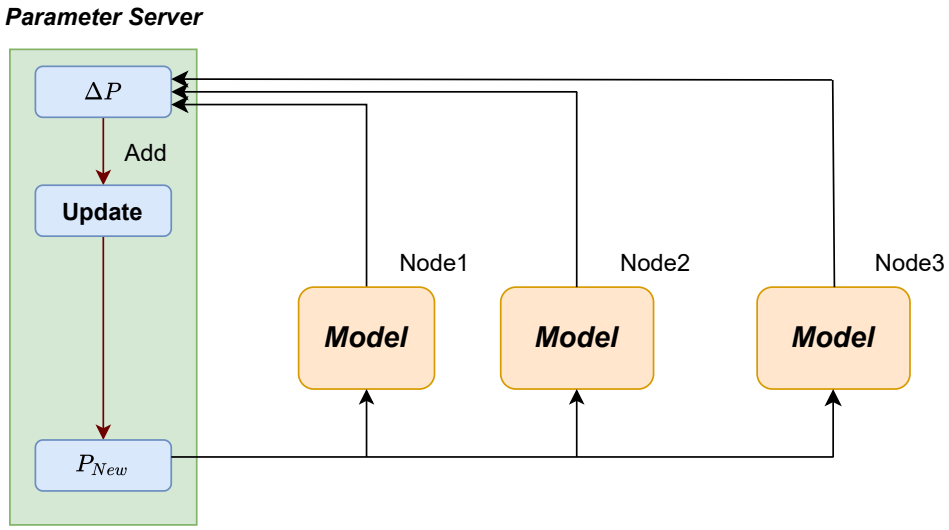


Figure 4.2: The mechanism of the synchronous approach. The parameter server will keep waiting for the gradients until all models in nodes have been updated. The Parameter Server provides a single gradient to update the models in different nodes.

puting power and communication costs of each node. Therefore, the training procedure of the synchronous approach is relatively slow;

The asynchronous methods can update the model after receiving the gradients from only one node instead of from all nodes. This approach is much quicker than the synchronous counterpart. However, there exists a common issue called stale gradients in the asynchronous approach. After one node has submitted its gradients to the server for model updating, the other nodes may still utilize the same gradients for training, which means those gradients are out of date. Although the asynchronous approach can quickly summarize the gradients, the model quality is usually degraded.

The traditional model parallelism method (Fig.4.4) divides the inner structure of the model into blocks of model layers and run different blocks of layers on different nodes in sequence, which is an intuitive way of allocating computer resources when the entire model and its data cannot be fitted into the memory of a single GPU. The batch is sequentially calculated on all the nodes $Node_1, Node_2, ..., Node_N$ for forwarding propagation. The backpropagation is performed in the reverse order, starting from $Node_N$ to $Node_1$. The advantage of model parallelism is that the model that cannot be fitted into a single node can be trained. However, the disadvantage is that a node relies on the results from its previous node, which increases the waiting time during the training. Fig.4.4 illustrates the model parallelism. In the figure, $step1$ shows $Node_2$ and $Node_3$ have to wait until they receive the gradients from $Node_1$. Such a mechanism may cause the blockages in a single node and consequently result

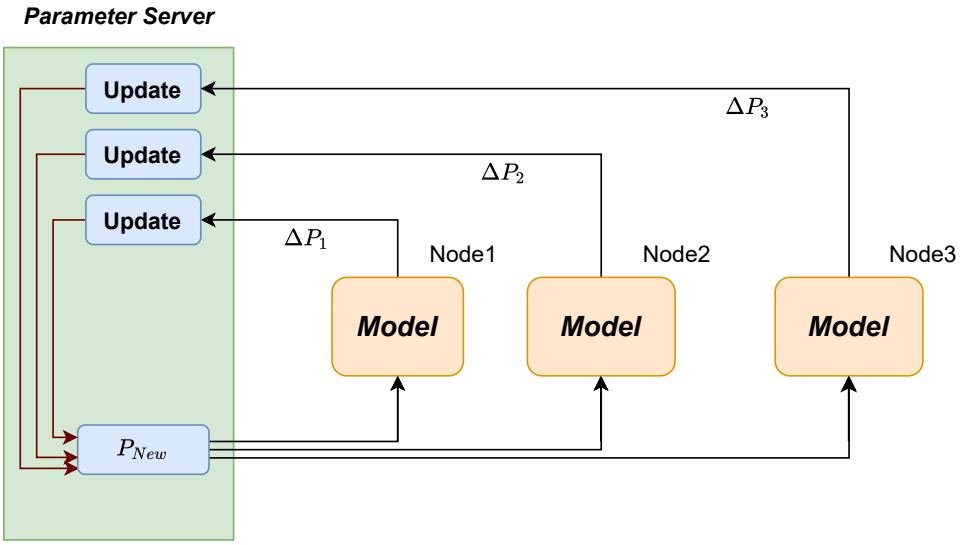


Figure 4.3: The mechanism of the asynchronous approach. The nodes send their own dependent gradients to the parameter server. When the parameter server receives any gradient, it dispatches the summarized gradient to update the models immediately rather than waiting for all the models.

in high latency of the training procedure. Moreover, this mechanism requires high communication capacity because each node has to transmit its gradients to next node.

To address the above-mentioned problems and reduce the running cost in the GAN training, we propose a new framework called BPGAN. Specifically, we design a division strategy for model parallelism, which reduces the time of training a model compared with the traditional data parallelization method. BPGAN depends on the execution workflow of the model training process. It allows us to train the models faster under the same conditions without modifying the internal structure of the models or augmenting the original datasets. It is based on model parallelism. BPGAN divides the training network into two blocks based on the computations in the execution workflow of the GAN. Different blocks will be run on distinct nodes, which change both the execution and communication modes for the better. Comparing to the traditional data parallelism method, BPGAN can reduce the communication cost without increasing other processing cost. In addition, the technique is generic and can be applied to any other GAN variants. We can even change the one-to-one mapping between the generators and the discriminators to quickly implement a model from the existing GAN model [148] [62], or combine the model in [57] with DCGAN [155] to form a new network by adopting different blocks.

The contributions of this chapter can be summarized as follows:

1) We develop a simple, communication-efficient model parallelism method

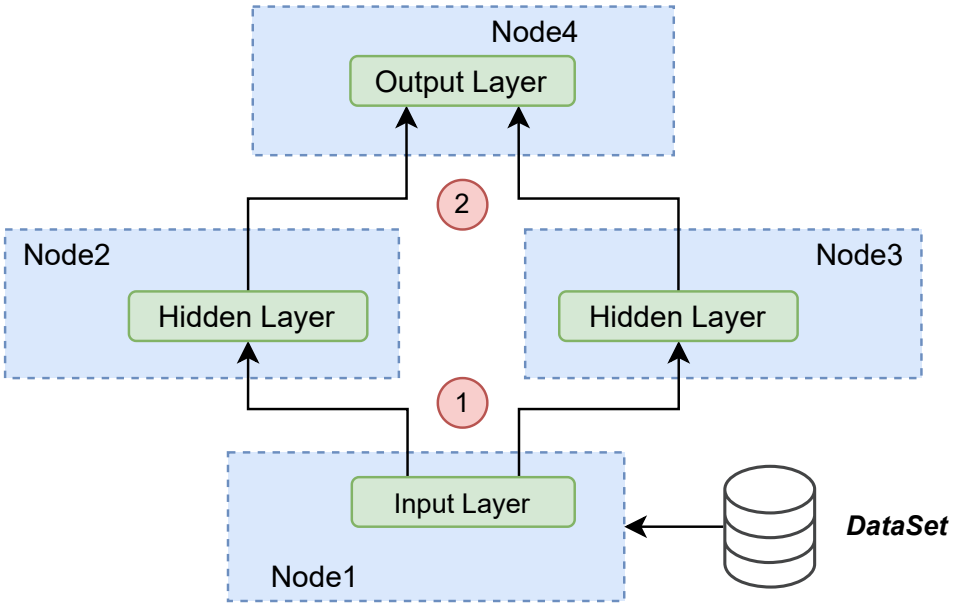


Figure 4.4: The illustration of model parallelism. The neural model is divided into several parts, and each part is deployed in different nodes. A node requires the parameters of previous layers.

to accelerate the training of GAN without the need of modifying the internal model structure. Our parallelization method can be applied to any GAN variants. After analyzing the execution workflow of the GAN training, we propose a novel training mechanism to reduce the total transmission cost in the training. Our method transforms part of transmission time into the execution time in the work nodes since running operations is faster than communicating the data. Our method can also be used to create new GAN structures by selecting the generators and the discriminators from different GAN variants, for example, a generator from DCGAN [155] and a discriminator from WGAN [57].

2) We conduct the theoretical analysis to compare our method with the traditional data parallelism method. We investigate the procedures in both the proposed method and the data parallelism method, and model their total training cost theoretically. The modelling shows the superiority of our approach rigorously.

3) We conduct the extensive experiments on the synthetic dataset, CIFAR-10, LFW and LSUN. The results show that our approach can reduce the training cost without sacrificing the quality of generated samples.

4.2 The Proposed Method

The traditional model parallelism for GAN splits the GAN network (either generator network or discriminator network) into multiple parts and run each

part on a different node. Though it allows training a large model, the traditional model parallelism method cannot be applied in practical applications due to the unstable waiting time and the large quantity of gradients and model weights that need to be transmitted between the nodes. As for the data parallelism framework, the model weights and gradients have to be transmitted between the parameter server and each of the work nodes, which incurs much communication overhead. Considering these drawbacks, we propose a new parallelization scheme called BPGAN to accelerate the running of the GAN framework.

In BPGAN, the whole GAN framework is divided into two parts, which we call Discriminator Block (DB) and Generator Block (GB). The structures of DB and GB are illustrated in Fig.4.5. In each block, we further divide the calculations in the execution workflow of GAN training into two types: First-order calculations and second-order calculations. The first-order calculation contains the operations that are performed on the parameters of GAN networks (either generator network or discriminator network, labelled "1" in Fig.4.5) to update model weights, including convolution, deconvolution, pooling, etc. The second-order calculations (encapsulated as the second-order computing space, labelled "2" in the figure) include the operations that are performed on the output of the first-order calculations, including the calculation of loss function (i.e., error), generating samples (for the generator network) and generating the labels (for the discriminator network). It is worth noting that most of the first-order calculations are the computations performed on the higher-dimensional data (e.g., calculating the model weights) in the neural network, compared with the second-order calculations, which are performed on the low-dimensional data such as the computation of the mean error, the total error of fake/real samples, the label of generation samples, etc. The second-order computing space in DB accepts the weights computed by the neural network of the discriminator to generate the labels and the error of discriminator network. The labels are sent to the other node of the node pair to assist the training of the corresponding generator. The error of the discriminator, which contains the errors for the fake and the real samples, is used to update the parameters of the discriminator.

Similarly, GB contains the first-order calculations performed for the generator network and the second-order computing space, which accepts the weights of the neural network of the generator to generate fake samples and calculating the errors for updating the generator network.

As described above, DB and GB are deployed to run on two different nodes. As shown in Fig.4.5 (labelled "3" in the figure), the communication between this pair of nodes includes the labels and the fake samples, the volume of which is very small compared with the parameters of the neural networks, which is

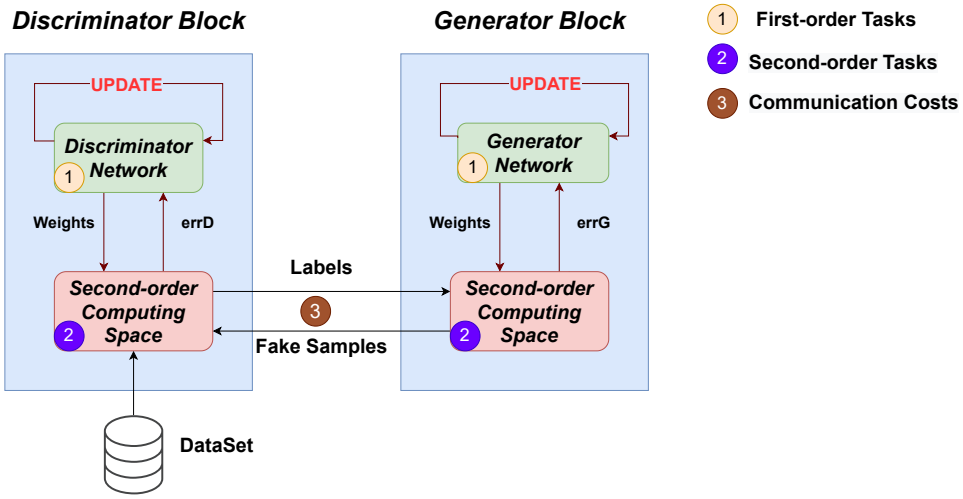


Figure 4.5: The network structure of the DB and GB.

the key we claim BPGAN can reduce the communication cost compared with the data parallelism approach (the detailed analysis on training cost will be presented in next section).

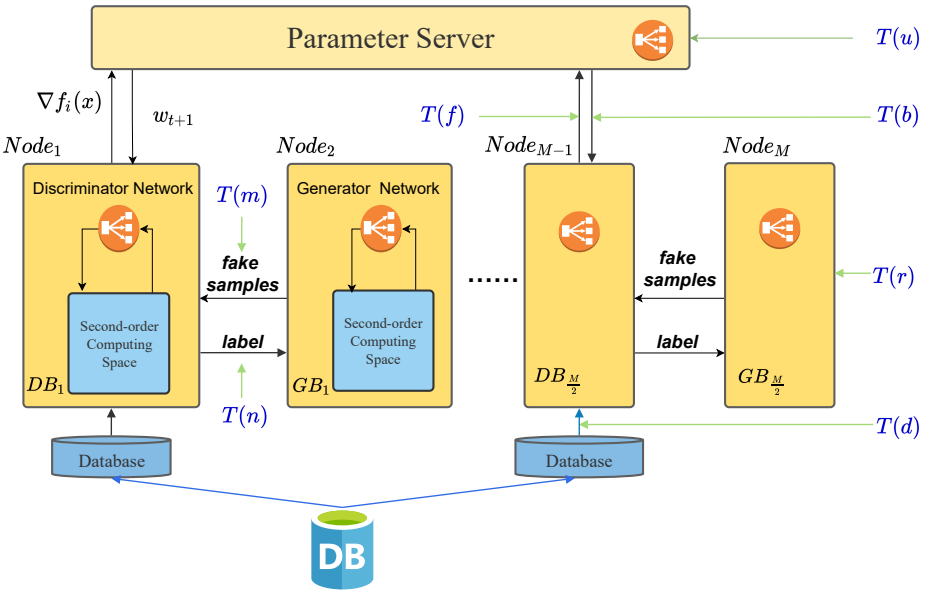


Figure 4.6: The workflow of our proposed method. Our method includes the transmission part and execution part within the training system. The nodes with DB updates the discriminator and the nodes with GB update the generator.

BPGAN splits the GAN framework into two blocks and needs to be run on at least two nodes. Fig.4.6 shows the structure and the execution work of BPGAN when it is deployed on M working nodes (M is an even number). In BPGAN, a subset of training data is loaded to the DB nodes (i.e., the work nodes where the DB block is deployed). After a DB node finishes a round of

local training, it sends the computed gradients (i.e., $\nabla f_i(x)$ in the figure) to the parameter server, where the gradients received from different DB nodes are used to update the model weights of the discriminator network. The updated weights of the discriminator network (i.e., w_{t+1}) are then sent back to the DB nodes. Based on the new model weights, the DB nodes start a new round of local training. In BPGAN, the generator network is only trained locally based on the update of the discriminator network. The advantages of BPGAN are as follows:

1) In BPGAN, a subset of the training data is only loaded to the DB nodes, i.e., half of all nodes. Fig.4.7 show the architecture and the execution flow of the traditional data parallelism approach. In the traditional data parallelism approach, each work node contains a full copy of the GAN framework, including the generator network and the discriminator network, and a subset of the training data has to be loaded to every work node. Therefore, BPGAN saves half of data loading cost.

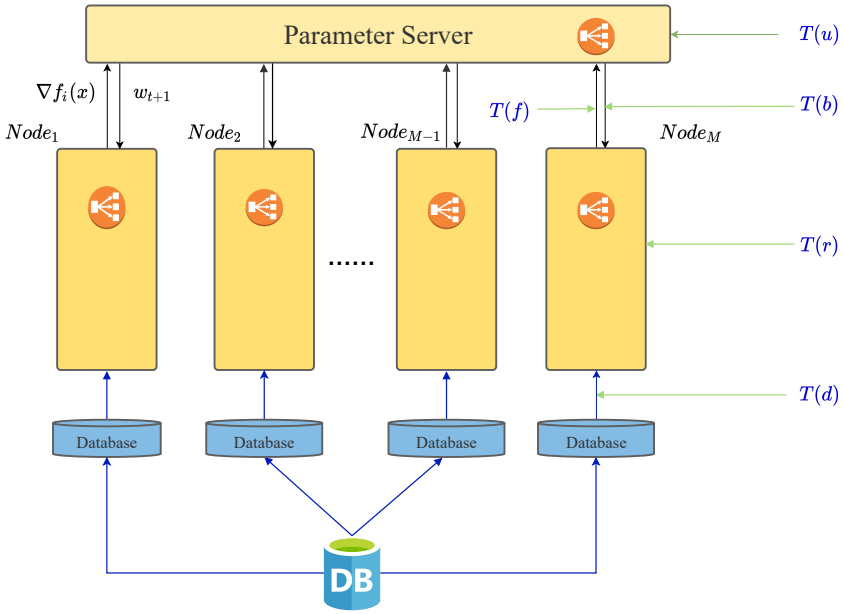


Figure 4.7: The training procedure of Data Parallelism (DP)

2) In the traditional data parallelism approach, the model gradients and weights need to be transmitted between the parameter server and every work node. In BPGAN, the gradients and weights are only transmitted between the parameter server and the DB nodes, which once again saves half of the communication cost.

3) In the traditional data parallelism, a work node has both discriminator and generator and therefore has to train the discriminator network and the generator network in sequence. In BPGAN, however, each node in the node pair calculates the gradients and weights of the discriminator or the generator

Algorithm 2: DB Node processing the DB for each iteration

Data: The training data $Data$, the label L, the discriminator Network $netD$, the error of real sample E_r, the error of fake sample E_f, the error of discriminator E_D, the fake samples $Fake$,

Result: Update the discriminator in DBNode

1 Loading the training data to DBNode $Data.to('DBNode')$;
2 Create label L;
3 Load the discriminator Network $netD(Data).to('DBNode')$;
4 Calculate error E_r;
5 Generate label L and noise and transmit label L to GBNode;
6 Receive the fake samples $Fake$ from GBNode;
7 Calculate the error E_f;
8 Total Error of discriminator $E_D = E_r + E_f$;
9 Update parameters of discriminator Network.

network in parallel since our method allows the DB node to send the labels to the GB node before it is maturely trained. This enables the GB node to start the training of the generator network in advance, and therefore accelerates the training process.

4) In traditional data parallelism, the gradients of both generator and discriminator in every work node are sent to the parameter server for aggregation. BPGAN eases the computation burden on the parameter server since only the gradients of the discriminator network are sent to the parameter server for aggregation in BPGAN.

All the above benefits brought by BPGAN only comes at the price of the light communication cost (i.e., fake samples and labels) in the node pair. In the traditional data parallelism method, the generators have to send the weights and gradients to the parameter server for aggregation. BPGAN transforms this communication into the fake samples and labels in the node pair, which have much less data volume than the weights and gradients of the entire network. Moreover, since the communication of fake samples and labels are transmitted within each node pair, the efficiency of our approach becomes more prominent as the number of nodes increases. Algorithm.2 and 3 show the pseudo-code of training DB and GB on different nodes.

4.3 Theoretical Analysis

In this section, we conduct the theoretical analysis on the training cost, including computation and transmission cost, involved in the traditional data parallelism approach and BPGAN. The analysis shows that BPGAN can efficiently reduce the training cost.

Algorithm 3: GB Node processing the GB for each iteration

Data: The generator network $netG$, the label L, the error of generator E_G, the fake samples $Fake$,

Result: Update the generator in GB Node

1 Create noise;
2 Load the generator network $netG(noise).to('GBNode')$;
3 Receive the label L from DB Node;
4 Calculate the loss of E_G;
5 Update parameters of generator network;
6 Generate the fake sample $Fake$ and transmit the $Fake$ to DB Node;
7 Calculate total error of generator E_G;
8 Update parameters of discriminator network.

The Analysis of Data Parallelism Method

Fig.4.7 shows the training process for the traditional data parallelism. To facilitate the analysis, we define the cost of various tasks as follows, in which $T(d)$, $T(b)$ and $T(f)$ are the communication cost while $T(r)$ and $T(u)$ are the computation cost.

$T(d)$: the time for loading the data to the nodes.
$T(b)$: the time for transmitting the model weights from Parameter Server to the work node.
$T(f)$: the time for transmitting the gradients from the work node to Parameter Server.
$T(r)$: the time for the work node updating the model.
$T(u)$: the time for Parameter Server updating the model.

Then we define the following variables related to the system settings. Note that we assume that in the optimal situation the discriminator network and the generator network have the same volume of model weights and model gradients, so we do not differentiate the weights or the gradients for the discriminator or the generator.

v: transmission speed
D: the volume of the training data
M: the number of work nodes in the system
t_D: the time spent by the discriminator in processing one unit of training data when training the discriminator network
t_G: the time spent by the generator in processing one unit of training data when training the generator network
t_S: the time spent by the parameters server in processing one unit of model data (gradients or weights)
g: the volume of the gradients of the discriminator network or the generator network
ω: the volume of the weights of the discriminator network or the generator network

With the above system settings, we can calculate various training cost involved in the traditional data parallelism.

$T_d = \frac{D}{v}$
$T_b = \frac{M \times (2\omega)}{v}$, where 2ω is for the gradient of both discriminator and the generator network
$T_f = \frac{M \times (2g)}{v}$, where $2g$ is for the gradient of both discriminator and the generator network
$T_r = \frac{D \times (t_D + t_G)}{M}$
$T_u = t_S \times (g + \omega) \times M$

The total training time of traditional data parallelism can then be modelled as:

$$\begin{aligned} T_{DP} &= T_d + T_b + T_r + T_f + T_u \\ &= \frac{D}{v} + \frac{M\omega}{v} + \frac{M \times g}{v} + \frac{D(t_D + t_G)}{M} + t_S(g + \omega)M \end{aligned} \tag{4.1}$$

The Analysis of BPGAN

Fig.4.6 shows the meachanism of the synchronous approach. In addition to the training cost defined above for the data parallelism approach, there are the following two costs in BPGAN.

$T(m)$: The time for transmitting the fake samples from GB node to the DB node.
$T(n)$: The time for transmitting the labels from the DB node to the GB node.

Compared with the data parallelism approach, there are the following differences in BPGAN: a) only half of the work nodes receive the training data; b) only the gradients and weights from the discriminators will be transmitted to and computed in the parameter server; c) the discriminator and the generator can perform the computation in parallel. Therefore, various training costs in BPGAN can be calculated as follows.

$T_d = \frac{D}{v}$
$T_m = \frac{M}{2} \times \frac{F}{v}$, where F is the volume of the fake samples and $\frac{M}{2}$ represents the number of the node pairs in the system
$T_n = \frac{M}{2} \times \frac{L}{v}$, where L is the volume of the labels
$T_b = \frac{M\omega}{v}$
$T_f = \frac{Mg}{v}$
$T_r = \frac{D \times t_D}{\frac{M}{2}}$
$T_u = t_S \times (g + \omega) \times \frac{M}{2}$

The total training time of BPGAN, denoted by T_{BP} is:

$$\begin{aligned} T_{BP} &= T_d + T_b + T_f + T_m + T_n + T_r + T_u \\ &= \frac{D}{v} + \frac{M\omega}{v} + \frac{Mg}{v} + \frac{M}{2v}(F + L) + \frac{Dt_D}{\frac{M}{2}} + t_S(g + \omega)\frac{M}{2} \end{aligned} \tag{4.2}$$

We assume that in the optimal situation the computing speed for the

discriminator network and the generator network is the same, i.e., $t_D \approx t_G$. We perform Eq.4.1 subtract Eq.4.2:

$$\begin{aligned}
T_{DP} - T_{BP} &= \frac{D}{v} + \frac{2M\omega}{v} + \frac{2Mg}{v} + \frac{D(t_D + t_G)}{M} + t_S(g+\omega)M \\
&\quad - [\frac{D}{v} + \frac{M\omega}{v} + \frac{Mg}{v} + \frac{M}{2v}(F+L) + \frac{Dt_D}{\frac{M}{2}} + t_S(g+\omega)\frac{M}{2}] \\
&= \frac{M\omega}{v} + \frac{Mg}{v} + \frac{D}{M}(t_D + t_G - 2t_D) + t_S(g+\omega)\frac{M}{2} - \frac{M}{2v}(F+L) \\
&= \frac{M\omega}{v} + \frac{Mg}{v} + t_S(g+\omega)\frac{M}{2} - \frac{M}{2v}(F+L) \\
&= \frac{M}{v}(\omega + g - \frac{F+L}{2}) + t_S(g+\omega)\frac{M}{2}
\end{aligned} \tag{4.3}$$

In Eq. 4.3, ω and g are both higher-dimensional data compared with the fake samples (F), which are for example images, and the labels (L), which are scalar values. Therefore we can conclude that $T_{DP} > T_{BP}$, which shows that BPGAN reduces the total training time. Moreover, as M increases and the models become bigger, the cost difference becomes bigger, which indicates that the advantage of BPGAN over the traditional data parallelism approach is more prominent.

4.4 Experiments

BPGAN reduces the training time of GAN by partitioning the discriminator and the generator and reducing the transmission cost. In the experiments, we implement four types of GAN structures with BPGAN and conduct the experiments on two realistic datasets. 1080Ti GPUs are used to train the networks in the experiments. Some effective metrics have been proposed to replace human evaluation.

4.4.1 GAN Structure

BPGAN allows combining various GAN components to form new GAN variations. In the experiments, we combine two standard GAN models (DCGAN and WGAN) to form two GAN variants, called DCWGAN and WDCGAN.

DCGAN is an extension to the standard GAN, except that it explicitly uses the convolutional and convolutional-transpose layers in the discriminator and the generator, respectively. The discriminator is made up of strided convolution layers, batch norm layers, and LeakyReLU activations. The input is 3x64x64 input images, and the output is a scalar probability of the input is from the real data distribution. The generator is comprised of convolutional-transpose layers, batch norm layers, and ReLU activations. The WGAN is

an extension to the GAN, which improves the training stability and provides a loss function that correlates with the quality of generated images. WGAN approximates the distribution of a given training dataset better. WGAN uses a linear activation function in the output layer of the critic model instead of the sigmoid function. It constrains the weights of the critic model to a limited range after each batch update.

We create two GAN variations named DCWGAN and WDCGAN to evaluate the performance of BPGAN. DCWGAN adopts the generator from DCGAN with the discriminator from WGAN. The WDCGAN adopts the generator from WGAN and the discriminator from DCGAN. These experiments are conducted to demonstrate the generality and robustness of BPGAN.

4.4.2 Datasets

We conducted the extensive experiments on two standard datasets: CIFAR-10 [106], LFW [84]. The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in the random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class. Fig. 4.8 shows the classes in the dataset, as well as 10 random images from each class.

LFW is a dataset of face photographs created for studying the problem of unconstrained face recognition. The data set contains more than 13,000 images of faces collected from the web. Each face has been labelled with the name of the pictured person. 1680 of the people pictured have two or more distinct photos in the data set. There are now four different sets of LFW images including the original and three different types of "aligned" images. The aligned images include "funneled images" (ICCV 2007), LFW-a, which uses an unpublished method of alignment, and "deep funneled" images (NIPS 2012).

4.4.3 Evaluating DCGAN

In this subsection, we parallelize DCGAN by our BPGAN (called BP-DCGAN) and by the traditional data parallelism approach (called DP-DCGAN), and conducted the extensive experiments on different datasets. Fig.4.9(a) and Fig.4.9(b) compare BP-DCGAN with DP-DCGAN, in which the CIFAR-10 and LFW datasets are used and the results are generated by 300, 600 and 1k epochs. The results show that BP-DCGAN takes much less the training time

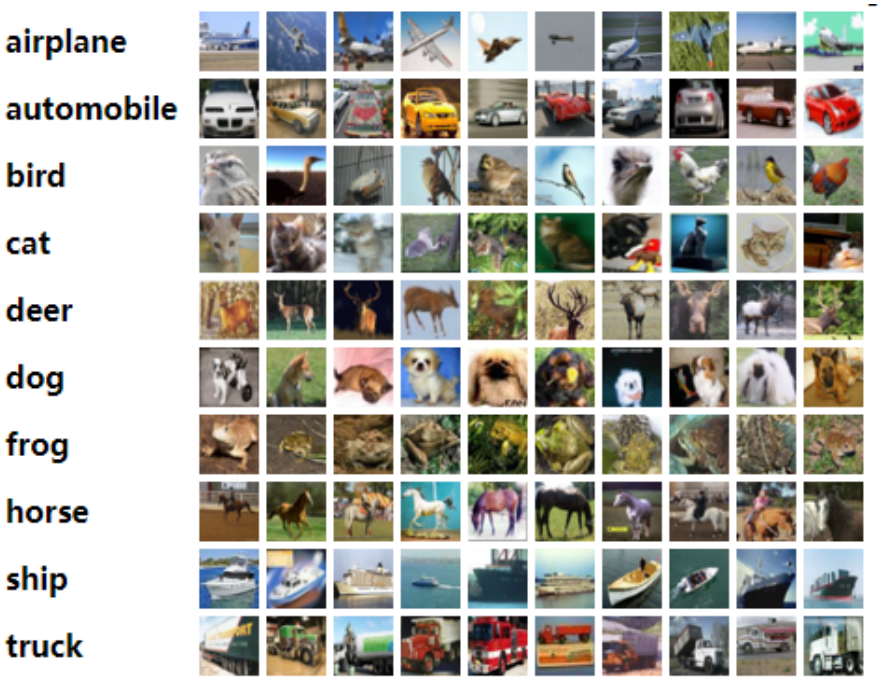
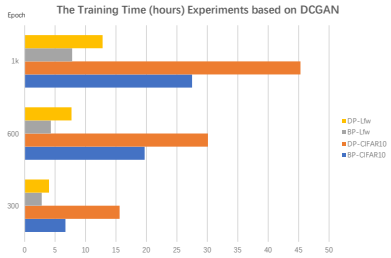


Figure 4.8: The CIFAR-10 Datasets

than DP-DCGAN.



Training Iterations	DCGAN CIFAR-10	DCGAN LFW
300	57%	30%
600	34.6%	22%
1k	24%	20%

Figure 4.9: a) The Training time of DCGAN parallelized by BPGAN and DP; the datasets are CIFAR-10 and LFW (left). b) The reduction percentage in training time achieved by BPGAN on CIFAR-10 and LFW (right).

Fig.4.9a shows that our method can reduce the total training time by nearly 1/3. Theoretically, our approach can reduce the gradient transmission cost by half. However, in practice, the discriminators and the generators contain different amounts of gradients. Therefore, the achieved improvement will be less than the theoretical prediction. Fig.4.9 also displays that when we increase the training iterations (e.g., from 300 epochs to 600 epochs), the increase in the training time of the BP-DCGAN framework is less than that of DP-DCGAN. This result indicates that BPGAN has better robustness than the DP approach. Fig.4.9(b) shows that the improvements that BP-DCGAN

achieves on CIFAR-10 and LFW. It can be seen that BP-DCGAN reduces the training time by higher percentages on CIFAR-10 than on LFW. Moreover, the improvement diminishes as the training iteration increases.

Table 4.1: Evaluating BP-DCGAN and DP-DCGAN on CIFAR-10 and LFW

Training Iterations (Epochs)	DCGAN CIFAR-10				DCGAN LFW			
	BP		DP		BP		DP	
	IS	FID	IS	FID	IS	FID	IS	FID
300	2.10	221	2.30	208	2.10	78.6	1.97	83.2
600	4.10	54.2	3.90	73	2.60	59.6	2.70	59.6
1k	4.40	54.5	4.20	55.4	2.80	54.3	2.60	58.6

Table 4.1 presents the quality of the samples generated by BP-DCGAN and DP-DCGAN in terms of the metrics IS and FID. It can be seen from the table that BP-DCGAN achieves the similar performance as DP-DCGAN. These results suggest that BPGAN can reduce the training time without sacrificing the quality of the generated samples (except one or two specific disturbance points caused by the random factors). The error of IS can be maintained within 10%, and the error of FID be within 5%).

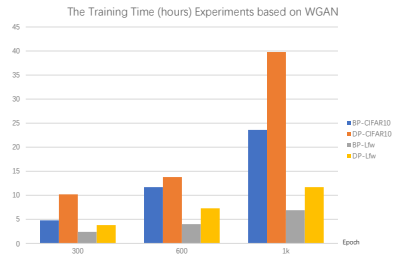
4.4.4 Evaluating WGAN

In this subsection, we parallelize WGAN by our BPGAN (called BP-WGAN) and the traditional DP (called DP-WGAN). WGANs are used to improve the stability and solve model collapse problem in the GAN training. Fig.4.10 compares the training time spent by BP-WGAN and DP-WGAN. The results once again show that BPGAN-WGAN reduce the training time effectively. Moreover, different from the results of BP-DCGAN, the improvements achieved by BP-WGAN are similar on CIFAR-10 and LFW (see Fig.4.10(b)). Also, as the training iteration increases, the improvement decreases at a slower pace compared with BP-DCGAN.

Table 4.2: Evaluation of the samples generated by BP-WGAN and DP-WGAN on CIFAR-10 and LFW

Training Iterations (Epochs)	WGAN CIFAR-10				WGAN LFW			
	BP		DP		BP		DP	
	IS	FID	IS	FID	IS	FID	IS	FID
300	2.8	116	2.9	118	2.1	88.9	2.2	92.3
600	3.2	100.7	2.6	127.5	2.4	70.7	2.5	78.3
1k	3.5	98.4	3.3	99.0	2.6	62.6	2.7	59.6

Table 4.2 shows the quality of the samples generated by BP-WGAN and



Training Iterations	WGAN CIFAR-10	WGAN LFW
300	52.9%	36.8%
600	45.2%	44.9%
1k	40.7%	41%

Figure 4.10: a) The training cost of BP-WGAN and DP-WGAN on CIFAR-10 and LFW (left). b) The improvement of BP-WGAN over DP-WGAN (right).

DP-WGAN. The difference in terms of IS is less than 10%. In terms of FID, the difference of samples is still small, Although the difference is a bit big (more than 25%) when the number of epochs is 600, the difference converges to a very small value (0.6%) when the number of epochs increases to 1K. These results suggest that BP-WGAN can reduce the training time while keeping the quality of generated samples.

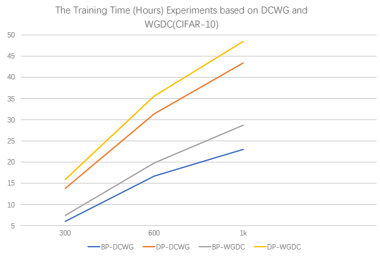
4.4.5 Evaluating the hybrid GAN variants

To show the generality of our framework, we created two hybrid GAN networks: DCWGAN and WDCGAN. DCWGAN is composed of a generator from DCGAN and a discriminator from WGAN, while in WDCGAN the generator is from WGAN and the discriminator from DCGAN. We then parallelize the two GAN variants using BPGAN and DP. The resulting versions are called BP-DCWGAN, DP-DCWGAN, BP-WDCGAN and DP-WDCGAN. Fig. 4.11(a) shows the training cost of these four variants as the number of epochs increases. The results once again show that BPGAN can deliver lower training cost than DP.

Table 4.3 shows the quality of the samples generated by the four GAN variants. Once again, the differences in terms of IS and FID between BPGAN and DP are small. The results indicate that BPGAN has excellent generalization ability and can be used to parallelize the hybrid GAN variants.

4.5 Conclusion

In this work, we propose BPGAN to build a novel parallelism framework. We carefully designed the architecture and crafted the execution workflow of the GAN training in BPGAN. As the result, the proposed BPGAN can be used to parallelize various GAN variants and reduce their training time on different datasets. At the same time, the quality of the samples generated by



Training Iterations	DCWG CIFAR1-0	WGDC CIFAR-10
300	56.5%	53.5%
600	46.8%	44.4%
1k	47%	34%

Figure 4.11: a) The training cost of BP and DP using Created GAN on CIFAR-10 (left). b) The improvements of BPGAN compared to the DP methods of the Training (right).

Table 4.3: Evaluating the quality of samples generated under BPGAN and DP on CIFAR-10

Training Iterations (Epochs)	DCWGAN CIFAR-10				WGDCGAN CIFAR-10			
	BP		DP		BP		DP	
	IS	FID	IS	FID	IS	FID	IS	FID
300	2.90	166.7	2.60	194.2	2.70	125	2.80	131
600	3.10	87.0	3.00	127.1	3.40	97.0	3.30	99.0
1k	3.40	68.0	3.30	65.0	3.80	74.2	3.70	79.4

BPGAN is not sacrificed. Although BPGAN offers an efficient new perspective to accelerate the GAN training, there is still the space to research further, such as optimizing the execution efficiency within the work nodes, which will be our future work. Note that our method will not show obvious benefit for the network with very big bandwidth. Our approach aims to speed up the training by optimizing the communication cost of the gradients and the weights between a working node and the parameter server. If the bandwidth is very big, there is no much space for optimization because the transmission cost is already very small. However, in reality, the network bandwidth of mobile devices is still limited and precious nowadays. Therefore, the realistic networking scenarios today can benefit significantly from the method proposed in this work.

Chapter 5

PrivacyGAN: A Lightweight, Privacy-aware GAN Framework

With the development of computer vision and artificial intelligence, the personal biological information has been regarded as a confidential resource. Attackers can invade users' privacy by spreading, diffusing, and aggrieving the private data. However, the service suppliers also need users' data to improve the service, such as updating the recommendation system, analyzing the users' habits, or supporting the marketing research. Therefore, a legal approach to accessing personal data from clients (e.g., Mobiles, Laptops) should be considered. In deep learning, generative models are widely used to simulate the original data and receive great success in producing images and videos. However, in practice, we will face the following problems: 1) the servers (or the suppliers) are not allowed to access the personal data directly (e.g., the cloud server should not access the data in mobile devices without special agreement); 2) the clients usually have limited personal information, training on a small amount of data can easily over-fit the models; 3) the clients may have limited computing resources (such as in mobile devices), which means that complex learning algorithms and models cannot be deployed on local devices. In order to address these issues, we propose a lightweight, privacy-aware GAN framework called PrivacyGAN to generate the samples similar to the sensitive data in the clients. We first design a Teacher-Student model to solve the over-fitting problem caused by the limited training data in the clients. Next, we propose a new deployment strategy to tackle the private issue. Finally, we adopt a novel objective function which we call Joint Restraint Learning Function (JRLF) to avoid the issue of "Simply Accepting (SA)". We conduct extensive experiments on real-world datasets, including CIFAR-10, lFW, and LSUN. The experimental results show that the model in the server can synthesize the clients' data efficiently without directly accessing the clients' data.

5.1 Introduction

In this chapter, we aim to generate the samples for the distributed architecture illustrated in Fig.5.1, where a collection of clients are connected to a server. In particular, we assume that the clients are resource-constrained mobile devices, such as mobile phones. Many systems in the real world has this distributed architecture, such as a cloud system. Some distributed machine learning paradigms are also run on such architecture, such as Federated Learning [137], where the clients train the local models on their local data and send the local models to the server while the server aggregates the local models from the clients to produce the global model. Data is a critical resource in developing artificial intelligence and computer vision system. Sensitive data include personal data containing private information, such as GPS location, weather forecasts, landscape photos, family or selfie photos, and the application's service time. The server can quickly acquire personal information such as home address, name and the surroundings by analyzing these personal data. Therefore, people usually do not want to share their private data. It is also against the law for the companies to acquire the clients' data with consent. The work in [162] demonstrates the importance of the high quality data. However, in real applications, the data in the clients may be too limited to perform effective local training (e.g., in the scenario of few-shot learning). One approach to solving the problem is to generate the data for the clients.

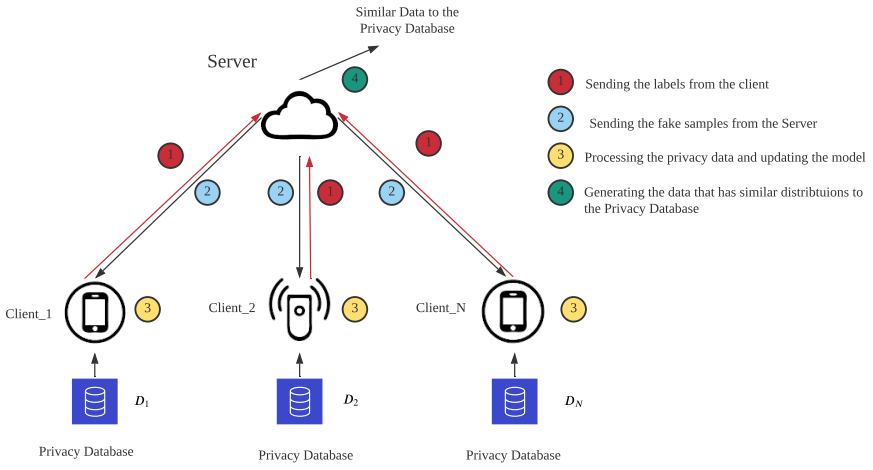


Figure 5.1: The structure of PrivacyGAN. The private data are only accessed by the clients. The training process is achieved by transmitting fake samples and labels between the server and the clients. The server generates the fake samples that mimic the distributions of the private data in clients

There are different approaches to generating data. Data Augmentation (DA) has been shown to be a useful regularization technique to increase both

quantity and the diverisity of training data. However, many data augmentation methods, such as translation, flip and rotation, are designed specifically for images. Moreover, the complex data augmentation approaches [208][97][28] are often computation-intensive. Table [28] shows that AutoAugment, a data augmentation technique, requires thousands of GPU hours even in the reduced data and network settings. Although the work in [121] proposed an efficient search method called Fast AutoAugement to improve the generalization performance of a given network while searching the augmentation policies significantly faster than AutoAugment, it still requires at least 1.5 hours, which are not bearable for mobile devices.

Dataset	AutoAug [28]	Fast AutoAug [121]
CIFAR-10	5000	3.5
SVHN	1000	1.5
ImageNet	15000	450

Table 5.1: The GPU hours spent by the method in [121] and the method in [28]. NVIDIA Tesla V100 is used in [121] while Tesla P100 is used in [28]. The datasets include CIFAR-10, Street View House Number (SVHN) and ImageNet.

Data privacy is another issue in the distributed architecture considered in this chapter, especially when we assume that the clients are mainly mobile devices. Private personal data, which contains the most valuable information, should be strictly guarded. Otherwise, there may arise security concerns. For example, the criminals may be able to bypass a face recognition system to attack a financial system, or put people's lives in danger with the development of auto-driving.

This chapter aims to explore an efficient lightweight method to generate high-quality samples for the private data in the devices with limited computation capacity. To achieve the goal, we must address several issues in the training process. Firstly, since the data in the clients are limited, the overfitting problem can easily occur. In order to solve this problem, we adopt the teacher-student model from the Few-Shot Learning field, which can be applied to generate the samples with only a limited amount of data that include the supervised information for the target distribution. The teacher-student model is a type of knowledge distillation technique, which trains a student model residing in the client to gain the similar performance to the teacher model in the server. Secondly, we need to train the network without acquiring sensitive personal data directly. The traditional methods for addressing the privacy issue include encrypting the data and transmitting them in a secure way [137][152][49][186].

However, the previous approaches described above have the following prob-

lems:

1) The Few-Shot Learning targets at generating a specific class of data. In our work, we need a GAN framework that can generate the resembling samples for any type of data.

2) The traditional security-related operations such as encryption and feature compression increase the complexity and the computation expenses. These extra payloads may not be suitable for the resource-constrained clients.

3) The student-model often heavily depends on the teacher model. However, it may not be appropriate in the scenario assumed in this chapter. In this chapter, different clients may have different personalized data. If the student models in all clients simply accept the knowledge shared by the teacher model, the student models will not generate the samples following the unique distribution in a client.

4) Although GAN has achieved impressive success in synthesizing data, the drawback is that it requires plenty of training data and intensive computation resources and its training stability cannot be guaranteed. This is a challenging issue in our scenario since the clients possess limited labelled data, and the resource-constrained devices have limited computation and communication capability.

To overcome these difficulties, we propose a novel GAN framework to generate the samples that resemble the private data in local clients without accessing the data directly. The proposed methodology contains three key components: 1) A teacher-student GAN model is proposed, in which the lightweight student GAN resides in the clients while the teacher GAN is located and trained in the server. The student GAN receives the gradients and weights provided by the teacher GAN. This technique can generate high-quality samples even with a limited quantity of training data. 2) A novel deployment mechanism allows the model on the server to extract the features and synthesize the distributions without accessing the private data. We transform the sensitive data into a fake sample and labels by a local discriminator. Considering the limited resource capability (computation and transmission cost) and model performance, we do not use encryption algorithms or compression approaches. These approaches usually increase the training load and affect the precision of the output, and 3) A joint restraint learning function is proposed to solve the "Simply Accepting" issue. This technique is applied to constraint the extent to which the student model depends on the teacher model.

Since the clients have limited data, we cannot simply deploy a GAN framework on the clients to generate the samples that mimic the clients' data. The limited amount of training data usually leads to instability and over-fitting. Therefore, we propose to use a global auxiliary Teacher-Student model to address the issue.

First, we use the global training data and the global GAN model to obtain the teacher model, which is used to guide the learning for the student model. Then, each client is assigned with a student GAN, which is responsible for training the student model with the limited sensitive data. The teacher GAN is a global model which is trained on a large dataset on the server. The well-trained teacher, also called a global discriminator, is used to share the knowledge and optimize the training outcome of the student model. The auxiliary teacher-student model can improve the quality of generated samples while avoiding the over-fitting problem.

Next, the server should train the network without contacting the personal data in the clients. Considering the discriminators and generators from GANs are relatively independent, they have their separate computation convolution layers. Moreover, one network (discriminator or generator) is updated through the labels or fake samples generated by the other network (generator or discriminator). Therefore, we proposed a novel deployment mechanism to protect data privacy by locating different components on different devices. The idea of the proposed method is that the discriminator networks reside on clients, while the generator networks on the server. The labels and fake samples generated by the networks are used for updating the networks through iterations. The advantage of doing so is that the discriminators can be made unavailable to the outside world and only the labels are accessible by the server. Compared to the traditional approaches, the clients in our method only undertake the computations involved in the discriminators. They do not need to transmit the heavy data (e.g., model gradients and weights) to the server, which is friendly to the resource-limited clients. Moreover, each client has its independent student model to generate the samples with distinguishing distributions. We deploy the global discriminator on the server. As for the student GAN, we deploy the discriminator in the clients and the generator in the server separately. In our mechanism, the discriminator in the student model is updated upon receiving the outcome (fake samples) from the matching generator in the server, while the generator in the student model is updated based on the labels sent by the discriminator. Our approach enables the server to capture the embedded features of the targeted data in the clients without accessing them.

Finally, we propose a JRLF to solve the SA problem. The SA problem means that the student fully believes in the teacher and simply accept all of its knowledge. Although the teacher GAN captures the features from the global dataset, the student GAN aims to synthesize the data in the local client. Believing in the teacher model ultimately may not capture the uniqueness of the data distribution in each local client. The traditional student model lacks judgement and cannot sift the gradients and weights of the shared lay-

ers. Simply accepting the parameters of the teacher model reduces the space distance between the samples generated by the student model and the global samples generated by the teacher model, which does not meet the requirements of generating the data for the unique clients. The essence of the problem is that our student and teacher models may target different data distributions. Therefore, we re-design a novel objective function (i.e., JRLF) for the training to constrain the impact of the information provided by the teacher model. In other words, the JRLF function is used to limit the training ability of the teacher given that the teacher does not fully represent the data distributions in the clients.

The main contributions of this chapter are summarized as follows:

1.We propose a Teacher-Student model consisting of a student GAN and a teacher GAN. The teacher shares the knowledge with the student to improve the generalization and robustness of the training. The proposed model a) improves the original training condition and b) generates the high quality samples without the over-fitting problem.

2.We propose an Decentralized Deployment method to address the problem that the server does not have the permission to access clients' data.

3.We propose a Joint Restraint Learning Function to limit the information that the student learns from the teacher. The features from the teacher model are not reliable, even sometimes harmful. Therefore, the strategy is proposed to control the level of the student model's dependency on the teacher's knowledge.

5.2 PrivacyGAN

5.2.1 The Teacher-Student Model and its Deployment Strategy

The limited training data usually result in over-fitting the models, which means the models lack the generalization. To solve this problem, we designed a global teacher GAN model and a local student GAN model. The teacher GAN model consists of a global discriminator D_{global} and a global generator G_{global}, which are both deployed on the server. The student GAN model includes a set of generators $G_{local} = [G_{local1}, G_{local2}, ..., G_{localN}]$ and a set of matching discriminators $D_{local} = [D_{local1}, D_{local2}, ..., D_{localN}]$ (N is the number of clients). (G_{locali}, D_{locali}) is the student GAN model for client i. D_{locali} is deployed and trained on the local clients while the generator G_{locali} is on the server. The student GAN model will receive the knowledge from the teacher model to overcome the issues of over-fitting and low-diversity. Fig.5.2 illustrates the deployment of our teacher-student model in PrivacyGAN. The teacher model (includes D_{global} and G_{global}) is used to extract the distributions of the global

dataset and train a mature global discriminator on the server. An independent student GAN model (i.e., (G_{locali}, D_{locali})) is allocated to each client. The generators from all the student models (i.e., $G_{locali}, 1 \leq i \leq N$) are deployed on the server, while the discriminator of a student GAN model (i.e., D_{locali}) is deployed on the corresponding client. The server is responsible for training the teacher GAN model (i.e., both D_{global} and G_{global}) and the generators (i.e., G_{locali}) from the student models. Each client is responsible for training the discriminator (i.e., D_{locali}) of its student model. D_{locali}, G_{locali} and D_{global} are updated together to reach a Nash Equilibrium.

In the training process, we first train the student GAN model using the local data in the clients. The fake samples generated by G_{locali} on the server are transmitted to the clients, and the labels generated by D_{locali} are sent to the server. D_{locali} and G_{locali} are then updated by the fake samples and the labels. For preserving the authoritativeness, the parameters of the teacher GAN model are fixed during the training, which means the teacher model does not modify its parameters, but only guides the training of the student GAN model. Next, D_{global} is trained using the global datasets on the server as the teacher model.

In each training round, the global discriminator D_{global} optimizes the G_{locali} to reduce the distribution distance between the generated samples and the global dataset. Then, client i receives the fake samples generated by G_{locali} and uses local training data to train D_{locali}, and transmits the predicted labels to the server. Next, G_{locali} is updated by the server based on the labels received from client i. Meanwhile, the global discriminator optimizes G_{locali} to reduce the space distance between the generated samples and the global dataset. It needs to be emphasized that the teacher model is not allowed to receive the gradients from G_{locali} due to the security requirement, which may affect the performance of the teacher model. In other words, D_{global}, D_{locali} and G_{locali} will be trained jointly to reach the Nash Equilibrium. Once the training ends, G_{locali} can be utilized to generate the samples for client i. The generated samples can also be stored for further updates and investigations.

5.2.2 The Advantages of the Decentralized Deployment Strategy

As described in the previous subsection, the components in our teacher-student model are decentralized and deployed on different clients and the server. The advantages of the deployment strategy are as follows:

First, it helps protect the privacy of the data in the clients, which is a key objective of PrivacyGAN.

Second, the deployment strategy reduces the computation load of the resource-limited clients while improving the generalization and generality of

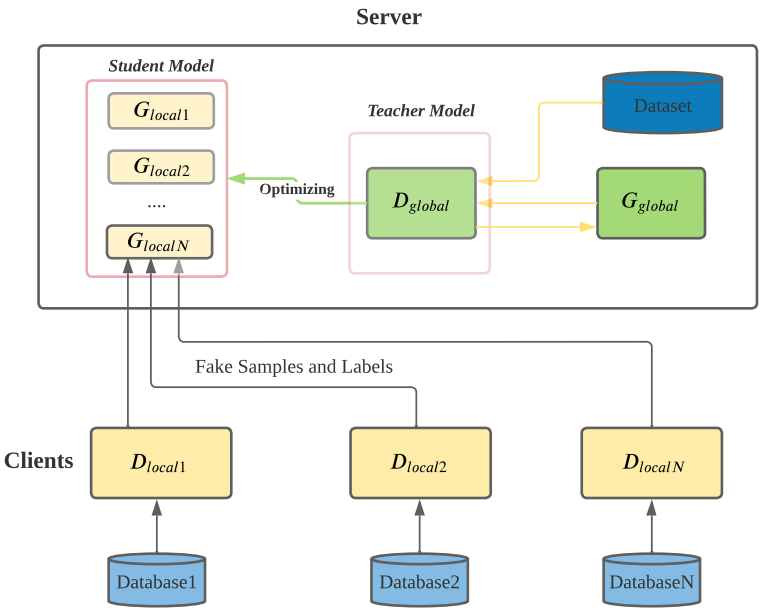


Figure 5.2: The workflow of PrivacyGAN. The global discriminator will be trained on server and then guide student models (including G_{local} and D_{local}) to learn distributions from limited private samples.

the training. In the deployment strategy, the generators of the clients are deployed in the server.

Third, there are no exchanges of model gradients and weights between the clients and the server, but only the fake samples and the labels, which reduces the communication cost significantly. Suppose we simply apply the traditional method of deploying the teacher and student GAN on the server and the clients. The teacher GAN on the server has to transmit its gradients and weights to the clients. The clients then train the student GAN to generate fake samples and send the fake samples to the server. When implementing DCGAN using Pytorch and the deep learning networks, the size of the model weights of the discriminator is nearly 10.6MB, which means in each iteration at least 10.6MB data have to be transmitted from the server to each of the client. In our deployment strategy, we only need to communicate the labels and the fake samples. The label is the scalar value, whose size is minimal. Assume a fake sample is an image with RGB. Its size ranges from 0.3MB to 0.8MB. Therefore, our deployment strategy reduces the communication cost significantly. The size of the model weights and fake image are from the weight file extracted from the trained DCGAN based on pytorch.

5.2.3 The Joint Restraint Learning Function

In our scenario, the teacher model and the student models target different datasets in the training. This introduces the SA problem if the students have the full confidence in the teacher and fully accept the knowledge shared from the teacher model. Since the SA problem may jeopardize the model performance or even lead the student model to abandon its distinction, we propose the JRLF to limit the optimization weights provided by the teacher model.

In the training process of D_{global}, D_{localN} and G_{localN}, we first update D_{locali} and G_{locali}. As discussed above, the training aims to optimize $V(D_{localN}, G_{localN})$ as defined below [54], where G is a differentiable function represented by a multiplayer perceptron. $D(x)$ represents the probability that x comes from the training data rather than P_g (P_g is the generated samples distributions).

$$\min_G \max_D V(D_{localN}, G_{localN}) = \mathbb{E}_{x\sim p_{\text{data}}\ (x)}[\log D_{localN}(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D_{localN}(G_{localN}(z)))]$$

After we have the mature G_{localN} and D_{localN} based on the training on the private data from the clients, we focus on updating G_{localN} with a fixed discriminator D_{localN} to initialize the parameters, which can be formalized as:

$$\min_G V(D_{localN}, G_{localN}) = \mathbb{E}_{x\sim p_{\text{data}}\ (x)}[D_{localN}(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D_{localN}(G_{localN}(z)))] \tag{5.1}$$

Next, we update $V(D_{global}, G_{localN})$. As described above, D_{global} is not allowed to be updated. Therefore, the training process is to minimize $V(D_{global}, G_{localN})$:

$$\min_G V(D_{global}, G_{localN}) = \mathbb{E}_{x\sim p_{\text{data}}\ (x)}[\log D_{global}(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D_{global}(G_{localN}(z)))] \tag{5.2}$$

In practice, we sample $p_{data(x)}$ from $S_{data} = [x_1, x_2, ..., x_m]$ (S_{data} is the distributions of training data) instead of performing the integral operation, and $p_G(x)$ (generated samples) from $S_G = [x_{G1}, x_{G2}, ..., x_{Gm}]$ (S_G is the distributions of generated samples). Therefore, the new objective function of minimizing $V(D,G)$ can be formulated as:

$$\min_G \max_D V(D,G) = \frac{1}{m}\sum_{i=1}^{m} \log D(x_i) + \frac{1}{m}\sum_{i=1}^{m} \log(1 - D(G(z_i))) \tag{5.3}$$

When we update the generator, the first item in Eq.5.3 is irrelevant to the generator. Therefore, the training of the generator can be simplified as:

$$V = \mathbb{E}_{z\sim P_z(z)}[\log(1-D(G(z)))] \tag{5.4}$$

In practice, however, the value of $D(x)$ is between 0 to 1, and the gradient of $log(1-D(x))$ at the beginning is relatively small, which lead to a very slow training process. Therefore, an improved loss is proposed as follows to solve the problem.

$$V = \mathbb{E}_{z\sim P_z(z)}[-\log(D(G(z)))] \tag{5.5}$$

In Eq.5.5, the training starts faster and then becomes slower at the later stage, which is more consistent with the general trend of the training. Moreover, the improved loss function is equivalent to the error of the data generated by the generator as a positive sample of the discriminator. In summary, the improved objective function can be implemented as:

$$\min_G \max_D V(D,G) = \frac{1}{m}\sum_{i=1}^{m} \log D(x_i) + \frac{1}{m}\sum_{i=1}^{m}[-\log(D(G(z_i)))] \tag{5.6}$$

Therefore, the training of discriminator and generator can be formalized as follows:

$$\max_D V(D,G) = \frac{1}{m}\sum_{i=1}^{m} \log D(x_i) + \frac{1}{m}\sum_{i=1}^{m}[-\log(D(G(z_i)))] \tag{5.7}$$

$$\min_G V(D,G) = \frac{1}{m}\sum_{i=1}^{m}[-\log(D(G(z_i)))] \tag{5.8}$$

The first term in Eq.5.7 means the error of the real data as a positive sample of the discriminator, and the second term means the error of the real data generated as a positive sample of the discriminator. The term in Eq.5.8 means the error of fake data generated by the generator as a positive sample of the discriminator. Combining Eq.5.6 and Eq.5.8, we add the error of the fake samples generated by G_{locali} as the positive samples of D_{global} to the original function. We also introduce two hyper-parameters α and β, subject to $\alpha+\beta=1$ and $\alpha,\beta>0$, as the weight of the students accepting the teacher's knowledge. Therefore, the new objective function, called the Joint Restraint Learning Function, to train G_{locali} on the server has the following form:

$$\min_G V(D,G) = \alpha \frac{1}{m}\sum_{i=1}^{m}[-\log(D_{locali}(G_{locali}(z_i)))] + \beta \frac{1}{m}\sum_{i=1}^{m}[-\log(D_{global}(G_{locali}(z_i)))]$$
$$= \alpha ErrorD_{locali} + \beta ErrorD_{global} \tag{5.9}$$

The first term $\alpha ErrorD_{locali}$ in Eq.5.9 means the error of the fake samples generated by G_{locali} as the positive samples of the D_{locali}, and the second item means the error of the fake samples generated by G_{locali} as the positive samples of D_{global}. α regulates the contribution provided by D_{locali} while β regulates the contribution provided by D_{global} . We can change the value of α and β to generate the required outcome through the experiments. In addition, when $\alpha = 1$, the student model will reject the knowledge, and the framework then recedes into a standard GAN.

5.3 Experiments

5.3.1 Datasets

Considering the personal data in clients are unlabeled, we begin by conducting the experiment on three standard datasets: CIFAR-10 [106], LFW [84] and LSUN [214] dataset, instead of on the Few-Shot-Learning datasets such as CIFAR-FS, Omniglot and ImageNet-1k. The Large-scale Scene Understanding (LSUN) classification dataset contains 10 scene categories, such as dining room, bedroom, chicken, outdoor church, etc. Each category includes a huge number of images for training, ranging from around 120,000 to 3,000,000. The validation data consists of 300 photos, and the test data has 1000 images for each category. For small datasets, i.e., CIFAR-10 and LFW, we pick the images from the original dataset at the ratio of 1:10 to construct a personal dataset for clients. For the large dataset, i.e., LSUN dining room dataset, we choose the images from the original dataset at the ratios of 1:10 and 1:100 to create two personal datasets for clients. Table.5.2 list the datasets used in the experiments.

Table 5.2: Local training datasets for different original datasets

Datasets	Original Datasets	Personal Datasets	
		1:100	1:10
CIFAR-10	60k	\	6k
Lfw	10k	\	1k
LSUN Dining Room	650k	6.5k	65k

5.3.2 Evaluation

We implemented DCGAN [155] using our PrivacyGAN framework. For the fair comparison, all training networks in DCGAN and WGAN-GP use the parameters similar as those in [155] and [57]: 1) the length of a latent vector is 100; 2) the learning rate is 1e-4 for both generator and discriminator, an Adam optimizer is used with β_1=0 and β_2=0.99 ; 3) the weights are randomly initialized from a normal distribution N $(0, 0.02)$; 4) the feature map size is 64 in both generator and discriminator. We use IS [160] and FID as the evaluation metrics.

CIFAR-10

We compare PrivacyGAN under different hyper-parameters with the standard GAN on CIFAR-10. As shown in Table.4.2, the adequate training data (the results from the full set of CIFAR-10 samples) can improve the model performance noticeably.

Table 5.3: The quality of the samples (From α=0.9 to α=0.5) generated by DCGAN and PrivacyGAN training on a full set of CIFAR-10 samples (60K) and a local dataset (6k)

	DCGAN CIFAR-10 (60K)	DCGAN CIFAR-10 Local Dataset	PrivacyGAN CIFAR-10 Local Dataset (6k) with different values of the α				
			0.9	0.8	0.7	0.6	0.5
IS	4.2	3.1	3.2	3.4	3.7	3.7	3.8
FID	57	107	102	90	69	81	79

Table 5.4: The quality of the samples (From α= 0.4 to α=0.1) generated by DCGAN and PrivacyGAN training on a full set of CIFAR-10 samples (60K) and a local dataset (6k)

	DCGAN CIFAR-10 (60K)	DCGAN CIFAR-10 Local Dataset	PrivacyGAN CIFAR-10 Local Dataset (6k) with different values of the α			
			0.4	0.3	0.2	0.1
IS	4.2	3.1	4.0	3.7	3.8	3.5
FID	57	107	79	62	69	72

We conducted a series of experiments with different values of α. The results are also listed in Table.5.3 and Table.5.4. The results demonstrate that our technique improves the performance of IS and FID. As α increases, the student model becomes increasingly dominant, and the teacher model

results in the improvement in solving the over-fitting problem. The teacher model prompts the generator to learn the diverse features in the global dataset. When α decreases, the student model can generate the samples with more uniqueness. Comparing the trend curve of IS and FID (shown in Fig.5.3), we find that PrivacyGAN can improve the quality and the diversity of the generated samples. It can be seen from Fig.5.3 that both IS and FID reach the optimal values at $\alpha = 0.4$. The difference is that the improvement of IS increases steadily from $\alpha = 0.9$ to $\alpha = 0.4$, which indicates the optimization procedure is relatively stable. But the improvement of FID oscillates around $\alpha = 0.4$. Fig.5.4 shows the samples produced by PrivacyGAN from a local CIFAR-10 dataset (6k).

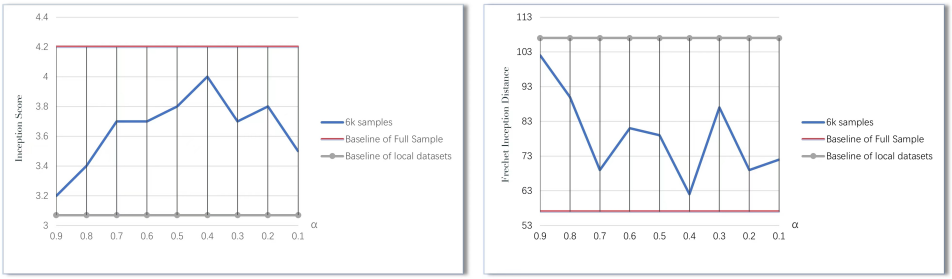


Figure 5.3: The IS scores (left) and the FID scores (right) of the samples generated by PrivacyGAN based on the full set of CIFAR-10 samples and a local CIFAR-10 dataset.

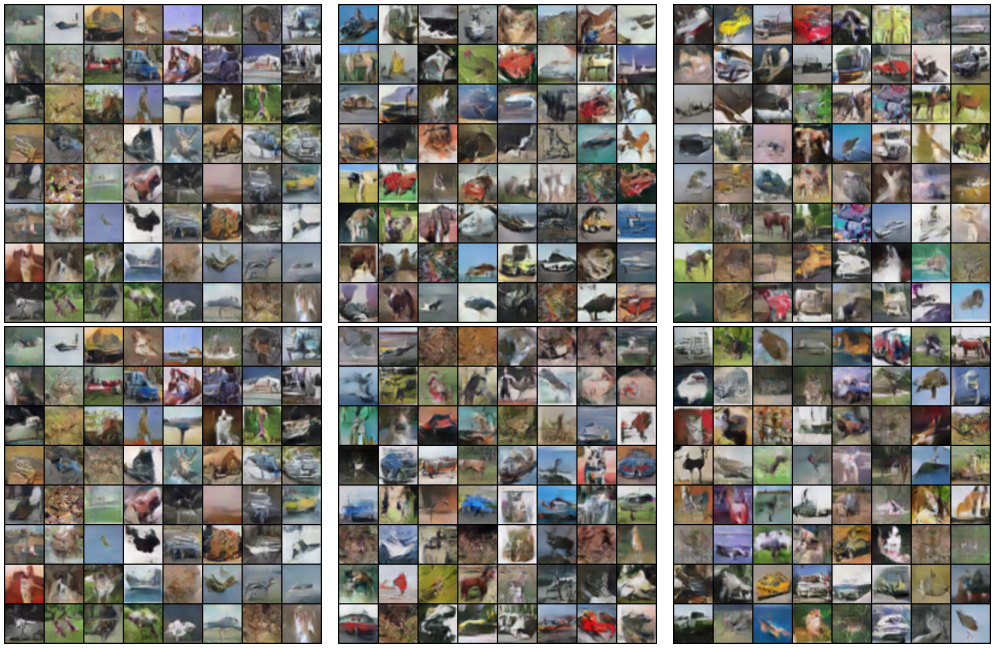


Figure 5.4: Images generated by PrivacyGAN on the local CIFAR-10 dataset (6k).

LFW

For the LFW dataset, we used the same architecture and experimental setup as for CIFAR-10. As shown in Table.4.3, we obtain the similar results as those for CIFAR-10, i.e., PrivacyGAN improves both IS and FID scores. Compared with the smooth increase in IS for CIFAR-10, our method achieves bigger improvements for LFW. From the results under different values of α, we can see our JRLF function optimizes the sample quality steadily compared with the results from CIFAR-10.

Table 5.5: The quality of the samples (From α=0.9 to α=0.5) generated by DCGAN and PrivacyGAN training on a full set of LFW samples (10k) and a local dataset (1k)

	DCGAN LFW (10K)	DCGAN LFW Local Dataset(1k)	PrivacyGAN LFW Local Dataset (1k) with different values of the α				
			0.9	0.8	0.7	0.6	0.5
IS	2.60	1.80	2.40	2.20	2.10	2.20	2.00
FID	46	345	122	247	170	204	197

Table 5.6: The quality of the samples (From α= 0.4 to α=0.1) generated by DCGAN and PrivacyGAN training on a full set of LFW samples (10K) and a local dataset (1k)

	DCGAN LFW (10K)	DCGAN LFW Local Dataset(1k)	PrivacyGAN LFW Local Dataset (1k) with different values of the α			
			0.4	0.3	0.2	0.1
IS	2.60	1.80	2.20	2.00	2.20	2.35
FID	46	345	226	176	189	188

It is noted that we got better IS scores at $\alpha = 0.9$ and $\alpha = 0.1$ (From Table.5.5 and Table.5.6). The phenomenon occurs when the training dataset is small, which implies that it does not take much effort to generate good samples for the small feature space. Fig.5.5 shows the samples produced by PrivacyGAN training on the local LFW dataset (1k).

LSUN

We evaluated our method on a large dataset LSUN. The enormous number of objects makes GAN training challenging because of the tendency to underestimate the entropy distributions. We conduct a series of experiments to demonstrate that our method improves the IS and FID. The experimental results are shown in Fig.??.

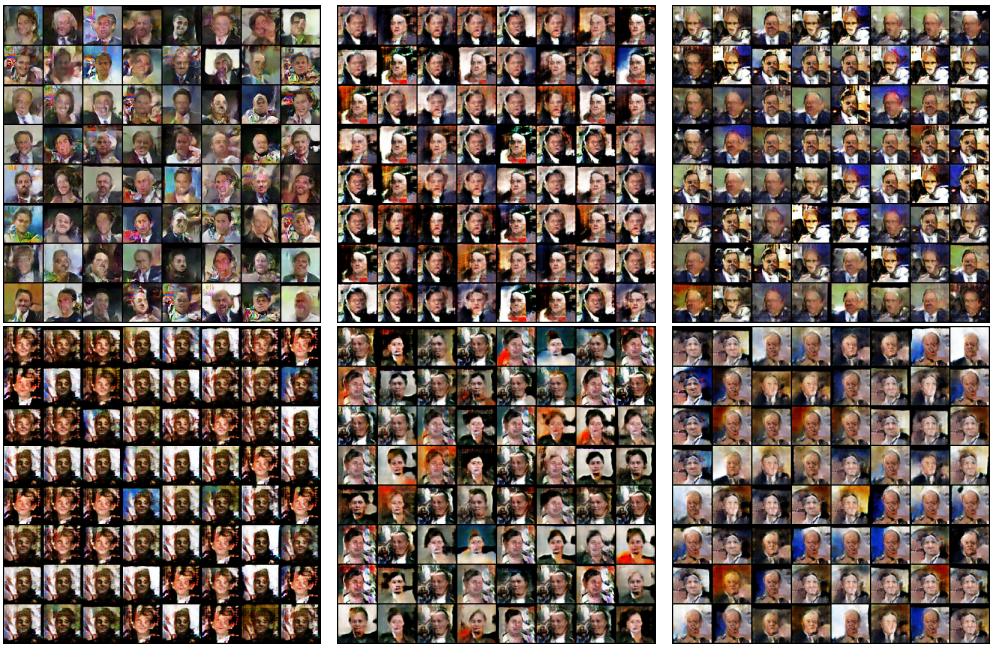


Figure 5.5: Images generated by PrivacyGAN on the 1k LFW dataset.

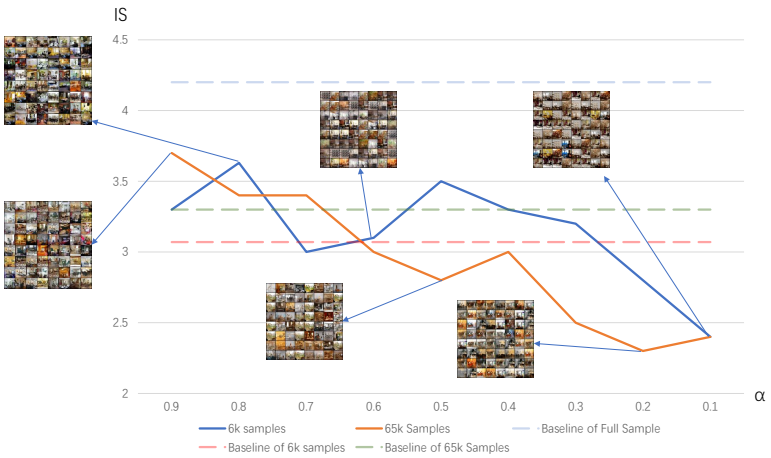


Figure 5.6: The IS of samples generated by DCGAN and PrivacyGAN training on a full LSUN dataset and the local datasets (6k and 60k). The baseline of different numbers of samples in the figure means the results obtained by the standard DCGAN. The samples in the figure are used for the visual inspection.

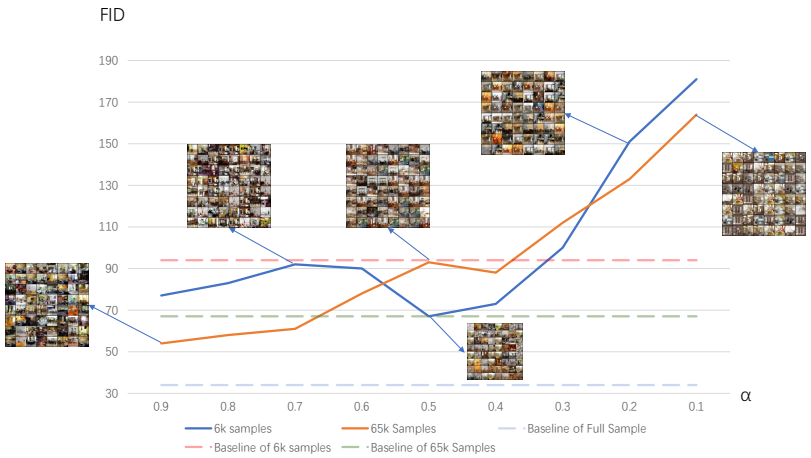


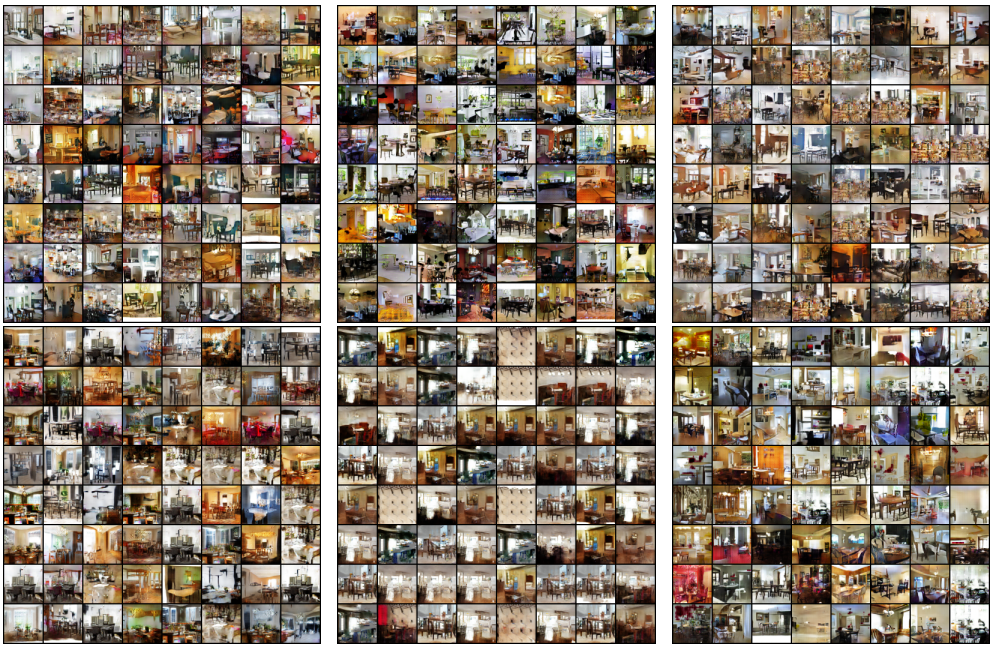
Figure 5.7: The FID of samples generated by DCGAN and PrivacyGAN training on a full LSUN dataset and the local datasets (6k and 60k). The baseline of different numbers of samples in the figure means the results obtained by the standard DCGAN. The samples in the figure are used for the visual inspection.

We can make the following observations from Fig.5.6 and Fig.5.7.

1) When the model is trained with small values of α, it creates the samples worse than the results from standard DCGAN. For the large dataset such as LSUN, increasing α equips the teacher model with high training weight, which makes the generated samples resemble the full dataset better than the local one.

2) Compared with the standard DCGAN, PrivacyGAN improves the samples' quality on the 6k local dataset compared with the output from the 65k local dataset. The results indicate that the improvement for the 6k local dataset is more robust than the improvement for the 65k local dataset. Since the 6k local dataset has fewer sample features than the 65k dataset, the results also indicate that there is more room to optimize the model when training on a dataset with less information. Although the improvement for the smaller dataset is more robust, a bigger dataset usually results in better optimization results eventually.

3) As α decreases, we can observe that although PrivacyGAN can improve the results, the model trained by PrivacyGAN on the LSUN dataset sometimes generates the worse samples than the standard DCGAN. This result is caused by insufficient training data. Therefore, for the large dataset, we need to set the weight of the student model to be a big value so as to improve the quality of the generated samples. Fig.5.8 shows the samples produced by PrivacyGAN training on the full LUSN dining room dataset.



(a) 6k dining room (b) 65k dining room (c) standard GAN from 6k,65k local

Figure 5.8: Images generated by PrivacyGAN on the LSUN dining room dataset.

5.4 Conculsion

In this chapter, we proposed a lightweight, privacy-aware GAN framework called PrivacyGAN to generate the high quality samples that resemble the private data in local clients. We proposed several optimization techniques, including the decentralized deployment strategy and an optimized loss function called JRLF, to synthesize the local data and improve the quality of the generated samples. The decentralized deployment strategy can reduce the communication cost between the server and the clients as well as reduce the computation load for the clients. The JRLF function solves the Simply Accepting problem in the teacher-student model. We conducted extensive experiments on multiple popular datasets to show the performance of our work. Although PrivacyGAN offers the efficient performance on generating personal data, there is still room to optimize the framework further, such as optimizing the structure of the teacher and the student models, which will be our future work.

Chapter 6

Conclusions

6.1 Conclusion

Improving the performance of the generative adversarial network is a challenging task and a long-term goal in the field of artificial intelligence and computer vision. The performance of GAN includes many aspects. 1) The quality of the generated samples can be judged by the diversity and the similarity of the samples, etc. 2) The training performance can be measured by the stability and the training time, etc. 3) The performance of GAN is also affected by the restrictions in real application scenarios, such as limited computing capacity of the nodes, the limited training data and the requirement of data privacy, etc.

6.1.1 MGGAN: Improving Sample Generations of Generative Adversarial Networks

Mode collapse is one of the most common issues in the training of the GAN. In chapter 3, we proposed a novel GAN framework called MGGAN to address this issue. In MGGAN, multiple generative groups are constructed to enrich the diversity of the generated samples. We also devised a dynamic adjustment strategy to escape from local optimums. Moreover, we demonstrated the importance of the swappping frequency which the generators and discriminators in different generative groups are regrouped. The generative groups with arbitrary regrouping frequency may aggravate the concussion or even cause the training hard to converge. We conducted the extensive experiments on different datasets to show the effectiveness of our method. The experimental results show that our method can obtain faithful samples and increase their quality and diversity. Moreover, we identified the appropriate swapping interval, which was of great importance for capturing the data distribution. We finally conducted the experiments to evaluate the quality of the generated samples in terms of IS and FID.

6.1.2 BPGAN: Accelerating the GAN Training by a Novel Model Parallelism Scheme

The training of GAN usually demands enormous time to extract valuable features due to the complexity of the network and the training datasets. In Chapter 4, we proposed BPGAN, a lightweight model parallelism framework for GAN, to accelerate the training. BPGAN requires at least a pair of working nodes to parallelize the GAN framework. The GAN framework is partitioned into two blocks (called Discriminator Block and Generator Block) based on the execution workflow. The computations are further divided into the first order computations, which are the operations performed directly on the network parameters, and the second order computations, which are the operations performed based on the results obtained by the first order computations. The partition reduce the communication cost and also the total training time. We conducted the theoretical analysis to compare the time cost between BPGAN and traditional data parallelism, which shows that BPGAN indeed reduces the training cost. We finally conducted the extensive experiments on a synthetic dataset, CIFAR, LFW and LSUN. The results show that our method can reduce the training cost without sacrificing the quality of the generated samples or increasing the complexity of the network.

6.1.3 PrivacyGAN: A Teacher-Student Framework based on GAN to Simulate Limited Privacy Data in Clients

Chapter 5 proposed PrivacyGAN, a teacher-student GAN model to produce the samples that resemble the sensitive data in local clients. We applied a teacher model to update the student model by sharing the knowledge of the teacher model. It avoids the overfitting issue resulted from the limited training data on the clients. We also proposed a decentralized deployment mechanism to capture the unique distributions of the clients' data without directly accessing the sensitive data. The deployment mechanism transforms the sensitive data into the fake images and labels through a local discriminator to update the generator on the server. Moreover, we designed a novel objective function to constraint the student's acceptance of the knowledge shared by the teacher. Simply accepting the knowledge from the teacher model may lose the unique diversity of the private data in the clients. Finally, we conducted a set of experiments to demonstrate the effectiveness of PrivacyGAN. The experimental results show that our PrivacyGAN can improve the quality and variety of the generated samples even with the limited training data. Notably, the acceptance weight should be set to be large values for the large dataset in order to improve the model performance.

Chapter 7

Future Work

7.1 Future Work of Research Contributions

Chapter 3 proposed a lightweight framework to tackle the mode collapse issue during the GAN training. The generative groups and dynamic adjustment strategy improve the generated samples' diversity and the training stability. In future, we plan to modify the structure of generative groups, such as adding a transformer component or design other tricks to improve the training performance. In addition, we will optimize the adjustment strategy to stabilize the training procedure for generating reliable fake samples. Moreover, we also plan to optimize the algorithm for identifying the regrouping intervals and develop a universal controlling parameter, which can reduce the randomness of the training process.

Chapter 4 investigated the impact of the model parallelism on the GAN training. We proposed a novel parallelism framework to speed up the training. In the future, we plan to extend our research in two folds.

i) Our method requires an even number of nodes to parallelize the GAN, limiting the framework's generality. Therefore, we plan to optimize the segmentation mechanism to apply our method in different settings.

ii) Our approach requires designing a specific execution partition for GAN models, which reduces the robustness and the generality. Hence, we plan to develop an improved method to minimize the efforts required to design the execution partition.

Chapter 5 proposed a lightweight, privacy-aware GAN framework to generate the samples that are similar to the sensitive data on the clients. We applied different strategies to address the over-fitting and privacy issues during the training. In future, our research will be extended to a more complex scenario: further reducing the computation requirement on the clients and the communication cost between the server and clients. We will apply some light encryption algorithms adopted in Federated learning to improve the per-

formance while protecting data privacy. Furthermore, more efficient teacher models can be developed to improve its guidance on the student models and reduce the communication costs. We will also try to collect the data from groups of people instead of one single person, which may increase people's willingness to share their private data.

7.2 Future Work of Thesis

In addition, another research direction following the above work is to develop a flexible framework that can adapt to other GAN variations. Different combinations of the GAN structure lead to different training mechanisms on working nodes. Moreover, we want to go further in applying the GAN to the application level by solving the issues caused by limited computing resources and bandwidth. Our next goal is to improve the performance on small mobile devices by optimizing the GAN models and algorithms.

Appendix A

More Experimental Results

In this Chapter, we present samples randomly generated by our proposed model.

A.0.1 MGGAN: Improving Sample Generations of Generative Adversarial Networks

Fig.A.1 to Fig.A.8 are experimental results from Chapter 3.

A.0.2 BPGAN: Accelerating the GAN Training by a Novel Model Parallelism Scheme

Fig.A.9 to Fig.A.14 are experimental results from Chapter 4.

A.0.3 PrivacyGAN:A Teacher-Student Framework based on GAN to Simulate Limited Privacy Data in Clients

Fig.A.15 to Fig.A.16 are experimental results from Chapter 5.

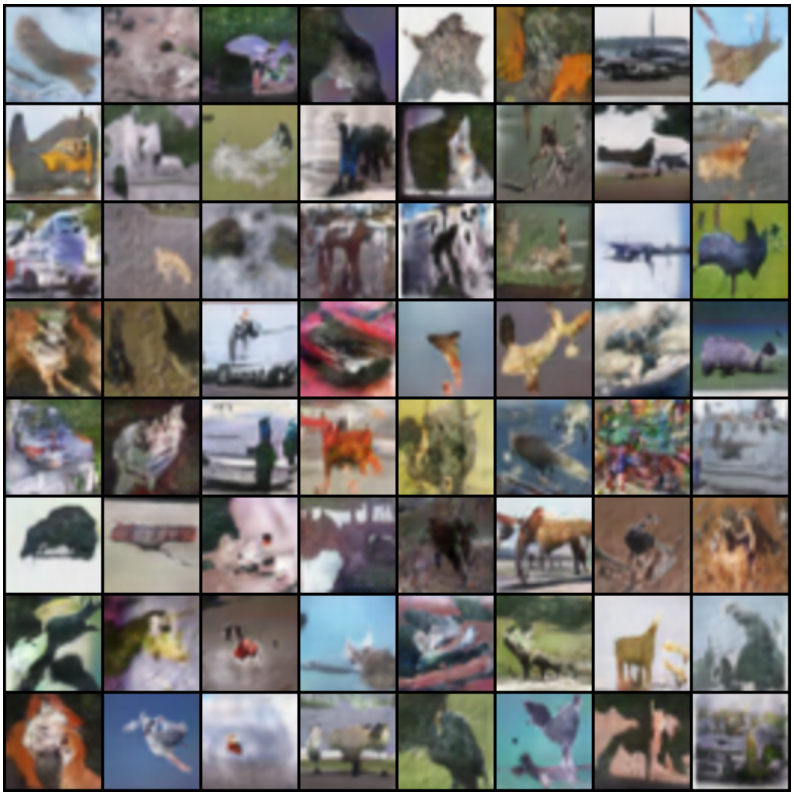


Figure A.1: Images generated by MGGAN (2GG) on the CIFAR-10 Dataset with epoch=1k and T=50

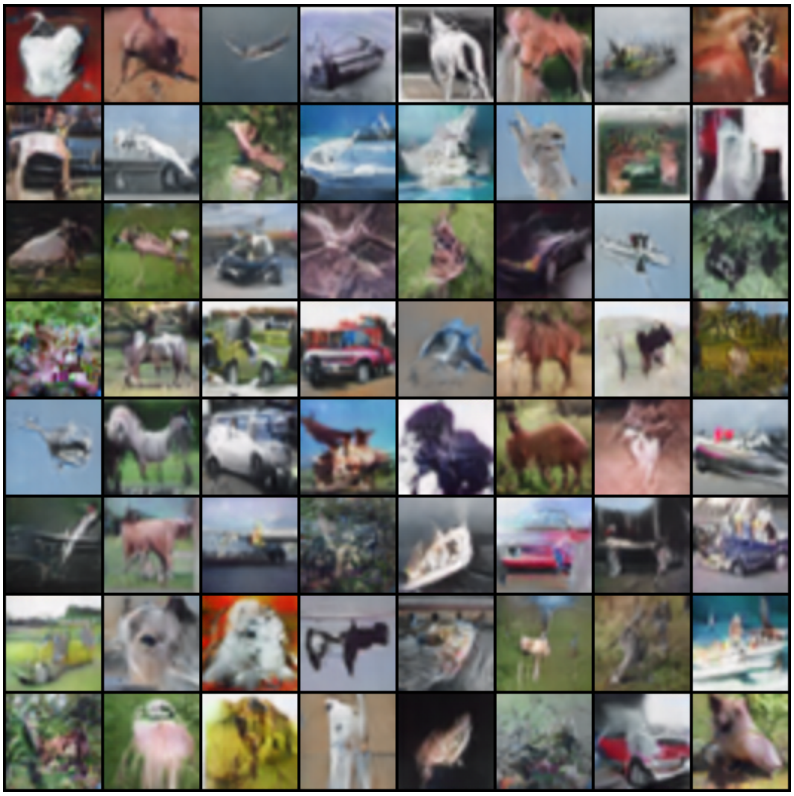


Figure A.2: Images generated by MGGAN (2GG) on the CIFAR-10 Dataset with epoch=1k and T=65

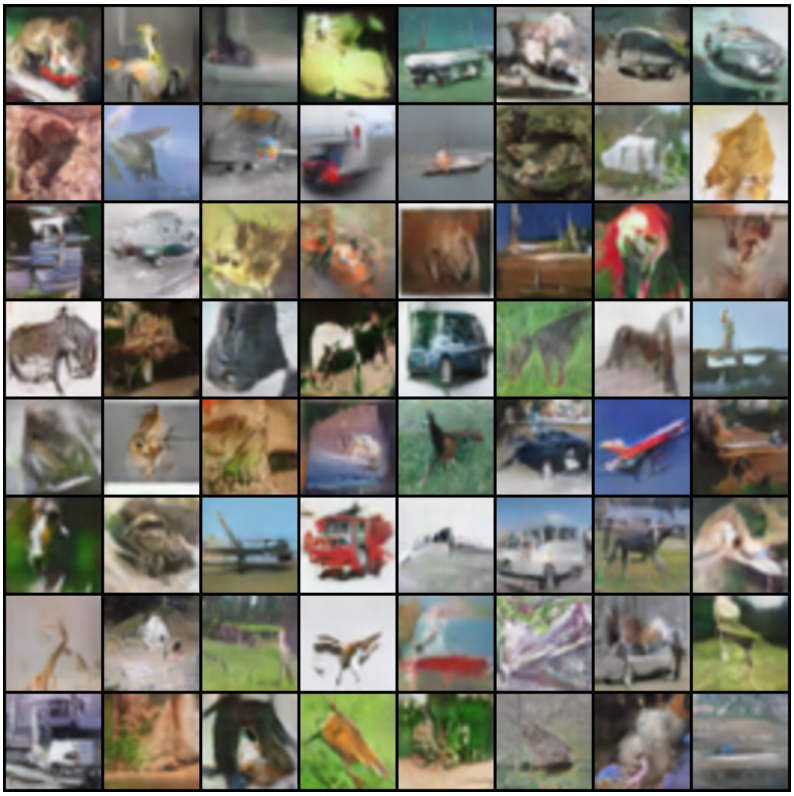


Figure A.3: Images generated by MGGAN (2GG) on the CIFAR-10 Dataset with epoch=1k and T=100

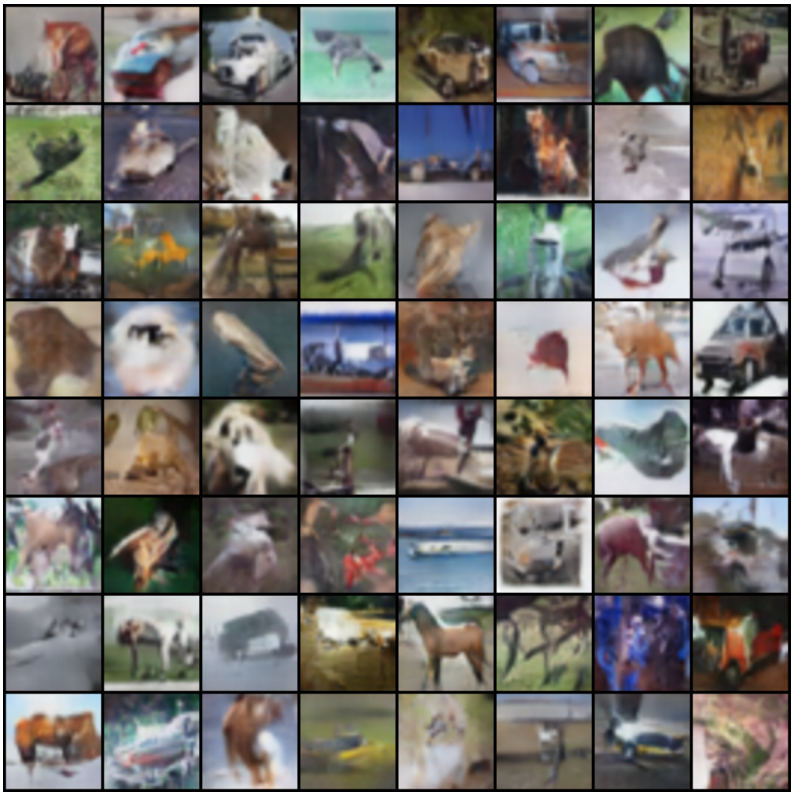


Figure A.4: Images generated by MGGAN (4GG) on the CIFAR-10 Dataset with epoch=1k and T=80

Figure A.5: Images generated by MGGAN (4GG) on the CIFAR-10 Dataset with epoch=1k and T=300

Figure A.6: Images generated by MGGAN (4GG) on the Lfw Dataset with epoch=1k and T=50

Figure A.7: Images generated by MGGAN (4GG) on the Lfw Dataset with epoch=1k and T=100

Figure A.8: Images generated by MGGAN (4GG) on the Lfw Dataset with epoch=1k and T=150

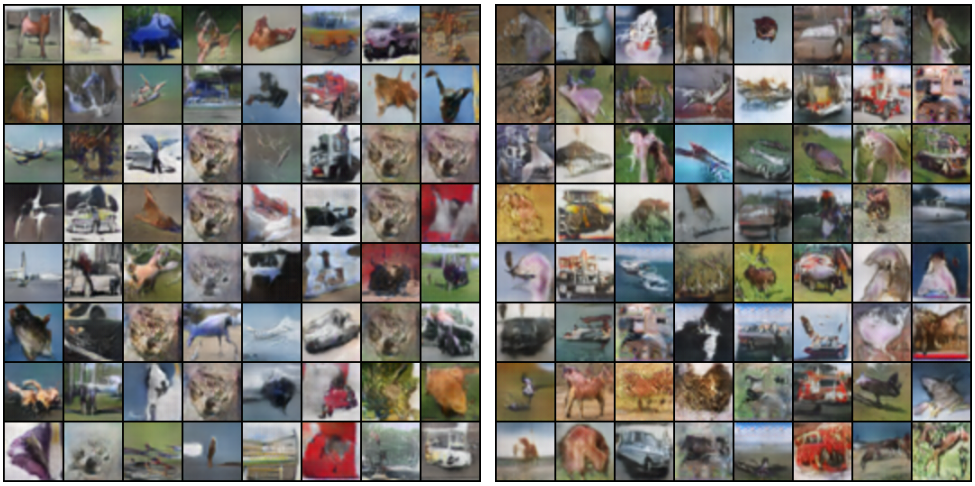


Figure A.9: Images generated by BPGAN on CIFAR Dataset with epoch=300. The DP (Left) and BPGAN (Right)

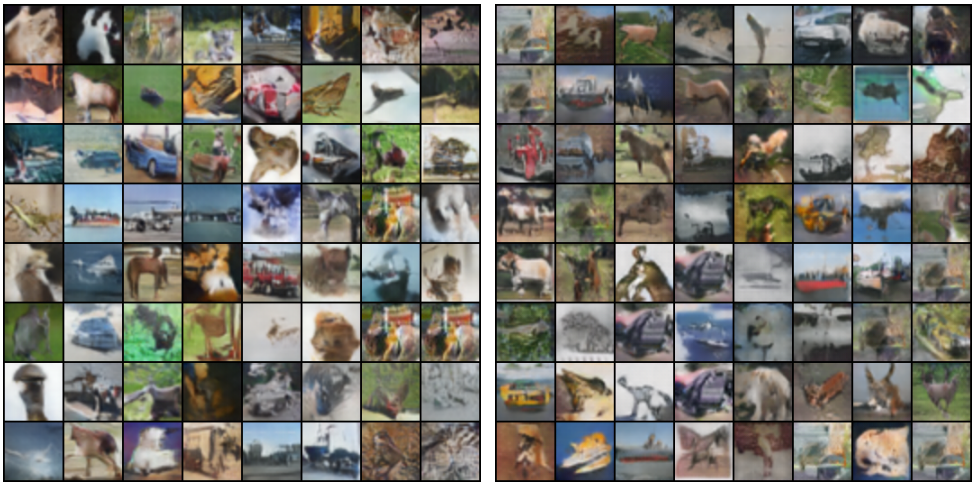


Figure A.10: Images generated by BPGAN on CIFAR Dataset with epoch=600. The DP (Left) and BPGAN (Right)

Figure A.11: Images generated by BPGAN on Lfw Dataset with epoch=600. The DP (Left) and BPGAN (Right)

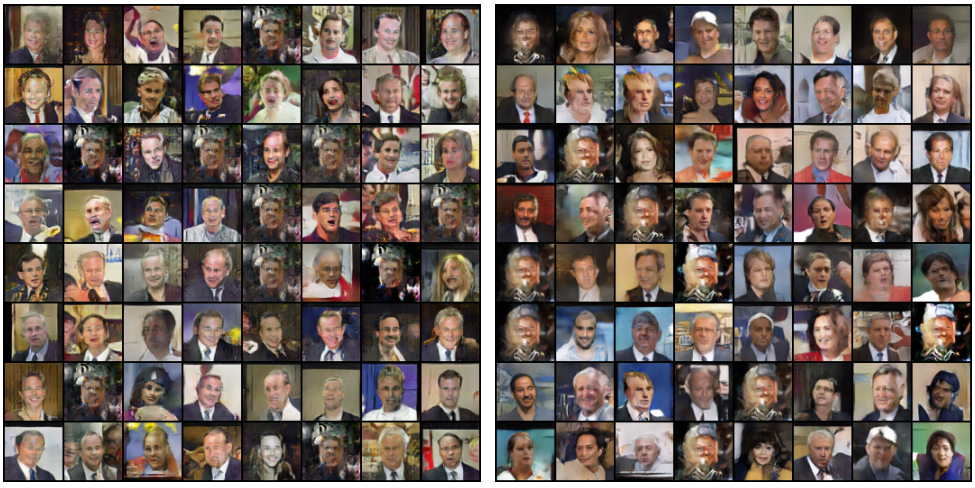


Figure A.12: Images generated by BPGAN on Lfw Dataset with epoch=1k. The DP (Left) and BPGAN (Right)

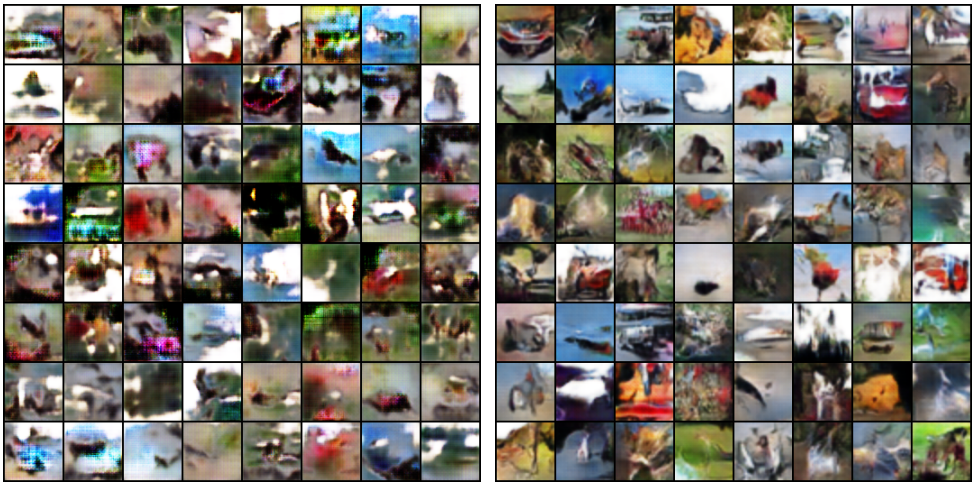


Figure A.13: Images generated by BPGAN on CIFAR Dataset with epoch=600. The DP (Left) and BPGAN (Right)

Figure A.14: Images generated by BPGAN on CIFAR Dataset with epoch=1k. The DP (Left) and BPGAN (Right)

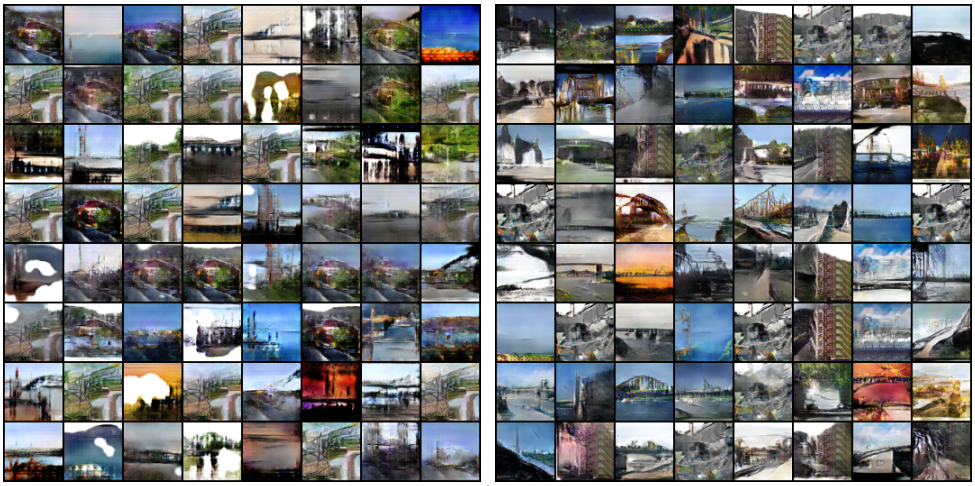


Figure A.15: Images generated by PrivacyGAN on LSUN Dataset with epoch=600. The a=0.4 (Left) and a=0.8 (Right)

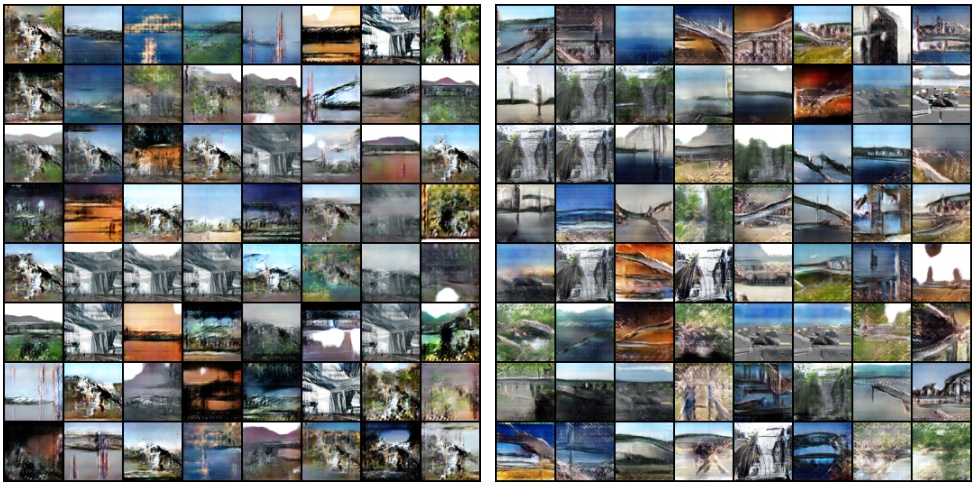


Figure A.16: Images generated by PrivacyGAN on LSUN Dataset with epoch=600. The a=0.1 (Left) and a=0.9 (Right)

Bibliography

[1] Adversarial Perturbations of Deep Neural Networks, pages 311–342. 2017.

[2] David H. Ackley, Geoffrey E. Hinton, and Terrence J. Sejnowski. A learning algorithm for boltzmann machines. Cogn. Sci., 9:147–169, 1985.

[3] Guillaume Alain, Yoshua Bengio, Li Yao, Jason Yosinski, Eric Thibodeau-Laufer, Saizheng Zhang, and Pascal Vincent. Gsns: generative stochastic networks. Information and Inference: A Journal of the IMA, 5(2):210–249, 2016.

[4] Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks, 2017.

[5] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In Doina Precup and Yee Whye Teh, editors, Proceedings of the 34th International Conference on Machine Learning, volume 70 of Proceedings of Machine Learning Research, pages 214–223. PMLR, 06–11 Aug 2017. URL `http://proceedings.mlr.press/v70/arjovsky17a.html`.

[6] Sanjeev Arora, Rong Ge, Yingyu Liang, Tengyu Ma, and Yi Zhang. Generalization and equilibrium in generative adversarial nets (gans), 2017.

[7] Lei Jimmy Ba and Rich Caruana. Do deep nets really need to be deep?, 2014.

[8] Haoli Bai, Jiaxiang Wu, Irwin King, and Michael Lyu. Few shot network compression via cross distillation, 2020.

[9] David Bau, Jun-Yan Zhu, Jonas Wulff, William Peebles, Hendrik Strobelt, Bolei Zhou, and Antonio Torralba. Seeing what a gan cannot generate, 2019.

[10] Sagie Benaim and Lior Wolf. One-shot unsupervised cross domain translation, 2018.

[11] Anant Bhardwaj, Souvik Bhattacherjee, Amit Chavan, Amol Deshpande, Aaron J Elmore, Samuel Madden, and Aditya G Parameswaran. Datahub: Collaborative data science & dataset version management at scale. arXiv preprint arXiv:1409.0798, 2014.

[12] Huang Bin, Chen Weihai, Wu Xingming, and Lin Chun-Liang. High-quality face image sr using conditional generative adversarial networks, 2017.

[13] Malik Boudiaf, Ziko Imtiaz Masud, Jérôme Rony, José Dolz, Pablo Piantanida, and Ismail Ben Ayed. Transductive information maximization for few-shot learning, 2020.

[14] Eric Breck, Neoklis Polyzotis, Sudip Roy, Steven Whang, and Martin Zinkevich. Data validation for machine learning. In MLSys, 2019.

[15] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis, 2019.

[16] Joan Bruna, Pablo Sprechmann, and Yann LeCun. Super-resolution with deep convolutional sufficient statistics, 2016.

[17] Sergi Caelles, Kevis-Kokitsi Maninis, Jordi Pont-Tuset, Laura Leal-Taixé, Daniel Cremers, and Luc Van Gool. One-shot video object segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 221–230, 2017.

[18] Tong Che, Yanran Li, Athul Paul Jacob, Yoshua Bengio, and Wenjie Li. Mode regularized generative adversarial networks. arXiv preprint arXiv:1612.02136, 2016.

[19] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets, 2016.

[20] Zitian Chen, Yanwei Fu, Yu-Xiong Wang, Lin Ma, Wei Liu, and Martial Hebert. Image deformation meta-networks for one-shot learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 8680–8689, 2019.

[21] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. A survey of model compression and acceleration for deep neural networks. arXiv preprint arXiv:1710.09282, 2017.

[22] Soumith Chintala, Emily Denton, Martin Arjovsky, and Michael Mathieu. How to train a gan? tips and tricks to make gans work. Github. com, 2016.

[23] Jang Hyun Cho and Bharath Hariharan. On the efficacy of knowledge distillation, 2019.

[24] Yoojin Choi, Mostafa El-Khamy, and Jungwon Lee. Towards the limit of network quantization, 2017.

[25] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation, 2018.

[26] J. Chun and T. Kailath. Generalized displacement structure for block-toeplitz, toeplitz-block, and toeplitz-derived matrices. In Gene H. Golub and Paul Van Dooren, editors, Numerical Linear Algebra, Digital Signal Processing and Parallel Algorithms, pages 215–236, Berlin, Heidelberg, 1991. Springer Berlin Heidelberg. ISBN 978-3-642-75536-1.

[27] Aidan Clark, Jeff Donahue, and Karen Simonyan. Adversarial video generation on complex datasets. arXiv preprint arXiv:1907.06571, 2019.

[28] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 113–123, 2019.

[29] Bo Dai, Sanja Fidler, Raquel Urtasun, and Dahua Lin. Towards diverse and natural image descriptions via a conditional gan, 2017.

[30] Christopher R Dance, Julien Perez, and Théo Cachet. Conditioned reinforcement learning for few-shot imitation. In International Conference on Machine Learning, pages 2376–2387. PMLR, 2021.

[31] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In CVPR09, 2009.

[32] Qianggang Ding, Sifan Wu, Hao Sun, Jiadong Guo, and Shu-Tao Xia. Adaptive regularization of labels, 2019.

[33] Zihan Ding, Xiao-Yang Liu, Miao Yin, and Linghe Kong. Tgan: Deep tensor generative adversarial nets for large image generation, 2019.

[34] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. arXiv preprint arXiv:1410.8516, 2014.

[35] Chris Donahue, Julian McAuley, and Miller Puckette. Adversarial audio synthesis. arXiv preprint arXiv:1802.04208, 2018.

[36] Yan Duan, Marcin Andrychowicz, Bradly C Stadie, Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-shot imitation learning. arXiv preprint arXiv:1703.07326, 2017.

[37] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. Adversarially learned inference. arXiv preprint arXiv:1606.00704, 2016.

[38] Ishan Durugkar, Ian Gemp, and Sridhar Mahadevan. Generative multi-adversarial networks, 2017.

[39] Nikita Dvornik, Cordelia Schmid, and Julien Mairal. Diversity with cooperation: Ensemble methods for few-shot classification. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 3723–3731, 2019.

[40] Mohamed Elfeki, Camille Couprie, and Mohamed Elhoseiny. Learning diverse generations using determinantal point processes. 2018.

[41] Gongfan Fang, Jie Song, Chengchao Shen, Xinchao Wang, Da Chen, and Mingli Song. Data-free adversarial distillation, 2020.

[42] Michael Fink. Object classification from a single example utilizing class relevance metrics. Advances in neural information processing systems, 17:449–456, 2005.

[43] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In International Conference on Machine Learning, pages 1126–1135. PMLR, 2017.

[44] Chelsea Finn, Kelvin Xu, and Sergey Levine. Probabilistic model-agnostic meta-learning. arXiv preprint arXiv:1806.02817, 2018.

[45] Chelsea Finn, Aravind Rajeswaran, Sham Kakade, and Sergey Levine. Online meta-learning. In International Conference on Machine Learning, pages 1920–1930. PMLR, 2019.

[46] Maayan Frid-Adar, Eyal Klang, Michal Amitai, Jacob Goldberger, and Hayit Greenspan. Synthetic data augmentation using gan for improved liver lesion classification. In 2018 IEEE 15th international symposium on biomedical imaging (ISBI 2018), pages 289–293. IEEE, 2018.

[47] Stanislav Frolov, Tobias Hinz, Federico Raue, Jörn Hees, and Andreas Dengel. Adversarial text-to-image synthesis: A review. arXiv preprint arXiv:2101.09983, 2021.

[48] Tommaso Furlanello, Zachary Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born again neural networks. In International Conference on Machine Learning, pages 1607–1616. PMLR, 2018.

[49] Adrià Gascón, Phillipp Schoppmann, Borja Balle, Mariana Raykova, Jack Doerner, Samee Zahur, and David Evans. Secure linear regression on vertically partitioned datasets. IACR Cryptol. ePrint Arch., 2016: 892, 2016.

[50] Jon Gauthier. Conditional generative adversarial nets for convolutional face generation. Class Project for Stanford CS231N: Convolutional Neural Networks for Visual Recognition, Winter semester, 2014(5):2, 2014.

[51] Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. Made: Masked autoencoder for distribution estimation. In International Conference on Machine Learning, pages 881–889. PMLR, 2015.

[52] Andrew Gibiansky. Bringing hpc techniques to deep learning.(2017). URL http://research. baidu. com/bringing-hpc-techniquesdeep-learning, 2017.

[53] Yunchao Gong, Liu Liu, Ming Yang, and Lubomir Bourdev. Compressing deep convolutional networks using vector quantization. arXiv preprint arXiv:1412.6115, 2014.

[54] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. Advances in neural information processing systems, 27, 2014.

[55] Jindong Gu and Volker Tresp. Search for better students to learn distilled knowledge. arXiv preprint arXiv:2001.11612, 2020.

[56] Jie Gui, Zhenan Sun, Yonggang Wen, Dacheng Tao, and Jieping Ye. A review on generative adversarial networks: Algorithms, theory, and applications, 2020.

[57] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans. arXiv preprint arXiv:1704.00028, 2017.

[58] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep learning with limited numerical precision. In International conference on machine learning, pages 1737–1746. PMLR, 2015.

[59] Alon Halevy, Peter Norvig, and Fernando Pereira. The unreasonable effectiveness of data. IEEE Intelligent Systems, 24(2):8–12, 2009.

[60] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149, 2015.

[61] Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks, 2015.

[62] Corentin Hardy, Erwan Le Merrer, and Bruno Sericola. Md-gan: Multi-discriminator generative adversarial networks for distributed datasets. In 2019 IEEE international parallel and distributed processing symposium (IPDPS), pages 866–877. IEEE, 2019.

[63] B. Hassibi, D.G. Stork, and G.J. Wolff. Optimal brain surgeon and general network pruning. In IEEE International Conference on Neural Networks, pages 293–299 vol.1, 1993. doi: 10.1109/ICNN.1993.298572.

[64] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[65] Byeongho Heo, Jeesoo Kim, Sangdoo Yun, Hyojin Park, Nojun Kwak, and Jin Young Choi. A comprehensive overhaul of feature distillation, 2019.

[66] Joeri R Hermans, Gerasimos Spanakis, and Rico Möckel. Accumulated gradient normalization. In Asian Conference on Machine Learning, pages 439–454. PMLR, 2017.

[67] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. arXiv preprint arXiv:1706.08500, 2017.

[68] G.E. Hinton. Boltzmann Machines: Constraint Satisfaction Networks that Learn. Carnegie-Mellon University, Department of Computer Science, 1984. URL `https://books.google.co.uk/books?id=IniqXwAACAAJ`.

[69] Geoffrey Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. Neural computation, 18:1527–54, 08 2006. doi: 10.1162/neco.2006.18.7.1527.

[70] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531, 2015.

[71] Quan Hoang, Tu Dinh Nguyen, Trung Le, and Dinh Phung. Multi-generator generative adversarial nets, 2017.

[72] Judy Hoffman, Eric Tzeng, Jeff Donahue, Yangqing Jia, Kate Saenko, and Trevor Darrell. One-shot adaptation of supervised deep convolutional models. arXiv preprint arXiv:1312.6204, 2013.

[73] Fred Hohman, Kanit Wongsuphasawat, Mary Beth Kery, and Kayur Patel. Understanding and visualizing data iteration in machine learning. In Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems, pages 1–13, 2020.

[74] J.L. Holt and J.-N. Hwang. Finite precision error analysis of neural network hardware implementations. IEEE Transactions on Computers, 42(3):281–290, 1993. doi: 10.1109/12.210171.

[75] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. arXiv preprint arXiv:2004.05439, 2020.

[76] Ruibing Hou, Hong Chang, Bingpeng Ma, Shiguang Shan, and Xilin Chen. Cross attention network for few-shot classification. arXiv preprint arXiv:1910.07677, 2019.

[77] Rein Houthooft, Richard Y Chen, Phillip Isola, Bradly C Stadie, Filip Wolski, Jonathan Ho, and Pieter Abbeel. Evolved policy gradients. arXiv preprint arXiv:1802.04821, 2018.

[78] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861, 2017.

[79] Qinghao Hu, Peisong Wang, and Jian Cheng. From hashing to cnns: Training binaryweight networks via hashing, 2018.

[80] Weiwei Hu and Ying Tan. Generating adversarial malware examples for black-box attacks based on gan. arXiv preprint arXiv:1702.05983, 2017.

[81] Zikun Hu, Xiang Li, Cunchao Tu, Zhiyuan Liu, and Maosong Sun. Few-shot charge prediction with discriminative legal attributes. In Proceedings of the 27th International Conference on Computational Linguistics, pages 487–498, 2018.

[82] Ziniu Hu, Ting Chen, Kai-Wei Chang, and Yizhou Sun. Few-shot representation learning for out-of-vocabulary words. arXiv preprint arXiv:1907.00505, 2019.

[83] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 4700–4708, 2017.

[84] Gary B Huang, Marwan Mattar, Tamara Berg, and Eric Learned-Miller. Labeled faces in the wild: A database forstudying face recognition in unconstrained environments. In Workshop on faces in'Real-Life'Images: detection, alignment, and recognition, 2008.

[85] Huaibo Huang, Ran He, Zhenan Sun, and Tieniu Tan. Wavelet domain generative adversarial network for multi-scale face hallucination. International Journal of Computer Vision, 127(6):763–784, 2019.

[86] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization, 2017.

[87] Ferenc Huszár. How (not) to train your generative model: Scheduled sampling, likelihood, adversary? arXiv preprint arXiv:1511.05101, 2015.

[88] Ben Hutchinson, Andrew Smart, Alex Hanna, Emily Denton, Christina Greer, Oddur Kjartansson, Parker Barnes, and Margaret Mitchell. Towards accountability for machine learning datasets: Practices from software engineering and infrastructure. In Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency, pages 560–575, 2021.

[89] Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5mb model size, 2016.

[90] Daniel Jiwoong Im, He Ma, Chris Dongjoo Kim, and Graham Taylor. Generative adversarial parallelization. arXiv preprint arXiv:1612.04021, 2016.

[91] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

[92] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1125–1134, 2017.

[93] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko.

Quantization and training of neural networks for efficient integer-arithmetic-only inference, 2017.

[94] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution, 2016.

[95] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. arXiv preprint arXiv:1710.10196, 2017.

[96] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 4401–4410, 2019.

[97] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. arXiv preprint arXiv:2006.06676, 2020.

[98] Jangho Kim, Minsung Hyun, Inseop Chung, and Nojun Kwak. Feature fusion for online mutual knowledge distillation, 2020.

[99] Minje Kim and Paris Smaragdis. Bitwise neural networks, 2016.

[100] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim. Learning to discover cross-domain relations with generative adversarial networks, 2017.

[101] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

[102] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114, 2013.

[103] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. Foundations and Trends® in Machine Learning, 12(4): 307–392, 2019. ISSN 1935-8245. doi: 10.1561/2200000056. URL `http://dx.doi.org/10.1561/2200000056`.

[104] Naveen Kodali, Jacob Abernethy, James Hays, and Zsolt Kira. On convergence and stability of gans, 2017.

[105] Laura Koesten, Kathleen Gregory, Paul Groth, and Elena Simperl. Talking datasets–understanding data sensemaking behaviours. International Journal of Human-Computer Studies, 146:102562, 2021.

[106] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[107] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 25:1097–1105, 2012.

[108] Mandar Kulkarni, Kalpesh Patil, and Shirish Karande. Knowledge distillation using unlabeled mismatched images, 2017.

[109] Y. LECUN. The mnist database of handwritten digits. http://yann.lecun.com/exdb/mnist/. URL https://ci.nii.ac.jp/naid/10027939599/en/.

[110] Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. In D. Touretzky, editor, Advances in Neural Information Processing Systems, volume 2. Morgan-Kaufmann, 1990. URL https://proceedings.neurips.cc/paper/1989/file/6c9882bbac1c7093bd25041881277658-Paper.pdf.

[111] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 4681–4690, 2017.

[112] Chuan Li and Michael Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks, 2016.

[113] Dan Li, Dacheng Chen, Baihong Jin, Lei Shi, Jonathan Goh, and See-Kiong Ng. Mad-gan: Multivariate anomaly detection for time series data with generative adversarial networks. In International Conference on Artificial Neural Networks, pages 703–716. Springer, 2019.

[114] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets, 2017.

[115] Mu Li, Li Zhou, Zichao Yang, Aaron Li, Fei Xia, David G Andersen, and Alexander Smola. Parameter server for distributed machine learning. In Big Learning NIPS Workshop, volume 6, page 2, 2013.

[116] Mu Li, David G Andersen, Jun Woo Park, Alexander J Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J Shekita, and Bor-Yiing Su. Scaling distributed machine learning with the parameter server. In 11th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 14), pages 583–598, 2014.

[117] Muyang Li, Ji Lin, Yaoyao Ding, Zhijian Liu, Jun-Yan Zhu, and Song Han. Gan compression: Efficient architectures for interactive conditional gans. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 5284–5294, 2020.

[118] Tianhong Li, Jianguo Li, Zhuang Liu, and Changshui Zhang. Few sample knowledge distillation for efficient network compression, 2020.

[119] Yuncheng Li, Jianchao Yang, Yale Song, Liangliang Cao, Jiebo Luo, and Li-Jia Li. Learning from noisy labels with distillation, 2017.

[120] Kevin J Liang, Chunyuan Li, Guoyin Wang, and Lawrence Carin. Generative adversarial network training is a continual learning problem, 2018.

[121] Sungbin Lim, Ildoo Kim, Taesup Kim, Chiheon Kim, and Sungwoong Kim. Fast autoaugment. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, Advances in Neural Information Processing Systems, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/6add07cf50424b14fdf649da87843d01-Paper.pdf.

[122] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. arXiv preprint arXiv:1312.4400, 2013.

[123] Yujun Lin, Song Han, Huizi Mao, Yu Wang, and William J Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. arXiv preprint arXiv:1712.01887, 2017.

[124] Iou-Jen Liu, Jian Peng, and Alexander G Schwing. Knowledge flow: Improve upon your teachers. arXiv preprint arXiv:1904.05878, 2019.

[125] Ming-Yu Liu and Oncel Tuzel. Coupled generative adversarial networks. arXiv preprint arXiv:1606.07536, 2016.

[126] Qing Liu, Lingxi Xie, Huiyu Wang, and Alan Yuille. Semantic-aware knowledge preservation for zero-shot sketch-based image retrieval, 2019.

[127] Rui Liu, Yixiao Ge, Ching Lam Choi, Xiaogang Wang, and Hongsheng Li. Divco: Diverse conditional image synthesis via contrastive generative adversarial network. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 16377–16386, 2021.

[128] Xuanqing Liu and Cho-Jui Hsieh. Rob-gan: Generator, discriminator, and adversarial attacker. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 11234–11243, 2019.

[129] Yuang Liu, Wei Zhang, and Jun Wang. Adaptive multi-teacher multi-level knowledge distillation. Neurocomputing, 415:106–113, 2020.

[130] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In Proceedings of the IEEE international conference on computer vision, pages 3730–3738, 2015.

[131] Yunteng Luan, Hanyu Zhao, Zhi Yang, and Yafei Dai. Msd: Multi-self-distillation learning via multi-classifiers within deep neural networks, 2019.

[132] Shuang Ma, Jianlong Fu, Chang Wen Chen, and Tao Mei. Da-gan: Instance-level image translation by deep attention generative adversarial networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 5657–5666, 2018.

[133] Yan Ma, Kang Liu, Zhibin Guan, Xinkai Xu, Xu Qian, and Hong Bao. Background augmentation generative adversarial networks (bagans): Effective data generation based on gan-augmented 3d synthesizing. Symmetry, 10(12):734, 2018.

[134] Yuqing Ma, Wei Liu, Shihao Bai, Qingyu Zhang, Aishan Liu, Weimin Chen, and Xianglong Liu. Few-shot visual learning with contextual memory and fine-grained calibration. In Christian Bessiere, editor, Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20, pages 811–817. International Joint Conferences on Artificial Intelligence Organization, 7 2020. Main track.

[135] Andrés Marafioti, Nathanaël Perraudin, Nicki Holighaus, and Piotr Majdak. Adversarial generation of time-frequency features with application in audio synthesis. In International Conference on Machine Learning, pages 4352–4362. PMLR, 2019.

[136] Giovanni Mariani, Florian Scheidegger, Roxana Istrate, Costas Bekas, and Cristiano Malossi. Bagan: Data augmentation with balancing gan. arXiv preprint arXiv:1803.09655, 2018.

[137] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data, 2017.

[138] Gaurav Menghani. Efficient deep learning: A survey on making deep learning models smaller, faster, and better, 2021.

[139] Gaurav Menghani and Sujith Ravi. Learning from a teacher using unlabeled data, 2019.

[140] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. arXiv preprint arXiv:1611.02163, 2016.

[141] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks, 2017.

[142] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784, 2014.

[143] Takeru Miyato and Masanori Koyama. cgans with projection discriminator, 2018.

[144] Andriy Mnih and Russ R Salakhutdinov. Probabilistic matrix factorization. In Advances in neural information processing systems, pages 1257–1264, 2008.

[145] Hossein Mobahi, Mehrdad Farajtabar, and Peter L. Bartlett. Self-distillation amplifies regularization in hilbert space, 2020.

[146] Marcin Moczulski, Misha Denil, Jeremy Appleyard, and Nando de Freitas. Acdc: A structured efficient linear layer, 2016.

[147] Anh Nguyen, Jeff Clune, Yoshua Bengio, Alexey Dosovitskiy, and Jason Yosinski. Plug play generative networks: Conditional iterative generation of images in latent space, 2017.

[148] Tu Dinh Nguyen, Trung Le, Hung Vu, and Dinh Phung. Dual discriminator generative adversarial nets. arXiv preprint arXiv:1709.03831, 2017.

[149] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans, 2017.

[150] Samir Passi and Steven Jackson. Data vision: Learning to see through algorithmic abstraction. In Proceedings of the 2017 ACM conference on computer supported cooperative work and social computing, pages 2436–2447, 2017.

[151] Henning Petzka, Asja Fischer, and Denis Lukovnicov. On the regularization of wasserstein gans, 2018.

[152] Le Trieu Phong, Yoshinori Aono, Takuya Hayashi, Lihua Wang, and Shiho Moriai. Privacy-preserving deep learning via additively homomorphic encryption. IEEE Transactions on Information Forensics and Security, 13(5):1333–1345, 2018. doi: 10.1109/TIFS.2017.2787987.

[153] Cheng Perng Phoo and Bharath Hariharan. Self-training for few-shot transfer across extreme task differences, 2021.

[154] Antonio Polino, Razvan Pascanu, and Dan Alistarh. Model compression via distillation and quantization. arXiv preprint arXiv:1802.05668, 2018.

[155] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434, 2015.

[156] Tiago Ramalho and Marta Garnelo. Adaptive posterior learning: few-shot learning with a surprise-based memory module. arXiv preprint arXiv:1902.02527, 2019.

[157] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. In International Conference on Machine Learning, pages 1060–1069. PMLR, 2016.

[158] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets, 2015.

[159] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. International journal of computer vision, 115(3):211–252, 2015.

[160] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. Advances in neural information processing systems, 29:2234–2242, 2016.

[161] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-gan: Protecting classifiers against adversarial attacks using generative models. arXiv preprint arXiv:1805.06605, 2018.

[162] Nithya Sambasivan, Shivani Kapania, Hannah Highfill, Diana Akrong, Praveen Paritosh, and Lora M Aroyo. "everyone wants to do the model work, not the data work": Data cascades in high-stakes ai. In proceedings of the 2021 CHI Conference on Human Factors in Computing Systems, pages 1–15, 2021.

[163] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In International conference on machine learning, pages 1842–1850. PMLR, 2016.

[164] Kripasindhu Sarkar, Lingjie Liu, Vladislav Golyanik, and Christian Theobalt. Humangan: A generative model of humans images. arXiv preprint arXiv:2103.06902, 2021.

[165] Eli Schwartz, Leonid Karlinsky, Joseph Shtok, Sivan Harary, Mattias Marder, Rogerio Feris, Abhishek Kumar, Raja Giryes, and Alex M Bronstein. Delta-encoder: an effective sample synthesis method for few-shot object recognition. arXiv preprint arXiv:1806.04734, 2018.

[166] Frank Seide, Hao Fu, Jasha Droppo, Gang Li, and Dong Yu. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns. In Fifteenth Annual Conference of the International Speech Communication Association, 2014.

[167] Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. Singan: Learning a generative model from a single natural image. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 4570–4580, 2019.

[168] Assaf Shocher, Shai Bagon, Phillip Isola, and Michal Irani. Ingan: Capturing and retargeting the" dna" of a natural image. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 4492–4501, 2019.

[169] Han Shu, Yunhe Wang, Xu Jia, Kai Han, Hanting Chen, Chunjing Xu, Qi Tian, and Chang Xu. Co-evolutionary compression for unpaired image translation. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 3235–3244, 2019.

[170] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In European conference on computer vision, pages 746–760. Springer, 2012.

[171] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.

[172] Vikas Sindhwani, Tara N. Sainath, and Sanjiv Kumar. Structured transforms for small-footprint deep learning, 2015.

[173] Jake Snell, Kevin Swersky, and Richard S Zemel. Prototypical networks for few-shot learning. arXiv preprint arXiv:1703.05175, 2017.

[174] Casper Kaae Sønderby, Jose Caballero, Lucas Theis, Wenzhe Shi, and Ferenc Huszár. Amortised map inference for image super-resolution. arXiv preprint arXiv:1610.04490, 2016.

[175] Akash Srivastava, Lazar Valkov, Chris Russell, Michael U Gutmann, and Charles Sutton. Veegan: Reducing mode collapse in gans using implicit variational learning. arXiv preprint arXiv:1705.07761, 2017.

[176] Flood Sung, Li Zhang, Tao Xiang, Timothy Hospedales, and Yongxin Yang. Learning to learn: Meta-critic networks for sample efficient learning. arXiv preprint arXiv:1706.09529, 2017.

[177] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision, 2015.

[178] Fabio Henrique Kiyoiti dos Santos Tanaka and Claus Aranha. Data augmentation using gans. arXiv preprint arXiv:1904.09135, 2019.

[179] Hao Tang, Dan Xu, Nicu Sebe, Yanzhi Wang, Jason J. Corso, and Yan Yan. Multi-channel attention selection gan with cascaded semantic guidance for cross-view image translation, 2019.

[180] Wei Tang, Gang Hua, and Liang Wang. How to train a compact binary neural network with high accuracy? In AAAI, 2017.

[181] Takeshi Teshima, Issei Sato, and Masashi Sugiyama. Few-shot domain adaptation by causal mechanism transfer. In International Conference on Machine Learning, pages 9458–9469. PMLR, 2020.

[182] Lucas Theis, Aäron van den Oord, and Matthias Bethge. A note on the evaluation of generative models. arXiv preprint arXiv:1511.01844, 2015.

[183] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive representation distillation, 2020.

[184] Karen Ullrich, Edward Meeds, and Max Welling. Soft weight-sharing for neural network compression, 2017.

[185] Benigno Uria, Marc-Alexandre Côté, Karol Gregor, Iain Murray, and Hugo Larochelle. Neural autoregressive distribution estimation. The Journal of Machine Learning Research, 17(1):7184–7220, 2016.

[186] Jaideep Vaidya and Chris Clifton. Privacy preserving association rule mining in vertically partitioned data. In Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '02, page 639–644, New York, NY, USA, 2002. Association for Computing Machinery. ISBN 158113567X. doi: 10.1145/775047.775142. URL https://doi.org/10.1145/775047.775142.

[187] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks, 2016.

[188] Vincent Vanhoucke, Andrew Senior, and Mark Z. Mao. Improving the speed of neural networks on cpus. In Deep Learning and Unsupervised Feature Learning Workshop, NIPS 2011, 2011.

[189] Hui Wang, Hanbin Zhao, Xi Li, and Xu Tan. Progressive blockwise knowledge distillation for neural network acceleration. In IJCAI, pages 2769–2775, 2018.

[190] Lin Wang and Kuk-Jin Yoon. Knowledge distillation and student-teacher learning for visual intelligence: A review and new outlooks. IEEE Transactions on Pattern Analysis and Machine Intelligence, page 1–1, 2021. ISSN 1939-3539. doi: 10.1109/tpami.2021.3055564. URL http://dx.doi.org/10.1109/TPAMI.2021.3055564.

[191] Naigang Wang, Jungwook Choi, Daniel Brand, Chia-Yu Chen, and Kailash Gopalakrishnan. Training deep neural networks with 8-bit floating point numbers. In Proceedings of the 32nd International Conference on Neural Information Processing Systems, pages 7686–7695, 2018.

[192] Peisong Wang and Jian Cheng. Fixed-point factorized networks, 2017.

[193] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Chen Change Loy, Yu Qiao, and Xiaoou Tang. Esrgan: Enhanced super-resolution generative adversarial networks, 2018.

[194] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. Generalizing from a few examples: A survey on few-shot learning. ACM Computing Surveys (CSUR), 53(3):1–34, 2020.

[195] David Warde-Farley and Yoshua Bengio. Improving generative adversarial networks with denoising feature matching. 2016.

[196] Jinliang Wei, Wei Dai, Aurick Qiao, Qirong Ho, Henggang Cui, Gregory R Ganger, Phillip B Gibbons, Garth A Gibson, and Eric P Xing. Managed communication and consistency for fast data-parallel iterative analytics. In Proceedings of the Sixth ACM Symposium on Cloud Computing, pages 381–394, 2015.

[197] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. Advances in neural information processing systems, 29:2074–2082, 2016.

[198] Davis Wertheimer, Luming Tang, and Bharath Hariharan. Few-shot classification with feature map reconstruction networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 8012–8021, 2021.

[199] Mitchell Wortsman, Kiana Ehsani, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. Learning to learn how to learn: Self-adaptive visual navigation using meta-learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 6750–6759, 2019.

[200] Jiaxiang Wu, Cong Leng, Yuhang Wang, Qinghao Hu, and Jian Cheng. Quantized convolutional neural networks for mobile devices. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 4820–4828, 2016.

[201] Jiqing Wu, Zhiwu Huang, Janine Thoma, Dinesh Acharya, and Luc Van Gool. Wasserstein divergence for gans, 2018.

[202] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V. Le. Self-training with noisy student improves imagenet classification, 2020.

[203] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks, 2017.

[204] Qiantong Xu, Gao Huang, Yang Yuan, Chuan Guo, Yu Sun, Felix Wu, and Kilian Weinberger. An empirical study on evaluation metrics of generative adversarial networks, 2018.

[205] Ting-Bing Xu and Cheng-Lin Liu. Data-distortion guided self-distillation for deep neural networks. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, pages 5565–5572, 2019.

[206] Abhay Yadav, Sohil Shah, Zheng Xu, David Jacobs, and Tom Goldstein. Stabilizing adversarial nets with prediction methods, 2018.

[207] Chenglin Yang, Lingxi Xie, Siyuan Qiao, and Alan Loddon Yuille. Training deep neural networks in generations: A more tolerant teacher educates better students. In AAAI, 2019.

[208] Guang Yang, Simiao Yu, Hao Dong, Greg Slabaugh, Pier Luigi Dragotti, Xujiong Ye, Fangde Liu, Simon Arridge, Jennifer Keegan, Yike Guo, et al. Dagan: Deep de-aliasing generative adversarial networks for fast compressed sensing mri reconstruction. IEEE transactions on medical imaging, 37(6):1310–1321, 2017.

[209] Hao Yang and Yun Zhou. Ida-gan: A novel imbalanced data augmentation gan. In 2020 25th International Conference on Pattern Recognition (ICPR), pages 8299–8305. IEEE, 2021.

[210] Ling Yang, Liangliang Li, Zilun Zhang, Xinyu Zhou, Erjin Zhou, and Yu Liu. Dpgn: Distribution propagation graph network for few-shot learning, 2020.

[211] Zichao Yang, Marcin Moczulski, Misha Denil, Nando De Freitas, Alex Smola, Le Song, and Ziyu Wang. Deep fried convnets. In Proceedings of the IEEE International Conference on Computer Vision, pages 1476–1483, 2015.

[212] Junho Yim, Donggyu Joo, Jihoon Bae, and Junmo Kim. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 7130–7138, 2017. doi: 10.1109/CVPR.2017.754.

[213] Jaemin Yoo, Minyong Cho, Taebum Kim, and U Kang. Knowledge extraction with no observable data. 2019.

[214] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. arXiv preprint arXiv:1506.03365, 2015.

[215] Xiaoli Yu, Yanyun Qu, and Ming Hong. Underwater-gan: Underwater image restoration via conditional generative adversarial network. In International Conference on Pattern Recognition, pages 66–75. Springer, 2018.

[216] Xin Yu and Fatih Murat Porikli. Ultra-resolving face images by discriminative generative networks. In ECCV, 2016.

[217] Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer, 2017.

[218] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks, 2017.

[219] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In International conference on machine learning, pages 7354–7363. PMLR, 2019.

[220] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks, 2019.

[221] Jinsong Zhang, Kun Li, Yu-Kun Lai, and Jingyu Yang. Pise: Person image synthesis and editing with decoupled gan. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 7982–7990, 2021.

[222] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 6848–6856, 2018.

[223] Yabin Zhang, Hui Tang, and Kui Jia. Fine-grained visual categorization using meta-learning optimization with sample selection of auxiliary data. In Proceedings of the european conference on computer vision (ECCV), pages 233–248, 2018.

[224] Fang Zhao, Jian Zhao, Shuicheng Yan, and Jiashi Feng. Dynamic conditional networks for few-shot learning. In Proceedings of the European Conference on Computer Vision (ECCV), September 2018.

[225] Shuxin Zheng, Qi Meng, Taifeng Wang, Wei Chen, Nenghai Yu, Zhi-Ming Ma, and Tie-Yan Liu. Asynchronous stochastic gradient descent with delay compensation. In International Conference on Machine Learning, pages 4120–4129. PMLR, 2017.

[226] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. arXiv preprint arXiv:1606.06160, 2016.

[227] Hao Zhu, Aihua Zheng, Huaibo Huang, and Ran He. High-resolution talking face generation via mutual information approximation, 12 2018.

[228] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A Efros. Generative visual manipulation on the natural image manifold. In European conference on computer vision, pages 597–613. Springer, 2016.

[229] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In Proceedings of the IEEE international conference on computer vision, pages 2223–2232, 2017.