Phoenix: Federated Learning for Generative Diffusion Model

Fiona Victoria Stanley Jothiraj

A thesis
submitted in partial fulfillment of the
requirements for the degree of

Master of Science

University of Washington

2023

Committee:
Afra Mashhadi
Brent Lagesse
Michael Stiber
William Erdly

Program Authorized to Offer Degree:
Computer Science & Software Engineering

University of Washington

Abstract

Phoenix: Federated Learning for Generative Diffusion Model

Fiona Victoria Stanley Jothiraj

Chair of the Supervisory Committee:
Afra Mashhadi
Department of Computing & Software Systems

Generative AI has made impressive strides in enabling users to create diverse and realistic visual content such as images, videos, and audio. However, training generative models on large centralized datasets can pose challenges in terms of data privacy, security, and accessibility. Federated learning is an approach that uses decentralized techniques to collaboratively train a shared deep learning model while retaining the training data on individual edge devices to preserve data privacy. This paper proposes a novel method for training a Denoising Diffusion Probabilistic Model (DDPM) across multiple data sources, using federated learning techniques. Diffusion models, a newly emerging generative model, show promising results in achieving superior quality images than Generative Adversarial Networks (GANs). Our proposed method *Phoenix* is an unconditional diffusion model that leverages strategies to improve the data diversity of generated samples even when trained on data with statistical heterogeneity (Non-IID data). We demonstrate how our approach outperforms the default diffusion model in a federated learning setting. These results are indicative that high-quality samples can be generated by maintaining data diversity, preserving privacy, and reducing communication between data sources, offering exciting new possibilities in the field of generative AI.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY

AIGC: Artificial Intelligence Generated Content

DDPM: Denoising Diffusion Probabilistic Models

GAN: Generative Adversarial Networks

VAE: Variational Autoencoder

CNN: Convolutional Neural Network

FID: Fréchet Inception Distance

IS: Inception Score

IID: Independent and Identically Distributed

GLIDE: Guided Language to Image Diffusion for Generation and Editing

CIFAR-10: Canadian Institute For Advanced Research Dataset

LSUN: Large-scale Scene UNderstanding Challenge

SGD: Stochastic Gradient Descent

ACKNOWLEDGMENTS

I would like to thank my thesis advisor, Professor Afra Mashhadi. This paper would have never been accomplished without her assistance and dedicated involvement in every step of the process. I would like to thank you very much for your support and understanding over the last year.

I would also like to show gratitude to my committee, including Professor Brent Lagesse, Professor Michael Stiber, and Professor William Erdly. I am very grateful for the technical and resource help I received that played an important part in the completion of this research work.

DEDICATION

To my loving husband, supportive parents, and family, your presence and encouragement have been my greatest motivation in pursuing this Master's thesis. With all my love and heartfelt gratitude, this accomplishment is dedicated to you.

Chapter 1

INTRODUCTION

In recent years, Generative AI (similar to AIGC or AI-Generated Content) has emerged as a rapidly growing field of research beyond the computer science community. Generative AI refers to generating high-quality synthetic content using advanced AI techniques of natural language processing (NLP), computer vision (CV), and machine learning (ML). Conventional AI models were created to perform classical, regression, or clustering tasks that helped analyze existing data. What sets generative AI is its distinctive ability to generate novel content based on the patterns and information learned from the training data. Some of the key factors that attributed to the rise in research and development of innovative tools like Stable diffusion [59], Imagen [61], ChatGPT or GPT-4 [50] are listed below,

Data Access. Deep learning typically requires large amounts of training data to improve the model performance. Complex tasks like Generative AI require even larger datasets, often amounting to terabytes or petabytes of data, to accomplish tasks like text-to-image, text-to-audio, and text-to-video generation. The increased accessibility of data, particularly through online platforms, digital archives, and databases, open data initiatives, etc. has played a pivotal role in driving the popularity and adoption of Generative AI for creative creation and consumption [77].

Neural Network Techniques. The widespread adoption of transformer-based architectures has strongly influenced the emergence of Generative AI. The earliest groundbreaking work on Transformers in 2017 [71] used attention mechanisms to capture important contextual information, thereby playing an important role in making progress in language generation and other generative tasks.

Computing Resources. In the past, simple machine learning tasks such as linear

classification or linear regression could be processed using CPUs alone, but more complex algorithms such as deep neural networks required significantly more computing resources. However, the advent of GPUs specifically optimized for rendering graphics in video games significantly improved the computational capability required to train complex neural network models. This significantly reduced the cost of high-performance GPUs, making them more affordable and accessible than ever before. This has led to major breakthroughs and accelerated the pace of research and innovation in artificial intelligence [77].

Deep generative models (DGMs), a deep learning approach in Generative AI are a special class of probabilistic neural networks that can estimate the likelihood of each image or observation to generate new samples based on the underlying distribution. Commonly used DGM techniques include generative adversarial networks (GANs), Variational Autoencoders (VAEs), and Diffusion Models. This method of unsupervised learning has potential applications, like data augmentation, code generation, text, video and audio synthesis, anomaly detection, etc. [9]. This paper focuses on better understanding this domain of DGMs and, specifically, Diffusion Models.

Although the results of generative AI are astonishing, many ethical concerns exist. There are concerns about intellectual property (IP) and ownership of generated content. Since generative AI models learn patterns and relationships from existing data, it is highly likely that copyrighted works are used without providing credits to the original creator. Another issue with such centralized AI technologies is the lack of protection of sensitive user data since global data is always stored in a centralized third-party server for training.

Pressing issues like data privacy and security prompted the development of a distribution training approach that differs from centralized training. Federated Learning (FL), an idea proposed by Google [43], aimed to create a decentralized collaborative learning scenario wherein the user data is trained locally without the need for data to be transferred to a central server. This secure training technique meant that several edge devices could be trained with rich representations of global data without any direct data interaction. The process involves training local data on each client and transmitting the learned parameters (model weights)

from each round of training to the global model for aggregation. Hence, Federated Learning (FL) offers the potential to create a high-performing global model without the need to collect training data in a central location.

This research paper presents a novel method to train a generative unconditional diffusion model in a distributed way for image synthesis. Specifically, we aim to address the problem of statistical heterogeneity in federated learning when local data is heterogeneous or non-IID (more information on IID and Non-IID Data). The paper first demonstrates generated samples' quality and mode coverage for state-of-the-art Generative Adversarial Networks (GAN) models. Next, two strategies are proposed to address statistical challenges when designing a diffusion model in a federated learning setting - Data sharing Strategy and Personalization & Threshold Filtering. The effectiveness of these strategies is evaluated through experiments to show the improvement in data quality and mode coverage over GANs. This work provides the first step in providing a promising solution to train diffusion models through federated learning for image synthesis tackling the statistical heterogeneity of clients. While security and privacy concerns are crucial for any distributed ML system, this work solely focuses on the model's ability to generate high-quality synthetic data. Hence, the work does not investigate the security and privacy-related approaches when designing and deploying an FL technique in a real-world setting. Nonetheless, it is important to acknowledge the importance of privacy and security when practically implementing generative diffusion models through federated learning, and future research should explore these issues in detail.

1.1 Motivation

1.1.1 Why Diffusion Models?

Denoising diffusion models, a new class of unsupervised generative models, has gained much interest over the last few years. They have been known to generate synthetic data with high quality and mode coverage despite their high computational resource requirements. A recent work by OpenAI [17] in 2021 provides compelling evidence that diffusion models

outperformed GANs on image synthesis by modifying their model architecture and sampling method. Moreover, DDPMs or diffusion models offer advantages over other generative models, such as GANs, due to their inherent capability to handle convergence issues and vanishing gradients, as described below.

Generative adversarial networks (GANs) are a class of generative models that learn to create new data instances that resemble the training data. GAN models contain two neural network models: a generator and a discriminator. The generator learns to generate good data, whereas the discriminator learns to distinguish the generator's fake data from real data. In other words, the two models pit against each other since the generator tries to produce really good fake samples. In contrast, the discriminator constantly penalizes the generator if it detects fake samples accurately. This often continues till the generator is too good to outwit the discriminator, thereby causing random or junk feedback to be sent from the discriminator. This causes the generator to train on random feedback, collapsing the overall model performance [2]. This causes failure to converge. On the contrary, if the discrimination is too good, then the generator could fail due to vanishing gradients, thereby making no progress in training. Additionally, the generator often repeatedly produces a limited set of outputs, as the discriminator fails to learn effective strategies to reject them, leading to the GANs getting trapped in a cycle where each iteration of the generator over-optimizes for a specific discriminator. This leads to mode collapse or lack of data distribution of generated samples. These drawbacks make GANs difficult to scale and apply to new interdisciplinary applications [64].

Real World Applications

Diffusion models are being used for diverse real-world applications [76] in computer vision, natural language processing, temporal data modeling, multi-modal learning, robust learning, and interdisciplinary applications. Specific applications within these tasks include image restoration, in-painting, video generation, anomaly detection, text-to-image generation, text-to-video generation, text-to-audio generation, molecular graph modeling, and medical image

reconstruction [15].

Multi-Modal Learning: Visual language models for Text-to-image generation [20] has garnered significant interest in recent times owing to their wide range of applications. Text-to-Image generation is the task of generating an image from a text prompt or description. Several popular works on text-to-image tasks are based on the diffusion model, including GLIDE [48], Imagen [61], Stable diffusion [59], and DALL-E2 [54].

The success of text-to-image diffusion-based models motivated the development of more relevant tasks like Text-to-Video generation. Imagen Video was among the first works that generated high-definition videos using cascaded diffusion models, a frozen T5 text encoder, and classifier-free guidance [76]. Similarly, Text-to-3D Generation has been another popular task for applications like gaming, entertainment, and simulation in robotics.

Life Science: Diffusion models are becoming increasingly popular in the medical domain with applications ranging from image-to-image translation, image reconstruction, segmentation, denoising, anomaly detection, etc. Researchers have leveraged diffusion-based medical imaging research across a variety of image modalities, including MRI, CT, PET, and Microscopic images, and across organs such as the brain, multi-organ, skin, heart, and eye.

In life science, graph-based diffusion models are being used to enhance molecule or protein generation. A study [30] explains the clinical importance of generative AI to create synthetic electronic health records that are similar in statistical properties to real data without compromising patient privacy. Studies have used diffusion models to generate histology images, translate MRI images to CT, and medical image reconstruction and enhancement. These models are also highly used in image segmentation tasks to understand better and diagnose vascular diseases. Since the diffusion models are designed on the basis of probabilistic distributions and create synthetic samples with a high degree of diversity and quality, it has been a preferred choice among the research community for generative AI-based medical tasks.

1.1.2 Why Federated Learning?

Federated learning enables the training of models in a distributed way, thereby reducing the risk of data breaches and ensuring data privacy. This secure distributed machine learning technique is being adopted in a number of applications involving autonomous vehicles, healthcare, finance, industry, IoT (Internet of Things), and urban services.

For instance, the medical community is highly concerned about the privacy of sensitive medical data. Using diffusion models to create synthetic data could violate the copyright laws of the originally trained data and compromise the privacy of training data. Therefore, a comprehensive survey of diffusion models in medical imaging [30] suggests combining the strength of federated learning and diffusion models. By training generative AI models on decentralized data, patients' sensitive data is not exposed but still captures the properties of data distribution of various clients participating in training. This means that high-quality data can still be generated with mode coverage.

Furthermore, the advantages of federated learning extend beyond healthcare. In autonomous vehicles, FL enables collaboration among multiple vehicles and data sources without sharing sensitive information, leading to enhanced model performance and robustness. In IoT and urban services, FL enables diverse organizations to leverage data from multiple sources while respecting privacy to generate synthetic data using diffusion models. This provided an opportunity to combine the strength of federated learning and diffusion models where organizations can generate diverse and high-quality synthetic data while preserving privacy and data security.

1.2 Proposed Solution

This research aims to address several important questions within the context of unconditional diffusion models trained in a horizontal federated learning setting. Since we utilize a federated learning setup, the data remains on the local devices, addressing data privacy and security issues. This makes our system the first of its kind to employ generative diffusion models

trained collaboratively in such a privacy-preserving manner.

We investigate the impact of statistical heterogeneity or imbalanced data among clients involved in the federated learning process. By understanding this impact, we propose effective strategies to overcome these challenges. Then, we compare the performance of our proposed approach, called Phoenix, with existing solutions like federated learning for GANs. Next, we quantitatively evaluate with metrics like precision to effectively measure fidelity and recall to measure diversity since they often correspond well with human judgment as mentioned in previous studies [17].

The name "Phoenix" is aptly chosen for our approach due to its transformative nature. Just as how the mythical bird rises from ashes, the diffusion model can be symbolized by its transition from chaos (pure noise) to creation (generated data).

1.3 Outline

The rest of the thesis is outlined as follows. Chapter 2 outlines the related works in the field of generative diffusion models and federated learning. Chapter 3 dives deep into the proposed methodology of federated learning for diffusion models that can handle IID and Non-IID data and its quantitative metrics. Chapter 4 focuses on the implementation, different experiments, and results. Chapter 5 focuses on the proposed work's overall discussion, limitations, implications, and future aspects.

Chapter 2

RELATED WORK

2.1 Generative Diffusion Models

"Creating noise from data is easy; Creating data from noise is generative modeling" [67]. Generative Artificial Intelligence is a new class of AI algorithms that aim to generate new content or output based on the vast amount of data that it has been trained on. Researchers have highly sought them due to their exceptionally high-quality data samples generated by the models. This new family of models, also known as deep generative models (DGMs), combines the power of generative models and deep neural networks. Some popular DGMs include variational autoencoders (VAEs), generative adversarial networks (GANs), and diffusion models. Research in 2015 [66] originally introduced the idea of diffusion probabilistic models that aim to learn how to reverse a gradual multi-step noising or decaying process. The diffusion probabilistic model or diffusion model is a parameterized Markov chain that produces samples after finite samples. Transitions of this chain are taught to learn how to reverse a diffusion process when the chain gradually adds noise to the data until all information is destroyed. The overall step consists of 2 processes:

- A predefined forward diffusion process q that gradually destroys the image by adding gaussian noise to it until only pure noise remains.
- A reverse learned denoising diffusion process p where a neural network is trained to denoise the image gradually till the actual image is recovered.

The DDPM Model [26] shows that extremely high-quality samples could be generated on the CIFAR-10 and LSUN datasets. On the unconditional dataset, they obtained an Inception score of 9.46 and an FID score of 3.17 on the train set and 5.24 on the test set. Their work

solely relied on predicting the noise found in the image, not the mean, and fixing the variance of the gaussian noise in the forward pass. Further improvements on the DDPM model [49] showed the importance of learning variances of the reverse diffusion process that allows for sampling with fewer forward passes without affecting sample quality.

2.2 Evaluating Diffusion Models

2.2.1 Loglikelihood

One of the widely used metrics in generative modeling is Loglikelihood which aims to capture all the modes of data distribution. [49] defines a new hybrid objective function that achieved its best log-likelihood by optimizing the Kullback–Leibler divergence (KL) divergence [28] between the Gaussian distributions. Another important parameter that boosts loglikelihood is changing the number of diffusion steps, T. Since the forward and reverse process occurs for a finite number of time steps T, this parameter controls the number of steps requires to end up with an isotropic Gaussian distribution at t=T in a gradual process.

2.2.2 Inception score

Inception score (IS) [63] [4] [5] is another metric to evaluate the quality of image generative models. The IS score uses the Inception V3 Network pre-trained on the ImageNet dataset [16] to calculate the single floating point number score. The metric evaluates the following criteria: (i) The generative model should be able to generate a diverse set of images from all the different classes in CIFAR-10 (ii) The generated images should contain clear objects and not blurry objects. i.e., the score measures how well a model captures the full class distribution while still producing individual samples that are convincing enough to belong to a single class. When both of these criteria are satisfied, the KL distance between the generated images and the conditional class distribution computed via the inception network is maximized. A KL distance yields a high inception score, indicating that the diffusion model can generate many distinct images.

2.2.3 Fréchet Inception Distance score

The Fréchet inception distance score or FID Score [68] [25] [6], for short, measures the Gaussian distribution between real and generated images. The metric assumes that the features extracted from the pre-trained Inception V3 Network have a Gaussian distribution. A lower FID score indicates the generated images are closer in distribution to the real images. Hence, an FID score closer to zero indicates the images generated by the diffusion model are high in quality.

2.2.4 Precision and Recall

Although proposed measures such as the Inception Score [63], Fréchet Inception Distance (FID) [68] have shown promising results in terms of visual fidelity of the generated samples, they do not account enough for the diversity of the images. The introduction of metrics such as precision and recall by [62] and [32] aims to solve this shortcoming. Precision denotes what fraction of the generated samples is realistic, and recall measures what fraction of the training set manifold the generator covers. Both these metrics can effectively measure the fidelity and diversity of the samples.

2.3 Generative Adversarial Networks (GANs)

One of the first generative models was proposed in 2014 [23] called the adversarial nets framework. The framework consists of a generative model that acts against an adversary discriminative model, which learns to distinguish whether a data sample is indeed from the original data distribution. It was analogous to a counterfeit team who produces counterfeit currency notes and tries to get away with it (i.e. generative model) and the police who detect the counterfeit currency notes (i.e., discriminative model). Both the generator and discriminator (Figure 2.1) are deep neural networks, and the output of the generator is fed as input to the discriminator. During backpropagation, the weights of the generator are updated through the discriminator's classification task.

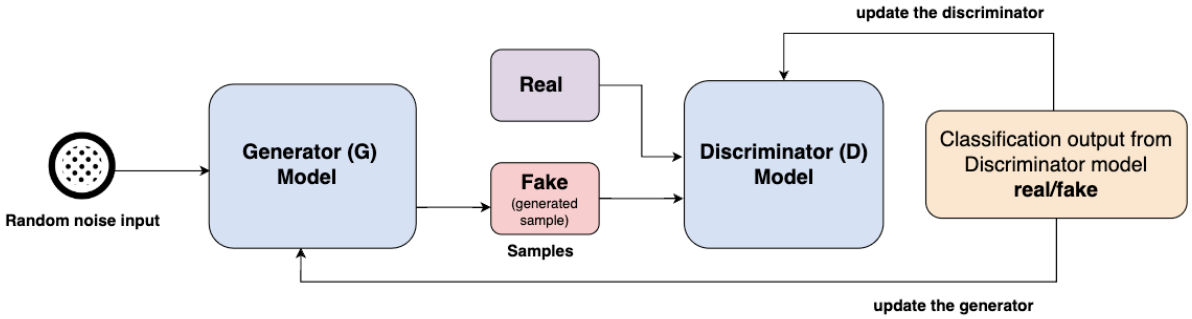


Figure 2.1: Basic architecture of a GAN model

2.4 Federated Learning

Federated learning (FL) [42] [41] is an emerging Machine learning technique that enables ML models to be trained in distributed data on edge devices without being stored on a central server. In FL, each client retains its local training dataset, ensuring that data is not transmitted to a central server. Instead, the clients (or individual devices) are trained on the neural network models, and only the computed model updates are communicated. The federated learning technique can thus significantly reduce the privacy and security risks in comparison to the traditional centralized technique. As per the White House Report [57], the FL technique best makes use of principles such as data minimization and security.

While Federated learning offers privacy and security benefits over traditional methods, it also faces certain challenges. The training data present in clients are typically personalized and hence not representative of the global data distribution. This poses a problem of non-IID data, which is described in detail in section 2.4.3. Since the clients act as edge devices, there is typically limited communication, or they are frequently offline; hence, some devices train faster than others.

2.4.1 Types of Federated Learning Systems

Based on the degree of overleaf of dataset features in different clients, FL methods are typically divided into horizontal federated learning, vertical federated learning, and federated transfer learning [74]. Figure 2.2 illustrates their key differences.

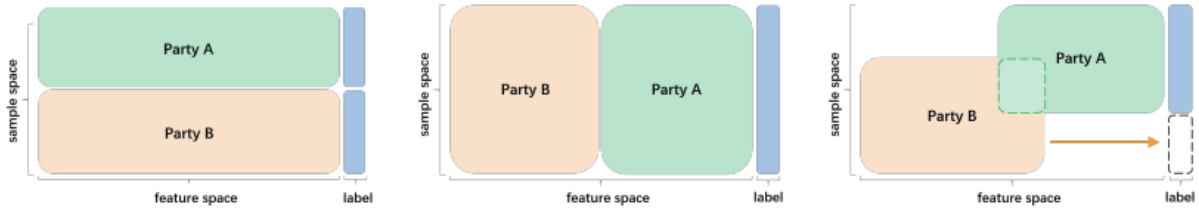


Figure 2.2: Types of federated learning systems [39] **Left**: Horizontal FL, **Middle**: Vertical FL, **Right**: Federated Transfer Learning

- **Horizontal Federated Learning:** Multiple edge devices or clients have the same type of data, i.e., they share the same feature space but differ in the type of samples they contain.
- **Vertical Federated Learning:** Multiple edge devices or clients have different data characteristics, i.e., they share the same samples but differ in the type of feature space.
- **Federated Transfer Learning:** Federated Transfer Learning is a mix of the above-mentioned methods where the datasets of each client differ in both the feature and the sample space.

2.4.2 Federated Optimization by Model Aggregation

After each round of the local model, the computed model weights are sent to the central server for aggregation. Various model aggregation techniques are used depending on the nature of the local data and the system behavior.

- The baseline aggregation technique FedSGD (Federated stochastic gradient descent) requires that each client locally takes one step of gradient descent on the current model using its local data. The server then takes a weighted average of the resulting models.

- The widely used aggregation technique FedAvg [42] is derived from the baseline implementation FedSGD where each client iterates the local update multiple times before performing the averaging step. However, this technique does not allow participating devices to perform variable amounts of local work. Another drawback of this technique is that devices that fail to compute E epochs within a specified time window (Stragglers) are dropped.

- FedProx [35], a more recently designed aggregation technique is more stable, has an accurate convergence behavior, and aims to solve system and statistical heterogeneity. In FedProx, variable amounts of work can be performed across client devices based on their availability of system resources and then aggregates the partial solutions sent from these stragglers. Another feature of FedProx is the addition of a proximal term to the objective function to improve the stability of the model. The proximal term penalizes the large deviation between the local and global models.

- The QFedAvg [36] algorithm achieves a more fair (or more uniform) accuracy distribution across devices in the federated network. High fairness is achieved by reweighting the objective function, i.e., assigning higher weights to devices with poor performance, so that the distribution of accuracies in the network shifts towards more uniformity. However, the reweighting is done dynamically since the performance of devices cannot be evaluated prior to training.

- FedOptim [56] introduces optimizers such as AdaGrad, Adam, and Yogi for adaptive optimization. Here, the server chooses a specific optimization function (one of ADAGRAD, YOGI, or ADAM), and the client chooses SGD. By doing this, the convergence

of the FL networks is greatly improved. Apart from the above-mentioned algorithms, there are several other model aggregation strategies. These techniques differ in how they combine the local model updates and adjust the learning rates for each client.

2.4.3 IID and Non-IID Data

It is common in a federated learning network to have clients with different data distributions. For instance, each patient's medical history can be treated as a distinct dataset in a healthcare dataset. The data within these datasets is likely to have been drawn from different distributions. Such forms of data are called Non-IID data or Non-Independent and identically distributed [24]. Although IID data is the simplest form of data distribution to deal with, it is unrealistic to assume that the data present in edge devices is IID since the clients are unique.

Several efforts are being carried out to deal with this highly skewed non-IID data. Survey research [81] explains that the most commonly used methods to handle non-IID data are the data-based, algorithmic, and system-based approaches. As the name suggests, the data-based approach manipulates the way client data is preprocessed before training. The data sharing method [80] uses a warm-up global model and a small portion of global data to be trained alongside the client data. This ensures that the diversity of generated images is not lost after training. Another method of data augmentation helps increase the diversity of training data by means of random transformations and knowledge transfer to mitigate local data imbalance and diversity issues in FL. The algorithmic-based approach tests various learning techniques like knowledge distillation, local fine-tuning, personalization layers, and multi-task learning to handle non-IID data. The system-based approach uses client-based clustering methods to enable a better understanding of client information in a heterogeneous data environment.

2.5 Federated Learning with GANs

Generative Adversarial Networks (GANs) as mentioned in section 2.3 [23] are generative models designed to create new data based on the training set. It comprises two neural networks - a generator and a discriminator that pit against each other till the discriminator model can no longer detect fake images created by the generator. In recent years, much research has been made to use GANs in a federated learning setting to personalize the task of clients. Recent research has explored the utilization of GANs in a federated learning setting to personalize client tasks and heterogeneous data distribution problems (Table 2.1). However, GANs have limitations such as mode collapse, convergence issues, and scalability challenges. To address these drawbacks, Diffusion Models have emerged as a more suitable approach for generative AI tasks, effectively overcoming these challenges.

Table 2.1: State-of-the-art works that use Federated learning with GANs

Model	**Year**	**Approach**	**Dataset**	**Metrics**
SDA-FL [37]	2022	GANs with Differential privacy	Fashion MNIST and CIFAR 10	FID, Accuracy
PerFED-GAN [12]	2022	Personalized models with differential privacy	CIFAR 10, FEMNIST, CIFAR 100	Relative test accuracy (RTA)
ACGAN [22]	2021	GAN with distributed data	CIFAR 10	FID, Visual examination, Generator and Discriminator Loss
cGAN [19]	2020	Conditional GANs that can generate images given the label	MNIST and CIFAR-10	Classification score and Earth Mover's Distance (EMD)

Chapter 3

METHODOLOGY

3.1 Neural Network Models

3.1.1 Diffusion Model

A denoising diffusion model is a type of generative model that transforms noisy data, typically originating from a Gaussian distribution, into high-quality samples of the desired data distribution. A neural network backed by a U-Net model architecture [60] is typically used to achieve this. This model learns to gradually denoise the data, resulting in a proper image.

The two-step process involved in diffusion models is illustrated in Figure 3.1 and is explained as follows:

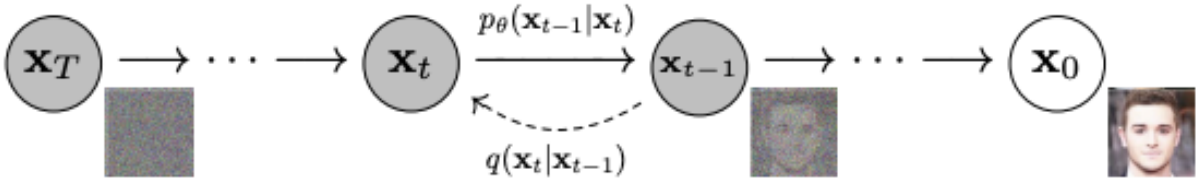


Figure 3.1: Directed Graphical Model of Diffusion Model [26]

- **Forward diffusion process**: Given an image x0, the forward process aims to add Gaussian noise at each time step t (pre-defined) till an isotropic Gaussian distribution is achieved. As the step size t increases, the image loses all its distinguishable features, eventually becoming complete noise.

  If $q(x_0)$ is the real data distribution of real images and a sample image from this distribution is $x_0 \sim q(x_0)$, then the forward diffusion process can be denoted as $q(x_t \mid$

x_{t-1}). Gaussian noise is added at every time step t based on a variance scheduler β, $0 < \beta_1 < \beta_2 < \beta_3 < .... < \beta_T < 1$ as

$$q(x_t \mid x_{t-1}) = N(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I) \tag{3.1}$$

The term $\sqrt{1-\beta_t}x_{t-1}$ is the mean of the Gaussian distribution denoted by μ_t and β_t is the variance of the Gaussian distribution denoted by σ_t^2 that is based on sampling $\epsilon \sim N(0, I)$. Hence the next image can be calculated by

$$x_t = \sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t}\epsilon \tag{3.2}$$

The forward diffusion process starts with an image x_0 and adds noise in T steps $x_1, ..., x_t, ..., x_T$ until only pure Gaussian noise remains.

- **Reverse diffusion process**: A neural network (typically U-Net) is trained to denoise the images containing pure noise till the actual image is retrieved.

  Based on Figure 3.1, if we knew the conditional distribution $p(x_{t-1} \mid x_t)$, we could effortlessly perform the reverse process to reconstruct the original image. However, we do not know $p(x_{t-1} \mid x_t)$ since that would require the knowledge of the distribution of all training images. Hence, we harness the power of Neural Networks to approximately learn the conditional probability distribution $p_\theta(x_{t-1} \mid x_t)$, where θ are the parameters to be learned by the neural network when it is updated by a gradient descent [58]. The neural networks learns the mean and variance of the conditional probability distributions to recover or reconstruct an image.

Hyperparameters

- **Variance Scheduler**: The rate at which noise is added to the images is controlled by the **variance scheduler** β. This pre-defined parameter can be a constant integer or chosen as a scheduler over T timesteps. Earlier works of Denoising Diffusion Probabilistic Models (DDPM) [26] use a linear schedule from $\beta_1 = 10^{-4}$ to $\beta_T = 0.02$.

While the linear noise schedule has shown to be effective for generating high-resolution images, its performance may be limited when applied to smaller image resolutions like 32x32 and 64x64. Furthermore, the linear schedule made the forward decaying process too noisy, which didn't improve sample quality. The Improved Denoising Diffusion Probabilistic Model (DDPM) proposed in 2021 [49] aims to address the problem of reduced performance (FID Score) when trained with a linear schedule. The improved DDPM proposed a cosine scheduler with a linear drop-off in the middle of the decaying process but changes minimally at the start and end of the process (i.e. t = 0 and t = T). By doing this, useful information is destroyed slower, thereby retaining more information during the reverse process for accurate prediction.

- **Diffusion Steps**: The number of **decaying or diffusion steps** is controlled by T. Earlier works [26] use the standard T = 1000; however, the improved version of DDPM [49] trained the models with 4000 diffusion steps. However, since the time taken to produce a single image takes several minutes on a GPU, they proposed reducing the number of sampling steps to 25, 50, 100, 200, 400, 1000, and 4000.

- **Learning Rate**: The rate at which the weights are updated during training is controlled by Learning Rate (lr) or step size. This pre-set hyperparameter ranges between 0.0 to 1.0. Careful selection of the learning rate is necessary for reaching the optimal solution since a large learning rate can cause faster convergence leading to a sub-optimal solution. Meanwhile, a small learning rate can cause the process to get stuck at a local minima (during gradient descent).

3.1.2 Conditional Vs. Unconditional Diffusion Models

Conditional and unconditional diffusion models are two distinct approaches within the field of generative modeling. Unconditional diffusion models are known to generate data samples without any specific conditioning or external factors. These models learn the underlying

probability distribution of the training data and generate new samples (based on seed value) that resemble the training data. On the other hand, conditional diffusion models contain additional information or conditioning factors during the generation process. These models consider external inputs, such as labels or text prompts to generate samples that are conditioned on the provided information. This enables the generation of data samples that conform to desired characteristics or constraints specified by the user.

The choice between conditional and unconditional diffusion models depends on the specific task and requirements at hand. In our work, we concentrate on utilizing unconditional diffusion models, as they serve as a starting point for future work to incorporate conditional factors.

3.1.3 U-Net Model

U-Net is a convolutional neural network (CNN) architecture originally created for precise medical image segmentation. However, U-Nets [60] is a well-suited architecture for diffusion models. This is due to the presence of bottlenecks and skip connections (residual connections) that provide representations of varied granularity and improve the gradient flow that is used to denoise images. The network architecture is illustrated in Figure 3.2. The architecture consists of a contracting path on the left and an expansive one on the right. The left side (or downsampling) path contains typical basic blocks found in a convolution neural network: repeated 3x3 unpadded convolutions followed by a rectified linear unit (ReLU) and a 2x2 max pool operation with stride 2. At every downsampling step, the number of feature channels is doubled (say, 64 to 128 to 256, and so on). On the right side (or upsampling) path, there are repeated 2x2 up-convolutions that halve the number of feature channels, followed by a concatenation, 3x3 convolution, and a rectified linear unit (ReLU). There are a total of 23 convolutional layers in this network. The architecture has a significant constraint regarding input data size, as it can only handle images with a maximum size of 572 x 572 pixels. This limitation poses challenges when generating larger-sized images.

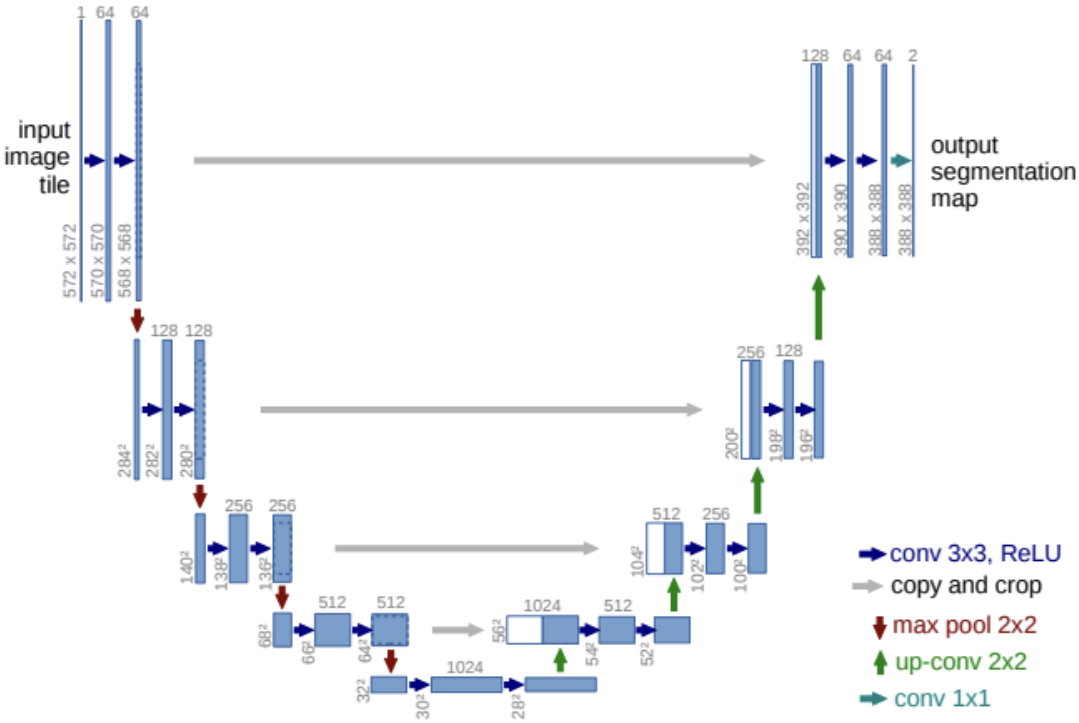


Figure 3.2: U-Net Architecture [60]

3.2 Phoenix Algorithm

We propose *Phoenix*, a novel diffusion neural network model that is specifically designed to be trained in a federated learning setting. The primary focus of our work is to explore the implications and effects of utilizing such a model on two distinct data scenarios: independent and identically distributed (IID) data and non-IID data.

3.2.1 Default Mode

The default mode of the *Phoenix* framework uses K clients with E number of local epochs and N server rounds. Initially, the image data to be used (CIFAR-10 in our example) is split into train and test data. Further, the training data is split into K clients; each has an equal distribution of classes in the case of IID Data and non-equal distribution of classes in the

case of Non-IID Data. All clients are trained locally with their own portion of data using diffusion models (with U-Net backbone model architecture). At the end of E local epochs, the weights are sent back to be averaged in the global server before the next round of training begins.

The algorithm depicted in Figure 3.3 can be explained as follows: The server initiates the process by initializing the weights of the diffusion model and establishing communication with the clients for t server rounds. During each round, the ClientUpdate procedure is executed on k clients. The local training data is partitioned into batches, and the diffusion model is trained for E epochs on each client. The optimization process employs gradient descent to update the model's weights, which are then sent back to the server for aggregation. Image samples are generated at specific epochs throughout the training process to enable visual quality inspection. These samples are saved locally on the device for further analysis and evaluation. This iterative procedure ensures that the diffusion model continually improves its performance and generates high-quality synthetic data.

3.2.2 Data Sharing Strategy

One way to solve the diversity issues faced by Non-IID data is to utilize a data-sharing strategy that uses a small subset of global data shared between all devices before federated learning training. By sharing a small subset of global data, the overlap of data distribution across clients can be increased, thereby improving the overall model performance. Prior experiments [80] show that when using a CIFAR-10 dataset trained on CNN models, they achieved a 30% increase in test accuracy when using 5% of the globally shared data.

The stepwise process of data sharing strategy for a CIFAR-10 dataset is explained as follows,

1. The CIFAR-10 training set, comprising 50,000 image samples, is divided into two parts: the Client (C) with 40,000 examples and the Server (S) with 10,000 examples.

2. The Client (C) part is split into four or ten clients chosen as the number of clients that

Algorithm : Federated Deep Generative Diffusion Model (FedDM)

Input: K clients are indexed by k; B is the training batch size; E is the number of local epochs, and η is the learning rate

ServerExecutes:

initialize weights w0
for each server round t = 1, 2, . . . **do** *// communication rounds*
m ← max(C · K, 1)
St ← (random set of m clients)
for each client k ∈ St **in parallel do**
w k t+1 ← ClientUpdate(k, wt)
mt ← Σ k∈St nk
wt+1 ← Σ k∈St nk / mt wk t+1
save weights

ClientUpdate(k,w): *// run on client k*
B ← (split Pk into batches of size B) *// Pk - set of indexes of data points on client k*
for each local epoch i from 1 to E **do**
for batch b ∈ B **do**

repeat *// diffusion model training*
x0 ~ q(x0) *// x is image on client k, q is the noisy image*
t ~ Uniform({1, . . . , T}) *// T is the total decay steps*
∈ ~ N (0, I)
Take gradient descent step on
$\nabla\theta \; \| \in - \in\theta(\sqrt{\alpha}^{-}tx0 + \sqrt{1 - \alpha}^{-}t, t)\| 2$
until converged

if (E + 1) mod 10 == 0 or (E == E - 1) **then** *// sampling at specified intervals*
xT ~ N (0, I)
for t = T, . . . , 1 **do**
z ~ N (0, I) if t > 1, else z = 0
$xt{-}1 = \sqrt{1}/ \alpha t \; (xt - 1{-}\alpha^{-}t / \sqrt{1}{-}\alpha t \; \in\theta(xt, t)) \; + \sigma tz$
end for
save x0

return weights to server

Figure 3.3: *Phoenix* default training algorithm

participate in the federated learning training process. The partition is usually a case of 1-class or 2-class non-IID Data. ie. The number of classes in a client is one or utmost two, which helps simulate the scenario of non-IID Data.

3. The Server (S) part is split into ten groups as a globally shared data (G) based on a factor β that is quantified by $\beta = \frac{|G|}{|C|} \times 100\%$. β can range from 2.5% to 25%.

4. Next, the global warmup model is trained on globally shared data (G) for the required number of epochs till it reaches the desired accuracy.

5. From the globally shared data (G), a fraction of data known as α is merged with every client part (C) for federated learning training. α of zero means that no data from G is shared with client, and α of 100 means that all the data from G is merged with every client.

The different kinds of experiments that can be tested with a data-sharing strategy are shown in Table 3.1, and the illustration of the same is shown in Figure 3.4

Table 3.1: Examples of experiments of data sharing strategy

β	**G**	**C**	α
5	2,000	40,000	0-100%
10	4,000	40,000	0-100%
15	6,000	40,000	0-100%
20	8,000	40,000	0-100%
25	10,000	40,000	0-100%

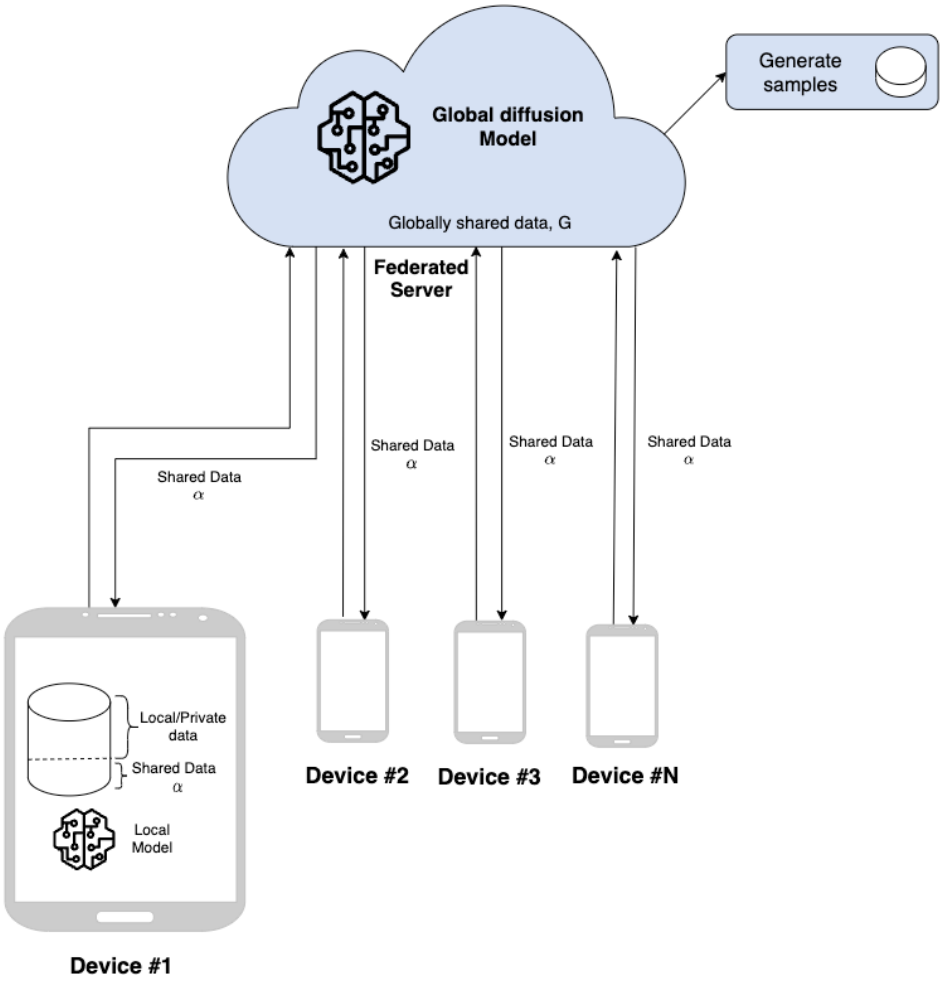


Figure 3.4: Illustration of Data sharing strategy in FL setting

3.2.3 Personalization layers & Threshold Filtering

Personalization layers

Statistical Heterogeneity [3] i.e., data distribution is different for different clients and typically impacts the performance of ML models. The personalization layer-based approach [46] is one popular way of handling different clients with very different data distributions. It aids in the customization of models to individual users or devices without compromising user privacy. To better capture the features/properties each client learns, the deep neural networks are divided into personalization and base layers. The base layers are typically co-trained by the clients and take part in the model aggregation stage, whereas the personalization layers [18] do not leave the individual clients.

We work with U-Net, a convolution neural network, as the backbone model architecture for diffusion models. U-Net contains a total of eight basic blocks: 4 blocks in the encoder side, upsampling, or contracting path. The block consists of two convolutional layers, a ReLU activation function, and a max pooling layer. Four basic blocks are on the decoder side, downsampling or expanding path of the U-Net. The block consists of upsampling layers, concatenation, and two convolutional layers. To better understand the effects of personalization layers, we consider the last basic block containing convolution layers to be the personalization layers.

Threshold Filtering

We propose another strategy to improve the overall model performance in the global model by means of a threshold filtering technique. In the context of federated learning, threshold filtering can be used to remove (or) disconnect underperforming clients from the active training process (Figure 3.5). Precision and recall are the performance metrics to determine if a client is performing below optimal levels. Clients that do not meet the pre-set/average values of the threshold are removed either temporarily or permanently. This can be achieved in two ways: excluding from a particular server communication round or removing from the

training altogether. There are several advantages to this method, as listed below,

- Improved Global Model Performance: By removing underperforming clients, the global model performance may be less impacted by noise or biased data.
- Faster Convergence: Underperforming clients in a training process require more server communication rounds to achieve optimal levels of performance. By removing such clients, the models are trained on high-quality, relevant data, thereby training efficient and faster convergence of the model.
- Optimized Resource Utilization: The time and resources spent in training clients that perform at suboptimal levels can be eliminated, which is a significant overhead in Federated learning.

In our method of threshold filtering, we use a warning mechanism to disconnect clients only if they underperform consistently for more than two local training. By implementing this two-strike approach of threshold filtering, the federated learning system can help improve performance, efficiency, and fairness while still maintaining its privacy and scalability advantages. This method of threshold filtering technique offers several other benefits as listed below,

- Robustness: We give all clients a second chance and do not penalize clients immediately for temporary performance drops due to factors such as fluctuations in data quality.
- Fairness: By allowing underperforming clients to participate in further training rounds before being removed, there is a fair evaluation of performance. This eliminates premature exclusion that could arise due to connection/communication/data issues.
- Dynamic Adaption: The constant monitoring of a federated learning system helps keep track of clients' performance even when the client population or data distribution changes.

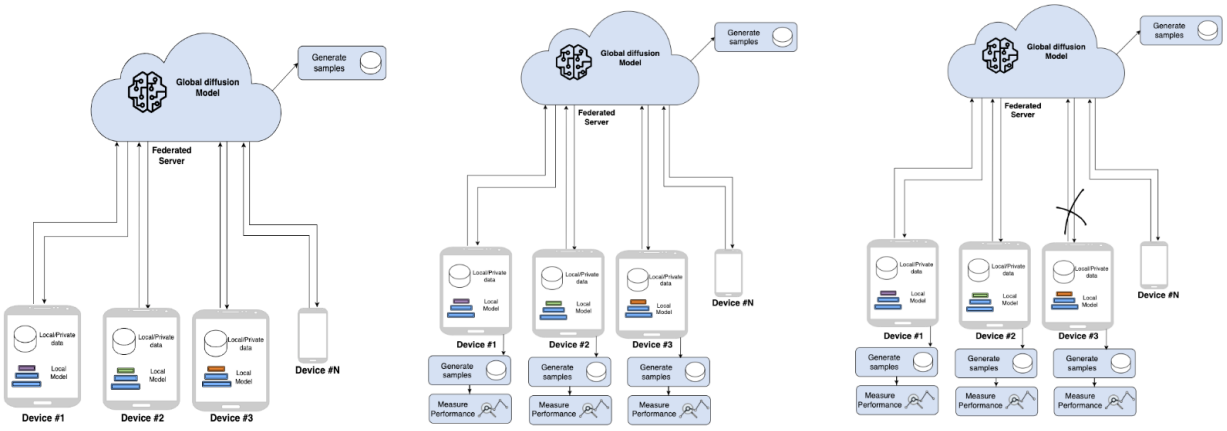


Figure 3.5: Illustration of Personalization layers & Threshold Filtering strategy in FL setting **Step 1**: Personalization layers, **Step 2**: Generate samples to monitor performance and **Step 3**: Disconnect underperforming clients

3.3 Dataset

The generative diffusion models were trained on the CIFAR-10 dataset [31]. The CIFAR-10 dataset consists of 60k samples in 10 different classes such as airplane, bird, cat, deer, dog, horse etc. These 3-channel RGB images of size 32*32 are split into 50k training images and 10k test images. The *Phoenix* framework trains the diffusion models using the 50k samples and evaluates the performance of these models using the 10k test images. CIFAR-10 is a commonly chosen benchmark dataset for evaluating neural network models in various domains due to various reasons as follows,

- Data was manually annotated by humans, which gives it credibility.
- Images are low resolution, allowing researchers to try their various methods/algorithms as fast as possible.
- Most of the computer vision tasks use CIFAR-10 as the benchmark data making it easier to compare proven methods.

3.4 Data Preparation

3.4.1 Setting up IID Data

The Independent and Identically Distributed (IID) data is simulated during training. The CIFAR-10 dataset is distributed among the k-clients as the diversity of image samples in each client is almost equal. Although the occurrence of IID data distribution in real-world applications is not practical, this is done to understand the behavior of diffusion model systems.

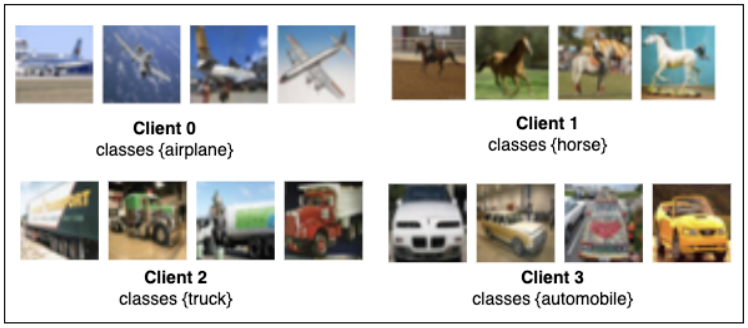


Figure 3.6: Example of 1 class Non-IID simulated data with Label Distribution Skew

3.4.2 Setting up Non-IID Data

There exist different ways to make data behave non-identically distributed across clients [34] [33].

Feature distribution skew: The feature distribution varies across parties. For instance, dogs may vary in fur color and pattern.

Label distribution skew: Given a specific label, each client owns highly different amounts of samples corresponding to that label. For instance, certain shoes are used only in a particular geographic location and not others.

Quantity skew: Given the dataset for a specific client, the amount of data between different labels is highly unbalanced.

The experiments in *Phoenix* use label distribution skew to simulate the scenario of [80]

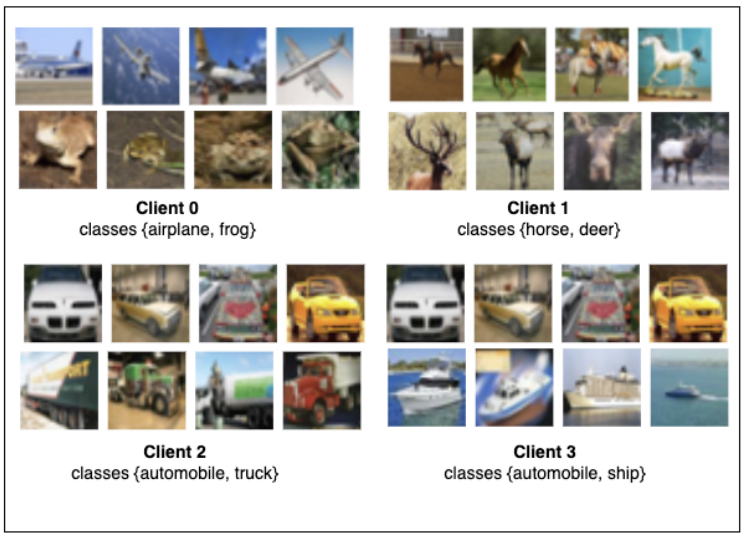


Figure 3.7: Example of 2 class Non-IID simulated data with Label Distribution Skew

1 class Non-IID data. A 1-class non-IID case makes sure that each client contains samples belonging only to a single class. Figure 3.6 illustrates an example of a 1 class non-iid simulated data using CIFAR-10. A 2-class non-IID case makes sure that each client contains samples belonging to two classes. Figure 3.7 illustrates an example of a 2-class non-iid simulated data using CIFAR-10.

3.5 Quality Metrics of Generated Images

Inception score: Inception Score (IS) is one of the most widely used metrics in generative modeling for images. The IS score is used to evaluate generative image quality as described by [63] in 2016. It uses a pre-trained Inception v3 Model (trained on ImageNet dataset). It aims to calculate the Kullback-Leibler (KL) divergence between probability distributions to measure how well they correlate well with human evaluation. Although the inception score measures the performance of generative models that correlate well with human judgment, they suffer from the fact that statistics of real-world samples are not used to compare against the statistics of these generated samples.

Fréchet Inception Distance score: The FID Score [25] introduced in 2018 is consistent with human judgment and disturbances found in images. This measure compares the distribution of real images (a.k.a ground truth) and the distribution of generated images. The comparisons are performed on the mean and standard deviation of the feature layers in Inception V3 Network and not a shallow pixel-by-pixel comparison of images. Since deep layers are used, they correspond well to real-world objects, thereby mimicking human perception of the similarity of images.

Precision and Recall: Although the FID score is the state-of-the-art metric for measuring the quality of generative adversarial networks, and likelihood-based models, it does not measure the variation of samples. A work done by NVIDIA in 2019 designed an improved precision and recall metric using k-nearest neighbors [32] [69].

A way to determine precision is to assess whether each generated image falls within the estimated manifold of real images. Another method involves calculating the probability that a randomly selected image from Pg (red) exists within Pr (blue) range. Essentially, Precision measures the proportion of generated images that possess a realistic quality.

Symmetrically, a way to determine recall is to assess whether each real image falls within the estimated manifold of generated images. Another method involves calculating the probability that a randomly selected image from Pr (blue) exists within Pg (red) range. Recall measures the fraction of training data covered by the generator (i.e. diversity or variation of generated images). Figure 3.8 illustrates the improved precision and recall metrics for distributions.

Classification Distribution: To better evaluate the diversity of generated samples, it is necessary to use measures like precision and recall that goes beyond just visual inspection. Another alternative measure we propose is the use of a pre-trained state-of-the-art classification model [73] LaNet that quotes a top-1 accuracy of 99.03% on the CIFAR-10 dataset using the NASNet search space.

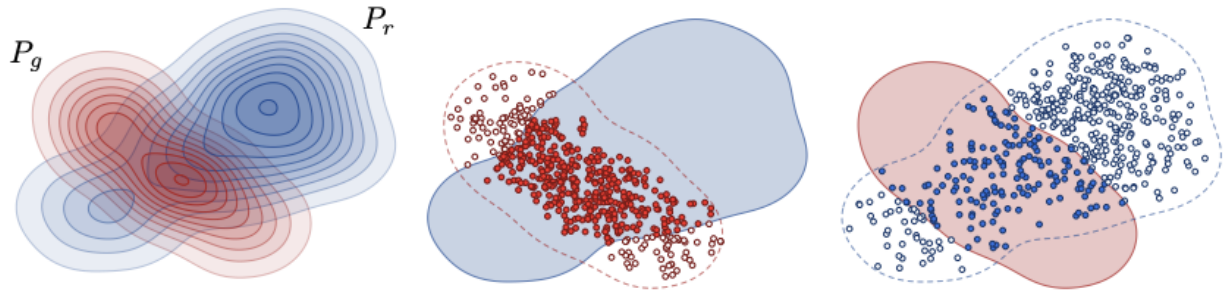


Figure 3.8: Precision and Recall for distributions [32] **Left**: Distribution of real images (blue) and the distribution of generated images (red), **Middle**: Precision, **Right**: Recall

We use the LANet model to classify k-generated samples and k-test set samples. By comparing the classification results of the generated samples with those from the test set, we can effectively measure the diversity of generated samples. If the generated samples are diverse and realistic, we would expect to see a similar distribution of classes between the two sets (generated and test set). However, if the generated samples are not as diverse as expected, we would expect to see a skewed distribution of classes. Overall, using a pre-trained classification model is an objective measure to evaluate generative models' performance.

3.6 ML Frameworks & Toolkits

The successful implementation of the *Phoenix* Algorithm heavily relies on the utilization of various Python based ML frameworks and libraries. The following libraries played an important role in different aspects of the implementation:

- **Flower** - A Federated Learning framework [7] that offers support to execute large-scale FL experiments and considers the varied, rich, heterogeneous FL device scenarios. This end-to-end FL framework enables development and seamless transition from experimental research using simulation to system research on a large number of edge devices.

- **HuggingFace Diffuser** - Diffusers [72] is a versatile library for pre-trained diffusion models, enabling the generation of images, audio, and 3D structures of molecules. It offers a modular toolbox with a focus on usability, simplicity, and customizability. The library comprises state-of-the-art diffusion pipelines for easy inference, interchangeable noise schedulers for fine-tuning generation speed and quality, and pre-trained models that can be combined to build customized end-to-end diffusion systems.

- **PyTorch** - A dynamic and flexible open-source machine learning library that is built upon the Torch framework. It serves as a powerful tool for a wide range of applications, including natural language processing, computer vision, and deep learning. PyTorch provides an intuitive and easy-to-use interface, making building and training machine learning models convenient. It's versatility and extensive documentation contribute to its popularity among researchers and practitioners [51]. Our work solely uses PyTorch during the data pre-processing phase and training.

- **Weights & Biases** - Weights & Biases (W&B) [8] is a comprehensive machine learning platform that helps developers build better models faster. With W&B's lightweight and interoperable tools, users can easily track experiments, version and iterate on datasets, evaluate model performance, reproduce models, visualize results, and share findings with others. Users can easily interact with Weights and Biases API to log runs of the desired project. It currently supports visualization of model training from Keras/Tensorflow, PyTorch, PyTorch Lightning, Transformers, XGBoost, and Sci-Kit Learn.

- **DCGAN** - The DCGAN is an open-source GitHub repository that implements a deep convolutional GAN model trained on the CIFAR10 dataset using PyTorch. Based on the reference [1], this work serves as our baseline comparison for evaluating GANs. The following chapter provides a concise overview of this implementation and its significance in our research.

Chapter 4

EXPERIMENTS AND RESULTS

4.1 Comparing with GANs

A recent work by OpenAI in 2021 [17] showed that denoising diffusion models could achieve higher image sample quality than the existing state-of-the-art generative models like GANs. We compare our proposed methods of diffusion model trained in a federated learning setting with that of GANs. Table 2.1 show some of the existing words that use federated learning with GANs. For the purpose of comparison, we chose the GAN architecture - DCGAN [53]. Deep convolutional generative adversarial networks or DCGAN for short, scales up GANs using CNNs which were unsuccessful before. DCGAN is a good baseline since it addresses several limitations of the original GAN architecture, and improves data quality and stability during training. Also, DCGAN is still one of the popular choices of GAN architectures for image generation tasks. Figure A.1 shows the full neural network of the DCGAN generator that was used to train and generate synthetic data based on the CIFAR-10 dataset. The core approach of DCGAN can be explained by the following modifications in the architecture topology as mentioned in their work [53],

- Batchnorm operation in both the generator and the discriminator networks
- ReLU activation in the generator networks for all layers except output which uses TanH
- LeakyReLU activation in all layers of the discriminator network
- Replacing pooling layers with fractional-strided convolutions in the generator
- Replacing pooling layers with strided convolutions in the discriminator
- Removing fully connected layers as architecture gets deeper

We experiment with three different training modes, centralized machine learning [1] and federated learning with IID data and federated learning with Non-IID Data, using DCGAN as our backbone architecture. We use the CIFAR-10 dataset containing 50,000 training data for these experiments. Pre-processing was performed on all training images to resize, normalize and convert the images to tensor. All models were trained using a mini-batch size of 128 and optimized using the Adam optimizer. We set the learning rate as 0.0002 and the momentum term β as 0.5 to stabilize training. For federated learning, we used a 2 class Non-IID simulated data [27] and set the following parameters prior to training: Server Rounds = 10, Aggregation Strategy = FedAvg, Training Set = 50k samples, Optimizer = Adam, lr=0.0002, Epochs = 50. The results of the training are tabulated in Table 4.1.

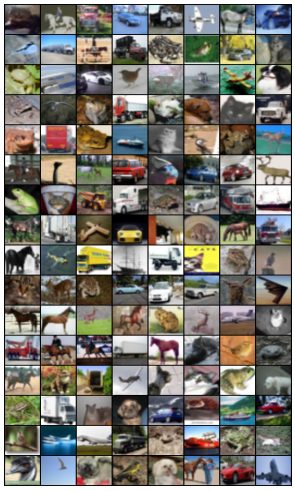
Additionally, in order to visualize the coverage of classes from synthesized data, we perform a classification task on the 10,000 generated samples. An image classification task using the LaNet model was performed on our generated sample since this model was also trained on the CIFAR-10 dataset. This open-source project from Facebook Research has been ranked #17 in image classification on CIFAR-10 with an accuracy of 99.03, containing over 44.1M parameters. We use the test set of the CIFAR-10 dataset containing 10,000 samples with approximately 1000 samples per class, such as airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. Our assumption is that the LaNet model's classification task on these 10,000 samples would also result in the same result - 1000 samples are predicted per class. This result serves as our baseline, and we aim to match the same distribution of synthesized images with our proposed models.

Based on our experiment results in Table 4.1, it appears that the centralized mode of training outperforms the federate learning technique in terms of metrics like FID, IS, Precision, and Recall. Furthermore, it seems that the performance of federated learning methods greatly depends on the type of data used. ie. The performance of federated learning on non-IID is worse than that with IID data since it is influenced by the uneven distribution of data across the participating devices. This can be visualized by the distribution of generated data (10k samples) for different modes as shown in Figure B.1a

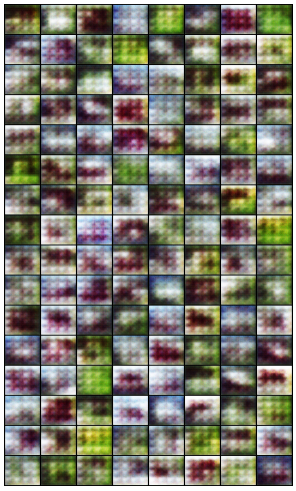
The experiments were conducted in the **NVIDIA Tesla K80 Accelerator** with a memory size of 12 GB which is optimal for DCGAN training. The overall time to train the centralized machine learning is approximately 5 hours, whereas the federated learning takes approximately 2 hours per server round of training. DCGANs contain 3.5 million parameters in the generator model and 2.7 million parameters in the discriminator model, totaling to approximately **6.2 million parameters**. This makes GAN models much faster to train and generate samples when compared to Diffusion Models. However, GAN contains a non-convex optimization problem, meaning that there are several local optima that the optimization algorithm can get stuck in. This means that the generator and discriminator models are unable to converge to a stable solution, and hence the performance can oscillate or drop off completely. The visual sample quality of a DCGAN model trained on the CIFAR-10 dataset is shown in Figure 4.1.

Table 4.1: DCGAN training on CIFAR-10 data

Type of Training	**FID**	**IS**	**Precision**	**Recall**
Centralized	0.5632	3.9822 ± 0.2800	0.9243	0.4951
Federated Learning - IID Data	1.9219	3.4588 ± 0.2313	0.8189	0.0744
Federated Learning - 2 Class Non-IID	6.9550	1.5390 ± 0.0388	0.1772	0.001

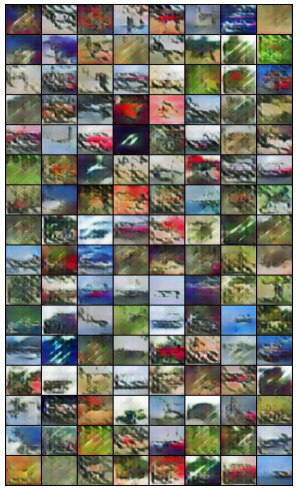


(a) Original Data from CIFAR-10

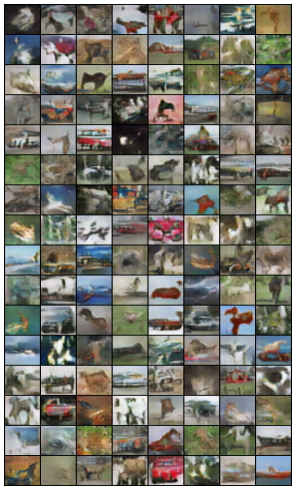


(b) Generated samples at Step 500

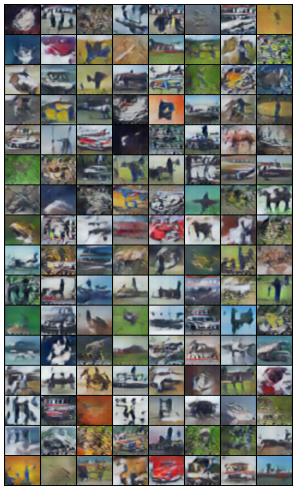
(c) Generated samples at Step 1000



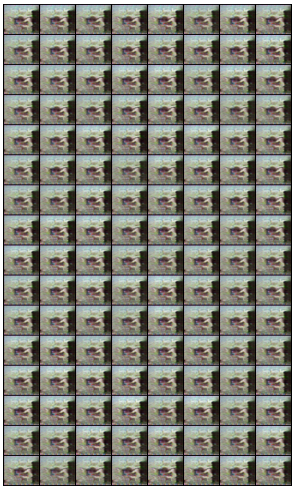
(d) Generated samples at Step 5000



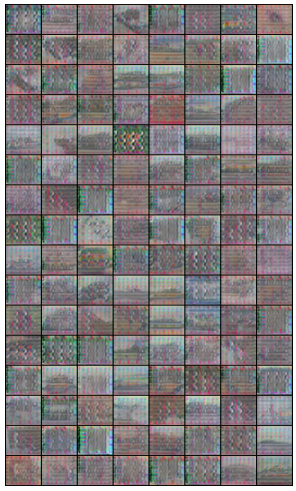
(e) Generated samples at Step 10000



(f) Generated samples at Step 15000



(g) Generated samples at Step 25000



(h) Generated samples at Step 40000

Figure 4.1: Samples when training the DCGAN from step 500 to 40000. Each image (except the first one, which is the original test data set from CIFAR-10 for comparison) corresponds to a set of samples generated at specific training steps. We observe that the generator model is gradually getting better at generating samples. However, its performance oscillates and is unable to converge.

4.2 Phoenix Algorithm

4.2.1 Default Mode

We conducted experiments by setting up the state-of-the-art Denoising Diffusion Probabilistic Models (DDPM) [26] [49] in a distributed manner. We use the hyperparameters and architecture topology as mentioned in their work,

- Four ResNet blocks are used for downsampling: three regular blocks and one with spatial self-attention. The layer widths are [C, 2C, 2C, 2C] where C is set to 128.
- Four ResNet blocks are used for upsampling: three regular blocks and one with spatial self-attention.
- Cosine scheduler is chosen as the scheduler to decay the images in the forward diffusion step
- The Adam optimizer is used for all experiments, with Mean Squared Error as the chosen loss function.
- The learning rate is set to 1×10^{-4}
- Performance metrics are computed using 10,000 samples, consistent with prior works.

The experiments were conducted in the **GeForce RTX 2080 Ti** with a memory size of 11 GB. The overall time to train the centralized machine learning is approximately 2 hours/epoch, whereas the federated learning takes approximately 11 hours per server round of training. DDPM model contains nearly **35 million parameters** which is six times the size of a DCGAN model. This makes diffusion models relatively slower to train and generate samples when compared to generative adversarial networks. The visual sample quality of a Diffusion Model trained on the CIFAR-10 dataset is shown in Figure B.2. Based on the results of the experiment in Table 4.2, it appears that the centralized mode of training outperforms the federate learning technique in terms of metrics like FID, IS, Precision, and Recall. Given

that the paper focuses on improving the sample quality of federated learning methods, we focus on those experiments over that of centralized methods. Since the performance of 2-class non-iid on federated learning is significantly poorer in terms of precision and recall, we propose two potential methods to improve the performance of the non-IID method when generating synthetic data.

Table 4.2: Diffusion Model default training on CIFAR-10 dataset

Type of Training	**FID**	**IS**	**Precision**	**Recall**
Centralized	0.7842	3.9429 ± 0.1620	0.8956	0.4717
Federated Learning - IID Data	0.1106	6.0631 ± 0.3753	0.8144	0.6628
Federated Learning - 2 Class Non-IID	0.6351	4.5087 ± 0.3275	0.3344	0.5453

4.2.2 Data Sharing Strategy

The data-sharing strategy is one of the proposed approaches to overcome the issue of statistical heterogeneity (non-iid problem) in generative models using federated learning. Here, the participating devices share a subset of their data with each other to create a more representative and diverse dataset prior to training [80]. We implemented this approach by having four and ten participating clients, each of which are experimented in detail below.

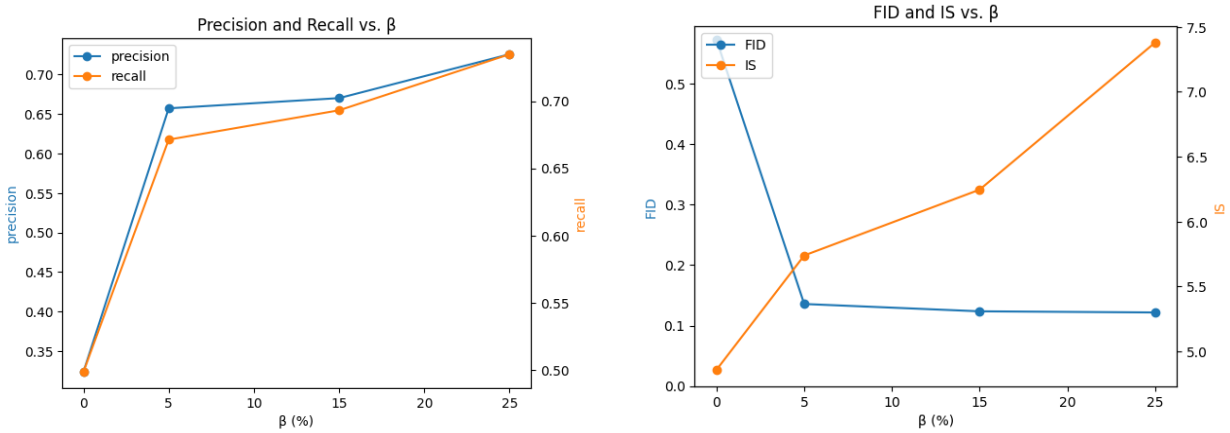
We divided the CIFAR-10 training dataset (50k samples) into a client (C) and server (S) part, each consisting of 40,000 and 10,000 samples, respectively. Next, we split the client part into non-overlapping partitions and randomly assigned these partitions to each participating client containing two classes of images per client. The server part is split into partitions (globally shared data G), and only a portion of the partition is used for global training which is quantified by β and can range from 2.5% to 25%. Once the global warmup model is trained, a fraction of G (quantified by α) is merged with the client part (C) for federated learning training.

Four Clients:

The experimental results presented in Table 4.3 and Figure 4.2 indicate that the performance of the model improves as the number of partitions in the server part (S) increases. One possible explanation is that when all the data from the server part (10k samples) are used for global warmup training, a higher number of samples per class are encountered during the training process. This increased diversity and representativeness of the dataset may lead to a better-performing global model. Additionally, the distribution of generated samples from the image classification 4.3 LaNet model suggests that as β increases, the slope of the curve gradually reduces. This is indicative that the Data Sharing Strategy with a high β value can generate samples with higher diversity than the default training mode. The visual sample quality of a Diffusion Model trained on the CIFAR-10 dataset with Data Sharing Strategy is shown in Figure B.3.

Table 4.3: Data sharing strategy experiment results with four clients. Number of decay steps T:1000, Server Aggregation Strategy: FedAvg, Scheduler: Cosine, Server Rounds: 10, 2 class Non-IID

$\beta(\%)$	$\alpha(\%)$	**Server**	**Client**	**Test Set**	**FID**	**IS**	**Precision**	**Recall**
5	100	2k	12k	10k	0.1358	5.7394 ± 0.1218	0.6572	0.6714
15	100	6k	16k	10k	0.1237	6.2467 ± 1782	0.6701	0.6932
25	100	10k	20k	10k	0.1218	7.3794 ±0.1368	0.7256	0.7349



(a) Precision and Recall vs. β

(b) Frechet Inception Distance score and Inception Score vs. β

Figure 4.2: Comparison of model performance in data sharing strategy with four participating clients with different β values

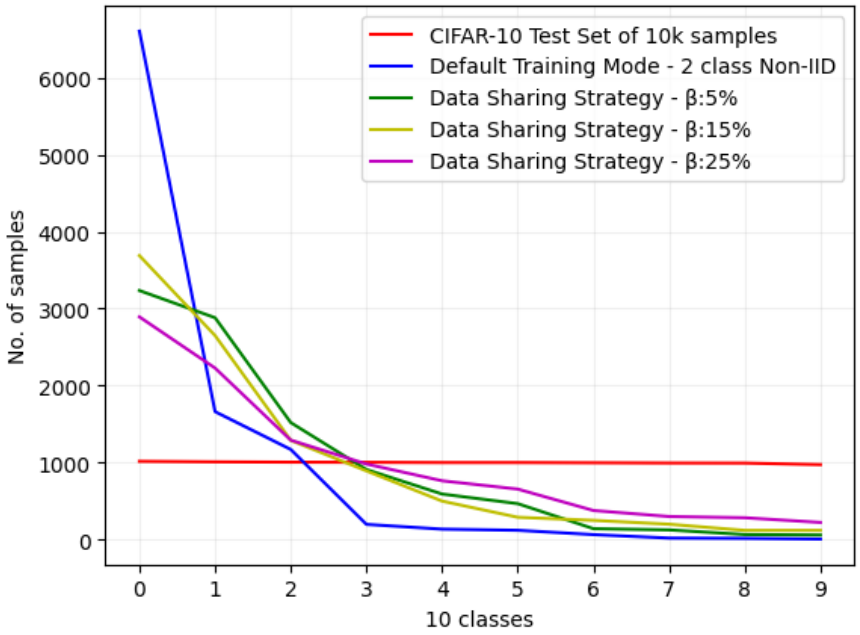


Figure 4.3: Distribution of generated sample classes from data sharing strategy with four clients for different values of β - Based on classification results from LaNet Model [73]

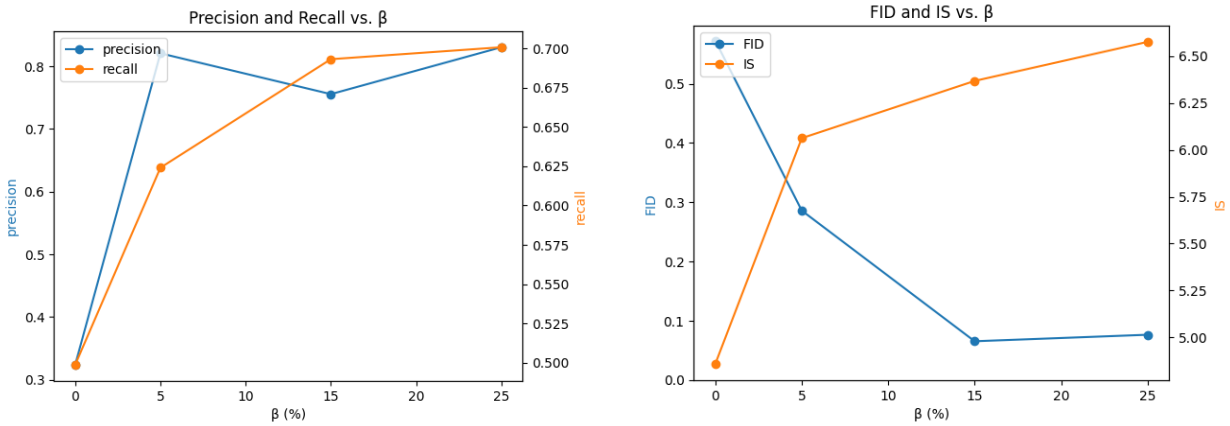
Ten Clients:

The data-sharing strategy experiment was further conducted on ten clients using the federated aggregation (FedAvg) strategy with a Cosine scheduler and 1000 decay steps. The experiment was conducted for ten server rounds and Table 4.4 shows the results of each round for different values of α and β. Here β represents the percentage of data from the server part (S) that was used for training at each client, and α represents the percentage of data from β used for training.

The FID (Fréchet Inception Distance) and IS (Inception Score), Precision, and Recall values were used to evaluate the quality and distribution of the generated images, with lower FID values indicating better image quality. Observing the results from Figure 4.4, it can be inferred that increasing the percentage of data shared among clients (i.e., increasing β) generally leads to better image quality, as indicated by the lower FID, and higher IS, Precision and Recall values. However, with the extreme 2-class non-IID data and a lower β of 5%, we can still achieve high precision and recall of 0.82 and 0.62, respectively, that outperforms the default training model with precision and recall of 0.33 and 0.54 respectively. This indicates that not much of the globally shared data is needed to improve the performance of synthesized images.

Moreover, it was observed that the entire server part (C) was not necessary to distribute to all clients to achieve improved performance. Instead, a random portion quantified by α as shown in Figure 4.5 could achieve similar performance. Since $\beta = 25\%$ yields the best performant model, we evaluated its combinations by modifying the value of α ranging from 25% to 100%. By distributing only 25% (i.e. additional 2.5k samples per client) of the server part (S) data, we could achieve precision and recall of 0.82 and 0.67, respectively. Additionally, by increasing the value of α to 100% (additional 10k samples per client), we achieved the best model through our data-sharing strategy with a precision and recall of 0.83 and 0.70. These results reveal that data distribution of class with a $\beta = 25$ and $\alpha = 100$ improves by over 30%, as illustrated in Figure 4.5

In conclusion, the results suggest that sharing a higher percentage of data among clients can improve image quality, but increasing the percentage beyond a certain threshold does not yield significant improvements. However, the optimal values of α and β may vary depending on the specific use case model and dataset.



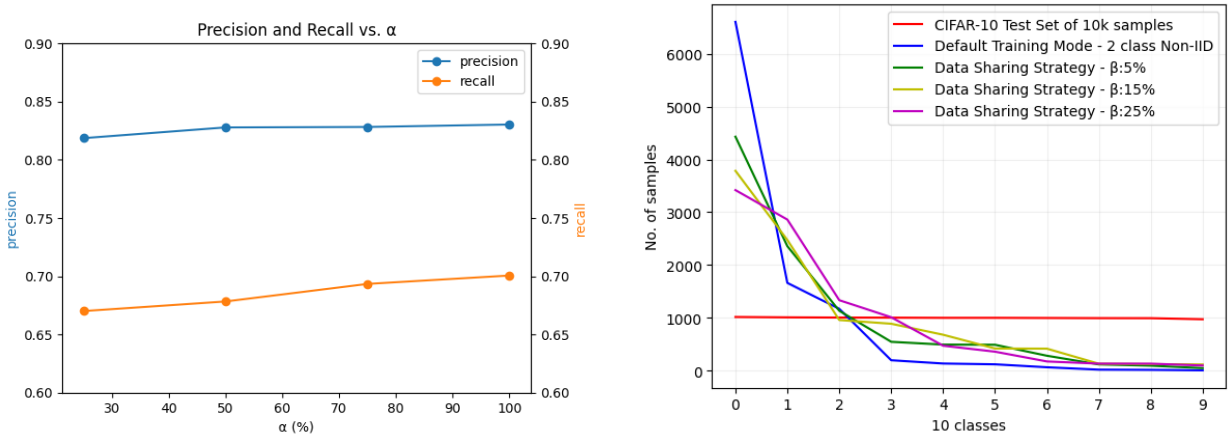
(a) Precision and Recall vs. β

(b) Frechet Inception Distance score and Inception Score vs. β

Figure 4.4: Comparison of model performance from data sharing strategy with ten clients with different β values

Table 4.4: Data sharing strategy experiment results with Ten clients

$\beta(\%)$	$\alpha(\%)$	**Server**	**Client**	**Test Set**	**FID**	**IS**	**Precision**	**Recall**
5	100	2k	6k	10k	0.2858	6.0624 ± 0.1583	0.8206	0.6240
15	100	6k	10k	10k	0.0653	6.3676 ± 0.1820	0.7555	0.6930
25	100	10k	14k	10k	0.0763	6.5758 ± 0.3126	0.8303	0.7006
25	25	10k	6.5k	10k	0.2920	5.5283 ± 0.6442	0.8186	0.6701
25	50	10k	9k	10k	0.2047	5.9563 ± 0.2638	0.8278	0.6783
25	75	10k	11.5k	10k	0.1435	6.0122 ± 0.4829	0.8282	0.6934



(a) Precision and Recall vs. α at β=25 (b) Distribution of generated sample classes

Figure 4.5: Performance graphs of Data sharing strategy experiment results with Ten clients

4.2.3 Personalization layers & Threshold Filtering

The *Phoenix* algorithm also leverages personalization and threshold filtering techniques to tackle statistical heterogeneity. By incorporating personalization layers during training, the federated learning model can better adapt to the participating devices, thereby providing personalized generated content. We divide the diffusion model into personalization layers and base layers. Since U-Net is the chosen model architecture for the reverse diffusion process, we choose the personalization layers from this model. The U-Net contains eight basic blocks with convolution layers, a max-pooling layer, and an activation function. In our experiments, we designate the last basic block with convolutional layers as the personalization layers [47].

Furthermore, in a scenario where the data is non-iid, it is possible that some samples provided by participating clients do not contribute value to the global server model. In the context of federated learning, threshold filtering is a technique to disconnect clients that exhibit subpar precision and recall during training. Since our experiments are simulation-based, new samples from clients are unlikely to be provided during training. Therefore, we permanently disconnect clients from participation in future rounds rather than temporarily disconnecting them. The details of our experiments are explained as follows,

Table 4.5: Diffusion model training with Personalization layers & Threshold Filtering on CIFAR-10 dataset with 2 class Non-IID

Client Drop approach	**FID**	**IS**	**Precision**	**Recall**
Lowest Precision	0.7607	5.3639 ± 0.3971	0.5920	0.7623
Threshold Precision: 0.7	1.2242	5.4681 ± 0.5142	0.5599	0.7614
Threshold Precision: 0.6	4.2548	2.6210 ± 0.1867	0.5337	0.259

- In Server Round 1, all the participating clients train like the default training mode. At the end of the training, the last basic block is stored locally as the personalization layer
- Starting from Server Round 2, the personalization layers are retrieved by each client before local training begins. Similar to Server Round 1, at the end of the training, the last basic block is stored locally as the personalization layer.
- From Server Round 5 onwards, after completing local training, each client generates 1000 samples and computes performance metrics such as precision and recall. These metrics are continuously logged to enable the server to make decisions based on the logged data.
- If a client's precision is lower than that of other participating clients in the same server round or falls below a predefined threshold value, the client is marked separately.
- These marked clients are closely monitored in subsequent rounds, and if their performance remains poor for at least two server rounds, they are disconnected.

The experiment was conducted with ten participating clients using the federated aggregation (FedAvg) strategy with a Cosine scheduler and 1000 decay steps. The experiment was conducted for ten server rounds, and Table 4.5 and Figure 4.6 shows the results of Diffusion model training with Personalization layers & Threshold Filtering on the CIFAR-10 dataset.

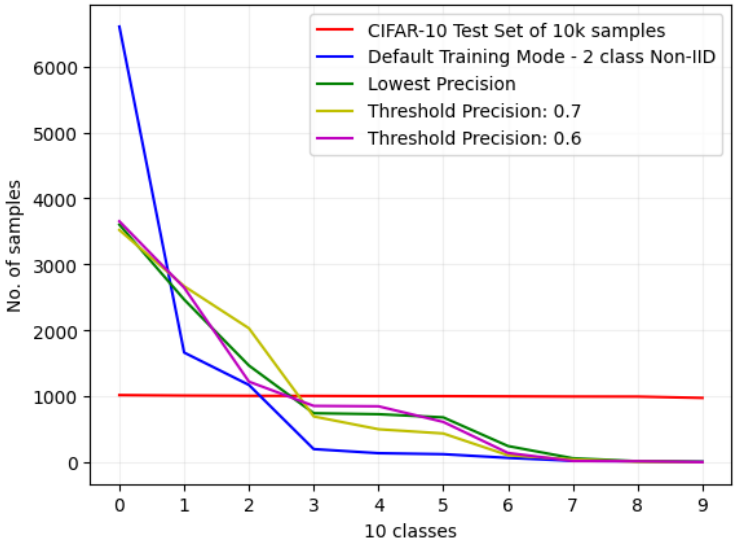


Figure 4.6: Distribution of generated sample classes using Personalization layers & Threshold Filtering technique - Based on classification results from LaNet Model [73]

The experiment results show that the client drop approach using the lowest precision yielded the best performance among the three approaches. This is primarily because the lowest precision approach provides all clients with equal opportunities for local training and only disconnects clients with the lowest precision in each server round. On the other hand, the other two methods employ pre-defined values, which may not be practical during training since precision and recall tend to fluctuate between server rounds. Hence, pre-defined value methods are not effective strategies for improving performance. Furthermore, the addition of personalization layers during training enhances the model's ability to adapt to the unique characteristics of each participating device. This is evident in its overall performance compared to the default training of diffusion models. The visual sample quality of a Diffusion Model trained on the CIFAR-10 dataset with Personalization layers & Threshold Filtering Strategy is shown in Figure B.4.

Chapter 5

DISCUSSION

5.1 Implications

This paper introduces a novel technique called the Phoenix for training unconditional diffusion models in a horizontal federated learning setting, which, to our knowledge, has not been previously explored. Its primary objective is to address the issue of mode coverage commonly encountered with non-IID datasets. Despite not being a sensitive dataset, we chose CIFAR-10 for our experiments since it serves as a robust baseline for visually understanding the behavior of different deep-learning models. To begin, we implemented DCGAN and DDPM, existing methods, within a federated learning framework using the Flower framework [7], along with PyTorch Library [51], DCGAN Library [1], Diffusers Library [72] and Weights & Biases Library [8] for experiment tracking. These implementations served as our benchmark for comparison. Next, we conducted comprehensive experiments and analysis on the two proposed strategies of Phoenix: Data Sharing Strategy and Personalization & Threshold Filtering to improve the diversity of synthesized images.

Our methods demonstrated superior model performance compared to existing techniques, such as Deep Convolutional GANs in Federated Learning and Diffusion Models in Centralized Mode. This performance improvement was particularly noticeable in non-IID data settings, as illustrated in Table 5.1 and Figure 5.1. An important point to highlight here is that the data-sharing approach achieved a significant performance boost by sharing a mere 4-5% of the overall data among clients. This minimal initial data sharing not only led to a significant boost in performance but also kept the communication overhead at a minimum. The personalization & threshold filtering techniques outperformed the comparison methods in terms of precision and recall. However, they did not excel in image quality compared to

the other proposed technique. Further exploration in this area as future work could unveil specific areas for improvement.

An additional observation that can be made is regarding the distribution of generated samples for the different methods, as depicted in Figures B.1a, B.1b, 4.3, 4.5, and 4.6. The ideal scenario for generated samples will be if they are closely aligned with the red line, which represents the CIFAR-10 Test set consisting of 10k samples equally distributed among the 10 classes. However, the unconditional diffusion model fails to reproduce this ideal scenario. Due to its random seed initialization for generating images, it is highly improbable for the model to effectively cover all the classes observed during training. However, we show how our proposed methods try to minimize the gap between a biased mode coverage and the ideal flat-line scenario.

Table 5.1: Performance comparison of generative model training methods using federated learning for non-IID data

Model/Technique	**FID**	**IS**	**Precision**	**Recall**
Deep Convolutional GAN (DCGAN)	6.9550	1.5390	0.1772	0.001
Denoising Diffusion Probabilistic Models (DDPM)	0.6351	4.5087	0.5453	0.3344
Data Sharing Strategy DDPM - **Ours**	0.0763	6.5758	0.8303	0.7006
Personalization & Threshold Filtering DDPM - **Ours**	0.7607	5.3639	0.592	0.7623

5.2 Limitation

5.2.1 Sampling Time

Diffusion models have a major drawback in terms of sampling steps and the long wall-clock time to sample synthetic data [11]. Since diffusion models are based on a Markov chain of processes that learns to noise and denoise an image through a fixed number of decay steps T (1000 or 4000), the same number of steps are required during the inference phase. This

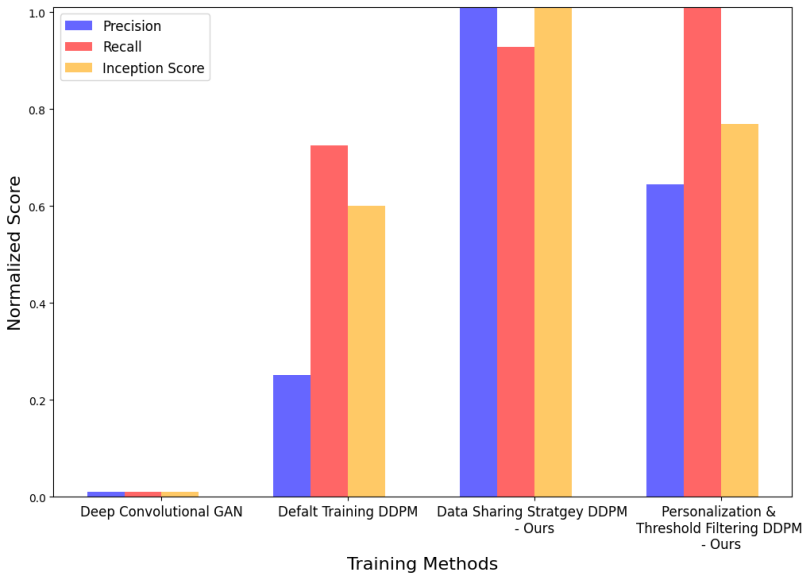


Figure 5.1: Performance Comparison of Different Training Methods

means that nearly 1000 steps are required to generate one image, which is substantially slower than GANs. Although a large number of steps are used to generate high-quality data in a controlled manner, it also comes with inherent disadvantages like increased computation complexity and longer processing times. However, several techniques are used to speed up the notoriously slow diffusion models. Some of the recent methods include

- Exponential Integrator Sampler (DEIS) [79] - Uses an efficient sampler that can be applied to any DM to achieve faster sampling with a limited number of network function evaluations.

- Latent space diffusion models [70] - Train diffusion models in the latent space rather than the data space.

- Denoising Diffusion GANs [75] - A diffusion model where the reverse denoising process is parametrized by conditional GANs to achieve significant speed-up.

5.2.2 Hardware Limitation

On-device training due to federated learning of diffusion models provides better personalization of models and improved user privacy and security. However, large diffusion models contain over a billion parameters [13], making it difficult to deploy in resource-constrained devices (edge devices) in federated learning settings. Our work employs diffusion models containing over **35 million parameters**, which is six times larger than the standard deep convolutional generative adversarial networks (DCGANs). Some of the ways to overcome the increased model size without affecting performance include

- Model Optimization - Using specialized kernels to enable faster execution, e.g., Group Norm, Gaussian Error Linear Unit (GELU), Partially fused softmax, Winograd convolution [13] etc.
- Model Compression [52] - Qualcomm AI Research performed full-stack AI optimizations, including post-training quantization to shrink the neural network model from floating point FP32 to integer INT8. This enhancement dramatically sped up the inference time to as less as 15 seconds to generate a 512*512 pixel image on a smartphone.

5.2.3 Standardization

Comparing deep learning models across different sources or applications can be challenging due to the lack of standardization. This issue becomes particularly relevant in the case of diffusion models, which are highly complex and specialized for different applications. Hence, they have variations in architectures, data, pre-processing steps, and training procedures. As a result, it becomes challenging to compare models or reproduce results reported by different research groups.

Our version of the neural network architecture of diffusion models is based on the state-of-the-art [26]. However, the choice of hyperparameters, pre-processing, and augmentation techniques of CIFAR-10 are quite different, making it difficult to establish a direct benchmark for comparison.

Better standardization practices of deep learning models, particularly in the context of federated learning, would greatly benefit the research community in terms of the reproducibility of results. This entails establishing unified approaches for evaluating deep learning models and datasets. Existing benchmarks for federated learning frameworks like UniFed [38] show the potential for extending similar benchmarks to generative modeling trained using federated learning techniques.

5.3 Future Work

While this paper has been a start to utilizing unconditional diffusion models in a federated learning setting for data privacy, security, and personalization, there are several avenues for future research and exploration. One direction involves extending to real-world applications in domains like finance and healthcare, where data is dispersed across multiple institutions and hence cannot be stored in a central server. Additionally, the development of robust evaluation metrics that go beyond just fidelity and diversity to better capture the novelty and fairness of generated samples is crucial for a comprehensive assessment of generative models. From an algorithmic standpoint, there is a need to delve deeper into privacy-preserving techniques such as differential privacy and encrypted data FL which are exciting areas of research. Additionally, diffusion models are computationally expensive, making them difficult to deploy in edge devices as part of federated learning. Hence, the development of smaller models or faster inference techniques is one possible example of future work.

Furthermore, ethical considerations can also be addressed through watermarking techniques for traceability and accountability and by exploring fairness, bias, transparency, and interpretability aspects of unconditional diffusion models in the context of federated learning. These investigations will help shape ethical guidelines and best practices for the responsible deployment of generative models in federated learning settings.

BIBLIOGRAPHY

[1] Dcgan-cifar10-pytorch. `https://github.com/Ksuryateja/DCGAN-CIFAR10-pytorch`, 2019. Accessed: Jan 01, 2023.

[2] Common problems in gans. `https://developers.google.com/machine-learning/gan/problems`, 2022. Accessed: Jan 01, 2023.

[3] Manoj Ghuhan Arivazhagan, Vinay Aggarwal, Aaditya Kumar Singh, and Sunav Choudhary. Federated learning with personalization layers. *arXiv preprint arXiv:1912.00818*, 2019.

[4] Shane Barratt and Rishi Sharma. A note on the inception score. *arXiv preprint arXiv:1801.01973*, 2018.

[5] Yaniv Benny, Tomer Galanti, Sagie Benaim, and Lior Wolf. Evaluation metrics for conditional image generation. *International Journal of Computer Vision*, 129:1712–1731, 2021.

[6] Eyal Betzalel, Coby Penso, Aviv Navon, and Ethan Fetaya. A study on the evaluation of generative models. *arXiv preprint arXiv:2206.10935*, 2022.

[7] Daniel J Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Javier Fernandez-Marques, Yan Gao, Lorenzo Sani, Kwing Hei Li, Titouan Parcollet, Pedro Porto Buarque de Gusmão, et al. Flower: A friendly federated learning research framework. *arXiv preprint arXiv:2007.14390*, 2020.

[8] Lukas Biewald. Experiment tracking with weights and biases. `https://www.wandb.com/`, 2020. Software available from wandb.com.

[9] Sam Bond-Taylor, Adam Leach, Yang Long, and Chris G. Willcocks. Deep generative modelling: A comparative review of VAEs, GANs, normalizing flows, energy-based and autoregressive models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11):7327–7347, nov 2022.

[10] Ali Borji. Pros and cons of gan evaluation measures: New developments. *Computer Vision and Image Understanding*, 215:103329, 2022.

[11] Hanqun Cao, Cheng Tan, Zhangyang Gao, Guangyong Chen, Pheng-Ann Heng, and Stan Z Li. A survey on generative diffusion model. *arXiv preprint arXiv:2209.02646*, 2022.

[12] Xingjian Cao, Gang Sun, Hongfang Yu, and Mohsen Guizani. Perfed-gan: Personalized federated learning via generative adversarial networks. *IEEE Internet of Things Journal*, 2022.

[13] Yu-Hui Chen, Raman Sarokin, Juhyun Lee, Jiuqiang Tang, Chuo-Ling Chang, Andrei Kulik, and Matthias Grundmann. Speed is all you need: On-device acceleration of large diffusion models via gpu-aware optimizations. *arXiv preprint arXiv:2304.11267*, 2023.

[14] Yujing Chen, Yue Ning, Martin Slawski, and Huzefa Rangwala. Asynchronous online federated learning for edge devices with non-iid data. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 15–24. IEEE, 2020.

[15] Florinel-Alin Croitoru, Vlad Hondru, Radu Tudor Ionescu, and Mubarak Shah. Diffusion models in vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[16] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[17] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021.

[18] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. *Advances in Neural Information Processing Systems*, 33:3557–3568, 2020.

[19] Chenyou Fan and Ping Liu. Federated generative adversarial learning. In *Pattern Recognition and Computer Vision: Third Chinese Conference, PRCV 2020, Nanjing, China, October 16–18, 2020, Proceedings, Part III 3*, pages 3–15. Springer, 2020.

[20] Stanislav Frolov, Tobias Hinz, Federico Raue, Jörn Hees, and Andreas Dengel. Adversarial text-to-image synthesis: A review. *Neural Networks*, 144:187–209, 2021.

[21] Liang Gao, Huazhu Fu, Li Li, Yingwen Chen, Ming Xu, and Cheng-Zhong Xu. Feddc: Federated learning with non-iid data via local drift decoupling and correction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10112–10121, 2022.

[22] Raghda Ghonima. Implementation of gans using federated learning. In *2021 Tenth International Conference on Intelligent Computing and Information Systems (ICICIS)*, pages 142–148. IEEE, 2021.

[23] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[24] Henrik Hellström, José Mairton B da Silva Jr, Mohammad Mohammadi Amiri, Mingzhe Chen, Viktoria Fodor, H Vincent Poor, and Carlo Fischione. Wireless for machine learning. *arXiv preprint arXiv:2008.13492*, 2020.

[25] 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. *Advances in neural information processing systems*, 30, 2017.

[26] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[27] Jeremy. Non-iid-dataset-for-personalized-federated-learning. `https://github.com/jeremy313/non-iid-dataset-for-personalized-federated-learning`, 2020. Accessed: March 01, 2023.

[28] James M. Joyce. *Kullback-Leibler Divergence*, pages 720–722. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[29] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

[30] Amirhossein Kazerouni, Ehsan Khodapanah Aghdam, Moein Heidari, Reza Azad, Mohsen Fayyaz, Ilker Hacihaliloglu, and Dorit Merhof. Diffusion models for medical image analysis: A comprehensive survey. *arXiv preprint arXiv:2211.07804*, 2022.

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

[32] Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. *Advances in Neural Information Processing Systems*, 32, 2019.

[33] Ang Li, Jingwei Sun, Binghui Wang, Lin Duan, Sicheng Li, Yiran Chen, and Hai Li. Lotteryfl: Personalized and communication-efficient federated learning with lottery ticket hypothesis on non-iid datasets. *arXiv preprint arXiv:2008.03371*, 2020.

[34] Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. Federated learning on non-iid data silos: An experimental study. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 965–978. IEEE, 2022.

[35] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450, 2020.

[36] Tian Li, Maziar Sanjabi, Ahmad Beirami, and Virginia Smith. Fair resource allocation in federated learning. *arXiv preprint arXiv:1905.10497*, 2019.

[37] Zijian Li, Jiawei Shao, Yuyi Mao, Jessie Hui Wang, and Jun Zhang. Federated learning with gan-based data synthesis for non-iid clients. In *Trustworthy Federated Learning: First International Workshop, FL 2022, Held in Conjunction with IJCAI 2022, Vienna, Austria, July 23, 2022, Revised Selected Papers*, pages 17–32. Springer, 2023.

[38] Xiaoyuan Liu, Tianneng Shi, Chulin Xie, Qinbin Li, Kangping Hu, Haoyu Kim, Xiaojun Xu, Bo Li, and Dawn Song. Unifed: A benchmark for federated learning frameworks. *arXiv preprint arXiv:2207.10308*, 2022.

[39] Yang Liu, Yan Kang, Tianyuan Zou, Yanhong Pu, Yuanqin He, Xiaozhou Ye, Ye Ouyang, Ya-Qin Zhang, and Qiang Yang. Vertical federated learning. *arXiv preprint arXiv:2211.12814*, 2022.

[40] 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.

[41] Sin Kit Lo, Qinghua Lu, Chen Wang, Hye-Young Paik, and Liming Zhu. A systematic literature review on federated machine learning: From a software engineering perspective. *ACM Computing Surveys (CSUR)*, 54(5):1–39, 2021.

[42] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[43] Brendan McMahan and Daniel Ramage. Federated learning: Collaborative machine learning without centralized training data. `https://ai.googleblog.com/2017/04/federated-learning-collaborative.html`, 2017. Accessed: August 01, 2022.

[44] Vaikkunth Mugunthan, Vignesh Gokul, Lalana Kagal, and Shlomo Dubnov. Bias-free fedgan: A federated approach to generate bias-free datasets. *arXiv preprint arXiv:2103.09876*, 2021.

[45] Ajinkya Mulay, Baye Gaspard, Rakshit Naidu, Santiago Gonzalez-Toral, S Vineeth, Tushar Semwal, and Ayush Manish Agrawal. Fedperf: A practitioners' guide to performance of federated learning algorithms. In *NeurIPS 2020 Workshop on Pre-registration in Machine Learning*, pages 302–324. PMLR, 2021.

[46] Bao-Long Nguyen, Tat Cuong Cao, and Bac Le. Meta-learning and personalization layer in federated learning. In Ngoc Thanh Nguyen, Tien Khoa Tran, Ualsher Tukayev, Tzung-Pei Hong, Bogdan Trawiński, and Edward Szczerbicki, editors, *Intelligent Information and Database Systems*, pages 209–221, Cham, 2022. Springer International Publishing.

[47] Cao Tt Cng Nguyn Bo Long. Meta-learning and personalization layer in federated learning. `https://github.com/baolongnguyenmac/fl-ml/tree/main`, 2022. Accessed: Jan 01, 2023.

[48] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*, 2021.

[49] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR, 2021.

[50] OpenAI. Gpt-4 technical report. `https://arxiv.org/pdf/2303.08774.pdf`, 2023.

[51] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[52] Qualcomm. World's first on-device demonstration of stable diffusion on android. `https://www.qualcomm.com/news/onq/2023/02/worlds-first-on-device-demonstration-of-stable-diffusion-on-android"`, 2023.

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

[54] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

[55] Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. *Advances in neural information processing systems*, 32, 2019.

[56] Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečnỳ, Sanjiv Kumar, and H Brendan McMahan. Adaptive federated optimization. *arXiv preprint arXiv:2003.00295*, 2020.

[57] White House Report. Consumer data privacy in a networked world: A framework for protecting privacy and promoting innovation in the global digital economy. *Journal of Privacy and Confidentiality*, 4(2), Mar. 2013.

[58] Niels Rogge and Kashif Rasul. The annotated diffusion model. `https://huggingface.co/blog/annotated-diffusion`, 2022. Accessed: August 01, 2022.

[59] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022.

[60] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

[61] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 35:36479–36494, 2022.

[62] Mehdi SM Sajjadi, Olivier Bachem, Mario Lucic, Olivier Bousquet, and Sylvain Gelly. Assessing generative models via precision and recall. *Advances in neural information processing systems*, 31, 2018.

[63] 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, 2016.

[64] Divya Saxena and Jiannong Cao. Generative adversarial networks (gans): Challenges, solutions, and future directions. *ACM Comput. Surv.*, 54(3), may 2021.

[65] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. Federated multi-task learning. *Advances in neural information processing systems*, 30, 2017.

[66] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2256–2265. PMLR, 07–09 Jul 2015.

[67] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[68] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[69] Youngjung Uh. Improved-precision-and-recall-metric-pytorch. `https://github.com/youngjung/improved-precision-and-recall-metric-pytorch`, 2019. Accessed: Jan 01, 2023.

[70] Arash Vahdat, Karsten Kreis, and Jan Kautz. Score-based generative modeling in latent space. *Advances in Neural Information Processing Systems*, 34:11287–11302, 2021.

[71] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[72] Patrick von Platen, Suraj Patil, Anton Lozhkov, Pedro Cuenca, Nathan Lambert, Kashif Rasul, Mishig Davaadorj, and Thomas Wolf. Diffusers: State-of-the-art diffusion models. `https://github.com/huggingface/diffusers`, 2022. Accessed: January 01, 2023.

[73] Linnan Wang, Saining Xie, Teng Li, Rodrigo Fonseca, and Yuandong Tian. Sample-efficient neural architecture search by learning actions for monte carlo tree search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):5503–5515, 2022.

[74] Jie Wen, Zhixia Zhang, Yang Lan, Zhihua Cui, Jianghui Cai, and Wensheng Zhang. A survey on federated learning: challenges and applications. *International Journal of Machine Learning and Cybernetics*, pages 1–23, 2022.

[75] Zhisheng Xiao, Karsten Kreis, and Arash Vahdat. Tackling the generative learning trilemma with denoising diffusion gans. *arXiv preprint arXiv:2112.07804*, 2021.

[76] Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Yingxia Shao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. Diffusion models: A comprehensive survey of methods and applications. *arXiv preprint arXiv:2209.00796*, 2022.

[77] Chaoning Zhang, Chenshuang Zhang, Sheng Zheng, Yu Qiao, Chenghao Li, Mengchun Zhang, Sumit Kumar Dam, Chu Myaet Thwal, Ye Lin Tun, Le Luang Huy, et al. A complete survey on generative ai (aigc): Is chatgpt from gpt-4 to gpt-5 all you need? *arXiv preprint arXiv:2303.11717*, 2023.

[78] Longling Zhang, Bochen Shen, Ahmed Barnawi, Shan Xi, Neeraj Kumar, and Yi Wu. Feddpgan: federated differentially private generative adversarial networks framework for the detection of covid-19 pneumonia. *Information Systems Frontiers*, 23(6):1403–1415, 2021.

[79] Qinsheng Zhang and Yongxin Chen. Fast sampling of diffusion models with exponential integrator. *arXiv preprint arXiv:2204.13902*, 2022.

[80] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

[81] Hangyu Zhu, Jinjin Xu, Shiqing Liu, and Yaochu Jin. Federated learning on non-iid data: A survey. *Neurocomputing*, 465:371–390, 2021.

Appendix A

NEURAL NETWORK ARCHITECTURE

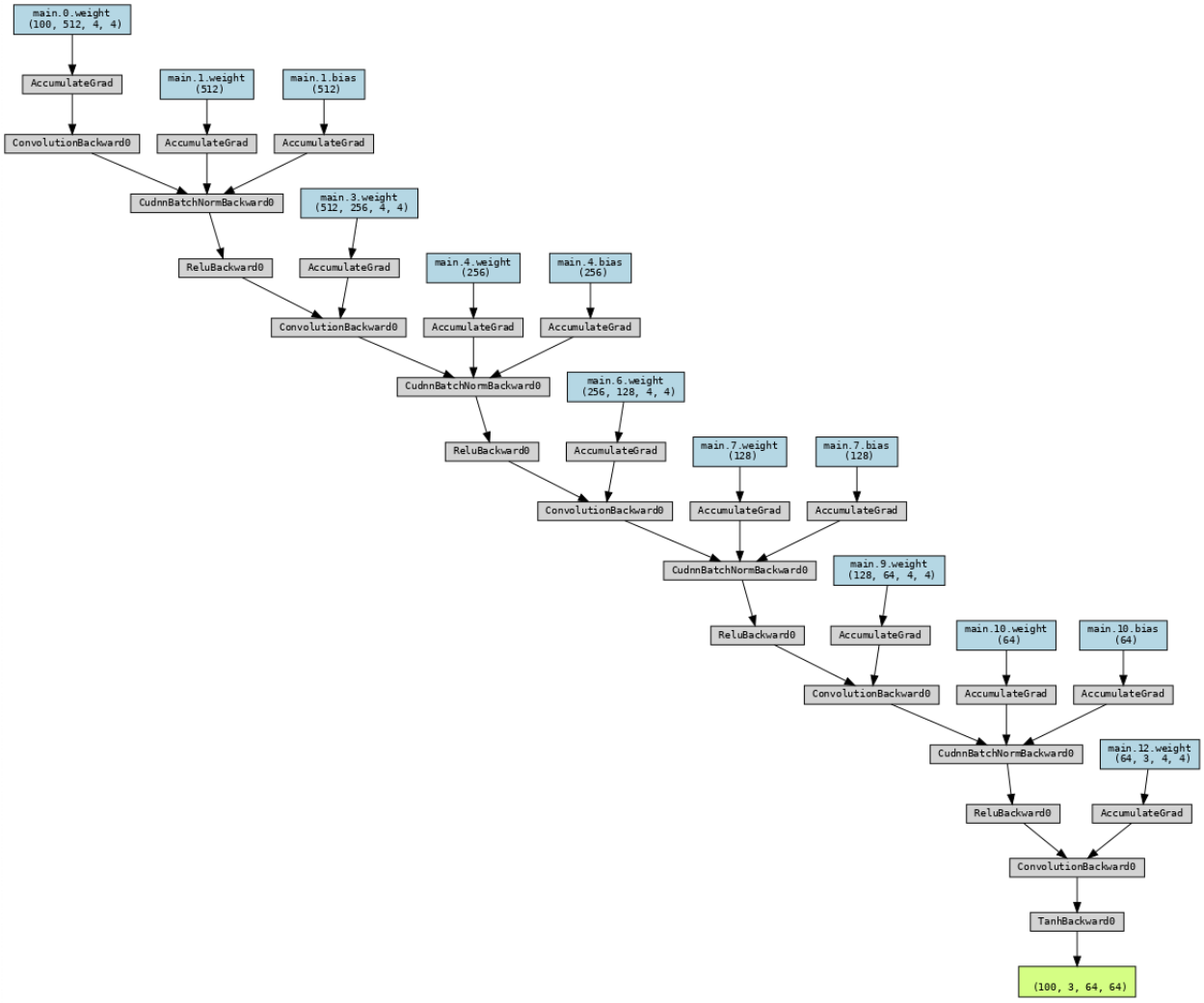


Figure A.1: DCGAN generator used for generating images from CIFAR-10 dataset

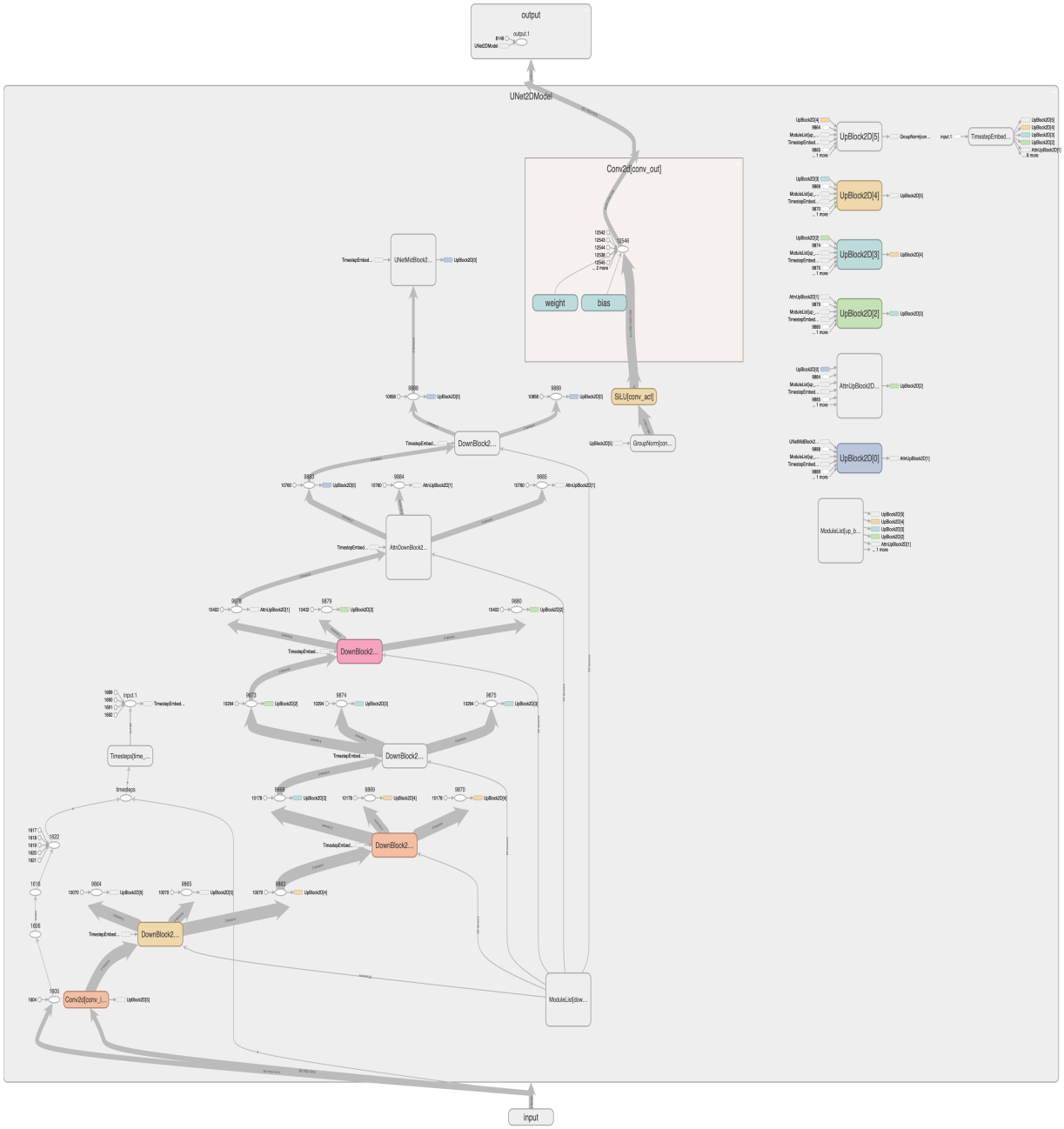
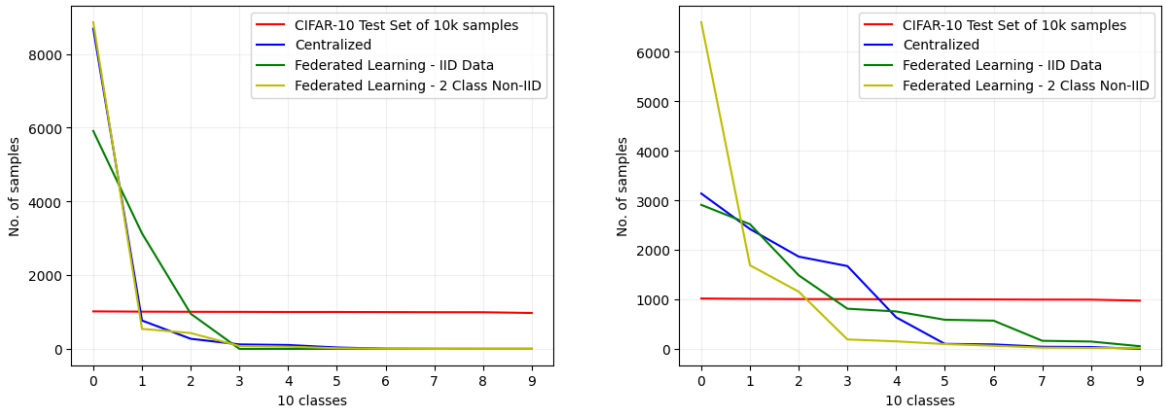


Figure A.2: Diffusion model used for generating images from CIFAR-10 dataset

Appendix B

EXPERIMENT RESULTS



(a) Distribution of generated sample classes from DCGAN

(b) Distribution of generated sample classes from Diffusion Model

Figure B.1: Comparison of generated sample class distributions from DCGAN and Diffusion Model for different training modes - Based on classification results from LaNet Model [73]

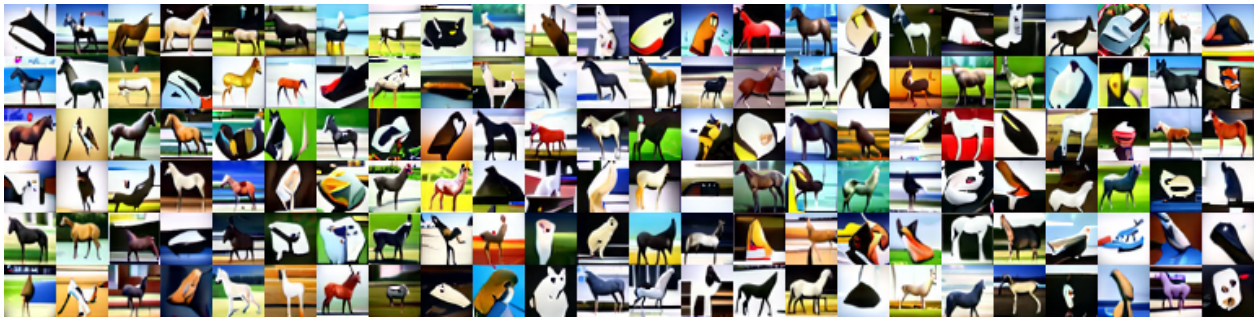


Figure B.2: Generated Samples from Default Diffusion Model trained using Federated Learning

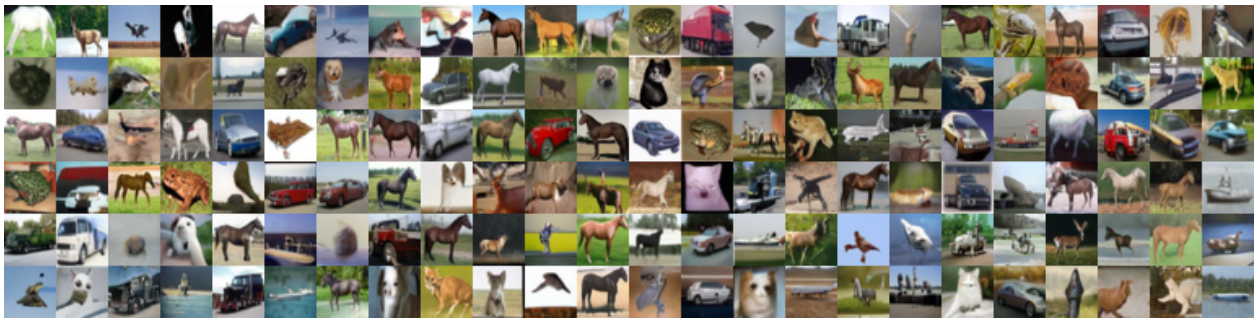


Figure B.3: Generated Samples from Phoenix Algorithm - Data Sharing Strategy

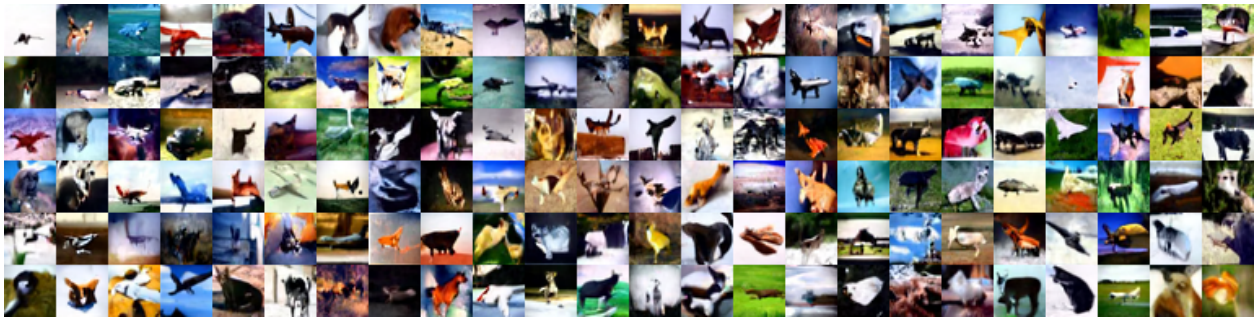


Figure B.4: Generated Samples from Phoenix Algorithm - Personalization and & Threshold Filtering