,

PrivacyVerse: Exploring Differential Privacy in Diverse ML Models

Project Report submitted by

Badavath Gayathri 420110

Maradana Surya Sai Indra 420205

Baliyarusimhula Pravallika 420112

Under the supervision of

Mrs. Sri Satya Monica Bandaru

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY, ANDHRA PRADESH

MAY-2024

PrivacyVerse: Exploring Differential Privacy in Diverse ML Models

Submitted in the partial fulfillment of the requirements the degree of

Bachelor of Technology

by

Badavath Gayathri	420110
Maradana Surya Sai Indra	420205
Baliyarusimhula Pravallika	420112

Supervisor:

Mrs. Sri Satya Monica Bandaru

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY, ANDHRA PRADESH

MAY-2024

PROJECT WORK APPROVAL

This project work entitled "**PrivacyVerse: Exploring Differential Privacy in Diverse ML Models**" was worked out by Badavath Gayathri (420110), Maradana Surya Sai Indra (420205), and Baliyarusimhula Pravallika (420112) is approved for the degree of Bachelor of Technology in Computer Science and Engineering

Examiners

Supervisor (s)

Chairman

Date:

Place: Tadepalligudem

DECLARATION

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Badavath Gayathri	Maradana Surya Sai Indra	Baliyarusimhula Pravallika
420110	420205	420112
Date:	Date:	Date:

Department of Computer Science and Engineering

NATIONAL INSTITUTE OF TECHNOLOGY, ANDHRA PRADESH

Certificate

This is to certify the thesis entitled ”PrivacyVerse: Exploring Differential Privacy in Diverse ML Models” submitted by Badavath Gayathri (420110), Maradana Surya Sai Indra (420205), Baliyarusimhula Pravallika (420112), to National Institute of Technology, Tadepalligudem in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering is a record of bonafide research work carried out by him/her under my supervision and guidance. This work has not been submitted elsewhere for the award of any degree

Mrs. Sri Satya Monica Bandaru
Computer Science and Engineering
N.I.T Andhra Pradesh
May 2024

Place: Tadepalligudem
Date:

Acknowledgement

We would like to express our sincere gratitude to the National Institute of Technology Andhra Pradesh for providing us with Mrs. Sri Satya Monica Bandaru, as our supervisor. Her extensive knowledge and guidance were invaluable, and we express our deepest appreciation for her continuous support and guidance throughout this project. Without her encouragement and feedback, this project would not have been possible. We are also grateful to our team members for their help and assistance throughout this project. Their collaboration and support were instrumental in achieving the goals of this project. Finally, we extend our thanks to our department and faculty members for their unwavering support and understanding throughout the course of this project. Their encouragement and belief in us have been a constant source of motivation and inspiration.

Badavath Gayathri	Maradana Surya Sai Indra	Baliyarusimhula Pravallika
420110	420205	420112
Date:	Date:	Date:

May 7, 2024

Table of Contents

List of Figures

List of Algorithms

List of Symbols and Abbreviations

DP	Differential Privacy
SGD	Stochastic Gradient Descent
GAN	Generative Adversarial Network
CGAN	Conditional Generative Adversarial Network
PATE	Private Aggregation of Teacher Ensembles

Chapter 1

Abstract

This project explores the implementation of Differential Privacy (DP) in multiple complex machine learning models. DP assures that the inclusion or exclusion of an individual's data has no major impact on the outcome of data analysis, hence protecting individual privacy in statistical queries. DP, a robust privacy-preserving mechanism, is increasingly vital in the era of data-driven decision-making, particularly in sensitive domains such as healthcare and finance. This paper studies the incorporation of differential privacy into distinct deep learning architectures, Trying to maintain a equivalence between privacy and accuracy. By analyzing the performance and trade-offs across different deep learning methods, this research contributes to the advancement of privacy aware machine learning techniques, fostering trust and confidentiality in data-driven applications.

Chapter 2

Introduction

Implementing Deep learning in network based models had became a trend in recent models, including image classification [1], facial recognition [2, 3], and so on many models had added DP into them. Training these networks involves a massive quantity of data, including sensitive user information. However there are some studies like model inversion [4] and membership inference attacks [5], had made the leakage of data easier. Many approaches have been presented to give privacy to user data, including k-anonymity [6], momorphic encryption [7], L-Diversity [8], and so on. Most of these methods attempt to encrypt data into cipher text or desensitize the data. However, they are ineffective against certain types of attacks.Suppose we have a dataset containing information about the ages of individuals. By querying the average age of the top 99 records and the top 100 records, we can infer the approximate age of the 100th person in the dataset. We call it Differential Attack.

Differential Privacy [9, 13] can prevent such Differential attacks.It gives privacy protection. It promises privacy by adding noise to the machine learning model before generating the output, resulting the non leakage of the data. Differential privacy ensures that an external user cannot get any kind of private information from the data records. Some papers [10, 11, 12] shows the usage of DP in some ML models. DP in a Deep Learning model [9] is done by implementing noise addition to gradient during training so that the output is differed. We explored on implementation of this privacy mechanism in Stochastic Gradient Descent, Generative Adversarial Networks, Conditional Generative Adversarial Network and PATE - a methodology in Differential Privacy.

Chapter 3

Literature Review

3.1 Introduction

In the era of big data, maintaining privacy while utilizing machine learning (ML) and deep learning (DL) techniques has become paramount. Differential privacy provides a framework for adding noise to datasets in such a way that the privacy of individual data points is guaranteed while still allowing for useful computations. This literature review delves into various applications of differential privacy in ML and DL, summarizing key methodologies and findings from recent research.

3.2 Advancements in Deep Learning Architectures

He et al. (2015) discuss the enhancement of neural network architectures, particularly through the use of rectified linear units, achieving surpassing human-level performance on image classification tasks [1]. Similarly, Hochreiter and Schmidhuber (1997) introduced Long Short-Term Memory (LSTM) networks, a cornerstone in the development of deep learning for processing sequences [19]. Lecun et al. (1998) further elaborate on gradient-based learning techniques applied to document recognition, marking a significant step forward in deep learning methodologies [20].

3.3 Generative Adversarial Networks (GANs) and Privacy

GANs have emerged as a powerful tool not only for generating synthetic data but also for enhancing privacy. Antipov et al. (2017) demonstrated the use of conditional GANs for realistic face aging applications [2], while Goodfellow et al. (2014) introduced the foundational framework of GANs that revolutionized the generation of synthetic data [21]. Recent advancements have integrated differential privacy within GAN frameworks to ensure data privacy while maintaining utility [22, 23].

Differential privacy has become a cornerstone of privacy-preserving data analysis. Chaudhuri et al. (2011) explored differentially private empirical risk minimization to ensure privacy without compromising the effectiveness of machine learning models [10]. Recent works by Papernot et al. highlight the potential of differential privacy in deep learning, particularly in semi-supervised settings [27, 30].

Model inversion and membership inference attacks pose significant threats to ML model integrity. Fredrikson et al. (2015) detailed these vulnerabilities and proposed basic countermeasures to mitigate risks associated with confidence information leakage [3, 4]. Shokri et al. (2016) also discussed how membership inference attacks could exploit machine learning models, highlighting the need for robust defensive strategies [5].

Beyond differential privacy, other techniques such as k-anonymity and l-diversity have been explored to protect user data. LeFevre et al. (2005) and Machanavajjhala et al. (2007) provide insights into these methodologies, which aim to enhance privacy guarantees beyond the capabilities of differential privacy [6, 8].

Emerging machine learning techniques continue to be explored for diverse applications. Myles et al. (2004) discuss decision tree modeling, while Su et al. (2012) and Alsabti et al. (1997) delve into the nuances of linear regression and k-means clustering, respectively [16, 17, 18]. These techniques underline the ongoing evolution and application of machine learning across different domains. This narrative organizes the provided references into a structured discussion, linking advancements in technology with ongoing challenges and solutions.

Chapter 4

Preliminaries

In this section, we review Differential Privacy, Generative Adversarial Network GAN, Conditional Generative Adversarial Network and PATE.

4.1 Introduction on Differential Privacy

A mathematical framework called Differential Privacy [10, 11] is used to show how much an individual's privacy is preserved in statistical databases. A Perfect privacy is applied to algorithms by aggregating databases. By adding noise to the outputs, inputs, Gradient, DP ensures privacy of data. Therefore, after making the results or machine learning models available to public, Attackers are not able to infer or predict any information for any private record.

Definition : $(DifferentialPrivacy)$ Training mechanism M_l of a machine learning model with a database D, can be said as (ε, δ)- differentially private, it should satisfy below inequality

$$P[M_l(d_1) \in R'] \leq e^{\varepsilon} P[M_l(d_2) \in R'] + \delta \tag{4.1}$$

Where d_1,d_2 $\in$D differ by only one sample and R'$\in$R subset from the output set, $M_l(d_1)$ and $M_l(d_2)$ are outputs of the model with inputs d_1 and d_2. ε describes how much privacy is added to the model and δ is privacy loss probability. It can be shown that the Above equation can also be written as to

$$\left| \log \left(\frac{Pr(M_l(d_1) = R')}{Pr(M_l(d_2) = R')} \right) \right| \leq \varepsilon \tag{4.2}$$

Theorem : Given a (ε, δ)-differentially private randomized algorithm $M_l : D \to R$ and a random mapping $f : R \to R'$, the composition of f and M $f \circ M_l : D \to R'$ is also (ε, δ)-differentially private.

To make the output of the function $f : D \rightarrow R$ private and secure, gaussian noise having its parameter values mean as 0 and variance to the scale of S_f, the sensitivity of the function f is added, whereS_f is the maximum absolute distance between $f(d_1)$ and $f(d_2)$. In formal notation:

$$S_f \equiv \max_{d_1 \sim d_2} |f(d_1) - f(d_2)| \tag{4.3}$$

Gaussian noise $N(0, S_f^2 \sigma^2)$ is added to the function f to make it differentially private. It's Mathematical notation

$$M_l(d_1) \equiv f(d_1) + N(0, S_f^2 \sigma^2) \tag{4.4}$$

4.2 GAN

Goodfellow et al.[25] introduced the Generative Adversarial Network (GAN), a popular method for image generating models. GAN's primary components are: discriminator, generator. Generator takes a noise as input and tries to generate data which is as likely as original data distribution. Discriminator takes input from real data and fake data and tries to separate the generated data from the real data set. Upon continous training The generator improves it's outputs quality, while the discriminator improves its model to seperate it. Architecture is shown in Figure 4.1

Let z be a noise that is taken from a random probability distribution $P_z(z)$,then $G(z)$ is the generator

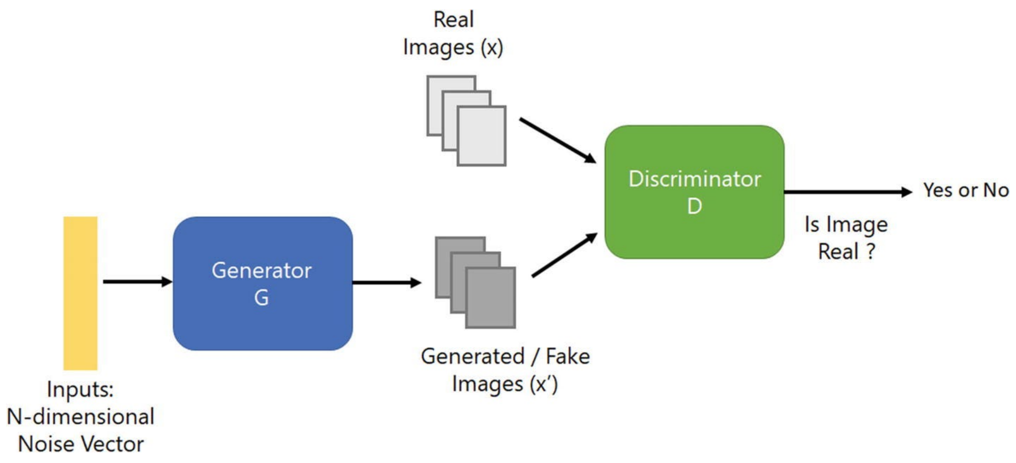


Figure 4.1: GAN Architecture [29]

network's output by taking z as input and x can be either an input from real data distribution $p_{data}(x)$ or output from the generator, then $D(x)$ is the discriminator network's output by taking x as input. Both generator and discriminator competes each other to beat one another. The objective function of this min-max game is:

$$\max_{D_{isc}} \min_{G_{en}} V(G_{en}, D_{isc}) = E_{z \sim p_z(z)}[\log(1 - D_{isc}(G_{en}(z)))] + E_{d \sim p_{\text{data}}(d)}[\log(D_{isc}(d))] \tag{4.5}$$

4.3 CGAN

Conditional GAN [26] extends GAN by conditioning both the Discriminator and Generator on ”y” information, which can be anything from class labels to data. CGAN Architecture is shown in Figure 4.2.
The objective of CGAN’s min-max game is :

$$\max_{D_{isc}} \min_{G_{en}} V(G_{en}, D_{isc}) = E_{z \sim p_z(z)}[\log(1 - D_{isc}(G_{en}(z|y)))] + E_{d \sim p_{\text{data}}(d)}[\log(D_{isc}(d|y))] \tag{4.6}$$

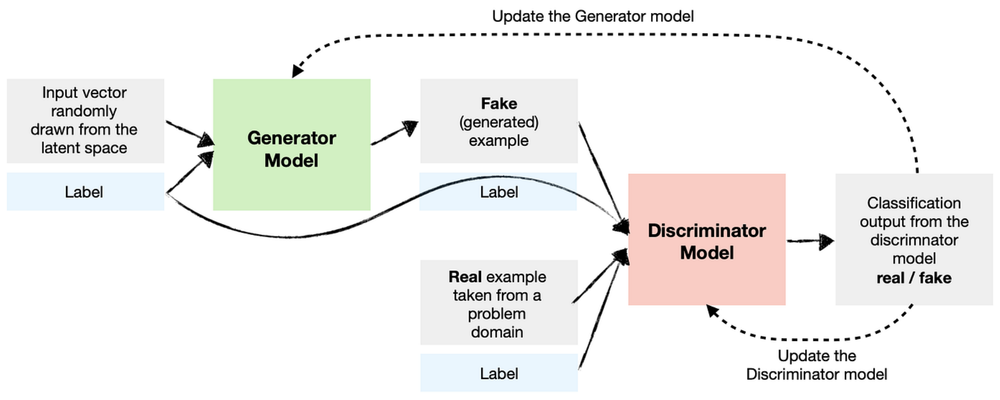

Update the Generator model
Input vector randomly drawn from the latent space
Label
Generator Model
Fake (generated) example
Label
Discriminator Model
Classification output from the discrimnator model real / fake
Real example taken from a problem domain
Label
Update the Discriminator model

Figure 4.2: Conditional Gan Architecture [28]

4.4 PATE

Neural Networks may accidentally remember personal data, potentially compromising privacy. Papernot et al. [27] presented (PATE) Private Aggregation of Teacher Ensembles algorithm, which offers better privacy guarantee. This method is a combination of training of multiple models trained on different datasets (subsets of training data). These multiple models wont get published, but they are used as ”teachers” to train ”student” model. Without any access on teacher model or parameters or training data, Students will produce output based on voting in between all the teacher’s output and adding noise to it. Figure 4.3 describes the framework.
The disjunct sensitive data of the teacher’s training should always have the PATE technique which

has to be involve which includes some examples like data collected from different users and also involving the aggregate aware answers to mainly focus the training of the student module.
A hard core differential policy which is applied by the privacy guarantee is to be disclosed only in

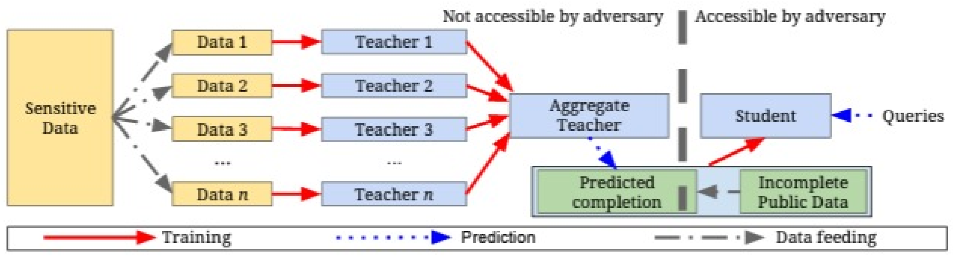


Figure 4.3: PATE Architecture [30]

the student module and this incorporation of the laplace noise for the aggregate gives a usage to the trained students.This pate model includes the knowledge of teacher's model aggregate results which inturn trains the student module which can avoid the attackers from directly having the access of the model or the private data ,even the parameters.This pate ,has been evolutionalised to offer flexible support for many models which even include models of deep neural networks. The training of pate goes as follows. 1st we prepare the training data for teachers model (dividing the training data into disjoint sets) using the upsampling method Algorithm 1 describes the upsampling method in pate.

Algorithm 1: Upsampling method

```
**Input:** Precision $p' \in N$, privacy budget $\varepsilon_x$ for each data point $x$
**Result:** $u_x$ upsampling factor for each data point x
$\varepsilon_1 \dots \varepsilon_i \leftarrow unique\varepsilon_x$ /* Get distinct budgets*/
**for** $Each\varepsilon_i$ **do**
    $\varepsilon' \leftarrow \varepsilon_i.10^p$ /*Upscale the budgets*/
**end**
**for** $Each\varepsilon_i'$ **do**
    $u_d \leftarrow \frac{\varepsilon_i'}{G}$
**end**
```

After that we assign weights to teachers model so that these weights are used in aggregation method which is used to decide the output of unlabeled input from the multiple outputs of the data. below Algorithm 2 explains the weighting algorithm used to add weights to each teacher.

Algorithm 2: Weighting method

```
Input: Privacy budget $\varepsilon_x$ for each data point $x$, number of teachers $n_j$ for each privacy
       group $g_j, j \to 1 \dots G$, total number of teachers $K$.
Result: $W_i$ Weight of teacher $t_i$
$\varepsilon \leftarrow \sum_{j=1}^{G} \varepsilon_j$
for Each privacy group $g_j$ do
    $\varepsilon'_j \leftarrow \frac{\varepsilon_j}{\varepsilon}$ /* Relative Privacy Budget */
    $\bar{n}_j \leftarrow \frac{n_j}{k}$ /* Relative group size */
    $\bar{w}_j \leftarrow \varepsilon'_j \cdot \bar{n}_j$
end
$W \leftarrow \sum_{j=1}^{G} \bar{w}_j$
for Each privacy group $g_j$ do
    $w_j \leftarrow \frac{\bar{w}_j}{W} \cdot k$ /* Make sum of weights to match k */
    for Each teacher $t_i$ with data from $g_j$ do
        $w_i \leftarrow w_j$
    end
end
```

Then comes the aggregation phase. Aggregation is the phase where the model finalizes the output from the multiple outputs that we will get from the teachers. PATE follows two algorithms. Confident GNMax aggregator and Interactive GNMax aggregator. Algorithm 3 shows the working of Confident GNMax and Algorithm 4 shows the working of Interactive GNMax

Algorithm 3: Confident-GNMax Aggregator

```
Input  : Input $x$, threshold $T$, noise parameters $\sigma_1$ and $\sigma_2$
Output: Confident teacher prediction
if $\max_i\{n_j(x)\} + \mathcal{N}(0, \sigma_2^2) \geq T$ then
    // Privately check for consensus
    return $\arg\max_j\{n_j(x) + \mathcal{N}(0, \sigma_2^2)\}$ // Run the usual max-of-Gaussian
end
else
    return No prediction // No confident prediction
end
```

where n_j is number of teachers from which x is came as output. This algorithm works by verifying if the maximum from the outputs is with in the threshold T or not, if it is, then the model adds the noise to the output which came maximum number of times.

Algorithm 4: Interactive-GNMax Aggregator

```
Input  : Input $x$, confidence $\gamma$, threshold $T$, noise parameters $\sigma_1$ and $\sigma_2$, total number of
         teachers $M$
Output: Aggregated prediction
Ask the student to provide prediction scores $p(x)$ ;
if $\max_j\{n_j(x) - M \cdot p_j(x)\} + \mathcal{N}(0, \sigma_2^2) \geq T$ then
    // Student does not agree with teachers
    return $\arg\max_j\{n_j(x) + \mathcal{N}(0, \sigma_2^2)\}$ // Teachers provide new label
end
else if $\max\{p_i(x)\} > \gamma$ then
    // Student agrees with teachers and is confident
    return $\arg\max_j p_j(x)$ // Reinforce student's prediction
end
else
    return No action // No action taken
end
```

Interactive GNMax is a method which interacts with student to finalise the output. The confidence γ will come from student which says how confident the student agrees with the teacher. $p_j(x)$ is probability of getting x as output and $n_j(x)$ is number of teachers from which x came as output. After the output came from the aggregator, the student takes input from the aggregator and use it for its training. We observe that the noise addition is done during the aggregation part. this ensures that during the training, the user wont have any access to the training data which helps us in achieving differential privacy to the data.

Chapter 5

Implementation

The implementing thought is to send the noise carefully into model and make it differentially private and so that we can avoid the data leakage. We make an additional work on the the objective function of the model which is adding noise to it during training, A control on the noise that we add should be there to get an accurate differential privacy. Also a specific strategy of separately clipping gradients for real and generated data losses. We Implemented RDP accounting technique to track the privacy loss that we spent during the training.During the training process, we added noise to the gradients computed during back propagation. A We inject noise during training but it can also

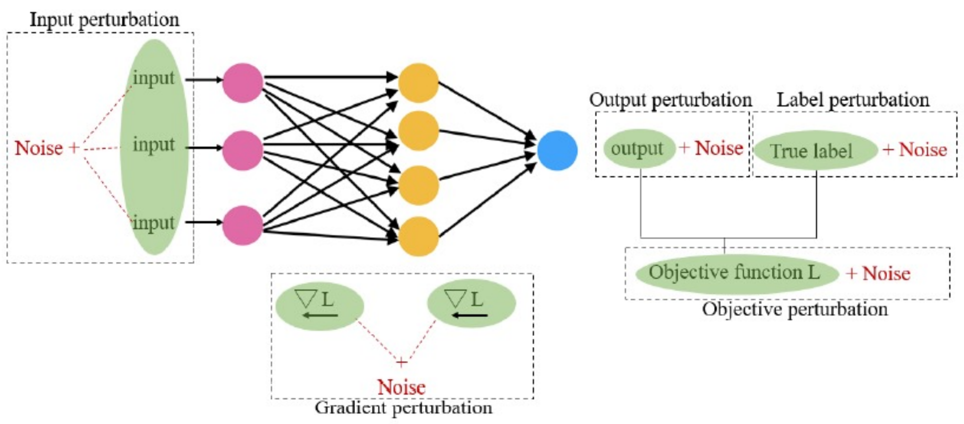


Figure 5.1: An overview on ways of adding noise in a Machine Learning Pipeline [28]

added to input, output, objective function, gradient descent or to ground truth labels as shown in Figure 5.1. We followed the method of adding noise to the gradient descent while training. The process followed in implementing DP in each of the model is

- Computing the Gradient of the model
- Gradient Clipping.
- Perturbation to the gradient
- Parameter Updates of the model

Computing Gradient Descent : Compute the gradient descent of the model based on its architecture

Norm Clipping : To prove the privacy guarantee, Each training requires a bounding to the influence of each individual example on final gradient we get while training. For that we clip each gradient in l_2 norm with a clipping threshold S

Adding noise : Once the gradient is clipped, before updating the parameters of the model, Gaussian noise $N(0, \sigma^2 S^2 I)$ is added to the gradient descent.

Updating Parameters : After the noise addition, the model training resumes
The core principle of training the model (choosing training data, Computing gradient descent, updating parameters) is followed as it is and additional work is that the gradient is clipped and noise is added to it in each model.

5.1 Differential Privacy in SGD

Below Algorithm 5 shows in detailed implement Differential Privacy in Deep Learning using stochastic gradient descent.

Algorithm 5: Differential Privacy in SGD

```
Input: Data samples $d_1, \dots, d_N$, loss func $L(X) = \frac{1}{N}\sum_i L(X; d_i)$, Clipping threshold C
Output: Parameters $X_T$, total privacy cost $(\varepsilon, \delta)$
Initialize $X_0$ to some random values
for $t = 1$ to $T$ do
    Pick a sample $D_t$ from the data samples with picking probability $D/N$
    Calculate the Gradient
    for each $i \in D_t$ do
        Calculate $g_t(d_i) = \nabla_x L(X_t; d_i)$
    end
    Clip the gradient
    for each $i \in D_t$ do
        $g_t(d_i) \leftarrow \text{clip}(g_t(d_i), -C, C)$
    end
    Add noise
    $\tilde{g}_t \leftarrow \frac{1}{D}\left(\sum_i g_t(d_i) + \mathcal{N}(0, \sigma^2 C^2 I)\right)$
    Update the parameters using SGD
    $X_t \leftarrow X_{t-1} - \eta_t \tilde{g}_t$
end
Output $X_T$
compute total privacy spent using moments account method
```

The traditional training method of the model is followed as it is with an additional work which is adding noise. The algorithm works by for an iteration Picking data points having picking probability of D/N, and then we calculate the gradient descent of the model using the data we picked. we make sure that the gradient does not dominate by clipping the gradients to C. where C is threshold. Once the gradient computation and clipping are done, the guassian noise is added to the final gradient which is further used to update the parameters using Stochastic Gradient Descent Method.

5.2 DP in Generative Adversarial Network

Generative Adversarial Network a Min-Max game where Discriminator and Generator tries to win on one another. It is found that implementing Differential privacy in Adversarial Networks is essential since training of discriminator requires a training data which consists of private information of users such as Medical data etc... We implement DP by sending gaussian noise to the discriminator GAN's training is done by a repetetive process of :

- Pausing the Generator and training the Discriminator
- Pauseing the Discriminator and training the Generator

Algorithm 6 shows the detailed implementation of Differential Privacy in GAN.

Algorithm 6: Differential Privacy in Generative Adversarial Nets

```
Input: $\alpha_{gen}$, generator's learning rate.$\alpha_{disc}$, discriminator's learning rate.$X$ Discriminator's
       parameters, $G$ Generator's Parameters. Data distribution $D$. $M$ batch size. $c_p$,
       parameter clipping threshold. $m$ batch size. $n_g$ number of generator iterations . $n_d$
       number of iterations for discriminator.$\sigma_n$ variance for gaussian noise. $c_g$ gradient
       clipping constant
Output: Generator $G$ Differentially privated.
Initialize $X_0$ - Disc parameters, $G_0$ - Gen parameters;
for $t = 1$ to $n_g$ do
    for $t' = 1$ to $n_d$ do
        Pick $\{d^{(i)}\}_{i=1}^{m} \sim p_{data}(d)$ a batch of real data points;
        Pick $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of generated data points;
        for each $i$ do
            $\delta x(d^{(i)}, z^{(i)}) \leftarrow \nabla_x f_x(d^{(i)}) - \nabla_x f_x(\hat{\theta}(z^{(i)}))$;
        end
        $\bar{\delta x} \leftarrow \frac{1}{m}\left(\sum_{i=1}^{m} \delta x(d^{(i)}, z^{(i)}) + \mathcal{N}(0, \sigma_n^2 c_g^2 I)\right)$;
        $x(t') \leftarrow x(t'-1) + \alpha_{disc} \cdot \mathrm{RMSProp}(x(t'), \bar{\delta x})$;
        $x(t') \leftarrow \mathrm{clip}(x(t'), -c_p, c_p)$;
    end
    Pick $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$, another batch of generated data points;
    $\delta G \leftarrow -\frac{1}{m}\sum_{i=1}^{m} \nabla_G f_w(\hat{G}(z^{(i)}))$;
    $G(t) \leftarrow G(t-1) - \alpha_{gen} \cdot \mathrm{RMSProp}(G(t-1), \delta G)$;
end
return $G$;
```

The traditional training of GAN is followed as it is with an additional work which is adding noise to the discriminator. As said above the training is done in a 2 process in each iteration. First we train the discriminator. We pick a batch of data from the real data distribution and a batch of data from the generated data from the generator. Then we compute the gradient with respect to both the real and generated data. Once the gradient computation is done, we add the gaussian noise to the gradient and we compute the final gradient of the data, once done, using RMSProps method, the parameters of the discriminator are updated, then we clip the parameters of the discriminator to make sure the unnecessary domination of any parameter on others. At this step the discriminator training finishes, then the generator training starts. We pick noise from the noise distribution and we compute the gradient with respect to each noise value we picked. After that, we update the parameters of the generator using RMSProp method.

5.3 DP in Conditional GAN

Conditional GAN works similar to GAN where conditional rendering is done in addition. Image generation is done conditionally. Implementing DP in CGAN is found essential as it requires (labeled)training data for training discriminator. Algorithm 7 shows the in detailed implementation of Differential Privacy in Conditional Generative Adversarial Network

The traditional methodology of training of CGAN is followed with an additional step which is addition of noise to the discriminator. Similar to how we did training in GAN we do the same to the CGAN to..

We pick a sample of data from the real data distribution and a sample of generated data from the generator's output. then we compute the losses from both real data and the generated data. After computing the losses, we compute the gradients with respect to the losses of generated and real data and clip them to avoid the unecessary domination of one gradient over others. After this step, we add the both real and generated gradients to compute the final gradient of the discriminator. In this step, we add guassian noise to the combination of gradients with respect to real and generated data's losses. Once done, we update the parameters of the discriminator using SGD method.. Training of discirminator completes here. Now we compute the loss of generator with respect to the noise which we took to generate the data which used in discriminator's training. Then, usinf ADAM Optimizer, we update the parameters of the model. Once done, we verify weather to continue or stop the training by computing the spent epsilon (DP Parameter). If this condtion satisfies, we stop the training else, we continue the training.

```
Algorithm 7: Differentially Private -CGAN
Input: Dataset $\{d_1, d_2, \ldots, d_N\}$, Labels $\{l_1, l_2, \ldots, l_N\}$,Gradient Clipping threshold $C$,batch
        size $b_s$, learning rate $lr$, DP parameters : epsilon $\varepsilon$, delta $\delta$, noise scale $\sigma$
Output: Differentially private Conditional Generator G
stop $\leftarrow$ $False$;
while $!stop$ and $step \leq max_step$ do
    From the data distribution $p_{data}(X)$ sample a random batch $(D_t, L_t)$ having batch size
      of $b_s$ with probability of sampling $bs/N$ ;
    From the noise distribution $p_z(Z)$ sample a noise batch $Z_t$ of size $b_s$;
    /* Pausing Generator and Training the Discriminator */;
    $d\_loss\_gen \leftarrow -\log(1 - D(G(Z_t)))$ - /*Generated data loss*/;
    $d\_loss\_real \leftarrow -\log(D(D_t))$ - /*Real data loss, $D_t$ data point*/ ;
    Compute gradients for disc. loss on gen. data $Z_t$ and clip them ;
    for each $i \in Z_t$ do
        Measure $grad^t_{d\_gen} \leftarrow \nabla_{\theta_d} d\_loss\_gen(\theta^t_d, z_i)$;
        $grad^t_{d\_gen} = grad^t_{d\_gen} / \max(1, ||grad_{d\_gen}||_2 C)$;
    end
    Compute gradients for disc loss on real data $p_{data}(X)$ and clip them ;
    for each $i \in D_t$ do
        Measure $grad^t_{d\_real} \leftarrow \nabla_{\theta_d} d\_loss\_real(\theta^t_d, d_i)$;
        $grad^t_{d\_real} = grad^t_{d\_real} / \max(1, ||grad_{d\_real}||_2 C)$;
    end
    Add both the computed gradients and add noise to it;
    $grad^t_d \leftarrow (1/bs) \sum (grad^t_{d\_real} + grad^t_{d\_gen} + \mathcal{N}(0, \sigma^2 C^2 I))$;
    Computing SGD for discriminator and updating the parameters;
    $\theta^{t+1}_d \leftarrow SGD(grad^t_d, \theta^t_d, lr)$;
    /* Update the DP metric RDP */;
    /* Pausing the Discriminator and Training the Generator */;
    $g\_loss \leftarrow -\log(1 - D(G(Z_t)))$;
    Find gradients for gen loss on noise distribution $Z_t$;
    $grad^t_g \leftarrow \nabla_{\theta_g} g\_loss(\theta^t_g, z_i)$;
    Computing ADAM optimizer for generator and updating the parameters;
    $\theta^{t+1}_g \leftarrow ADAM(grad^t_g, \theta^t_g)$;
    if delta spent $> \delta$ —— epsilon spent $> \varepsilon$ then
        terminate $\leftarrow$ True;
    end
end
```

5.4 PATE in CGAN

Implementation of PATE architecture is done for CGAN to implement differential privacy in CGAN. To follow the principle of CGAN and also the methodologies of PATE (Student, Teacher and Aggregation), Thinking of all the above senarios the below figure 5.2 is used. This model uses discrimators as teachers and generator as child. As mentioned above, following the principle of PATE, we take 10 teachers(discriminators) and one student(generator). We divide the training dataset (MNIST in our case) into 10 disjoint subset and train each of them. The model works as follows,

The input from training data (unlabeled) is sent to teachers(discriminators). The 10 outputs from each teacher is sent to differentially private aggregator (Which chooses a label to the input which is sent to teachers based on outputs from the teachers and adds laplacian noise to it). The output is a noisy image with label given to it. This output from aggregator is sent to Generator (In traditional Conditional GAN, the input to the generator must be a noisy image and a label corresponding to which the generator should generate the image which is exactly happening here), using which the generator generates an image which is sent to discriminators(teachers) to find weather the image is real or generated one. Algorithm 8 shows the implementation of PATE in CGAN

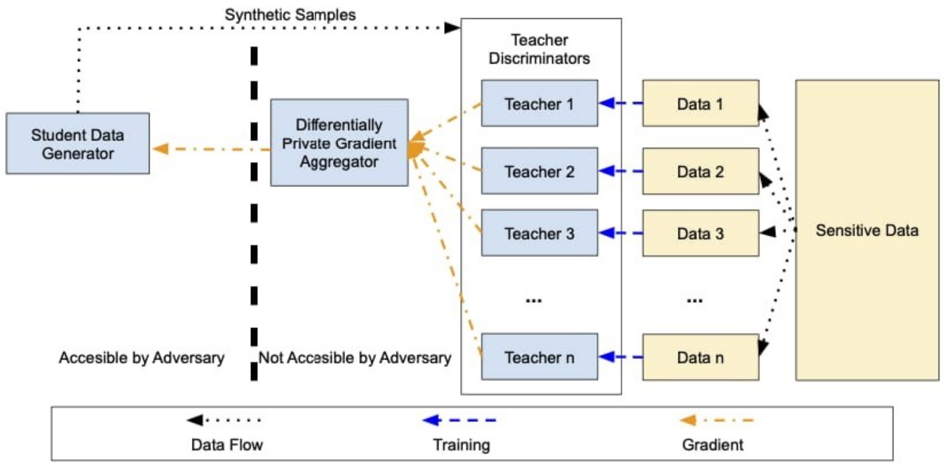


Figure 5.2: Architecture of CGAN using PATE

Algorithm 8: PATE mechanism for Conditional GAN with noisy image output

```
Input  : Input data $x$, generator $G$, Trained discriminators $D_1, D_2, \ldots, D_n$, noise parameters
         $\sigma_1$ and $\sigma_2$, threshold $T$, confidence $\gamma$
Output: Aggregated predictions, updated discriminator parameters
for Each iteration do
    // Training generator
    Sample minibatch of noise vectors $\{z_1, z_2, \ldots, z_m\}$ from noise prior ;
    Generate fake samples $\{x'_1, x'_2, \ldots, x'_m\}$ using generator $G$ with noise vectors
      $\{z_1, z_2, \ldots, z_m\}$ ;
    Update generator $G$ using the adversarial loss and update rule ;
    // Aggregating predictions using Confident-GNMax
    for Each discriminator $D_j$ do
        Compute discriminator's prediction confidence $C_j$ for each generated sample $x'_i$ ;
    end
    if $\max_j\{C_j\} + \mathcal{N}(0, \sigma_2^2) \geq T$ then
        Aggregate predictions using Confident-GNMax and update discriminator
          parameters ;
        // Generate noisy images with labels
        Sample minibatch of noise vectors $\{z'_1, z'_2, \ldots, z'_m\}$ from noise prior ;
        Generate noisy images with labels $\{(x''_1, y_1), (x''_2, y_2), \ldots, (x''_m, y_m)\}$ using generator
          $G$ with noise vectors $\{z'_1, z'_2, \ldots, z'_m\}$ ;
        // Update generator using the noisy images
        Update generator $G$ using the adversarial loss and update rule with noisy images
          $\{(x''_1, y_1), (x''_2, y_2), \ldots, (x''_m, y_m)\}$ ;
    end
    else if $\max\{C_i\} > \gamma$ then
        Update discriminator parameters based on student's prediction ;
    end
    else
        No action ;
    end
end
```

Chapter 6

Experimentation and Results

Dataset: MNIST is a specialised dataset which mainly consists of training samples and the test samples that are present at a varied collection of images with the dimensionalites of 28*28 grayscale images of the specified hand written digits right from 0 till 9 which are followed by their corresponding own labels.This MNIST is the benchmark dataset which gets the best results especially in the machine learning and computer vision related tasks that are mainly used for developing and also for testing algorithms in the domain of digit recognition tasks.This dataset in detail has 60 k training images and 10k test images which serves as the standard for mainly performing the computer vision ,image classification tasks in specific.

Model Architecture: In Differentially Non Private, Private CGAN and CGAN with standard DP use a vanilla CGAN architecture. Discriminator and Generator have 2 fully connected layers and a ReLU activation function.

In Differentially Private GAN. Discriminator and Generator have 2 Fully Connected layers and a activation functions Relu and Tanh

Input to Generator and Discriminator: Generator receives noise z from a noise distribution as input and any corresponding label y in case of CGAN. The label y and the actual training sample x are the discriminator inputs.

Generator receives a random noise sample z as input for the GAN. Real training sample x is one of the discriminator inputs.

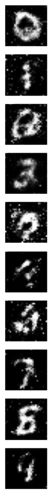
Differential Privacy Implementation: Tensor-Flow Privacy is a Python library used in differentially private CGAN and GAN models.The models can generate fake data by using MNIST training data set. For privacy measures, we use Renyl Diff. Privacy (RDP) metric.

Testing and Performance Metric: We used Area under the ROC curve to evaluate the performance of classifiers. Figure6.6 shows the results we got from the experimentation

Objective: Outputs of implementation of Differential Privacy in SGD, GAN, CGAN and Pate Architecture is shown in below figures.

Clustering of Differentially Private images from CGAN After adding differential Privacy to the Images, We thought of what can we do with the differentiaally private images that got generated from Differentially Private CGAN. Therefore came an idea of clustering the images using t-SNE and PCA. Both methodologies are used for clustering purposes but they have different approaches. t-SNE is used for better visual representation where PCA is used for getting better results. figure 6.4 and figure 6.5 shows the outputs of the implementation of PCA and t-SNE clusters

Outputs of our work Figure 6.1b shows the output we got while implementing DP in Gan and CGAN. Figure 6.2 and 6.3 shows the output of implementing pate methodology for CGAN.


(a) DP in CGAN

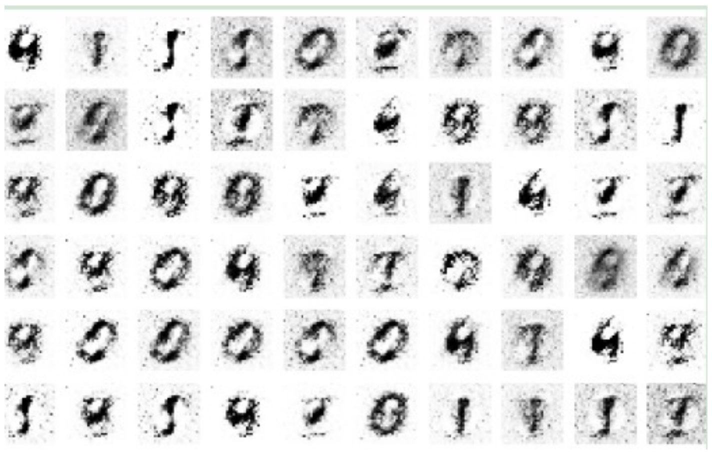

(b) DP in GAN

Figure 6.1: DP in Variants of GAN

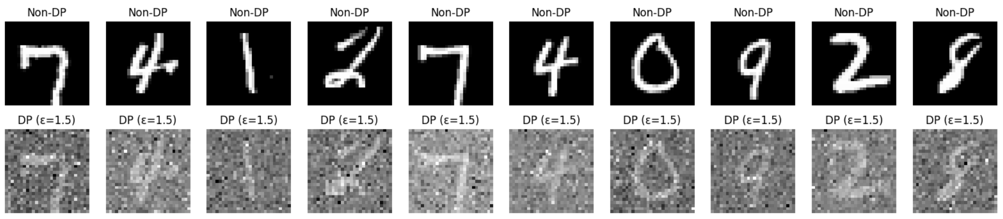


Figure 6.2: PATE in CGAN

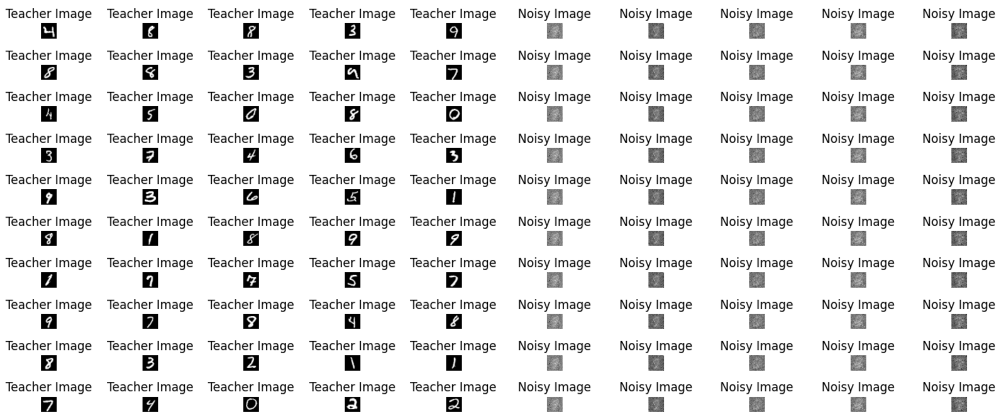


Figure 6.3: PATE in CGAN

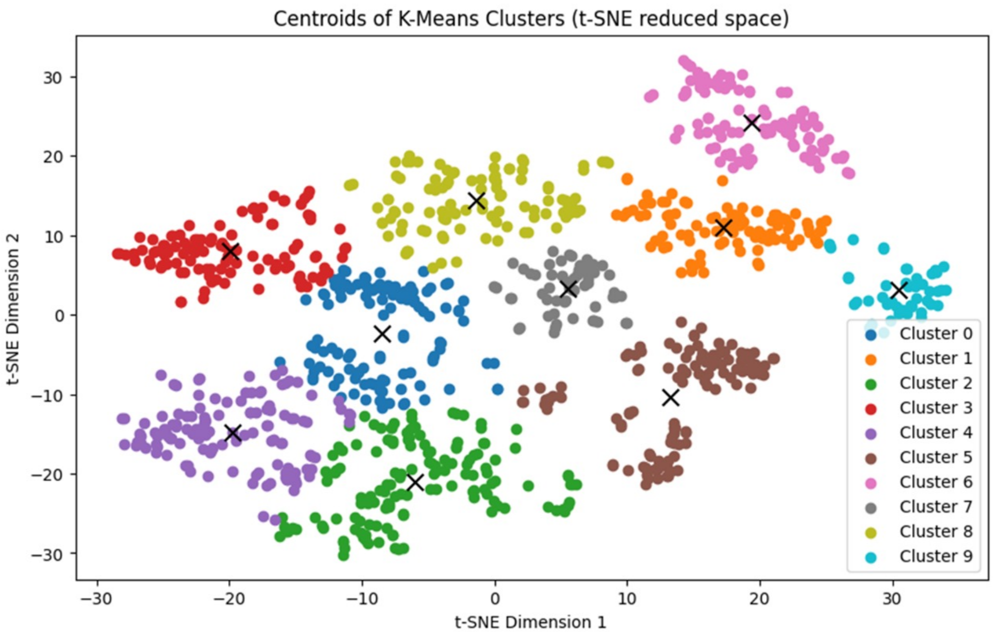


Figure 6.4: Clustering using t-SNE

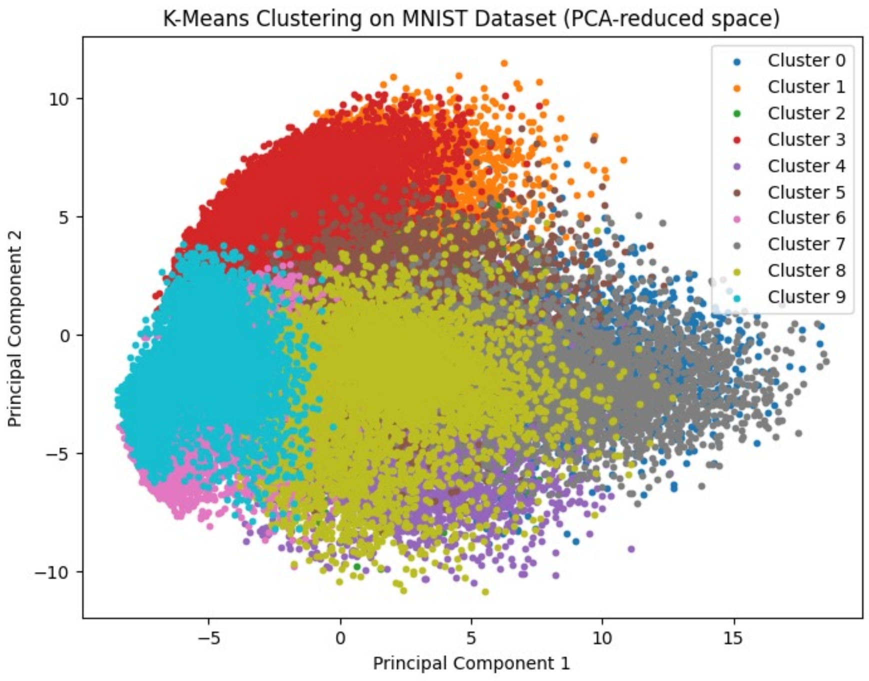


Figure 6.5: Clustering using PCA

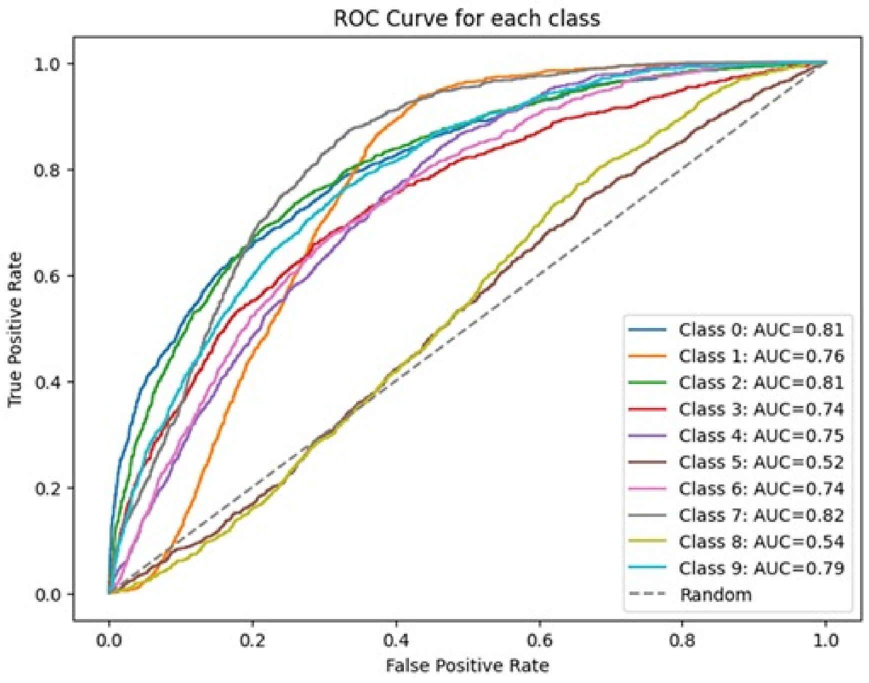


Figure 6.6: ROC Curve

Chapter 7

Conclusion

In conclusion, this thesis has provided a comprehensive examination of the implementation of differential privacy within diverse deep learning methodologies. Through meticulous experimentation and analysis, we have elucidated the intricate interplay between privacy preservation and model performance across a range of applications. Our findings underscore the importance of considering differential privacy as a foundational element in the design and deployment of deep learning systems, particularly in domains where data confidentiality is paramount. Moving forward, the insights gleaned from this study can inform the development of more robust and privacy-aware machine learning solutions, fostering trust and accountability in data-driven decision-making processes. As technology continues to evolve, it is imperative to prioritize privacy safeguards to ensure the ethical and responsible utilization of data in the pursuit of innovation and societal progress.

Bibliography

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: surpassing human-level performance on imagenet classification," in Proceedings of the 2015 IEEE International Conference on Computer Vision, ICCV 2015, pp. 1026–1034, Santiago, Chile, December 2015.

[2] G. Antipov, M. Baccouche, and J. Dugelay. Face aging with conditional generative adversarial networks. In 2017 IEEE International Conference on Image Processing (ICIP), pages 2089–2093, Sep. 2017

[3] M. Fredrikson, S. Jha, and T. Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, pages 1322–1333. ACM, 2015.

[4] att Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security - CCS '15, New York, New York, USA, 2015. ACM Press.

[5] Reza Shokri, Marco Stronati, and Vitaly Shmatikov. Membership inference attacks against machine learning models. CoRR, abs/1610.05820, 2016.

[6] I. K. LeFevre, D. DeWitt, and R. Ramakrishnan, "Efficient fulldomain kanonymity," in Proceedings of the ACM SIGMOD International Conference on Management of Data, Baltimore, MD, USA, June 2005.

[7] C. S. Gu, "Fully homomorphic encryption from approximate ideal lattices," Ruan Jian Xue Bao/Journal Software, vol. 26, pp. 2696–2719, 2015.

[8] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam, l-Diversity: Privacy Beyond, Discovery, vol. 1, p. 146, 2007.

[9] T. Ha, T. K. Dang, T. T. Dang, T. A. Truong and M. T. Nguyen, ”Differential Privacy in Deep Learning: An Overview,” 2019 International Conference on Advanced Computing and Applications (ACOMP), Nha Trang, Vietnam, 2019, pp. 97-102, doi: 10.1109/ACOMP.2019.00022.

[10] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate, “Differentially private empirical risk minimization,” Journal of Machine Learning Research: JMLR, vol. 12, pp. 1069–1109, 2011.

[11] K. Chaudhuri and C. Monteleoni, “Privacy-preserving logistic regression,” Proceedings of the 21st International Conference on Neural Information Processing, pp. 289–296, Vancouver, Canada, December 2008.

[12] J. Zhang, Z. Zhang, X. Xiao, Y. Yang, and M. Winslett, “Functional mechanism,” Proceedings of the VLDB Endowment, vol. 5, no. 11, pp. 1364–1375, 2012.

[13] A. E. Ouadrhiri and A. Abdelhadi, ”Differential Privacy for Deep and Federated Learning: A Survey,” in IEEE Access, vol. 10, pp. 22359-22380, 2022, doi: 10.1109/ACCESS.2022.3151670.

[14] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. Deep models under the gan: Information leakage from collaborative deep learning. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, New York, NY, USA, 2017. ACM.

[15] Daniel Lowd and Pedro Domingos. Naive bayes models for probability estimation. In Proceedings of the 22nd international conference on Machine learning - ICML ’05, New York, New York, USA, 2005. ACM Press.

[16] Anthony J. Myles, Robert N. Feudale, Yang Liu, Nathaniel A. Woody, and Steven D. Brown. An introduction to decision tree modeling. Journal of chemometrics, 18(6):275–285, 2004.

[17] Xiaogang Su, Xin Yan, and Chih-Ling Tsai. Linear regression: Linear regression. Wiley interdisciplinary reviews. Computational statistics, 4(3):275–294, 2012.

[18] Khaled Alsabti, Sanjay Ranka, and Vineet Singh. An efficient k-means clustering algorithm. 1997.

[19] S. Hochreiter and J. Schmidhuber. Long short-term memory. Neural computation, 9(8):1735–1780, 1997.

[20] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE. Institute of Electrical and Electronics Engineers, 86(11):2278–2324, 1998.

[21] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. arXiv [stat.ML], 2014.

[22] Liyang Xie, Kaixiang Lin, Shu Wang, Fei Wang, and Jiayu Zhou. Differentially private generative adversarial network. arXiv [cs.LG], 2018.

[23] Chugui Xu, Ju Ren, Deyu Zhang, Yaoxue Zhang, Zhan Qin, and Kui Ren. Ganobfuscator: Mitigating information leakage under gan via differential privacy. IEEE transactions on information forensics and security, 14(9):2358–2371, 2019.

[24] James Jordon, Jinsung Yoon, and M. Schaar. Pate-gan: Generating synthetic data with differential privacy guarantees. ICLR, 2018.

[25] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D.Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In Advances in neural information processing systems, pages 2672–2680, 2014.

[26] M. Mirza and S. Osindero. Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784, 2014.

[27] N. Papernot, I. Goodfellow, M. Abadi, K. Talwar, and ´ U. Erlingsson, "Semi-supervised knowledge transfer for deep learning from private training data," in Proceedings of the 5th International Conference on Learning Representations, ICLR 2017, pp. 1–16, Toulon, France, April 2019).

[28] https://towardsdatascience.com/cgan-conditional-generative-adversarial-network-how-to-gain-control-over-gan-outputs-b30620bd0cc8

[29] https://blog.ldtalentwork.com/2020/10/20/how-to-generate-unique-architectures-using-gans/

[30] Nicolas Papernot and Shuang Song and Ilya Mironov and Ananth Raghunathan and Kunal Talwar and Úlfar Erlingsson. Scalable Private Learning with PATE arXiv preprint arXiv:1802.08908, 2018

[31] Hai, Zhang. (2023). Privacy-Preserving SGD on Shuffle Model. Journal of Mathematics, 2023:1-16. doi: 10.1155/2023/4055950

[32] Jiancheng, Lin., Yanqing, Yao. (2023). Differentially Private Generative Model with Ratio-Based Gradient Clipping. Lecture Notes in Computer Science, 535-549. doi: 10.1007/978-3-031-20096-040

[33] (2023). Differential Privacy Meets Neural Network Pruning. doi: 10.48550/arxiv.2303.04612

[34] Kamil, Adamczewski., Mijung, Park. (2023). Differential Privacy Meets Neural Network Pruning. arXiv.org, abs/2303.04612 doi: 10.48550/arXiv.2303.04612