Privacy Preserving Machine Learning

by

Dhruvilkumar Doshi

A project report submitted in conformity with the requirements
for the degree of Master of Science, Information Technology

Department of Mathematical and Physical Sciences
Faculty of Graduate Studies
Concordia University of Edmonton

PRIVACY PRESERVING MACHINE LEARNING

DHRUVILKUMAR DOSHI

Approved:

Supervisor : Dr. Baidya Nath Saha, Ph. D Date

Committee Member Date

Dean of Graduate Studies: Dr. Alison Yacyshyn, Ph. D Date

Privacy Preserving Machine Learning

Dhruvilkumar Doshi
Master of Science, Information Technology
Department of Mathematical and Physical Sciences
Concordia University of Edmonton
2021

Abstract

A wide range of application sectors is progressively using machine learning (ML). A successful ML model often requires a huge amount of training data and powerful computational resources. Due to the potential risks of highly sensitive information being leaked, the need for and use of such enormous volumes of data raise serious privacy concerns. In addition, the changing regulatory environments that increasingly restrict access to and use of privacy-sensitive data present significant obstacles to fully utilizing the power of ML for data-driven applications. There are several techniques for achieving privacy in ML. Homomorphic Encryption (HE) is a public key cryptographic scheme. HE can perform inference on encrypted data, so the model owner never sees the client's private data and, therefore, cannot leak it. HE is computationally expensive and restricted to certain kinds of calculations. Federated Learning (FL) is a collaborative machine learning method with decentralized data and multiple client devices. During the FL process, each client trains a model on their data set and then sends a model to the server, where a model is aggregated to one global model and then again distributed over clients. Split Learning (SL) is a distributed and private deep learning technique that is used to train neural networks over multiple data sources while mitigating the need to share raw labeled data. This research will provide insight into the trade-off between performance and security for HE, SL, and FL among various ML and DL algorithms.

Keywords:Homomorphic Encryption, Split Learning, Federated Learning, Privacy.

Contents

Chapter 1

Introduction

1.1 Background

Our search queries, browsing history, purchase transactions, videos we watch, and movie preferences are a few types of information collected and stored daily. This data collection occurs on the streets, in our homes and offices, and our computers and mobile devices. Such private data is being used for a variety of machine learning applications. Machine learning (ML) is increasingly utilized for various applications, from intrusion detection to recommending new movies. Some ML applications require private individuals' data. For ML algorithms to extract patterns and create models, such confidential data is uploaded to centralized sites in clear text. The issue extends beyond the dangers of having all of this sensitive data exposed to insider risks within these organizations or external threats in the event that the companies holding these data sets were breached. Additionally, even if the data was anonymized, or if the data and ML models themselves were unavailable and just the test results were made public, it is still possible to learn additional information about the private data sets.

1.2 Problem Statement

ML is used in a wide range of application sectors. For example, in areas of application like computer vision, natural language processing, and speech or audio recognition, recently constructed deep neural networks, commonly referred to as deep learning (DL), have shown considerable advances in model performance and accuracy.[1]–[3]. Another cooperative machine learning method for creating high-quality models while training data is dispersed over numerous decentralized devices is emerging federated learning (FL). [4], [5]. FL has demonstrated its potential in several application areas, including intelligent manufacturing, vehicular networks, and healthcare. [6]–[8].

Although these models have demonstrated significant success in AI- or ML-driven applications, they face several difficulties, such as beginning to itemize an object. Absence of significant computing capabilities, large amounts of data available for model training. For an ML system to perform well, both the training and inference stages need to be supported by powerful computational resources and a lot of training data. Existing commercial ML infrastructure service providers like Amazon, Microsoft, Google, and IBM have put a lot of effort into developing infrastructure as a service (IaaS) or machine learning as a service (MLaaS) platforms with reasonable rental fees to address the need for computing resources with high-performance CPUs and GPUs, large memory storage, etc. Clients with limited resources might use ML-related IaaS or MLaaS to initially maintain and train their models before offering data analytics and prediction services through their applications. Another difficulty for ML systems is the availability of enormous quantities of training data. It makes sense that more training data would lead to an ML model performing better; therefore, collecting vast amounts of data—often from different sources—is necessary. However, due to the dangers of personal or private information being leaked, the building and usage of ML models, as well as the collection and use of data, create severe privacy problems. In light of recent data breaches, for instance, there are now far greater privacy worries around the extensive gathering and use of personal data.[9], [10]. An adversary can also infer confidential information by exploiting an ML model via various inference attacks such as membership inference attacks [11]–[15], model inversion attacks [16]–[18], property inference attacks [19], [20], and privacy leakage from gradients exchanged in distributed ML scenarios [21], [22]. For instance, an attacker can determine whether patient-specific data were used in the training of an ML model for HIV by a membership inference attack. Current laws like the Health Insurance Portability and Accountability Act (HIPPA) and more recent laws like the European General Data Protection Regulation (GDPR), Chinese Cybersecurity Law, California Consumer Privacy Act (CCPA), etc., further restrict the availability and use of privacy-sensitive data. Adopting ML models for practical applications is significantly hampered by these privacy worries and issues. It is essential to develop creative privacy-preserving ML (PPML) solutions to address the growing privacy concerns associated with employing ML in applications where users' privacy-sensitive data, such as electronic health/medical records, location information, etc., are kept and processed. Recently, there have been more attempts made to conduct PPML research, including the construction of novel new privacy-preserving approaches and architectures for ML systems, as well as the integration of current anonymization strategies into ML pipelines. Recent studies on ML, including Federated Learning, such as those in

[23]–[28], show or analyze the unique privacy and security challenges in ML or FL systems in part. Each PPML technique now in use either partially addresses privacy problems or is only useful in specific circumstances. A unified or comprehensive picture of PPML solutions does not exist. For instance, adopting differential privacy in ML systems can lead to model utility loss and reduced model accuracy. Similar substantial communication or calculation overhead is experienced when using secure multi-party computation technologies. Sending a lot of intermediate data, like jumbled tables of circuit gates, results in communication overhead. Adopting advanced cryptosystems causes computation overhead [29], [30]. Several earlier studies, including those that deal with the systematization of knowledge [31] or surveys/analyses [32]–[34], have examined ML security challenges, such as those of stealing the ML models, injecting Trojans, and availability of ML services and accompanying countermeasures. Although privacy is the main concern, there is still a lack of systematization in knowledge appraisal and discussion.

Chapter 2

Literature Review

There are different architectures and paradigms used in Privacy Preserving Machine Learning for training, testing, or both.

2.1 Federated Learning

2.1.1 Introduction

Federated learning, referred to as collaborative learning, is an ML paradigm that uses several distributed edge devices or servers that keep local data samples to train an algorithm without transferring the data samples. This method differs from more typical decentralized approaches, which frequently presume that local data samples are uniformly distributed, as well as traditional centralized machine learning techniques, where all local datasets are uploaded to a single server.

FL and distributed learning have a close relationship. Distributed computing and distributed storage make up a traditional distributed system. In some ways, distributed computation is similar to the original proposed FL of model update for Android clients. Although FL placed a lot of emphasis on privacy preservation, the most recent research in distributed machine learning also gives privacy-preserving distributed systems a lot of attention. Numerous computers in various places can be connected and managed by a central server to perform distributed processing through a communication network. Each computer completes various components of a single task. As a result, FL concentrates on developing a collaborative model without privacy leaks, while distributed processing is primarily focused on expediting the processing step.

Federated learning enables several players to develop an identical, reliable machine learning model without sharing data, enabling the resolution of crucial concerns such

as data privacy, security, access rights, and heterogeneous data availability. Defense, telecommunications, the internet of things, and pharmaceutical industries are just a few of the sectors where it has applications.

2.1.2 Related Work

In a federated learning environment, various entities (clients) work together to solve a machine learning problem under the supervision of a central server or service provider. To accomplish the learning purpose, tailored updates meant for immediate aggregation are utilized in place of each client's raw data, which is retained locally instead of being exchanged or transferred.

Federated Learning has been significantly used in many applications as follows many pieces of research. The Recurrent Neural Network has been used for Mobile Keyboard Prediction using federated learning [35], [36].

2.1.3 Applications

Applications for mobile devices

FL has been paid much attention to by the researchers since the concept was first put forward by Google to predict users' input from Gboard on Android devices. Further improvement for prediction on a keyboard has been made through [36], [37], and [35]. Emoji prediction is also gaining popularity [38]. In addition, bringing the FL model to smart devices to predict human trajectory [39] or human behavior [40] is also a potential application.

Although the storage and processing capability of mobile devices are increasing quickly today. Due to communication bandwidth restrictions, it is challenging to meet the rising quality demand from mobile users. In order to avoid network congestion, the majority of comprehensive providers prefer to offer a service environment at the cellular network's edge that is close to the consumer rather than integrating cloud computing and cloud storage into the core network. Mobile edge computing (MEC) is the name given to this technology, although it also carries a more serious danger of data leakage. . One possible solution is the combination of FL and MEC, [41] investigate an 'In-Edge AI' framework that combine FL based on deep reinforcement learning with the MEC system and further optimizes resource allocation problems. Further, [42] devoted to utilizing FL on MEC. They created a privacy-aware service placement strategy to deliver high-quality service by caching the desired service on the edge server near the customers.

Mobile devices in this context include both standard smartphones and gadgets used

in IoT environments. One of the key IoT application areas is the smart home. Devices in smart home architecture will upload some associated data to cloud servers to better understand consumers' preferences, which could result in a data breach. Therefore, [43] presents a sufficient secure federated architecture to build joint models. Similarly, a Federated multitask learning framework was developed to learn users' behavior patterns using smart home IoT. Furthermore, [42] proposed a data fusion approach based on FL for robot imitation learning in robot networking. This technique could be used with self-driving automobiles to create guidance models and anticipate different emergencies.

Applications for industrial engineering

It makes sense for industrial engineering to adopt FL's applications, given its success in protecting data privacy. Since there are some restrictions imposed by laws and regulations that prevent direct access to data in these places, we can exploit these scattered datasets to gain limitless benefits, but only when FL is utilized in these places. To the best of our knowledge, following the rise of and maturation of FL, it could have popularization and application prospects in data-sensitive fields for industrial engineering widely. Consider environmental protection as an example. [44], the need for inconvenient interchangeable monitor data, created a novel environmental monitoring frame based on federated region learning (FRL). Thus, for the collaborative model to work better, monitoring data scattered from many sensors might be used. Tasks requiring visual inspection are likewise covered by FL [45]. It could not only assist in resolving the issue of insufficient defective samples for detecting flaws in manufacturing jobs but also provide manufacturers with privacy guarantees. [46] bring FL to acquire diversiform representation from federated tasks for better grounding applications. Apart from image detection and representation, FL is suitable for malicious attack detection in communication systems composed of Unmanned Aerial Vehicles (UAVs) [47]. Since FL's difficulties and the characteristics of UAVs, such as uneven data distribution and unstable communication conditions, are extremely compatible. Due to the increasing popularity of electric vehicles. (2019) developed a system for federating energy demand forecasting for diverse charging stations to avoid energy transmission bottlenecks. Additionally, [48] used FL to transactions owned by many banks in order to efficiently detect credit card fraud, which is also a big contribution to the financial industry. Utilize an industrial grade federated framework based on Latent Dirichlet Allocation for text mining [49]. It has successfully passed the evaluation on real data for sentiment analysis and spam filtering. To summarize, FL enables data owners to broaden the scope of data applications and improve model

performance through iteration among different entities. In the future, FL technology will also support more industries to become more intelligent. The incorporation of FL in AI will build a federal ecosystem without data privacy concerns.

Application in HealthCare

FL has enormous potential in the healthcare industry as a revolutionary way to protect data privacy. Although each medical institute may have a large amount of patient data, it may not be sufficient to build their own prediction models [50]. Combination of FL and disease prediction is one of the good solutions to break down the barriers of analysis throughout different hospitals. Electronic health records (EMR) contain lots of meaningful clinical concepts, [51] gave an attempt to use tensor factorization models for phenotyping analysis to obtain information concealed in health records without sharing patient-level data. It could be regarded as the first attempt at FL application in the medical industry. Pfohl, Dai, and Heller (2019)[52] explored differentially private learning for EMR in a federated setting. And they further demonstrated the performance is comparable with training in a centralized setting. During the training process, there is not any form of data or parameter transmission among hospitals' databases. Besides this, data consolidated from multiple remote clients into a central server is encoded in advance, and the decoder will be abandoned at the end of training.

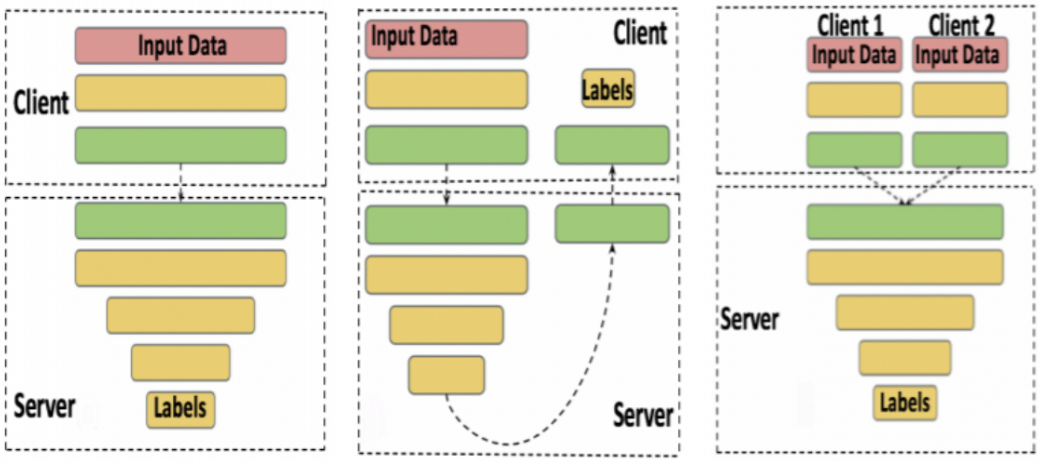
2.2 Split Learning

Without having to directly share raw labeled data, SplitNN is a private and distributed deep learning technique for training deep neural networks across different data sources. One of the biggest problems with machine learning models is data sharing. The emergence of methods like federated learning, differential privacy, and split learning has significantly addressed difficulties with data silos, privacy, and legislation. The method addresses issues like data silos, sharing, etc. The most significant aspect of Split Neural Networks (SplitNN) is that they do not exchange model specifications or raw data with partner institutions. The setups take into account real-world situations such as organizations holding several patient data modalities, centralized and local health organizations working together on various projects, and learning without labeling or sharing raw patient data[53]. The trade-offs between split learning's performance and resource efficiency were compared to those of federated learning and large batch synchronous stochastic gradient descent, among other techniques.

Split Neural Network Working

The issue of training a model across various data entities is addressed by SplitNN. Split learning divides the model into several portions, each of which is trained on a separate client. For instance, the training data may be located on a supercomputer or on a number of clients taking part in the collaborative training. The model's clients cannot, however, "see" the data of one another.

Before delivering the data to train the model, techniques are used to encode it into a new space. The network's training is carried over by moving the weights of each section's final layer to the adjacent (or following) part because the model is divided into numerous sections, each of which is trained on a distinct client. As a result, no raw data is transmitted between customers; instead, the next client is given the weights of the final layer, also known as the cut layer, of each segment.



(a) Simple vanilla Split Learning

(b) Split Learning without Label Sharing

(c) Split Learning for vertically distributed datasets

Figure 2.1: Split Learning (Image referenced from online source[54])

Simple Vanilla Split Learning

This is the most basic configuration of SplitNN, as seen in 2.1a. In this environment, for example, each client (let's say a radiology center) trains a partial model up to a particular layer known as the "cut layer." The outputs from the cut layers are then delivered to a server, which completes the training without access to the customers' raw data (for instance, radiological pictures).

Without exchanging raw data, this ends a round of forwarding propagation. At the server, the gradients are currently being back-propagated from the sliced layer to the last layer. Finally, radiology client centers receive the gradients from the sliced layers.

The radiology client centers have finished the remaining backpropagation. The SplitNN is trained using only its own raw data after repeating this approach.

Split learning without label

The network is wrapped around the end layers of the server's network, as seen in the image above 2.1b, and the output is transmitted back to client entities. The clients create the gradients from the final layers while the server still has the majority of its layers. Later, backpropagation is performed using this without sharing the appropriate labels. For instance, the labels contain extremely private information about patients' status. Distributed deep learning works well with this configuration.

Split learning for vertically partitioned data

This kind of setup enables learning distributed models across numerous institutions that hold various patient data modalities without disclosing or sharing the data. The configuration of SplitNN is suited for multi-modal multi-institutional collaboration, as seen in the image above 2.1c. For example, radiology organizations want to collaborate with pathology test centers and a server for disease diagnosis. Therefore, the radiology centers holding imaging data modalities train a partial model up to the cut layer. The pathology testing facilities that receive patient test results train a partial model up to its own cut layer in a similar manner.

The outputs from these two centres' cut layers are then combined and delivered to the illness diagnosis server, which trains the rest of the model. To train the distributed deep learning model without sharing each other's raw data, these stages are repeated.

2.3 Homomorphic Encryption

2.3.1 Introduction

Three states of data exist: at rest, in transit, and in use. The first two of these are the subjects of most encryption. Data that is at rest or in transit is not actively altered, which explains this. When it is decrypted, its value is the same as when it was encrypted.

In contrast, data that is currently being used lack this characteristic. Nearly all calculations performed on ciphertexts would alter the value of the matching plaintext. It is challenging to make sure that the plaintext changes in the "correct way."

Any connections between the plaintext and the associated ciphertext are destroyed by encryption techniques. An effective encryption method generates ciphertext that is

identical to a random number. Using the right key to decrypt a given ciphertext is the only way to figure out which plaintext belongs to that particular ciphertext.

There must be a connection between plaintexts and ciphertexts in order to execute mathematical operations on encrypted data. The result of adding or multiplying two ciphertexts together must be the same as doing the same thing to the two plaintexts before encrypting them.

At the same time, this relationship must be carried out in a way that conceals it from outsiders. The encryption is compromised if observing mathematical operations on ciphertexts discloses details about the related plaintexts.

It is quite challenging to achieve these interrelated aims of strong encryption and accurate mathematical computations on ciphertexts. The algorithms that have succeeded in achieving this are homomorphic encryption methods.

2.3.2 Applications of Homomorphic Encryption

Homomorphic encryption is significant because it enables calculations to be made on encrypted data. This implies that data processing can be delegated to a third party without having to have faith in the security of the data. It is not possible to extract the original data without the correct decryption key.

The capacity to process encrypted data has the potential to address a number of significant business issues encountered by businesses in all sectors.

Supply Chain Security

The majority of businesses rely on reliable third parties for various aspects of their operations. To perform their duties, these vendors, contractors, and other third parties frequently need access to confidential and sensitive company information.

The dangers of insecure supply chains have been highlighted by recent incidents, which also show how hackers would aim for the weakest link in the chain to accomplish their goals. This means that giving a partner access to sensitive data could expose a company to a costly and harmful data breach.

A business can help protect itself against these supply chain dangers by using homomorphic encryption. A data breach provides no harm to the organization if all data given to reputable third parties for processing is encrypted. This reduces the risk associated with outsourcing key data processing for an organization.

Regulatory Compliance

The legislative environment for data protection has become more complicated in recent years. New laws, including the EU's Data Protection Regulation (GDPR), have given individuals new rights while also imposing new obligations and limitations on enterprises.

The GDPR rule requiring that EU residents' data remain inside the EU or in nations or organizations with equal data security standards is one that many firms are having trouble complying with. One of the primary GDPR justifications for data flows between the EU, and the US was rejected by the Schrems II judgment in 2020, which generated issues for many US companies that do business with EU citizens.

The provisions of laws like the GDPR are expressly stated not to apply to encrypted data. A business might possibly store and process data using homomorphic encryption outside the EU and only decode it on servers in regions that adhere to GDPR regulations.

Private Data Analytics

Many businesses rely on data analytics to generate revenue. By gathering user data, analyzing it, and then selling it to other parties for targeted advertising, companies like Facebook are able to offer "free" services.

The monetization of personal data is debatable, though. Many consumers are upset that businesses are creating detailed profiles of them without giving them any visibility or control over the information being gathered and used.

To address this issue, homomorphic encryption offers a viable solution. A business like Facebook may use homomorphic encryption to execute the necessary data analyses without having access to or the capacity to view the original data. Users having access to the encryption keys opens the door to the possibility of targeted, private advertising.

2.3.3 Types of Homomorphic Encryption

Creating an encryption technique that permits an endless amount of data additions or multiplications is the aim of homomorphic encryption. The final output of the process should be the encrypted version of the ciphertext that would be generated if identical operations were carried out on the corresponding plaintexts.

The difficulty of creating such an encryption algorithm is the problem. As a result, there are many unique "types" of homomorphic encryption that categorize algorithms according to how near they are to the desired outcome.

Partially Homomorphic Encryption

Algorithms for partially homomorphic encryption enable an indefinite number of repetitions of a specific operation. An algorithm might, for instance, be additively homomorphic, which means that adding two ciphertexts yields the same result as encrypting the total of the two plaintexts.

Algorithms for partially homomorphic encryption are reasonably simple to create. In actuality, certain widely used encryption algorithms happen to be somewhat homomorphic.

For example, the RSA algorithm is multiplicatively homomorphic. The reason for this is that encryption in RSA is based on exponentiation:

$$C = m^x \mod n \tag{2.1}$$

where x is the secret key and m is the message.

The rules of exponents say that

$$a^n * b^n = (ab)^n \tag{2.2}$$

n̂. This means that multiplying two ciphertexts encrypted with the same key is equivalent to raising the product of the plaintexts to the power of the secret key. Therefore, RSA is multiplicatively homomorphic.

Somewhat Homomorphic Encryption

Somewhat homomorphic encryption is the next level up from partially homomorphic encryption. Instead of allowing an infinite number of one operation, a relatively homomorphic encryption algorithm enables a finite number of any operation.

Any combination of up to five additions or multiplications may be supported by a moderately homomorphic encryption algorithm, for instance. The sixth operation of either type, though, would produce an unreliable outcome.

An essential step toward completely homomorphic encryption is the use of somewhat homomorphic encryption techniques. Making an algorithm that supports the addition and multiplication of ciphertexts is more challenging than making one that supports limitless addition or multiplication of ciphertexts (even for a limited number of operations).

Fully Homomorphic Encryption

The pinnacle of homomorphic encryption is fully homomorphic encryption. An endless amount of ciphertext additions or multiplications are permitted by a completely homomorphic encryption method without changing the integrity of the product.

There are currently fully homomorphic encryption techniques. In actuality, Craig Gentry created the first fully homomorphic encryption method in 2009. Since then, several algorithms have been created that enhance the first one.

Chapter 3

Methodology

In this report, Neural Network is implemented using Federated Learning, Split Learning, and Homomorphic Encryption. The neural network using Pytorch is also implemented to use as a standard. The code is in the Jupyter notebook and has used several python libraries like numpy, pandas, matplotlib, PySyst, duet, sklearn, and Pytorch.

3.1 PySyft and PyGrid

Using PySyft, a peer-to-peer network of data owners and curators called PyGrid may collaborate to train AI models on distributed data. (Data never leaves the device). Additionally, PyGrid serves as the main server for both model- and data-driven federated learning. PyGrid consists of two parts:

Domain: A single domain node may be connected to a single computer system or a group of computer systems. It is in charge of giving data owners the ability to govern their data and serving as a gatekeeper for data scientists who need to do computations on private data stored on devices. The National Statistics Office (NSO) of several nations, for instance, wants to exchange cancer data. The cancer data can then be organised into a domain for research. As seen in 3.1, several research disciplines can incorporate more domains based on study subjects like Trade data, Climate change, etc.

Network: Application used to track and send commands to many Pygrid domains. The primary library of OpenMined and the privacy ecosystem, PySyft, uses federated learning, differential privacy, and encrypted computing such as secure multiparty computation and homomorphic encryption to isolate private data from model training.

PyGrid can be quickly deployed using the command-line utility HaGrid (HAppy Grid or Highly Available Grid).

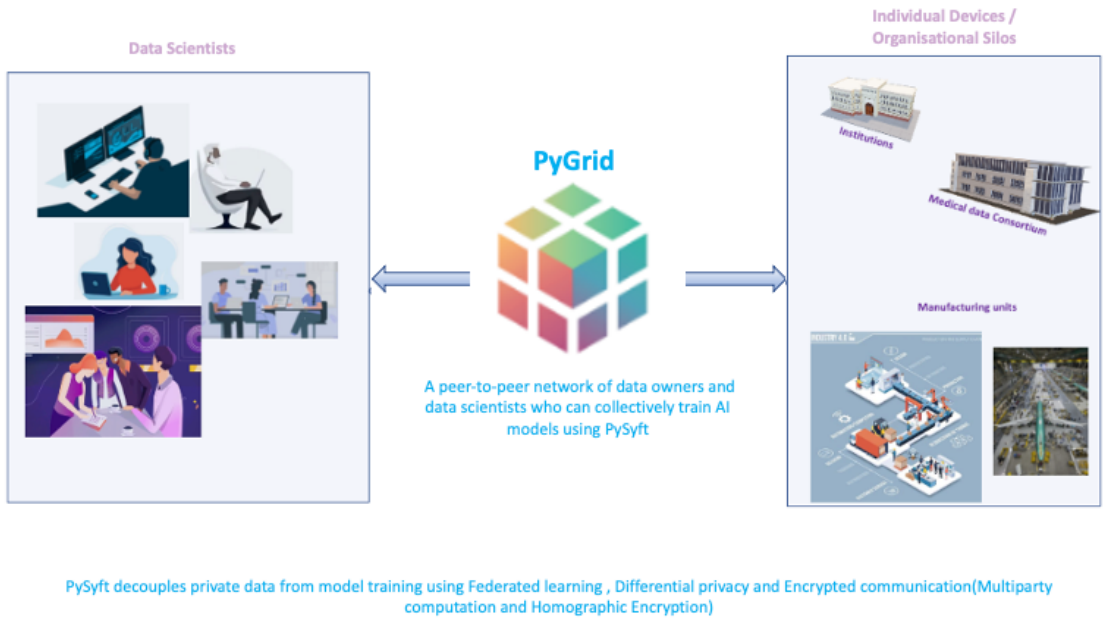


Figure 3.1: Image was referenced from online blog [55]

3.1.1 Visualising a Domain in Remote Data Science Infrastructure.

The domain node is in charge of enabling data custodians to manage their data. It serves as a gatekeeper to control access to the data for data scientists. Through the use of homomorphic encryption and a federated learning framework, remote access to the database may be made private. To prevent privacy leaks, each query on data in the database is responded to using differential privacy approaches. In the above diagram 3.2, data scientists are shown evaluating the domains using PyGrid. Users can create accounts on PyGrid, set a privacy budget, authorise data scientists to compute with the data, and check to see if any requests from data scientists need to be approved. The computational infrastructure of the data curator is then used to store the experimental results.

Observe how, depending on how an organisation decides to build domains, different disciplines (Left domain: Heart attack and covid) could be investigated under a domain or a single type of research (Right domain: Covid) could be conducted under a domain.

The network node is a server that is located outside of the institution of any data owner. If specific conditions are met, many domains may be joined to a single network node. Through PyGrid, the Network nodes will be interconnected and offer services like dataset search and project approval across domains.

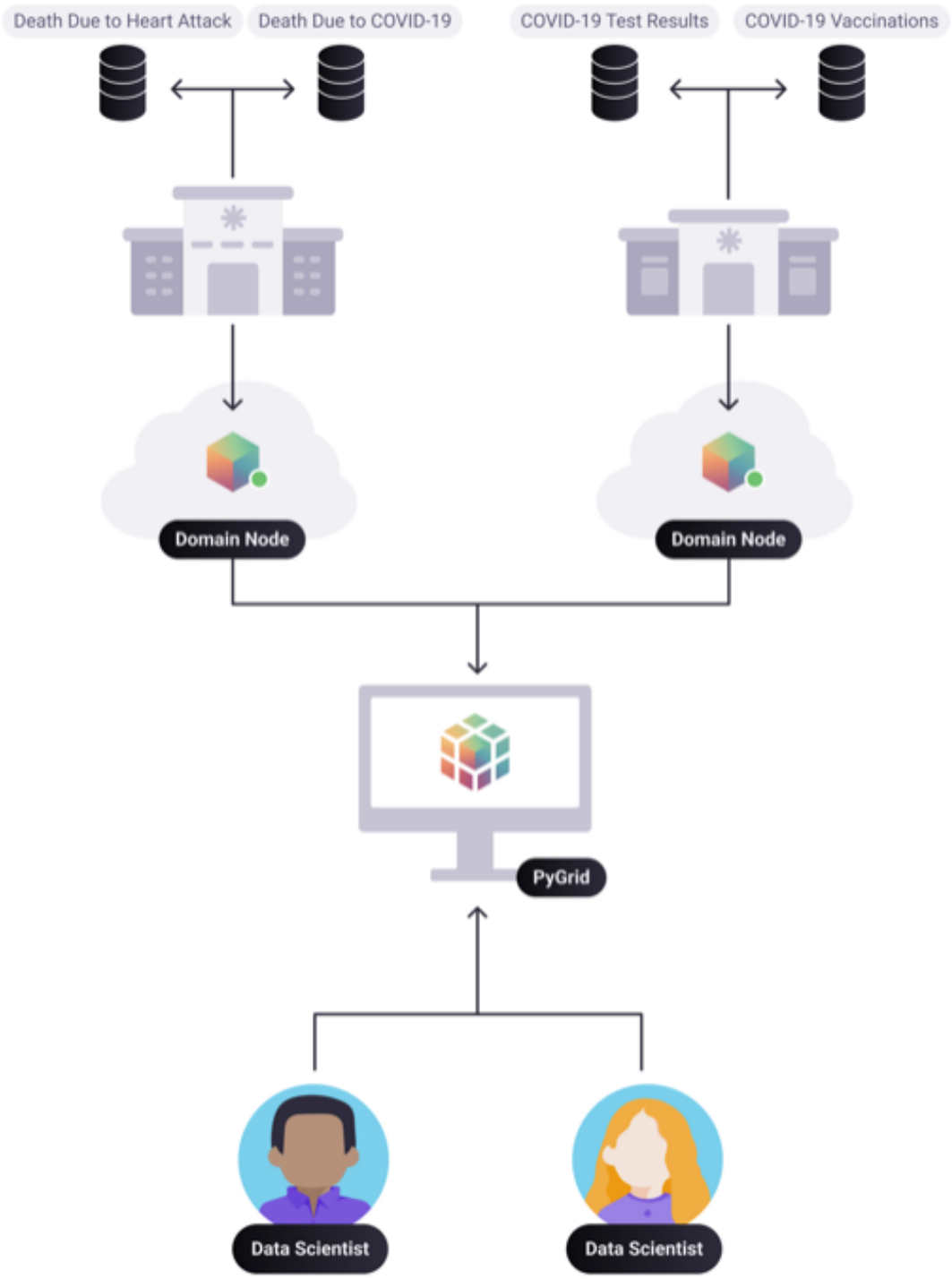


Figure 3.2: Image was referenced from online blog [56]

3.1.2 Model/Data-centric FL in Remote Data Science

Model-centric FL and Data-centric FL are the two types of federated learning (FL) that can be used for remote data science.

Model-Centric FL (Software-centric approach) has been the preeminent paradigm for AI development for decades. In this approach, locally generated and decentralised data from user devices or silos (institutions, banks, hospitals, etc.) is used to train the shared central model, while local models at the data owner's end forward the updated weights to the shared central model. Improvising on building a top-notch central model that trains the decentralised local data is the key goal. The data is still decentralised, but the model is stored in PyGrid.

Data-Centric FL guarantees maintaining constant code and concentrating on producing high-quality data for a specific issue. It is essential to employ a data-centric strategy when there is a paucity of data or when there are challenging data versions that need to be improved in order to produce high-quality data. The difficulties that result in various data versions are explained by the examples below:

- Using X-Rays, a computed tomography (CT) scan produces images. It is a helpful diagnostic technique for finding liver masses, cancer, heart disease, emphysema, fractures, and disorders with the bones and joints. To effectively and consistently identify injuries or diseases, more photos are required. The difficulty comes from consistently identifying an injury or disease from the photos under the patient's current medical circumstances. For instance:

  1. Hospitals deploy new CT scan devices incompatible with existing CT diagnostic tools.
  2. New Software updates for the CT scan device may have errors in detecting previously identified injuries incorrectly.

  To overcome such obstacles, manual analysis of the wounds or illnesses may be necessary.

- Applications that address more dynamic issues may need to be updated hourly or even more frequently, such as filtering unlawful content on social media or fraud detection.

The issues that DCAI (Data-centric AI) must take into account have considerably changed as a result of this continuous execution approach. Real-world customers most likely want a tool to generate fresh datasets every day automatically rather than choosing a diversified training dataset just once. Users might prefer algorithms that

can manage a dynamic taxonomy and use older data for certain classes while using newer data for others rather than establishing classes once.

When less data is available, Data-Centric FL focuses on improving the data (when there is less data available) by collaborating similar data that is made available from multiple sources, which in turn improves the quality of inference.

For instance, 95

3.2 Homomorphic Encryption

Homomorphic encryption allows you to perform mathematical or logical operations on the encrypted data. For example, suppose there are two numbers $m1$ and $m2$, and they have been encrypted using public key encryption scheme with a private key *priv* and a public key *pub*. We get two ciphertexts $c1 = Epub(m1)$ and $c2 = Epub(m2)$. Usually, For anyone without access to the private key needed for decoding, encryption seeks to make all encrypted numbers appear to be random numbers.

However, some associations are kept while using homomorphic encryption. If we use a homomorphic encryption method that enables addition, there will be a function add which anyone can perform on $c1$ and $c2$ such that the result, $addpub(c1, c2)$, will always decrypt to the sum of $m1$ and $m2$:

$$D_{priv}(add_{pub}(Epub(m1), Epub(m2))) = m1 + m2 \tag{3.1}$$

Partial homomorphic encryption techniques that can only perform a minimal number of addition or multiplication operations on the encrypted data have existed for a very long time. Fully homomorphic encryption methods have been developed over the previous ten years, enabling arbitrary calculations on encrypted data.

Pascal Paillier created the Paillier cryptosystem in 1999. It is a partially homomorphic encryption technique that permits two different forms of computation:

- addition of two ciphertexts
- multiplication of ciphertext by a plaintext number

3.2.1 Public key encryption scheme

The public key encryption scheme has three stages:

1. generates a public-private key pair
2. encrypt a number
3. decrypt a number

3.2.2 Helper functions

1. $gcd(x, y)$ outputs the greatest common divisor of x and y.
2. $lcm(x, y)$ outputs the least common multiple of x and y.

3.2.3 Key generation

Key generation works as follows:

1. Pick two large prime numbers p and q, randomly and independently. Confirm that $gcd(pq, (p1)(q1))$ is 1. If not, start again.
2. Compute $n = pq$.
3. Define function $L(x) = \frac{x-1}{n}$.
4. Compute λ as $lcm(p1, q1)$.
5. Pick a random integer g in the set $\mathbb{Z}^*_{n^2}$ (integers between 1 and n^2).
6. Calculate the modular multiplicative inverse $\mu = (L(g^\lambda \mod n^2))^{-1} \mod n$. If does not exist, start again from step 1.
7. The public key is (n, g). Use this for encryption.
8. The private key is λ. Use this for decryption.

3.2.4 Encryption

Encryption can work for any m in the range $0 \leq m < n$:

1. Pick a random number r in the range $0 < r < n$.
2. Compute ciphertext $c = g^m r^n \mod n^2$.

3.2.5 Decryption

Decryption presupposes a ciphertext created by the above encryption process so that c is in the range $0 < c < n2$:

1. Compute the plaintext $m = L(c^\lambda \mod n^2) \cdot \mu \mod n$.

Chapter 4

Results and Discussions

There are two parts of the implementation where different life cycles of Machine Learning are focused. Machine Learning is divided into two phases, namely training and testing. There are four implementations for the comparison as follows:

- **Federated Learning for private training** : In this use case, the data scientist doesn't have access to training data which is stored on the data owner side. However, once the training is completed, the model has been downloaded to the data scientist server and used to predict the test dataset.
- **Split Learning for private training** : For this use case, the data scientist has access to the target variable of training data but doesn't have access to dependent variables or features of the training dataset. However, once the training is completed, the model has been downloaded to the data scientist server and used to predict the test dataset.
- **Homomorphic encryption for private prediction**: In this use case, The training has been done without any encryption, and prediction is processed on encrypted data.
- **Pytorch Model**: For reference, The model using the PyTorch library was implemented without any encryption or privacy.

4.1 Dataset and Data preprocessing

4.1.1 Data Overview

Dataset is used to predict salary using various attributes. Dataset was taken from the open source website. It has five columns as follows:

1. Age

2. Groups
3. Active Lifestyle
4. Healthy Eating
5. Gender
6. Salary

4.1.2 Data preprocessing

The standard procedure for cleaning and processing datasets was followed.

- Data Cleaning: All the rows with null values were removed and checked for any other anomaly.
- Data processing: One hot encoding was done for two categorical values: Gender and groups. The other three features were normalized.

4.2 Neural Network

The neural network consists of three layers as follows :

- Input Layer of 7 neurons
- Hidden Layer of 4 neurons
- Output Layer of 1 neuron.

There were two ReLU activation layers used at the end of the hidden layer and output layer. The Least Absolute Deviation was used as the loss function. The stochastic gradient descent was used for training.

4.3 Training

The hyper-parameters and configuration for the neural network were consistent throughout all paradigms. Initially, Every model was trained for 200 epochs and then was trained for the optimized loss at different epochs.

4.3.1 Federated Learning

The model was initialized at Data Scientist (DS) server and was sent to Data Owner (DO) server. After every epoch, The DO sends the loss to the DS, and then DS calculates the gradient and then sends it back to the DO for updating the model. At the end of the training, The DS only has access to the loss function rather than the data itself. The following figure 4.1 depicts the training of Federated Learning.

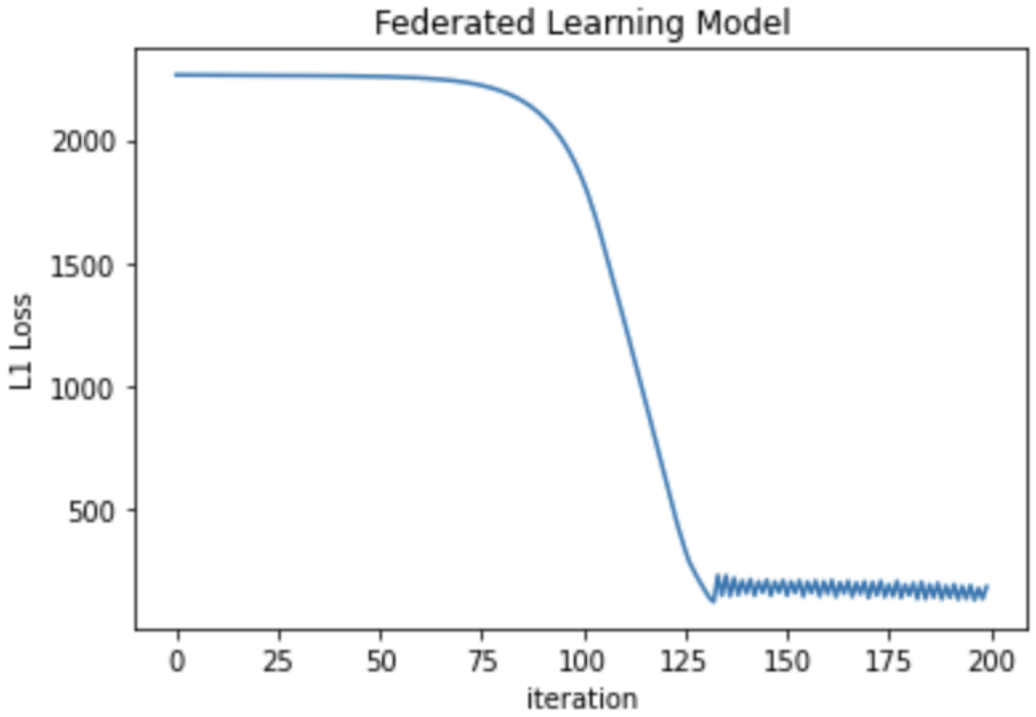


Figure 4.1: Federated Learning model training Epoch vs. Loss

4.3.2 Split Learning

The model was initialized at the Data Scientist DS server. The neural network was split into two parts. The first part contains the input and hidden layer, and the second part contains the output layer. The first part was sent to DO, and the second part was kept with DS. These parts act as an individual neural network. After every epoch, The DO sends the activation to the DS. The DS uses the activation as input for the second neural network for prediction and calculates the loss and gradients. The backpropagation starts from the last layer, which is on the DS server, and propagates to the first layer on the DO server. During the entire process, DS receives gradients from DO, so it doesn't get access to raw data. The following figure 4.2 depicts the training of Split Learning.

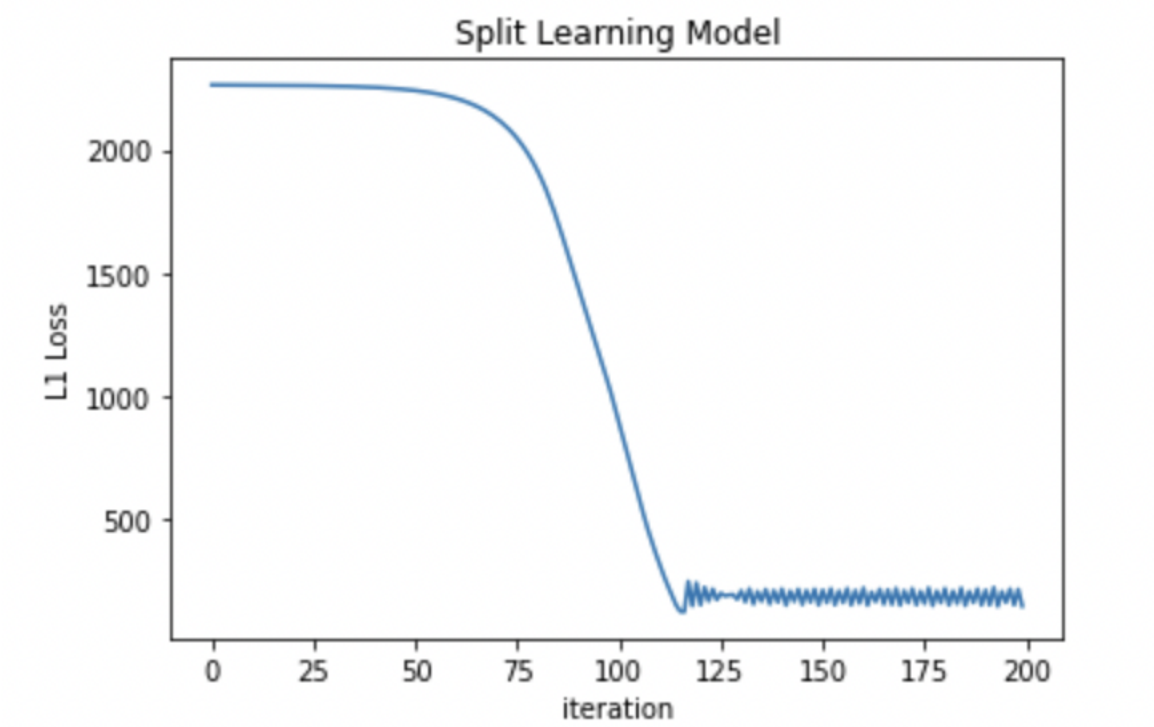


Figure 4.2: Split Learning model training Epoch vs. Loss

4.3.3 Pytorch and Homomorphic encryption

Since the Pytorch model is used just as a reference point, It was trained with the access of training data. Similarly, Homomorphic encryption is used for private prediction, not for private training; It was trained similarly to the PyTorch model. The following figure 4.3 depicts the training of the Pytorch model.

4.4 Results

There are four metrics calculated for the comparison as follows:

- Mean Squared Error
- Root Mean Square Error
- Mean Absolute Error
- R2 score

The results can be seen in the below figure 4.4. According to the results, Federated Learning and Split Learning are performing almost as well as a Pytorch Model. However, Homomorphic encryption is performing very poorly compared to the different

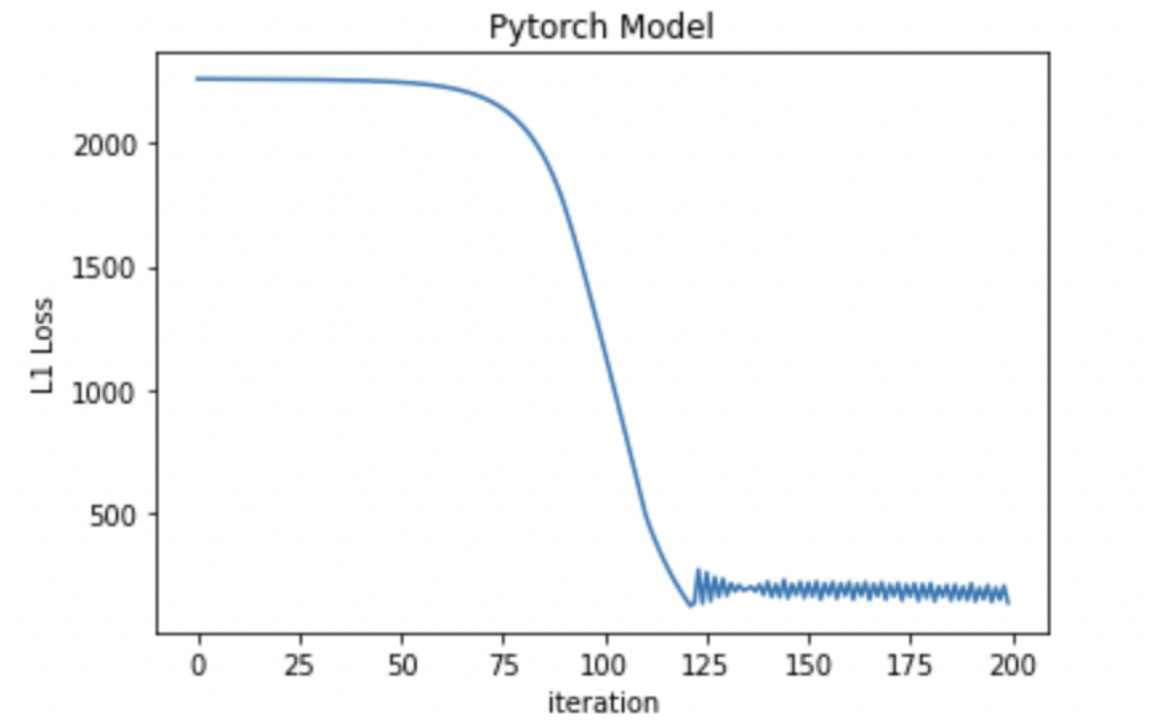


Figure 4.3: Pytorch model training Epoch vs. Loss

models. For this particular use case, Non-polynomial functions were not used since they are not supported in homomorphic encryption.

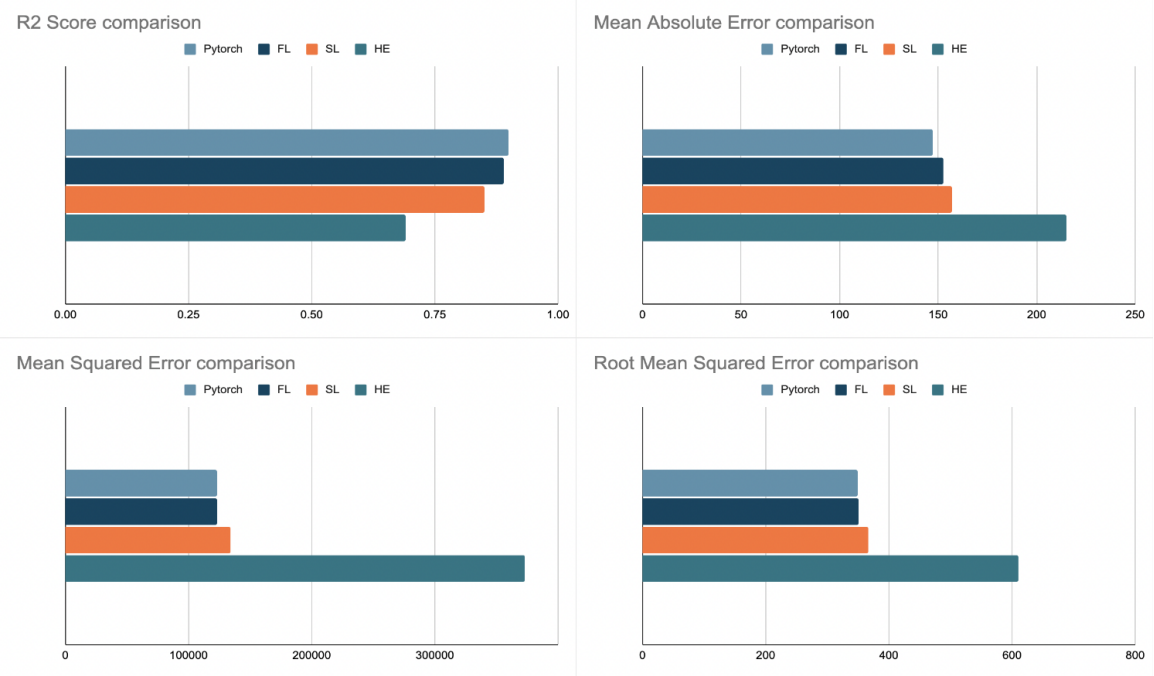


Figure 4.4: comparison of different models using various metrics

Chapter 5

Conclusion and Future Works

5.1 Conclusion

According to the results, Federated Learning and Split Learning's performances are almost identical to that of the simple PyTorch model. However, the computation cost and communication cost increase in both paradigms. In the testing environment, There were two virtual environments created in the same machine, and communications were done with the help of a duet. Since the experiment was done on is the same machine, communication costs were negligible compared to real-world scenarios where the dataset size would very large and distributed among different servers.

For private prediction, Homomorphic encryption was performing poorly compared to the original model. In the original model, the Non-polynomial function, i.e., the ReLU activation function, was used after the hidden and output layer. The non-polynomial functions are not supported in Homomorphic encryption, and hence it affects the accuracy. Homomorphic encryption uses more computation for prediction than its alternatives. If the non-polynomial function were processed with the help of Secure Multi-party communication, then it would significantly increase the communication cost.

5.2 Future Works

In this study, private training was successfully implemented with satisfactory results. There are certain points that will enlighten this study. Some of the future studies that can be done:

- Performing private training to different health datasets using different types of neural networks like Convolutional Neural networks, Recurrent Neural networks, etc.

- Using Spilt Learning for private prediction. The Split Learning in this experiment was implemented in such a way that DS had access to the target variable. For future works, It should be implemented in such a way that DS doesn't have access to the target variable and DO doesn't have access to the entire model.
- Studying and using secure multi-party computation along with Homomorphic encryption to achieve better accuracy along with security.
- Implementing differential privacy for private training and comparing its score with other models.
- In this experiment, The accuracy of different schemes was compared. Future study also needs to compare its performance.

Bibliography

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–44, May 2015. DOI: 10.1038/nature14539.

[2] J. Heaton, "Ian goodfellow, yoshua bengio, and aaron courville: Deep learning: The mit press, 2016, 800 pp, isbn: 0262035618," *Genetic Programming and Evolvable Machines*, vol. 19, Oct. 2017. DOI: 10.1007/s10710-017-9314-z.

[3] I. Rahwan, M. Cebrian, N. Obradovich, *et al.*, "Machine behaviour," *Nature*, vol. 568, pp. 477–486, Apr. 2019. DOI: 10.1038/s41586-019-1138-y.

[4] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, A. Singh and J. Zhu, Eds., ser. Proceedings of Machine Learning Research, vol. 54, PMLR, 20–22 Apr 2017, pp. 1273–1282. [Online]. Available: https://proceedings.mlr.press/v54/mcmahan17a.html.

[5] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtarik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," in *NIPS Workshop on Private Multi-Party Machine Learning*, 2016. [Online]. Available: https://arxiv.org/abs/1610.05492.

[6] J. Xu and F. Wang, "Federated learning for healthcare informatics," *Journal of Healthcare Informatics Research*, vol. 5, pp. 1–19, 2021.

[7] S. Samarakoon, M. Bennis, W. Saad, and M. Debbah, "Distributed federated learning for ultra-reliable low-latency vehicular communications," *IEEE Transactions on Communications*, vol. 68, no. 2, pp. 1146–1159, 2020. DOI: 10.1109/TCOMM.2019.2956472.

[8] M. Hao, H. Li, X. Luo, G. Xu, H. Yang, and S. Liu, "Efficient and privacy-enhanced federated learning for industrial artificial intelligence," *IEEE Transactions on Industrial Informatics*, vol. PP, pp. 1–1, Oct. 2019. DOI: 10.1109/TII.2019.2945367.

[9] P. Rosati, P. Deeney, M. Cummins, L. van der Werff, and T. Lynn, "Social media and stock price reaction to data breach announcements: Evidence from us listed companies," *Research in International Business and Finance*, vol. 47, pp. 458–469, Jan. 2019. DOI: 10.1016/j.ribaf.2018.09.007.

[10] N. Vemprala and G. B. Dietrich, "A social network analysis (sna) study on data breach concerns over social media," in *HICSS*, 2019.

[11] J. Li, N. Li, and B. Ribeiro, "Membership inference attacks and defenses in supervised learning via generalization gap," *ArXiv*, vol. abs/2002.12062, 2020.

[12] J. Jia, A. Salem, M. Backes, Y. Zhang, and N. Z. Gong, "Memguard: Defending against black-box membership inference attacks via adversarial examples," *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019.

[13] D. Bernau, P.-W. Grassal, J. Robl, and F. Kerschbaum, "Assessing differentially private deep learning with membership inference," *ArXiv*, vol. abs/1912.11328, 2019.

[14] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 3–18, 2017.

[15] M. Nasr, R. Shokri, and A. Houmansadr, "Machine learning with membership privacy using adversarial regularization," Aug. 2018. DOI: 10.1145/3243734.3243855.

[16] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," ser. CCS '15, Denver, Colorado, USA: Association for Computing Machinery, 2015, pp. 1322–1333, ISBN: 9781450338325. DOI: 10.1145/2810103.2813677. [Online]. Available: https://doi.org/10.1145/2810103.2813677.

[17] Z. He, T. Zhang, and R. B. Lee, "Model inversion attacks against collaborative inference," in *Proceedings of the 35th Annual Computer Security Applications Conference*, ser. ACSAC '19, San Juan, Puerto Rico, USA: Association for Computing Machinery, 2019, pp. 148–162, ISBN: 9781450376280. DOI: 10.1145/3359789.3359824. [Online]. Available: https://doi.org/10.1145/3359789.3359824.

[18] X. Wu, M. Fredrikson, S. Jha, and J. F. Naughton, "A methodology for formalizing model-inversion attacks," *2016 IEEE 29th Computer Security Foundations Symposium (CSF)*, pp. 355–370, 2016.

[19] K. Ganju, Q. Wang, W. Yang, C. A. Gunter, and N. Borisov, "Property inference attacks on fully connected neural networks using permutation invariant representations," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '18, Toronto, Canada: Association for Computing Machinery, 2018, pp. 619–633, ISBN: 9781450356930. DOI: 10.1145/3243734.3243834. [Online]. Available: https://doi.org/10.1145/3243734.3243834.

[20] C. Zhou, Y. Gao, A. Fu, *et al.*, *Ppa: Preference profiling attack against federated learning*, Feb. 2022.

[21] L. Zhu, Z. Liu, and S. Han, *Deep leakage from gradients*, Jun. 2019.

[22] B. Zhao, K. R. Mopuri, and H. Bilen, "Idlg: Improved deep leakage from gradients," *ArXiv*, vol. abs/2001.02610, 2020.

[23] P. Kairouz, H. B. McMahan, B. Avent, *et al.*, "Advances and open problems in federated learning," 2019. [Online]. Available: https://arxiv.org/abs/1912.04977.

[24] S. Pouyanfar, S. Sadiq, Y. Yan, *et al.*, "A survey on deep learning," *ACM Computing Surveys (CSUR)*, vol. 51, pp. 1–36, 2019.

[25] Q. Li, Z. Wen, Z. Wu, *et al.*, "A survey on federated learning systems: Vision, hype and reality for data privacy and protection," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1, 2021. DOI: 10.1109/TKDE.2021.3124599.

[26] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, Jan. 2019, ISSN: 2157-6904. DOI: 10.1145/3298981. [Online]. Available: https://doi.org/10.1145/3298981.

[27] L. Lyu, H. Yu, and Q. Yang, "Threats to federated learning: A survey," *ArXiv*, vol. abs/2003.02133, 2020.

[28] X. Yin, Y. Zhu, and J. Hu, "A comprehensive survey of privacy-preserving federated learning: A taxonomy, review, and future directions," *ACM Comput. Surv.*, vol. 54, no. 6, Jul. 2021, ISSN: 0360-0300. DOI: 10.1145/3460427. [Online]. Available: https://doi.org/10.1145/3460427.

[29] C. Gentry, "Fully homomorphic encryption using ideal lattices," ser. STOC '09, Bethesda, MD, USA: Association for Computing Machinery, 2009, pp. 169–178, ISBN: 9781605585062. DOI: 10.1145/1536414.1536440. [Online]. Available: https://doi.org/10.1145/1536414.1536440.

[30] D. Boneh, A. Sahai, and B. Waters, "Functional encryption: Definitions and challenges," in *Proceedings of the 8th Conference on Theory of Cryptography*, ser. TCC'11, Providence, RI: Springer-Verlag, 2011, pp. 253–273, ISBN: 9783642195709.

[31] N. Papernot, P. Mcdaniel, A. Sinha, and M. P. Wellman, "Sok: Security and privacy in machine learning," *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 399–414, 2018.

[32] M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar, "Can machine learning be secure?," ser. ASIACCS '06, Taipei, Taiwan: Association for Computing Machinery, 2006, pp. 16–25, ISBN: 1595932720. DOI: 10.1145/1128817.1128824. [Online]. Available: https://doi.org/10.1145/1128817.1128824.

[33] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning," Mar. 2019. DOI: 10.1109/SP.2019.00065.

[34] M. Barreno, B. Nelson, A. Joseph, and J. Tygar, "The security of machine learning," *Machine Learning*, vol. 81, pp. 121–148, Nov. 2010. DOI: 10.1007/s10994-010-5188-5.

[35] A. Hard, K. Rao, R. Mathews, *et al.*, *Federated learning for mobile keyboard prediction*, Nov. 2018.

[36] M. Chen, R. Mathews, T. Ouyang, and F. Beaufays, *Federated learning of out-of-vocabulary words*, 2019. DOI: 10.48550/ARXIV.1903.10635. [Online]. Available: https://arxiv.org/abs/1903.10635.

[37] D. Leroy, A. Coucke, T. Lavril, T. Gisselbrecht, and J. Dureau, "Federated learning for keyword spotting," May 2019, pp. 6341–6345. DOI: 10.1109/ICASSP.2019.8683546.

[38] F. Beaufays, K. Rao, R. Mathews, and S. Ramaswamy, *Federated learning for emoji prediction in a mobile keyboard*, 2019. [Online]. Available: https://arxiv.org/abs/1906.04329.

[39] J. Feng, C. Rong, F. Sun, D. Guo, and Y. Li, "Pmf: A privacy-preserving human mobility prediction framework via federated learning," *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 4, no. 1, Mar. 2020. DOI: 10.1145/3381006. [Online]. Available: https://doi.org/10.1145/3381006.

[40] K. Sozinov, V. Vlassov, and S. Girdzijauskas, "Human activity recognition using federated learning," Dec. 2018, pp. 1103–1111. DOI: 10.1109/BDCloud.2018.00164.

[41] X. Wang, Y. Han, C. Wang, Q. Zhao, X. Chen, and M. Chen, "In-edge ai: Intelligentizing mobile edge computing, caching and communication by federated learning," *IEEE Network*, vol. 33, no. 5, pp. 156–165, 2019. DOI: 10.1109/MNET.2019.1800286.

[42] L. Huang, X. Feng, C. Zhang, L. Qian, and Y. Wu, "Deep reinforcement learning-based joint task offloading and bandwidth allocation for multi-user mobile edge computing," *Digital Communications and Networks*, vol. 5, no. 1, pp. 10–17, 2019, Artificial Intelligence for Future Wireless Communications and Networking, ISSN: 2352-8648. DOI: https://doi.org/10.1016/j.dcan.2018.10.003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352864818301469.

[43] U. Aïvodji, S. Gambs, and A. Martin, "Iotfla : A secured and privacy-preserving smart home architecture implementing federated learning," May 2019, pp. 175–180. DOI: 10.1109/SPW.2019.00041.

[44] B. Hu, Y. Gao, L. Liu, and H. Ma, "Federated region-learning: An edge computing based framework for urban environment sensing," in *2018 IEEE Global Communications Conference (GLOBECOM)*, 2018, pp. 1–7. DOI: 10.1109/GLOCOM.2018.8647649.

[45] X. Han, H. Yu, and H. Gu, "Visual inspection with federated learning," in *Image Analysis and Recognition*, F. Karray, A. Campilho, and A. Yu, Eds., Cham: Springer International Publishing, 2019, ISBN: 978-3-030-27272-2.

[46] F. Liu, X. Wu, S. Ge, W. Fan, and Y. Zou, "Federated learning for vision-and-language grounding problems," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, pp. 11 572–11 579, Apr. 2020. DOI: 10.1609/aaai.v34i07.6824. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/6824.

[47] N. I. Mowla, N. H. Tran, I. Doh, and K. Chae, "Afrl: Adaptive federated reinforcement learning for intelligent jamming defense in fanet," *Journal of Communications and Networks*, vol. 22, no. 3, pp. 244–258, 2020. DOI: 10.1109/JCN.2020.000015.

[48] W. Yang, Y. Zhang, K. Ye, L. Li, and C. Xu, "Ffd: A federated learning based method for credit card fraud detection," in *BigData Congress*, 2019.

[49] Y. Wang, Y. Tong, and D. Shi, "Federated latent dirichlet allocation: A local differential privacy based framework," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, pp. 6283–6290, Apr. 2020. DOI: 10.1609/aaai.v34i04.6096. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/6096.

[50] G. Szegedi, P. Kiss, and T. Horváth, "Evolutionary federated learning on eeg-data," in *ITAT*, 2019.

[51] R. R. Holman, R. L. Coleman, J. C. N. Chan, *et al.*, "Effects of acarbose on cardiovascular and diabetes outcomes in patients with coronary heart disease and impaired glucose tolerance (ace): A randomised, double-blind, placebo-controlled trial," *The Lancet Diabetes Endocrinology*, vol. 5, no. 11, pp. 877–886, 2017, ISSN: 2213-8587. DOI: https://doi.org/10.1016/S2213-8587(17)30309-1. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2213858717303091.

[52] S. R. Pfohl, A. M. Dai, and K. Heller, *Federated and differentially private learning for electronic health records*, 2019. DOI: 10.48550/ARXIV.1911.05861. [Online]. Available: https://arxiv.org/abs/1911.05861.

[53] P. Vepakomma, O. Gupta, T. Swedish, and R. Raskar, *Split learning for health: Distributed deep learning without sharing raw patient data*, 2018. DOI: 10.48550/ARXIV.1812.00564. [Online]. Available: https://arxiv.org/abs/1812.00564.

[54] Anish Agarwal, Analytics India Mag, *Council post: Split learning—distributed deep learning method without sensitive data sharing*, 2022. [Online]. Available: https://149695847.v2.pressablecdn.com/wp-content/uploads/2022/05/dsads.png.

[55] Sukanya R, Openmined org., *Remote data science part 2: Introduction to pysyft and pygrid*, 2022. [Online]. Available: https://blog.openmined.org/content/images/2022/06/image.png.

[56] ——, *Remote data science part 2: Introduction to pysyft and pygrid*, 2022. [Online]. Available: https://blog.openmined.org/content/images/2022/06/image-1.png.