Vertical Federated Learning – An Application for Financial Institutions

Ng Kwok Yuen, Jackey
Master of Science in Data Science
City University of Hong Kong
Kwokyng5-c@my.cityu.edu.hk

Siu Him Kan, Steve
Master of Science in Data Science
City University of Hong Kong
shkan9-c@my.cityu.edu.hk

July 13, 2024

ABSTRACT

Machine learning has become a mainstream technology for organizations across industries to craft innovative and intelligent solutions that improve customer experience and drive business growth. Among other things, such as algorithms, tools, techniques and computing facilities, data is often the most important and valuable raw material to fuel the development of machine learning models. The collection, usage and retention of data are often constrained by privacy, regulatory and legal concerns. In a highly regulated industry, like the banking industry in Hong Kong, banks are often bound by the codes of practice, regulations, and laws from both local and overseas jurisdictions to protect the customers' information. Banks may be subject to financial penalties, disciplinary sanctions, and other forms of reprimands by the regulators or law enforcement agencies if they fail to comply with the codes, regulations, or laws. Therefore, banks are hesitant to collaborate with organizations outside their own industry for projects involving the sharing of data, such as joint machine learning projects. In this report, we explore and apply various Federated Learning techniques with privacy preserving technologies to solve a practical credit assessment problem where both banks and non-banks can be involved to jointly develop credit risk models while maintaining data privacy.

1. Introduction

Over the last decade, machine learning has become a mainstream technology for organizations from diverse industries to make sense of their own data and create highly personalized services for gaining competitive advantages in their respective domains. Increasingly, organizations across industries are also collaborating to create innovative solutions using machine learning technology to drive business growth and better customer service.

In the financial industry, the de facto central bank and regulator, Hong Kong Monetary Authority (HKMA), is spearheading different initiatives and efforts to promote cross financial institutions and cross industry collaboration and innovation, such as MCRA (Multiple Credit Reference Agencies) as a coordinated and collaborative efforts from the local FI (financial institutions) to establish the next generation data infrastructure enabling multiple credit agencies in Hong Kong (previously the TU, TransUnion, is the only credit agency in Hong Kong). CDI (Commercial Data Interchange) is another initiative designed and led by HKMA to enable cross industry data sharing and solution innovation.

Both MCRA and CDI are essentially industry-wide and cross-industry data sharing initiatives which are led by the regulator and trusted by all participants. Trust is built upon the authority of the regulator and the strict rules laid down by the regulator governing the conduct of the participants. Such arrangements, on one hand, can help facilitate the collaboration for the established organizations, however, on the other hand, the complicated rules may become a deterrent factor for smaller organizations, for example, start-ups to join and participate in the initiatives which will undermine the full potential of the initiatives to promote Fintech development in Hong Kong. Furthermore, both the MCRA and CDI platforms, albeit the security protections in place for secured data transmission, are built to facilitate the data sharing and collaboration among participants in plaintext format which will in turn raise the following concerns and difficulties in the practical and real-life applications:

1. The data might have been collected long before the platforms are available. The original Terms and Conditions (T&C) and Privacy Statements to which the customers consented may not cover the scenario in which the data might be transferred and used by a third party for a purpose potentially irrelevant to the consented purpose. This may significantly elevate the cost of using the data as customers' consent might have to be collected again via different channels.

2. Data is arguably the most important asset for an organization; therefore, an organization will be very reluctant to share their data with other organizations other than for specific and mutually beneficial purposes. However, sharing the data in plaintext will deprive the data owning organization of the controls over the data once it has been transmitted to the other parties. This problem is particularly prominent for organizations within the same industry where they are intensely competing in every aspect.

3. MCRA is a platform where banks will contribute customers' transaction data to credit agencies appointed by the regulator for providing the industry-wide credit reference service. This is a closed-loop data sharing platform within the banking industry. CDI, which is supposed to be a cross-industry data sharing platform, currently only allows banks and government departments to be the data consumers. The organizations falling outside these two categories can only play the role of data provider. The HKMA did have other initiative seeking to facilitate the sharing of the bank's data to other organizations, such the Open API framework. However, the complexity of the onboarding and due diligence process for organizations seeking to use the banks' transactional APIs have largely limited the real-life use cases to product enquiry (e.g., interest rate enquiry), product or service application referral (e.g., credit card application and business account opening referral) and loyalty points redemption (e.g., Citibank's pay with points scheme). More use cases of the Open API initiatives can be found in [5]. This unidirectional nature of most data sharing arrangements can undoubtedly help protect the reputation of the regulated industry on one hand, however, it will hinder the wider application and collaboration for the FinTech development in Hong Kong on the other.

4. Generally, banks in Hong Kong have business in other parts of worlds or in the other side of the border. They are not only subject to the Code of Banking Practice (CoBP), Banking Ordinance, Personal Data (Privacy) Ordinance (PDPO) and the supervisory rules from the HKMA (SPMs) as they are currently in force in Hong Kong but are also subject to Personal Information Protection Law (PIPL) and General Data Protection Regulation (GDPR) if they have business in the Mainland China or countries in the European Union, for examples. Failure to comply with any of these codes, regulations and laws will lead the banks to financial penalties, disciplinary actions and other forms of reprimands imposed by the regulators and law enforcement agencies. Therefore, the lack of an established and regulator-approved arrangement for banks to share data with organizations outside the industry has given the banks' very little incentive to collaborate with organizations outside the regulatory regime, especially, the start-ups to jointly develop machine learning projects where data sharing among banks and other organizations are involved.

Data privacy is at the center of the above challenges in the banking industry. If we can address the data privacy issue, we can promote and facilitate the collaboration for joint machine learning project developments within and across the industry. Our work seeks to explore and apply various techniques in the Federated Learning domain as well as privacy preserving technologies to address the data privacy issue for the joint machine learning project development.

1.1 Related Works

As proposed by [6], Federated Learning (FL) is a machine learning setting where the goal is to train a high- quality centralized model while training data remains distributed over a large number of clients.

A typical Conventional Federated Learning (FL) works by having individual parties engaged in local model training with proprietary data, followed by an iterative process of parameter and weight sharing with a centralized server to construct an aggregated model which is typically orchestrated by the Aggregator. This typical setup is described in [7] with a graphical illustration reproduced in Figure 1 below.

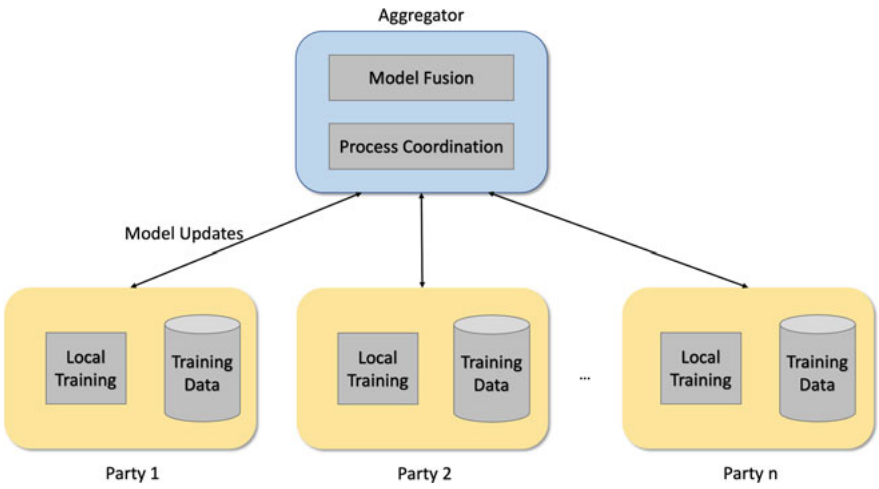

Figure 1 - Federated Learning Overview

To meet the goal of FL, different approaches were proposed to address the challenges involved in the process.

Proposed by McMahan et al. (2017) in [2], Federated Averaging (FedAvg) algorithm, is one of the popular FL methods where the server-side model updates through the aggregation of local stochastic gradient descents (SGD) provided by each participating party. Within each iteration, a weighting mechanism is implemented to calibrate the contribution of each party based on their respective sample sizes. However, the implementation of the standard FedAvg algorithm remains a difficult task primarily due to the complexities and data leakage concerns involved in the exchange of model parameters and metadata.

Vepakomma proposed the Split Learning approach (SplitNN) [15] which is flexible enough to cater for different configurations of FL including both horizontal FL and vertical FL. The author also compared the performance of SplitNN and FedAvg in CIFAR 10 and CIFAR 100 datasets using VGG and Resnet-50 architectures. The SplitNN outperformed FedAvg in terms of model accuracy and computation resources consumed. The author considered the approach safe in terms of data privacy protection because only the intermediate results of the cut-layers would be passed to the Aggregator for the global model training while the raw data would never leave the participating parties. However, it was shown in [16] that in the two-party vanilla FL configuration, the non-label party is able to infer the label with confidence by exploiting the norm-based scoring

function. One of the approaches adopted in our work will leverage the idea of SplitNN in a vertical FL configuration in which at least two participating parties and an Aggregator will be involved.

In an industry setting, Xu et al. (2021) in [11] proposed the FedV approach which offered superior data privacy preservation capabilities by leveraging functional encryption schemes. It supports a wide variety of machine learning models including the artificial neural network (ANN). In our work, we are focusing on the neural work model mainly due to its proven performance for many applications and its flexibility in terms of model design. In this context, the FedV approach resembles the SplitNN approach in many aspects other than the use of functional encryption schemes for model fusion as well as the gradient updates. To evaluate and demonstrate the performance of FedV in our work, we adopted LWE (Learning with Error)-based functional encryption schemes providing the inner product functionality under the Multi-Input Functional Encryption (MIFE) and Single-Input Functional Encryption (SIFE) configurations. The LWE encryption scheme was proposed by Regev in [12] whose hardness is based on the difficulty of the lattice problem. The lattice problem is known to be hard even in the face of quantum computers although Yilei [13] claimed that he discovered a way to solve the lattice problem in polynomial time with the help of a quantum computer. The claim was later dismissed by Yilei himself after identifying a bug in the algorithm. Therefore, the LWE remains the de facto quantum safe encryption scheme.

Several functional encryption schemes based on LWE were later proposed in [8,9 and 10] to securely calculate the inner product of two given vectors. Our implementation of the SIFE and MIFE schemes are based on the works of [9] and [10]. It is also worth noting that at the time of writing, there is no python implementation of the LWE-based MIFE scheme although there is a helpful python library [17] implementing the DDH-based MIFE scheme and LWE-based SIFE scheme. As part of our work, we developed a python library for the LWE-based MIFE with reference to [10].

Federated Reconstruction Framework (FedRecon) is another approach which can address the challenges of FL. Proposed by Singhal et al. (2021) in [3], FedRecon is a model-agnostic framework designed to allow meta learning and collaborative training in large-scale cross-device settings Singhal et al. (2021). In this framework, the server model is divided into local and global layers, which are trained by each client independently using two distinct data subsets: the Query and Support set. Specifically, the Support set updates local parameters through rounds of gradient descent while keeping the global parameters fixed. Conversely, the Query set is used to update the global parameters with the local layers held constant Singhal et al. (2021). Unlike FedAvg, FedRecon returns gradients to the server for aggregation rather than all model parameters. By design, this framework addresses the communication constraints and privacy concerns inherent in the standard FedAvg and other algorithms in our works, which aligns with our core objective of ensuring data privacy preservation. This framework also reduces the need to store model states and allows the reconstruction of local parameters with local data at training and testing time. We borrow this Query and Support split technique in one of our algorithms to form our own version

of FedRecon with each Financial Institutions collaboratively training a global model without sharing all parameters. We will elaborate on the assumptions and model algorithms in the later sections.

Some existing works aim to address the limitations in partial participation support and communication efficiency. The Federated Communication compressed AMSGrad with Max Stabilization (FedCAMs) framework proposed by Yujia Wang. (2023) in [1], introduces a mechanism to compress all gradients along with a cumulative compression error between the communication of the clients and server. A key feature of FedCAMs is its ability to maintain stale cumulative compression errors for clients not selected in the current global training round. This design choice enhances practicality and communication efficiency. In our work, however, full participation is assumed to reduce the complexity of our modelling problem as well as to be relevant to our problem domain in which the banks and other organizations will work in a commercial arrangement and therefore, they will be disciplined enough to participate in the training as expected.

1.2 Contributions

To demonstrate the gravity of the federated learning problem in a practical sense and in a practitioner-oriented approach we adopted to solve the problem, we shall make use of a credit assessment use case and a publicly available dataset in our work which can hopefully simulate a widespread problem of representative value in the Hong Kong banking industry.

Our main contributions in this work include:

1. Develop a repeatable methodology or decision framework to address the privacy issue arising from the joint machine learning projects in the banking industry.

2. Develop a reusable library or framework for multi-input and single-input functional encryption to achieve the goal of data privacy preservation.

2 The Big Problem

Concisely, banks are just intermediaries between depositors and borrowers and convert short-term liabilities (such as the deposit in your saving accounts) into long-term assets (such as the mortgage loan you may take to finance your dream home). Banks make profit from the difference in the interest which the banks pay to the depositors for their money in the saving accounts and the interest which the banks are paid by the borrowers for their loans taken out from the banks. The most important problem to this business model is how the banks can repay the depositors given the fact that not all borrowers will repay their loan on time or in full. Banks develop and maintain their own credit risk models for approving and disapproving loan applications to ensure that the banks will have sufficient funds to meet the expected demand of the depositors. Banks usually take into consideration the regulatory requirement on capital reserve, the banks' credit risk appetites, loss recovery ratio, probability of loss and exposure at loss to develop their credit risk models. An accurate credit risk model can, therefore, help a bank to maximize their profit while remaining comfortably resilient to credit risk exposure. To build an accurate model, banks do not only need the customer data from their own databases but also need the data from other parties, such as credit bureaus, other banks, companies outside the industry (e.g. logistic companies in the case of trade financing) and other governmental and non-governmental parties providing such information as inflight disputes, legal proceedings and past conviction records as part of the risk assessment and due diligence. In the recent Fintech development, banks also started considering using alternative data in their credit decisions, such as sales transaction records and other behavioral traits of the borrowers, which are typically held by organizations outside the banking industry.

In this work, we shall work on a simplified credit risk modelling problem in a banking context while preserving the gravity of the essential issues and challenges faced by the banks in the industry. In this regard, we select a publicly available dataset in relation to credit risk modelling and deliberately divide the data into multiple partitions with non-overlapping features to essentially create a scenario of Vertical Federated Learning (VFL).

In this VFL scenario, we assume that a bank together with two other parties, which can be organizations inside or outside the banking industry, are working on a common machine learning project to develop a credit risk model which predicts the probability of default of consumer loans. We further assume that the data held by the three parties are distinct in the sense that no single party will have access to the full set of features to train the credit model. We reckon that by following the approach we developed in this work, banks and other organizations involved in the project will be able to jointly develop effective machine learning models while addressing the underlying security and privacy concerns. Although the approach outlined in this report is constrained to a typical tri-partite problem, the underlying principles will remain unchanged, and the outlined approach will be flexible enough to address scenarios where more parties are involved in the common project.

2.1 The Dataset

The dataset we selected for this work is from Kaggle [4] which contains 32,581 records with 11 feature variables and 1 target variable in total. The dataset can be used to model the consumer credit risk and predict the Probability of Default (PD) for consumer loans. The following table describes the feature and target variables in the dataset.

Table 1 - Description of the Credit Dataset

Variable	Description	Data Type
Person Age	The age of the borrower	Numeric
Person Income	The income of the borrower	Numeric
Person Employment Length	The length of the borrower's employment history	Numeric
Loan Amount	The amount of the loan	Numeric
Loan Interest Rate	The interest rate of the loan	Numeric
Loan Income Ratio	Loan as a percentage the borrower's income	Numeric
Credit History Length	The length of the credit history for the borrower	Numeric
Home Ownership	Whether the borrower has ownership over his home	Categorical
Loan Purpose	The purpose of loan	Categorical
Loan Grade	The grade of the loan determined at the time when the loan was granted	Categorical
Default On File	Whether the borrower has any prior record of loan default	Categorical
Loan Status	Whether the loan is defaulted or not	Categorical

For the majority of our algorithms, we divide the dataset into three distinct partitions with no overlapping features as summarized in the following table.

Table 2 - Data Partitioning

Partition	Variables in the partition	No. of Variable After One-Hot Encoding
1	1. Person Age	7

	2. Person Income 3. Loan Income Ratio 4. Person Employment Length 5. Credit History Length 6. Default On File	
2	1. Loan Amount 2. Loan Interest Rate 3. Loan Purpose	8
3	1. Home Ownership 2. Loan Grade	11

Each partition of data will be held by a party in the common machine learning project. The target variable can be held by one of these parties.

In addition to the data partitioning, we also apply the following data pre-processing techniques to improve the model performance:

1. Normalize all numeric data variables using Standard Scaler.

2. Perform one-hot encoding on all categorical data variables.

3. Remove all records with null or missing values in any of the data variables.

For algorithm 3.3 and 3.4, a different set of data schema were used, and the following data pre-processing techniques were applied to allow better convergence:

1. Normalize all numeric data variables using Robust Scaler.

2. Performed Label Encoding on all categorical data variables.

3. Feature Engineering including logarithmic transformations and feature interactions.

The resulting dataset contains 28,638 data records in which 6,203 records are defaulted loans and 22,435 records are good loans.

2.2 The Problem Setup

In this section, we will highlight the assumptions underpinning the practical problem we are trying to solve with different approaches and techniques. The approaches and techniques we are exploring in this work will solve the problem in different ways and address the underlying concerns

to various extents. We shall analyze the pros and cons of different approaches and techniques and derive a set of recommendations that can be practically followed by the participants of the joint machine learning projects to achieve the desired outcomes.

The general assumptions for the problem are as follows:

1. There are three parties participating in the joint development of the credit risk model.

2. There is not a single party who can access the full set of features for model training.

3. None of the three parties are willing to reveal the raw data to facilitate the model training.

4. The final product of the joint machine learning project will be credit risk models capable of predicting the probability of default of a prospective loan application.

5. The performance of the models will be assessed with the Area Under the ROC Curve (AUC) and the F1-Score metrics.

The general objectives we are seeking to achieve are as follows:

1. Produce a credit model with acceptable performance.

2. Address the privacy concerns from all parties regarding their raw data.

Since Artificial Neural Network (ANN) is arguably the most flexible way to develop machine learning models with good performance, we have focused our works on building ANN models that have good prediction capabilities while addressing privacy concerns. Other methods and models, such as Support Vector Machine (SVM) and Tree Models (e.g., XGBoost) can be considered and further investigated in future work.

2.3 The Approaches

The following subsections give an overview on the different approaches we adopted to tackle the problem. The details, including the algorithm and model designs, will be discussed later in the report.

2.3.1 Local Learning

In this approach, we assume that a single party is entrusted by the other two parties to train the model locally. The trained model can then be distributed to the other two parties for the application. This assumption is violating the second and third general assumptions of the problem, however, we opine that including this approach in our analysis will help make our proposed methodology more complete.

2.3.2 Split Learning

In the split learning approach, we assume that there is a fourth party (we call it aggregator) in the project who will be responsible for (1) aggregating the partial models from all other parties in the project, (2) calculating gradients for model parameter update and (3) disseminating the gradients to the parties involved in the training.

To be able to perform the task (2) above, we also assume that the aggregator will have access to the target variables (i.e., the labels) so that it will be able to calculate the loss and gradients accordingly.

To simplify the setup and mimic the machine learning project executed in a controlled manner, we further assume that all parties will be participating in all rounds of model training.

2.3.3 FedAvg with Linear Projection

In this approach, (1) full participation is required at each global training round, and (2) we assume that each client has access to the loan statuses metadata. We also (3) require each participant to perform local training, (4) provide the updated model weights to the aggregator and (5) store their own linear projection layers for the next round of training. The aggregator in this approach simply performs a weighted aggregation with all the provided weights and returns the updated parameters back to each client at the beginning of the next global training around.

2.3.4 FedRecon with Linear Projection

Our implementation of FedRecon slightly deviates from the original idea proposed by Singhal et al. (2021). In this approach, we (1) treat each participant as heterogenous clients contributing to the training of a global model without sharing all model parameters with each other. Similar to 2.3.3, we also assume each client has full access to loan statuses information and we require (2) all clients to (3) fully participate and (4) store their linear projection weights at each global training round.

In large-scale scenarios, both (3) and (4) must be relaxed and other communication efficient techniques mentioned in Section 1.1 should be considered to avoid staleness clients and the degrade in model performance.

2.3.5 Vertical Federated Learning with FedV

In this approach, on top of the assumptions made under 2.3.2, we need to add the fifth party (we call it the Trusted Party, TP) in the project who will be responsible for (1) generating the public and secret keys, (2) distributing the secret keys to the right parties, and (3) encrypting the weight

vector from the aggregator. Please note that TP will not be involved in the model training. Aside from the keys for encryption / decryption, the only data which the TP can access is the weight vector from aggregator which is just the weights used to aggregate the model outputs from the three parties in the project.

The remaining part of this report will elaborate in detail the algorithms, model designs, model performance and privacy protection for each of the above approaches. Finally, we will summarize our methodology to address the Vertical Federated Learning problem in the banking industry in general.

3 Algorithms for Different Approaches

This section provides an analysis and discussion of each algorithm supporting our approaches to solve the big problem as outlined in the previous section.

3.1 Local Learning

The local training approach is nothing but making all the raw data available to locally train a model. The aggregator will have all the raw data necessary to train a local model from other parties, therefore, the second and third general assumptions of the problem statement are violated in this approach. However, a locally trained model with all relevant data should arguably have the best performance as compared to other approaches which constrain the training process in diverse ways to preserve data privacy. Therefore, the model trained under this approach can serve as the topline against which the performance of models trained under other approaches can be meaningfully compared. The participating parties in this approach will execute the Extract and Sort procedure to extract and sort the relevant data records from their own database based on a list of identifiers ($id_n \in \mathbb{I}$). The aggregator will then execute the *Aggregate & Train* procedure to aggregate the raw data and train a local model which is then used for predicting the probability of default for perspective loan applications.

The algorithm 3.1 is for the participating parties to extract the relevant data from their own database based on a list of identifiers. The extracted data will then be passed to the Aggregator for aggregation and model training. It is practically assumed that the identifiers should have been desensitized. For example, the HKID can be used as the identifier to correlate the relevant data across all participating parties. The HKID can be desensitized by using a one-way function, for instance, a secure hash function. The desensitized identifiers from all participating parties can then be passed to a selected party or the Aggregator to determine a list of identifiers which are common to all parties. The list of identifiers is denoted as $\mathrm{id_n} \in \mathbb{I}$. The full set of data records for a party is denoted as $\mathcal{D}_\mathrm{i}$. The r_n denotes a single data record under $\mathcal{D}_\mathrm{i}$. A data record r_n should contain, among other features, an identifier which can be used to uniquely identify the record.

ALGORITHM 3.1 LOCAL LEARNING

1	***procedure Aggregate & Train*** ($\mathcal{D}_K \in \mathcal{D}$)	***procedure Extract & Sort*** ($id_n \in \mathbb{I}$)
2	*let* $\mathcal{D}_{agg} \leftarrow \{\}$	$r_n \in \mathcal{D}_i, i \in K, n \in N$
3	***for*** *each* $i \in K$ ***do***	$\mathcal{D}_s \leftarrow \{\}$
4	$\mathcal{D}_{agg} \leftarrow \mathcal{D}_{agg} \cup \mathcal{D}_i$	***for*** *each* r_n *in* $\mathcal{D}_i$ ***do***
5	***end for***	***if*** id_n *is in* r_n ***then***
6	*train the local model with* $\mathcal{D}_{agg}$	$\mathcal{D}_s \leftarrow \mathcal{D}_s \cup r_n$
7	***end procedure***	***end if***
8		***end for***
9		$\mathcal{D}_s \leftarrow sort(\mathcal{D}_s) by\ the\ order\ of\ id_n$
10		***end procedure***

The following diagram (Figure 2) presents an overview of the model design adopted in the local learning approach. On the left of the diagram, there are three parties sharing the raw data to the aggregator on the right. Since no party will have the full set of features of the data, their data will be combined and fed to the input layer of the Aggregator's ANN model. The feature size of the input layers is equal to the sum of the feature sizes from the participating parties. The light blue rectangular boxes represent layers in the ANN models while the numbers in the boxes indicate the output feature size of the respective layers. The Aggregator will also be responsible for calculating the loss between the predictions and the true labels as well as the gradients for the model weight updates. On the right side of the diagram, the loss and gradient calculations are presented with the dotted boxes and green arrows. The green arrows at the bottom of the diagram gives an indication on the direction as well as the where the gradients will be propagated for the model weight updates. In the diagram below, the gradients will be propagated backward to update the model weights from the last output layers to the first input layers of the model on the Aggregator's side. Unless otherwise stated, the same legend will be used for the model overview diagrams presented for other approaches.

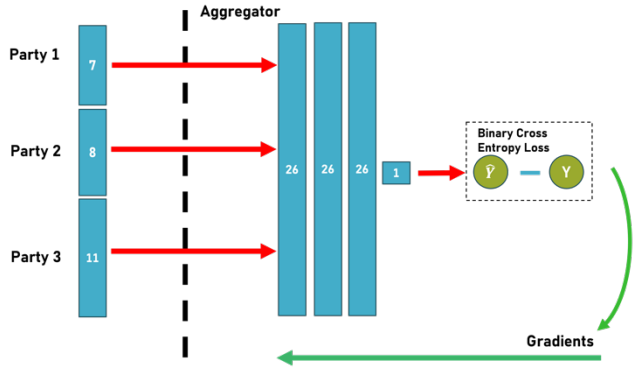

Figure 2 - Overview of Local Learning

3.2. Split Learning

The split learning approach is largely based on the ideas proposed in [15] in which the participating parties will not share the raw data to each other or even to the Aggregator. Instead, the parties will use their raw data to train their respective local models. The output of these local models (i.e., the intermediate results) will be fed to the Aggregator which will continue training a global model with the intermediate results. The Aggregator will also calculate the loss and gradients which will be sent back to the participating parties for updating the local models. In the whole training process, the raw data never leave the participating parties except for the label information which may have to be shared to an independent party, the Aggregator for the purpose of calculating the loss and gradients.

Label information may contain very private and sensitive information. In the context of our problem, the loan default status is considered a confidential information to the customers who borrow loans from a bank. If the bank wants to enhance its credit risk model with data from other organizations and it owns the loan status information, the bank must not reveal the information to the participating organizations unless prior consents have been obtained from the customers, otherwise the bank may be subject to the reprimands from regulators. As a matter of fact, it is usually not in the bank or other organizations' best interest to reveal such valuable information to the others for the sole purpose of training a better performing model.

In this approach, we assume that the Aggregator will be given the label information. However, this approach should still be considered secure and safe for the following reasons:

1) The label party may play the role of Aggregator and therefore, eliminating this additional party.

2) The Aggregator is only given the label information, which is only 1 or 0 to indicate a defaulted loan and a good loan respectively. It has no access at all to the raw training data including but not limited to the identifiers which can be used to relate the original training data to the given labels. Therefore, the Aggregator will not be able to recover any meaningful information from the data label alone.

In a limited two-party setting, however, it has been shown in [16] that label information can be leaked to the non-label party even though only the intermediate computation results and the gradient updates are exchanged. In the two-party setup, it was assumed that the non-label party had access to the full training dataset and the gradients calculated by the servers. It was showed that the non-label party was able to infer the label information with confidence through the norm-based scoring functions. However, our approach is not vulnerable to the proposed attacks in [16] since the data in our approach are vertically partitioned among all participating parties. There is no single party having all training data.

The Algorithm 3.2 highlight the key procedures used in this Split Learning Approach. The Aggregator execute the **Aggregate procedure** on the intermediate results of the cut layers passed by the participating parties. The gradients calculated from the loss will be split according to dimensions of the cut layers and the **Update Participants procedure** will be executed to update the local models of the participating parties.

ALGORITHM 3.2 SPLIT LEARNING

```
1   procedure Aggregate (D_c)
2       for each round from 1 to T do
3           ŷ ← predict with Global Model (D_c)
4           loss ← binary cross entropy loss (ŷ, y)
5           ∇w_c, ∇w_g = grad(Loss, Global Model)
6           Update Global Model (η∇w_g)
7           ∇w_c1, ∇w_c2 ... ∇w_ck = splitd (∇w_c)

    procedure Update Participants (∇w_cK)
        for each i ∈ K do
            Update Local Model_i (η_i ∇w_ci)
        end for
    end procedure
```

8 *Update Participants* (∇w_{c1}, ∇w_{c2} ... ∇w_{ck})
9 ***end for***
10 ***end procedure***

Figure 2 is the reproduction of the diagrammatic illustration of the setup in [16] which can be contrasted with Figure 3 which describes the overall setup of our Split Learning approach.

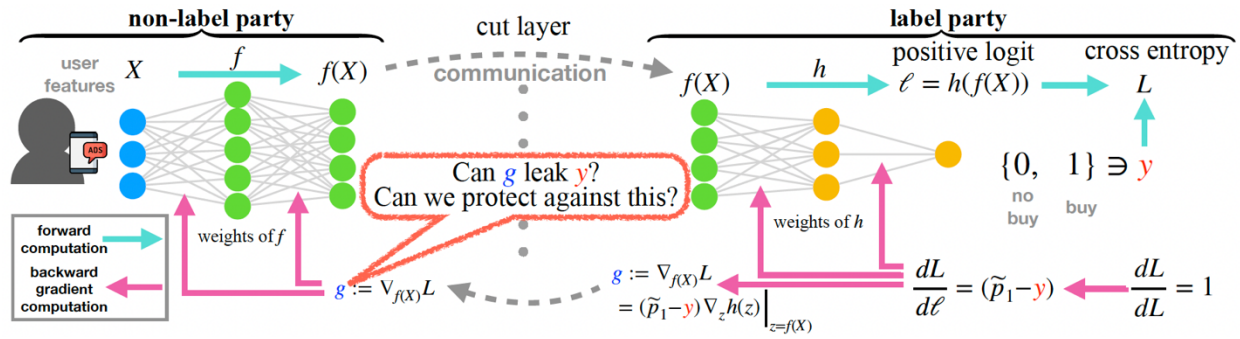

Figure 3 - Setup in (Li 2022)

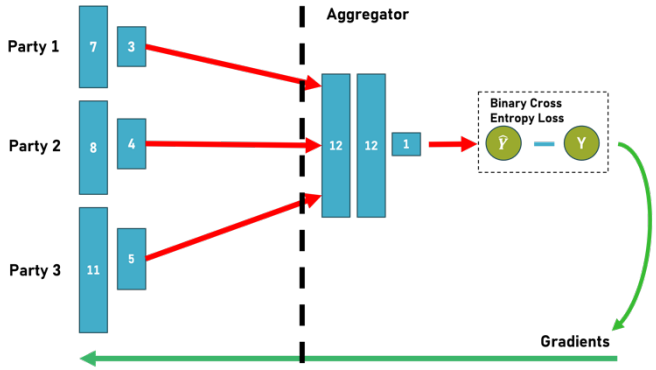

Figure 4 - Split Learning Overview

3.3 Federated Averaging (FedAvg) with Linear Projection

We also implemented a modified version of FedAvg as our baseline model. This adaptation addresses the challenges posed by Vertical Federated Learning, where participating financial institutions may have varying feature sets. Unlike other approaches, the Aggregator is not provided with the label information and its sole mission is to perform weighted aggregation of model parameters. Similarly, parties are participating in the joint model training process, but they will have their own augmented version of the credit risk model.

To allow better convergence using this framework, we reduced the total number of parties and modified our original schema specifications as summarized in Table 3. Label encoding is used here over One Hot Encoding to reduce sparsity and feature interaction effects are also fine-grained

with this framework. Feature engineering techniques include but are not limited to logarithmic transformations and feature multiplications.

Table 3 – Data Partitioning 2

Partition	Variables in the partition	No. of Variable After Feature Engineering
1	1. Person Age 2. Person Income 4. Person Employment Length 5. Credit History Length 6. Default On File 7. Home Ownership	10
2	1. Loan Amount 2. Loan Interest Rate 3. Loan Purpose 4. Loan Income Ratio 5. Loan Grade	5

At each training iteration, the bank augments the sever model by concatenating their own linear projection layer. This combined model is then used to perform local gradient decent operations. Upon completion of local training, updated parameters are returned to the server for weighted aggregation, excluding their own projection layer. By design, projection layers can add a simple layer of data protection whereas the original FedAvg requires the sharing of all model parameters. Details of this algorithm is illustrated in Algorithm 3.3 and Figure 5.

Model architecture and training design wise, a random set of banks is selected at each global round. The server model consists of 4 linear layers with ReLU as the activation function to introduce non-linearity. The number of network neurons is 128 / 64 / 32 / 1 with a sigmoid function in the final layer to map the output into a probability distribution.

In our experiments, we initially selected roughly 1-3 rounds of local epochs and 30 grounds of global epochs with an identical learning rate of 0.001 and batch size of 32. Algorithm 3.3 illustrates the core components and procedures used in this Federated Averaging (FedAvg) with Linear Projection approach. Details of our experiments will be elaborated further in Section 4.3.

In general, this approach attempts to:

1) Evaluate the feasibility of FedAvg at extreme cases of data governance and compliance

2) Experiment the idea of fine-tuning with FL with the introduction of linear projection and local trainings

ALGORITHM 3.3 FEDERATED AVERAGING WITH LINEAR PROJECTION LAYER

Algorithm 3.3 is the modified version of the standard Federated Averaging framework with the introduction of Linear Projection Layer. The K clients are indexed by k. E is the number of local epochs and T is the number of global epochs. η is the learning rate and l is the weights of linear projection layers. Data volume is indexed by n.

procedure S

  for each round from t = 1,2,.....T **do**

    $\widehat{S_t} \leftarrow$ (Random Set of K clients)

    for each client in $\widehat{S_t}$ in parallel **do**

      $w_{t,k},\ n_{t,k} = ClientUpdate(k, w_{t,k})$

    end for

    $n_t = \sum n_{t,k}$

    $w_{t+1} \leftarrow \sum \frac{n_{t,k}}{n,t}\ w_{t,k}$

  end for

end procedure

procedure ClientUpdate (k, w)

  $B \leftarrow$ (split D into batches of size B)

  for each local epoch $e = 1, 2, \ldots, E$ **do**

    for each batch $b \in B$ **do**

      if $l_{t-1,k}$ exists **then**

        $w,\ l_{t,k} \leftarrow w - \eta \nabla \ell(w,\ l_{t-1,k}, b, e)$

      else

        Initialize $l_{t,k}$

        $w,\ l_{t,k} \leftarrow w - \eta \nabla \ell(w,\ l_{t,k}, b, e)$

      end if

    end for

  end for

  Store $l_{t,k}$ for next global round

  $n_k \leftarrow |D_k|$

return w, n_k to server

end procedure

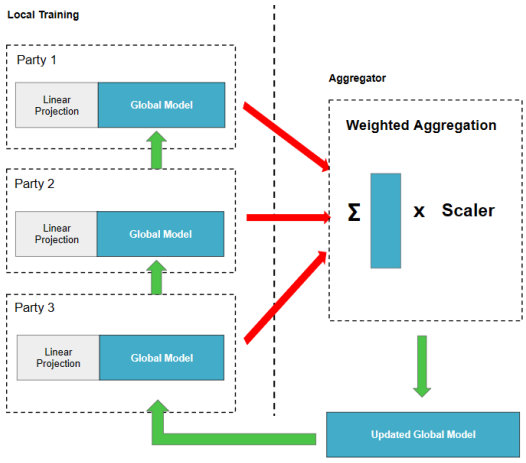

Figure 5 - FedAvg with Linear Projection Overview

3.4 Federated Reconstruction (FedRecon) with Linear Projection

Our implementation of FedRecon aligns with the methodology outlined in Section 3.3. We also used the modified schemas illustrated in Table 3 and required each client to concatenate their linear projection layers with the server model to perform local training. The primary distinction between Section 3.3 and 3.4 lies in the data sharing and model training mechanisms:

1) FedAvg framework trains with all the available data and share the entire model with the aggregator.

2) FedRecon framework partition the data and the model to local and global components and only share global gradients with the aggregator.

To elaborate further, each client also uses two subsets of the original dataset: Query and Support sets to train global and local parameters of the server model respectively, which puts more considerations for the problem of data leakage that FedAvg lacks consideration for. Visual representations of this workflow are illustrated in Figure 6 and 7. Details of this algorithm is illustrated in Algorithm 3.4.

In our experiments, we created these subsets with a 7:3 ratio of the original training dataset making the Query set larger in sample size. This design attempts to allow better client contributions towards the global parameters and improve convergence. SMOTE and under sampling techniques were also used to improve class balance after data partitioning. Without resampling, however, the Query set has approximately 7,000 loan records while the Support set has roughly 3,000 records.

Model architecture wise, we added layer normalizations and linear projection to the framework. Specifically, the classification, normalization and projection layer were selected as local parameters while the remaining layers and activation functions (ReLU) were trained exclusively with the Query set.

Local training using the Support Set is also known as the ***Reconstruction Process***. In our case, we performed local gradient descent techniques within the process. Server-wise, we initially selected a slightly higher learning rate of 0.05 and enforced 30 rounds of global training. Details of our experiments will be elaborated further in Section 4.4.

In general, this approach:

1) Is motivated by the assumption that each institution has their own copy of the target labels and unique training data.

2) Relaxes the requirement of model parameters sharing, which by design offers an improvement over the naïve FedAvg approach with regards to data security and privacy.

3) Introduces local and global parameters along with data partitioning to combat data leakage through parameter sharing.

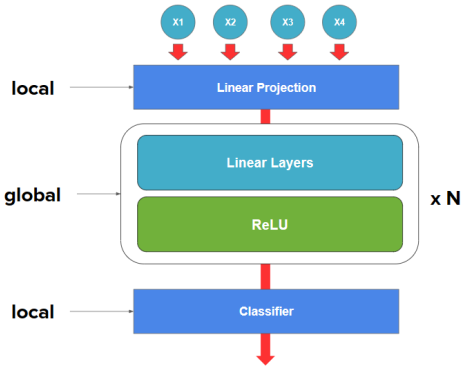

Figure 6 – Model Design for participants

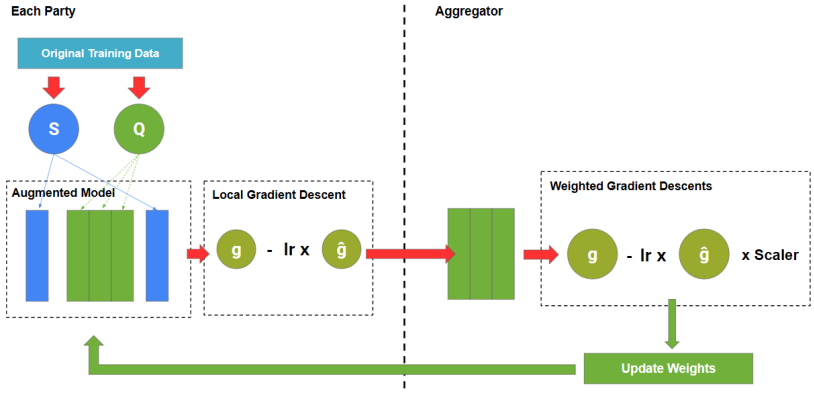

Figure 7 – FedRecon with Linear Projection Overview

ALGORITHM 3.4 FEDERATED RECONSTRUCTION WITH LINEAR PROJECTION LAYER

Algorithm 3.4 is the Federated Reconstruction with Linear Projection Layer. Clients are indexed by k. E is number of local epochs and T is the number of global epochs. η is the learning rate and l is the weights of linear projection layers for each client k. In our experiments, the learning rate η is static and identical for both the global and local gradient descent. Training data for each client is indexed by D with s and q subscripts representing the Support and Query subsets respectively. The data partition technique is indexed by S. Partitioning techniques can be as simple as random sampling that returns two distinct data subsets with equal or different lengths. Data volume is indexed by n.

procedure S
  for each round from t = 1,2,... to T **do**
    $\widehat{S_t} \leftarrow$ (Random Set of K clients)
    for each client k in $\widehat{S_t}$ in parallel **do**
      $w_{t,k}, n_k \, ClientUpdate(k, w_t)$
    end for
    $n_t = \sum n_{k,t}$
    $w_{t+1} \leftarrow w_t - \eta \sum \frac{n_{t,k}}{n_t} \, w_{t,k}$
end for
end procedure

procedure ClientUpdate (k, w)
  $D_{s,k}, D_{q,k} \leftarrow S(D_k)$
  for each local epoch $e = 1, 2, \ldots, E$ **do**
    for each batch $b \in B$ **do**
      if $l_{t-1,k}$ exists **then**
        $w_{local,t}, l_{t,k} \leftarrow w_{local,t} - \eta \nabla \ell(D_{s,k}, l_{t-1,k}, b, e)$
      else
        Initialize $l_{t,k}$
        $w_{local,t}, l_{t,k} \leftarrow w_{local,t} - \eta \nabla \ell(D_{s,k}, l_{t,k}, b, e)$
      end if
      $w_{global,t}, l_{t,k} \leftarrow w_{global,t} - \eta \nabla \ell(D_{q,k}, w_{local,t}, b, e)$
    end for
    Store $w_{local,t}$ and $l_{t,k}$ for next global round
    $n_k \leftarrow |D_{q,k}|$
  return $w_{global,t}, n_k$ to server
end procedure

3.5 Vertical Federated Learning with FedV

For this final approach, we largely follow the FedV framework as proposed in (Xu 2021) [11]. Aside from the participating parties, the framework also requires the roles of Aggregator and Trusted Party (TP) to be present in the training process. The Aggregator will orchestrate the end-to-end training process with all involved parties while TP is mainly responsible for (1) the management of the functional encryption and decryption keys, and (2) inspecting the validity of the weight vectors produced by the Aggregator through its Inference Prevention Module (IPM). Notably, under the FedV framework, the Aggregator will have no access to the unencrypted inputs (including raw features and the intermediate results) from the participating parties. The protocol and the functional encryption schemes adopted in the framework also guarantee that the Aggregator will not be able to recover or decrypt the inputs from the participating parties even though it uses the inputs for the partial model aggregation and the calculation of the loss and gradients.

The FedV framework has two major components which will be discussed in detail in the following subsections:

1. Private Entity Resolution (PER)

2. Federated Vertical Secure Gradient Descent (FedV-SecGrad)

3.5.1 Private Entity Resolution

PER serves the purpose of securely matching data samples in individual party's dataset so that the common data samples can be extracted and sorted for model training. The procedure is similar to the *Extract & Sort* procedure as defined in Algorithm 3.1, however, instead of matching sorting the data samples locally within one party, the PER is a collaborative and distributed effort orchestrated by the Aggregator.

The algorithm 3.5.1 starts with the participating parties which generate a list of Cryptographic Long-Term Keys (CLK) from their individual datasets. The CLK can be the value of a secure hash function whose inputs include features which, combined, can uniquely identify each data sample, such as the aggregation of name, date of birth and address, or simply the universally accepted identifier, such as HKID. The CLK will be able to relate the data samples across the datasets of all parties without revealing the data privacy. The CLKs of each party are passed to the Aggregator which finds the interception of the CLKs from all parties. The interception refers to the list of data samples which are common across parties and can be used for the model training. The Aggregator, after identifying the interception, will generate sorting vectors for the parties. A sorting vector is capable of re-arranging the data samples of a party. The sorting vectors ensure that the common data samples are sorted according to an agreed order. The number of common data samples $|\mathcal{M}|$

is also returned so that the participating parties will know how much data samples to be extracted as well as the order in which the samples should be arranged to support the model training.

ALGORITHM 3.5.1 PRIVATE ENTITY RESOLUTION		
	PARTY	**AGGREGATOR**
1	***procedure generate id list***	***procedure Match & Sort*** ($\mathcal{I}_s^1, \mathcal{I}_s^2, \dots \mathcal{I}_s^k$)
2	$\mathcal{I}_s \leftarrow \{\}$	$\mathcal{M} \leftarrow \{\mathcal{I}_s^1 \cap \mathcal{I}_s^2 \cap \dots \cap \mathcal{I}_s^k\}$
3	***for*** *each* $id_i \in S$ ***do***	$define\ \mathcal{v}_1, \mathcal{v}_2, \dots \mathcal{v}_k$
4	$id_{ci} = CLK(id_i)$	***for*** *i from 1 to k* ***do***
5	$\mathcal{I}_s \leftarrow \mathcal{I}_s \cup id_{ci}$	$\mathcal{v}_i = sort(\mathcal{I}_s^i, \mathcal{M})$
6	***end for***	***end for***
7	***return*** $\mathcal{I}_s$	***return*** $\mathcal{v}_1, \mathcal{v}_2, \dots \mathcal{v}_k, \lvert\mathcal{M}\rvert$
8	***end procedure***	***end procedure***
9		
10		

3.5.2 Federated Vertical Secure Gradient Descent

FedV-SecGrad performs the aggregation of partial model outputs, and the calculation of the loss and gradients for the model updates. The FedV-SecGrad is a two-phase process which performs feature dimension aggregation and sample dimension aggregation. The use of Multi-Input Functional Encryption (MIFE) and Single-Input Functional Encryption (SIFE) in the two-stage process ensures that the Aggregator has no access to plaintext inputs from the participating parties.

3.5.2.1 Functional Encryption Schemes

In our work, we selected the "Learning With Error (LWE)" functional encryption scheme to perform the MIFE and SIFE for two practical reasons. The first reason is about the scheme's security. The LWE scheme can be reduced to a lattice problem which is an NP-hard problem (i.e., a problem with no polynomial time algorithm to solve) and is known to be secure against quantum computation. As such, it is likely that the security of our work can survive in the post-quantum era. The other reason is the efficiency and precision required in our problem. In our work, we also explored the Decisional Diffie-Hellman (DDH) based scheme. The scheme works very efficiently in the encryption phase, however, since there is no known efficient algorithm to recover the results of large value using discrete logarithm, the scheme is extremely slow in the decryption phase. Further, the DDH is known to be vulnerable to the attacks by quantum computation. The Shor's algorithm is able to solve the DDH problem efficiently in quantum polynomial time of ($O(log|\mathcal{G}|)$) for a cyclic group $\mathcal{G}$ as confirmed in [14].

The function to be applied in our work under LWE based MIFE and SIFE schemes is the inner product which is one of the most frequently used functions in the machine learning domain. The two schemes serve two different purposes under the FedV-SecGrad process. For the MIFE scheme, it is used in the feature dimension aggregation where the Aggregator will apply the inner product function for the intermediate results from the participating parties and a weight vector so that the

intermediate results can be fused according to the assigned weights. For the SIFE scheme, it is used in the sample dimension aggregation. After calculating binary cross entropy loss, the Aggregator will apply the inner product function to the data samples and the loss to obtain the gradients for back propagation. The data samples refer to the inputs to the last layer of the local model of the participating parties to produce the intermediate results. In both MIFE and SIFE schemes, the Aggregator will have no way to know or decrypt the intermediate computation results and the data samples from the participating parties. However, as the Aggregator is responsible for generating the weight vector to fuse the intermediate results and calculating the gradients for the back propagation, it will have knowledge on the weight vector and gradients. Therefore, the weight vector and gradients are not considered secret as the inner product calculation is taken place within the control of the Aggregator. We will show that the MIFE scheme is a natural extension to SIFE. Thus, we will first describe the SIFE scheme and extend our discussion to cover the MIFE scheme.

The SIFE scheme works as follows:

Initial Setup

The initial setup defines the key parameters used in functional encryption scheme, including but not limited to the master public key and master secret key. The master public key is used to generate the actual public and private keys for the encryption and decryption. The master public key and private keys should be held by the Trusted Party or severally by the participating parties. For the simplicity of illustration, we assumed that the keys are managed by the Trusted Party. The setup includes the following:

$Define\ A\ \in \mathbb{Z}_q^{m\times n}, S \in \mathbb{Z}_q^{l\times n}, \varepsilon \sim N(0,\sigma) \in \mathbb{Z}_q^{m\times l}, r \xleftarrow[rand]{} \{0,1\}^m$
$\Delta = \frac{q}{p}, where\ q\ and\ p\ are\ big\ prime\ numbers\ and\ q \gg p$
$\mathbb{Z}_q\ is\ the\ cyclic\ group\ over\ the\ prime\ number\ q$
$\mathbb{Z}_q^{m\times n}\ denotes\ a\ m \times n\ matrix\ whose\ elements\ are\ drawn\ from\ \mathbb{Z}_q$
$\mathbb{Z}_q^{n}\ denotes\ a\ vector\ of\ dimension\ n\ whose\ elements\ are\ drawn\ from\ \mathbb{Z}_q$

$Master\ Public\ Key\ (MPK) = AS^T + \varepsilon, A$
$Master\ Secret\ Key\ (MSK) = S$
$secret\ key\ (s_i) \in \{s_1, s_2, \dots s_l\} \in S, where\ s_i \in \mathbb{Z}_q^n$

Encryption

A participating party may request the Trusted Party for the public keys for encrypting its inputs. By design, each element in the input vector can be encrypted with a distinct public key. For an input vector of l dimension$[x_1, x_2, \dots x_l]$, the party may request l public keys each of which can

be used to encrypt an element in the vector. Upon receiving the public keys, the party can perform the following steps to encrypt its input vector $x = [x_1, x_2, \dots x_l]$.

Choose an r to add randomness in the ciphertext.
$ct_0 = Ar, Ar \in \mathbb{Z}_q^n$

Encrypt the elements of the input vector individually
$ct_i = MPK_i \cdot r + x_i\Delta = Ars_i{}^T + \varepsilon_i r + x_i\Delta, ct_i \in \mathbb{Z}_q$

Sum up the encrypted input elements

$$ctx = \sum_{i=1}^{l} ct_i$$

The ciphertext consisting of (ct_0 and ctx) will be passed to the Aggregator.

Key Generation

The Aggregator may request the Trusted Party to encrypt its weight vector $[y_1, y_2, \dots y_l]$ with the secret keys. The dimension of the weight vector must be equal to the dimension of the input vector from the participating party as it should be for the ordinary inner product operation. The Trusted Party will perform the following steps:

$y = [y_1, y_2, \dots y_l]$ $is\ the\ weight\ vector$
$sk = S \cdot y, sk \in \mathbb{Z}_q^n$

The sk will be passed to the Aggregator for decrypting the inner product result.

Decryption

The Aggregator will make of the ciphertext (ct_0 and ctx) and sk to calculate the inner product of $< x, y >$. The calculation is a decryption process, but instead of decrypting ciphertext to plaintext, it decrypts the result of the inner product operation. It works as follows:

$$< x, y > \approx \left(\sum_{i=1}^{l} ct_i y_i - ct_0 sk\right) \div \Delta = \left(\sum_{i=1}^{l} (As_i{}^T r y_i + \varepsilon_i r y_i + x_i y_i \Delta) - ArSy\right) \div \Delta$$

$$= \left(\Delta \sum_{i=1}^{l} x_i y_i + \sum_{i=1}^{l} \varepsilon_i\, r y_i + ArSy - ArSy\right) \div \Delta = < x, y > + \frac{1}{\Delta} \sum_{i=1}^{l} \varepsilon_i\, r y_i \approx < x, y >$$

$\frac{1}{\Delta}\sum_{i=1}^{l} \varepsilon_i\, r y_i$ is a very small term relative to $< x, y >$, therefore, after rounding, the result will be equal to $< x, y >$. The weight vector y is generated by the Aggregator, therefore, it is not considered secret under the regime of the Aggregator. Therefore, the $y = [y_1, y_2, \dots y_l]$ can be used in the calculation in plaintext.

The SIFE scheme can be extended to support MIFE for the inner product operation. However, unlike SIFE which calculates the inner product of a weight vector and an input vector which comes from a single party, the MIFE scheme computes the inner product of weight vectors and input vectors from all parties. This allows the fusion of the parties' local model outputs by the Aggregator. The high-level idea is as follows:

Assuming that we have k parties, and they have the following input vectors:

$x_1 = [x_{11}, x_{12}, \dots x_{1l}]$

$x_2 = [x_{21}, x_{22}, \dots x_{2l}]$

...

$x_k = [x_{k1}, x_{k2}, \dots x_{kl}]$

The Aggregator will generate k corresponding weight vectors as follows:

$y_1 = [y_{11}, y_{12}, \dots y_{1l}]$

$y_2 = [y_{21}, y_{22}, \dots y_{2l}]$

...

$y_k = [y_{k1}, y_{k2}, \dots y_{kl}]$

The Aggregator will calculate the inner product of the weight vectors, and the input vectors as follows (without knowing the plaintext of the input vectors $x_1, x_2, \dots x_k$):

$$inner\ product = \sum_{i=1}^{k} \sum_{j=1}^{l} (x_{ij} \times y_{ij})$$

The above can be implemented with through the following process.

Initial Setup

The initial setup of the MIFE scheme is identical to the SIFE scheme except that the setup in the MIFE will also generate one-time pads for individually encrypting the elements of the input

vectors. For the simplicity of the presentation, the identical parts of the setup are not restated here.

$u \xleftarrow[rand]{} \mathbb{Z}_q^{k \times l}$

Encryption

The encryption under MIFE is a two-stage process. The first stage performs an elementwise encryption using a technique like one-time pad, while the second stage performs the SIFE encryption for individual input vectors.

Apply the one-time pad to the input

$$c = \sum_{i=1}^{k} \sum_{j=1}^{l} x_{ij} + u_{ij}$$

The input vectors will become:

$c_1 = [c_{11}, c_{12}, \dots c_{1l}]$
$c_2 = [c_{21}, c_{22}, \dots c_{2l}]$
...
$c_k = [c_{k1}, c_{k2}, \dots c_{kl}]$

Second Stage
The one-time pad encrypted input vectors will be individually encrypted under the SIFE scheme and will pass the following ciphertexts to the Aggregator:

$SIFE.Encrypt(c_1) = ct_1, ctx_1$
$SIFE.Encrypt(c_2) = ct_2, ctx_2$
...
$SIFE.Encrypt(c_k) = ct_k, ctx_k$

Key Generation

In addition to the operations of the Key Generation under the SIFE scheme, the Aggregator will also request the Trusted Party to pair $u \; and \; y$.

Paring

$$z_y = u \cdot y = \sum_{i=1}^{k} \sum_{j=1}^{l} u_{ij} \times y_{ij}$$

Perform the Key Generation in SIFE individually for the weight vectors through the Trusted Party.

$SIFE.KeyGen(y_1) = sk_1$
$SIFE.KeyGen(y_2) = sk_2$
...
$SIFE.KeyGen(y_k) = sk_k$

The generated keys $sk_1, sk_2, \ldots sk_k$ will be passed to the Aggregator for decrypting the inner product result.

Decryption

The decryption will also be a two-step process. The first step is to decrypt the SIFE scheme, and the second step is to decrypt the one-time pad.

Decrypt the SIFE scheme

$$SIFE.Decrypt(ct_1, y_1) =< c_1, y_1 >= \sum_{j=1}^{l}((x_{1j} + u_{1j}) \cdot y_{1j})$$

$$SIFE.Decrypt(ct_2, y_2) =< c_2, y_2 >= \sum_{j=1}^{l}((x_{2j} + u_{2j}) \cdot y_{2j})$$

...

$$SIFE.Decrypt(ct_k, y_k) =< c_k, y_k >= \sum_{j=1}^{l}((x_{kj} + u_{kj}) \cdot y_{kj})$$

$$inner\ product\ (intermediate) = \sum_{i}^{k}\sum_{j=1}^{l}((x_{ij} + u_{ij}) \cdot y_{ij})$$

Decrypt the one-time pad by getting rid of u_{ij}

$$\begin{aligned} inner\ product &= \sum_{i}^{k}\sum_{j=1}^{l}((x_{ij} + u_{ij}) \cdot y_{ij}) - z_y \\ &= \sum_{i}^{k}\sum_{j=1}^{l}((x_{ij} + u_{ij}) \cdot y_{ij}) - \sum_{i=1}^{k}\sum_{j=1}^{l} u_{ij} \times y_{ij} \\ &= \sum_{i}^{k}\sum_{j=1}^{l}(x_{ij} \times y_{ij} + u_{ij\times}y_{ij}) - \sum_{i=1}^{k}\sum_{j=1}^{l} u_{ij} \times y_{ij} \\ &= \sum_{i}^{k}\sum_{j=1}^{l} x_{ij} \times y_{ij} + \sum_{i=1}^{k}\sum_{j=1}^{l} u_{ij} \times y_{ij} - \sum_{i=1}^{k}\sum_{j=1}^{l} u_{ij} \times y_{ij} \end{aligned}$$

$$= \sum_{i}^{k} \sum_{j=1}^{l} x_{ij} \times y_{ij}$$

The decryption process is performed by the Aggregator which should have the knowledge of y but not x. As illustrated above, the only information to which the Aggregator can directly access are $< x, y >$ and $< c_1, y_1 >, < c_2, y_2 >, \ldots < c_k, y_k >$. The former is the final result we want to calculate while the latter is the intermediate results. With the one-time pad encryption and LWE functional encryption scheme, the original input x cannot be recovered from these two pieces of information.

It is worth noting that the operations in the LWE schemes are all based on integers due to the modulus arithmetic operation over the cyclic group. For the calculation involving decimal numbers, we must scale the numbers in question such that it can be represented as integers. For example, the number 3.1415926535897 can be applied with a scaling factor of 8 such that it will become 314159265 ($round(3.1415926535897 \times 10^8)$). The resulting number can be scaled back after the decryption process. In this case, 314159265 can be scaled back to 3.14159265 by $314159265 \div 10^8$. The precision of number is cut short by the scaling process. Although we can increase the scaling factor to preserve greater precision, however, by the design of the scheme, the larger the input is, the larger the cyclic group is required. An element of an input vector should be in the cyclic group of the smaller prime number p, $v \in \mathbb{Z}_p$ to hold the encryption scheme. The larger the cyclic group is required, the greater the resources and longer the time will also be required for the encryption and decryption process. For all practical reasons, we limited the scaling factor to 8 in our work for this report.

With the SIFE and MIFE schemes, we can proceed to implement the FedV-SecGrad which consist of two parts, namely the Feature Dimension Aggregation and Sample Dimension Aggregation, which will be elaborated below.

3.5.2.2 Feature Dimension Aggregation

The Feature Dimension Aggregation in a nutshell is to aggregate the model outputs from the participating parties. If we have k parties, there will be k local models. The outputs of the models are defined as $p_1, p_2, \ldots p_k$. Each of the outputs is a dot product of model weights w and inputs (samples) x. They are further elaborated as follows:

$p_1 = w_1 {x_1}^T$
$p_2 = w_2 {x_2}^T$
...
$p_k = w_k {x_k}^T$

$where$
$w_1, w_2, \dots w_k \in \mathbb{R}^{m \times n}$,
$x_1, x_2, \dots x_k \in \mathbb{R}^{m \times 1}$,
$\mathbb{R}\ is\ the\ set\ of\ real\ numbers,$
$m\ is\ the\ batch\ size\ and\ n\ is\ feature\ size$

The objective of the feature dimension aggregation is to fuse these model outputs with another weight vector v on a per sample basis, such that

$$aw = \sum_{i=1}^{k} p_i v_i = \sum_{i=1}^{k} w_i x_i v_i$$

$aw \in \mathbb{R}^{m \times 1}$

In this step, to ensure that the Aggregator will not have access to the information of wx, MIFE scheme is used to perform the above aggregation. In the FedV framework, it is also suggested there should be an *Inference Prevention Module* (IPM) built in the Trusted Party so that any attempts by the Aggregator to infer the module output wx by way of manipulating the weight vector v can be detected and prevented. For example, a curious Aggregator may assign the weight vector to be $v = [1,0,0 \dots 0]$ to gain the knowledge of $w_1 x_1$, however, the IPM will reject the Key Generation request with this kind of weight vectors.

The aggregated output aw can then be passed to the sigmoid function $\sigma(aw)$ to predict the labels $\hat{y}$. With the true labels y and $\hat{y}$, the Aggregator can calculate the binary cross entropy loss and its gradient for the back propagation.

$$Binary\ Cross\ Entropy\ (y, \hat{y}) = \mathcal{L} = -[y \times log(\hat{y}) + (1 - y) \times \log(1 - \hat{y})]$$

$$\frac{\partial \mathcal{L}}{\partial w} = \nabla w = (\hat{y} - y)x$$

As shown above, we need the samples x to calculate ∇w for the gradient descent process. In this regard, we need the Sample Dimension Aggregation.

3.5.2.3 Sample Dimension Aggregation

To prevent the Aggregator from gaining knowledge of samples x, we will use the SIFE encryption scheme to securely calculate the inner product of $(\hat{y} - y)$ and x on a per sample basis.

x_{ij} denotes the j-th features of the i-th sample, and ∇w_j denotes the j-th weight gradient. To calculate the gradient ∇w_j, we will perform the following calculation:

$$\nabla w_j = \frac{1}{s}\sum_{i=1}^{s}(\hat{y}_i - y_i)x_{ij}$$

Using SIFE, we can rephrase the calculation as follows:

$$\nabla w_j = \frac{1}{s}SIFE((\hat{y}_i - y_i), x_{ij})$$

The ∇w can then be used to update the local models of the participating parties. The model design in this approach is illustrated in the following diagram. The ∇w follows the green path to update the parameters of the local models of the participating parties in the backward direction.

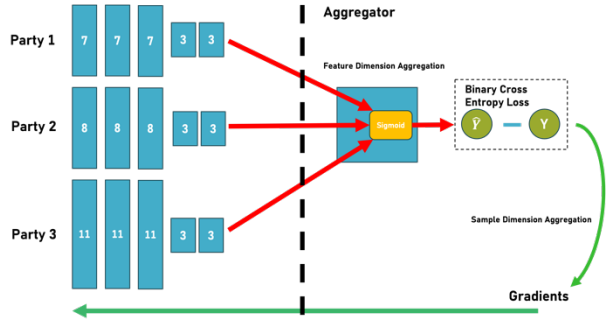

Figure 5 - FedV Overview

For this approach to be secure, it is observed that the training batch size must be larger than 1. This is because, if the batch size is equal to one, i.e., one training sample per batch, the Aggregator is able to recover the exact training sample as the Aggregator knows the result of the inner product of the data sample and the corresponding loss as well as the loss itself. This is essentially like solving a formula with one unknown. When the batch size is larger than 1, the Aggregator will be required to solve a formula with two or more unknowns, therefore, there will be indefinite number of solutions to the formula. As such, the original training samples cannot be recovered easily by the Aggregator.

4. Results and Discussions

This section presents the performance of the models developed through the different approaches described in the previous section, and their comparative analysis to highlight the strength and weakness of each approach. This will form the basis upon which our recommendations will be presented in the next section. The performance of the models is evaluated and compared against the F1 and the Area Under the ROC Curve (AUC) metrics. The metrics are so chosen because the problem is concerned with a binary classification problem with imbalanced labels. The normal accuracy metric will have bias towards to the popular labels. In our dataset, the good loans to defaulted loans ratio is roughly 4 to 1. If we blindly predict all inputs as good loans, we can easily achieve a decent accuracy of close to 80% which tells nothing about the robustness of the model performance. The F1 score, instead, combines the precision (how many identified bad loans are correct) and recall (how many bad loans can be correctly identified), and strikes a much better balance than the accuracy metric. Similarly, the AUC metric also robustly measures the model performance by means of True Positive Rate and False Positive Rate.

4.1 From Local Learning, Split Learning to FedV

We group these three approaches, i.e., Local Learning, Split Learning and FedV under this subsection as they share a similar training process and the same goal of producing a single global model for label prediction.

In this section, we compare the training and validation performance of these approaches in terms of the F1 Score, AUC Score, Binary Cross Entropy Loss (BCELoss) and the training time. Due to practical reasons, which will be elaborated later, we have divided the FedV approach into two configurations. The two configurations are identical in every aspect except that one configuration is enabled with the functional encryption schemes (i.e., MIFE and SIFE) while the other configuration is without the schemes. We call the configuration with the schemes “FedV with FE” and the other “FedV without FE” to differentiate them in our discussion below.

The general configurations for model training are as follows:

1. Number of training epochs is 20,000 for Local Learning and Split Learning without batching

2. Number of training epochs is 30 for FedV with a batch size of 5 or 2

3. 20,047 training samples except for “FedV with FE”

4. 8,591 test samples for validation

5. The optimizer is Rectified Adam

6. The learning rate is 0.001 with a weight decay of 0.00005

7. Apple M3 Max 128GB 14-inch MacBook Pro is used for the model training

For “FedV with FE”, the number of training samples is about 1% of the full dataset, i.e., 285. The significant reduction in the training samples is due to a practical reason. It took 38 minutes to complete the training of 30 epochs and 285 training samples, therefore, it is estimated that it will take around 63 hours to complete the same training as “FedV without FE”. Although the training samples is impractically small, the result did show that functional encryption scheme did not adversely affect the training performance in terms of the BCELoss and F Score despite the scaling technique applied in the encryption and decryption process. The details will be elaborated later in this report.

4.2 Comparative Analysis on Local Learning, Split Learning and FedV

Figure 6 to 9 below shows the training performance in terms of F-Score and BCELoss over the training epochs for the local learning, split learning, “FedV without FE” and “FedV with FE” approaches. Figure 10 shows validation performance for the model trained in the local learning approach. The details of the validation performance evaluation will be discussed later in this section.

It is worth noting that the training time is significantly longer in “FedV with FE” than “FedV without FE”. The two approaches are identical except that one used the MIFE and SIFE for the model weight aggregation and gradient calculation as detailed in the previous section, while the other just performed the calculation using plaintext numbers. Therefore, we can conclude that the computing overhead of the functional encryption scheme is very significant. It may not be practical in real-life to adopt this kind of functional encryption scheme for serious model training which will involve hundreds of thousands of training samples unless specialist hardware accelerators can be deployed to significantly shorten the encryption and decryption time. Nonetheless, the result does confirm that the scaling technique used in the functional encryption schemes under the FedV approach does not significantly affect the training performance as there is a clear and steady trend of the declining loss shown in Figure 9 as the training epochs progress. The Best BCELoss and the Best F1 Score shown in Table 3 for “FedV with FE” as compared to “Fed V without FE” also confirm that the performance impact is minimal.

Among all these approaches, local learning, as expected, has the best training performance as there is not any overhead of Federated Learning introduced in this approach. The Split Learning approach comes second. The good performance may be explained by its model design as more parameters can be passed from the participating parties to the Aggregator through the data exchange across the cut-layers than that was done in the FedV approach. The increase in the number of parameters passed to the Aggregator can indirectly facilitate more interactions among local models of the participating parties which make it more like local training and therefore, can capture more interrelationships between features, and improve the score and loss.

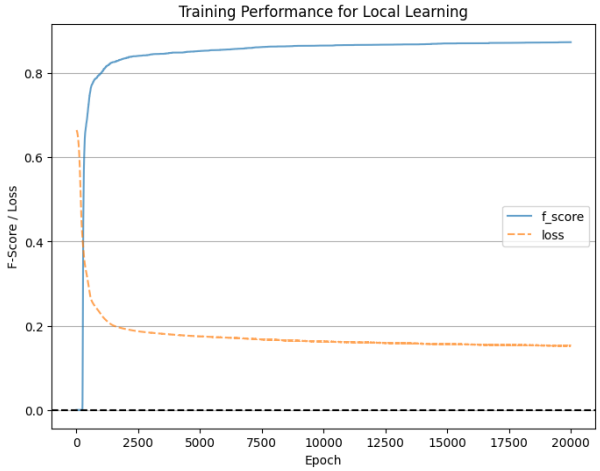

Figure 6 - Local Training

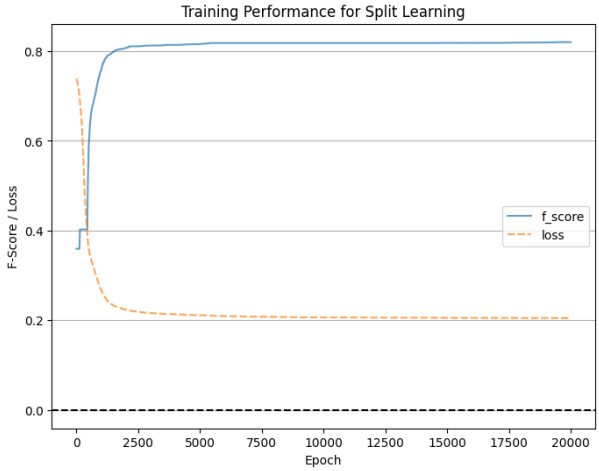

Figure 7 - Split Learning

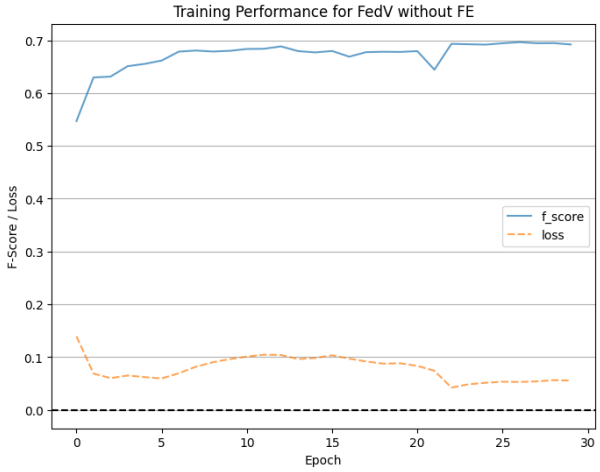

Figure 8 - FedV

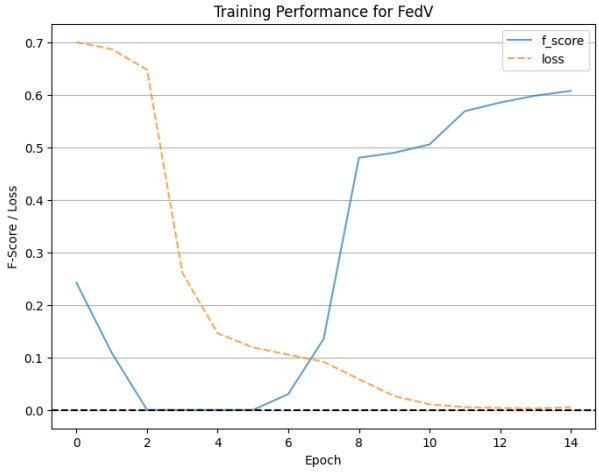

Figure 9 - FedV with Functional Encryption

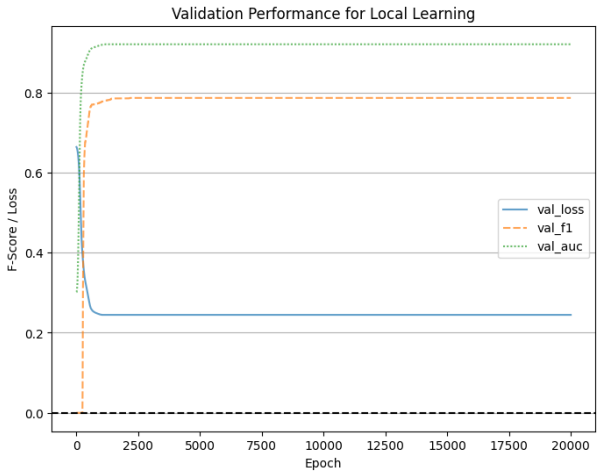

Figure 10 - Validation Performance for Local Learning

Table 3 below, shows the best training F1 score, best BCELoss and the training time for each of the approaches.

Table 3 - Training Performance

Comparison on the Training Performance

	Best F1 Score	Best BCELoss	Training Time
Local Learning	0.8736	0.1531	2 mins 40.6s
Split Learning	0.8200	0.2048	3 mins 3.2s
FedV without FE	0.6964	0.0424	1 min 30.9s
FedV with FE[1]	0.6071	0.0030	44 mins 8.9s

Remarks: 1. The "Fed V with FE" approach only run for 30 training epochs with 1% data of the dataset as the training data.

Table 4 below shows the validation performance for three different approaches, namely local training, split learning and "FedV without FE". The validation makes use of the unseen test samples, which is around 30% data of the dataset, to evaluate the models developed in the three approaches in terms of their F1 Score and AUC Score.

The approach "FedV with FE" is, however, excluded from the validation performance comparison since the model under the approach is only trained with 1% data of the dataset. The result of the under-trained model may distort the true performance of the FedV approach. On the other hand, the model trained under the "FedV without FE" approach should be in theory, the best performing model that the FedV approach can ever produce because the functional encryption scheme only improves the security of the training process but not the model performance. Therefore, by comparing the performance of the models developed under the three approaches, we can evaluate and compare their true potential. The models of the three approaches were trained and tested under the configurations mentioned in the section 4.1.

Table 4 - Validation Performance

Comparison on the Validation Performance

	F1 Score	AUC Score
Local Learning	0.7617	0.8951
Split Learning	0.7853	0.9141
FedV without FE	0.7059	0.8880

In terms of both AUC score and F1 score, the model developed through Split Learning has the best performance while local training and FedV comes at second and third respectively. It is surprising to see that the model developed locally does not have superior performance against the other two

models. There is no sign of over-fitting as shown in Figure 6 and 10. The explanation may have to go with the model design. The model design may be sub-optimal to the dataset in the local learning approach while, in the split learning approach, the model design may be better fit to the dataset, and therefore, achieving better performance. It is observed that the FedV approach has a comparable AUC score to the local learning and split learning approaches, however, its F1 Score is notably lower than the other two. This may be explained by the model design of the FedV approach which has fewer parameters for aggregation on the Aggregator's end. This may lead to the loss of some information on the interrelationship between the features and the participating parties' local models.

4.3 Federated Averaging (FedAvg) with Linear Projection

In this framework, each client will concatenate their linear projections with the global model to form a client-side model and their performance after 30 rounds of global training is as follows:

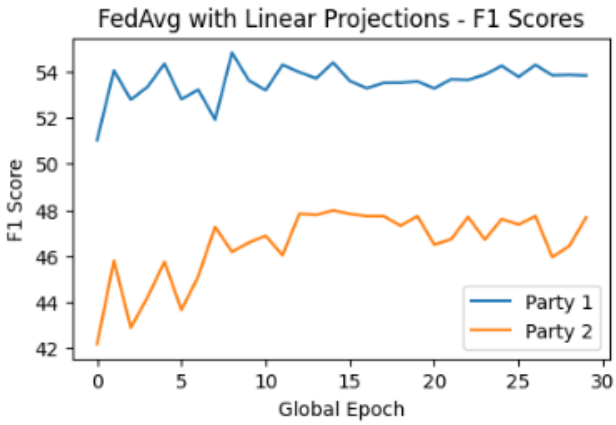

Figure 11 - FedAvg (F1-Score)

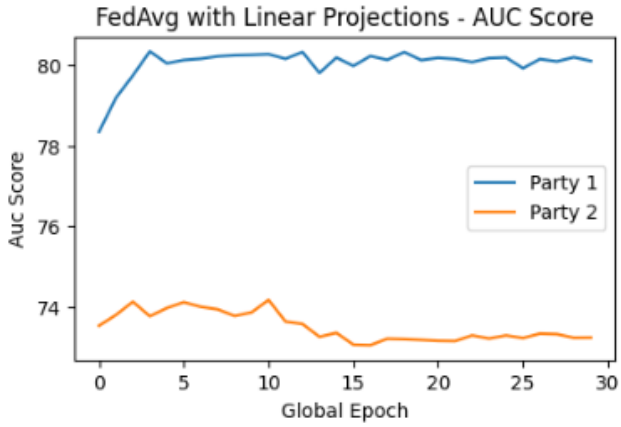

Figure 12 - FedAvg (AUC Score)

F1 Score wise, parties experienced an average performance increase of 9% approximately. Party 1 with personal metadata has an augmented model that generalizes better than party 2 with loan related information. After roughly 15 rounds of global epochs, however, model performance for both parties seems to stagnate and fails to improve. We will elaborate more on this behavior later in Section 4.4.1.

As mentioned in Section 3.3, we selected a slightly lower learning rate of 0.001 and 2 rounds of local epochs. In our experiments, allowing each client to conduct more rounds of local training did not necessarily improve model performance of the global model. Not to mention that the training time grows linearly with the selection of epochs. With our final setups, it takes approximately 34 minutes to complete 30 global rounds of training. Aggregator wise, selecting a learning rate between 0.9 and 0.1 for weighted aggregation will return inconsistent results. We reckon that the aggregator needs to take smaller gradient steps when combining parameters.

As mentioned in Section 3.3, we reduced the number of participating parties and adopted a refined set of data schemas for the training process to enhance convergence. This adjustment proved to be a critical factor for this framework. In particular, parties with vastly different data distribution will be merely noise and will either fail to produce robust local models and/or make negligible contributions to the global training of the server model (Figure 13 and 14).

This finding underscores the importance of careful party selection and data homogeneity in federated learning environments to ensure meaningful collaborative learning outcomes. This is also applicable to FedRecon detailed in Section 4.4. We will compare these results to those of FedRecon in Table 5 in Section 4.4.1.

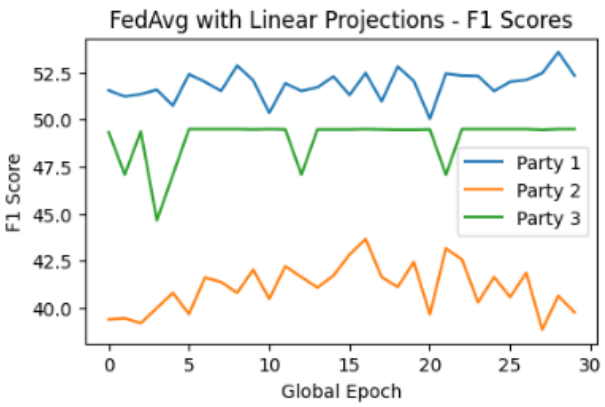

Figure 13 – FedAvg (F1 Score) with 3 Parties

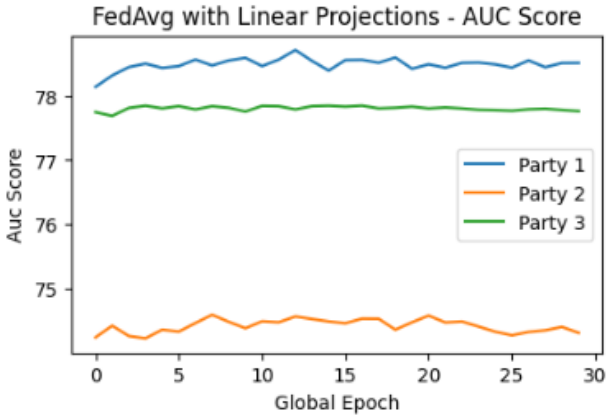

Figure 14 – FedAvg (AUC Score) with 3 parties

4.4 Federated Reconstruction (FedRecon) with Linear Projection

Similarly, each client will have their own augmented version of the global model in this framework and performance after 30 rounds of global training is as follows:

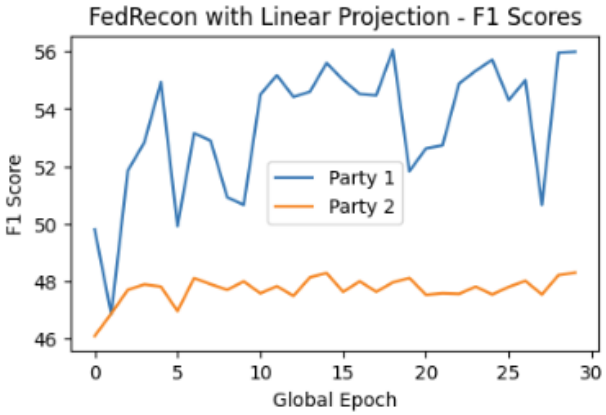

Figure 15 - FedRecon (F1 Score)

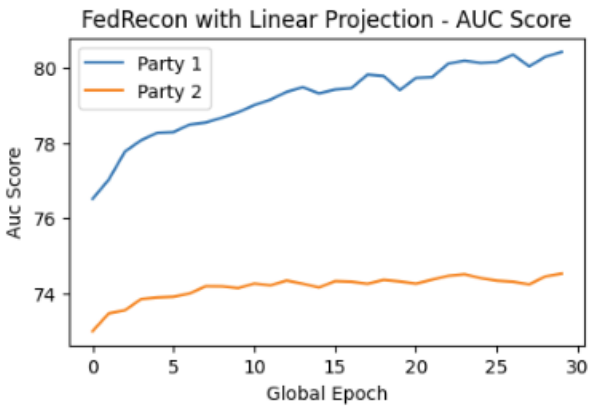

Figure 16 - FedRecon (AUC Score)

Our experiments reveal a disparity in performance enhancement across the participating parties. Party 2, which had access to loan metadata, experienced minimal improvement in performance metrics, with only a 4.5% marginal increase in F1.

In contrast, Party 1, which possessed personal information but lacked loan specifications, demonstrated the most significant benefits, showing approximately 12% improvement. In our experiments, we observed that class rebalancing plays a crucial role in the FedRecon framework. The partitioning of data inherent to FedRecon influences the convergence patterns of client models. In particular, we achieved the most, in terms of performance uplifts, by employing Random Under Sampling techniques to combat this problem.

We also observed that client-side learning rates below 0.01, combined with 5-9 rounds of local epochs, yielded the best results. Conversely, for the aggregator, gradient descents with slightly

higher learning rates between 0.1 and 0.5 returned inconsistent performance. Therefore, we recommend using more local epochs and lower client-side learning rates for future experiments involving FedRecon with an increased number of global epochs. In our experiments, we also tested a simple decaying learning rate technique for local training with both Query and Support sets. Local convergences slightly improved in earlier training rounds, but we reckon static learning rates should be used in FL settings that have limited local epochs.

Additional experiments were conducted to compare local training and FedRecon using 30 training epochs. Using another set of data schema, we obtained accuracy results illustrated in Figures 17 and 18. FedRecon's model architecture is inherently more complex than that of traditional local models, primarily due to its unique requirement to partition parameters into global and local components. This increased sophistication means that local training methods, even when utilizing the same model architecture as FedRecon, may not achieve comparable performance levels. Not to mention the model convergences varies under both training methods. While these findings provide valuable insights into the relative capabilities of FedRecon and local training, they should be viewed as reference points rather than conclusive performance benchmarks.

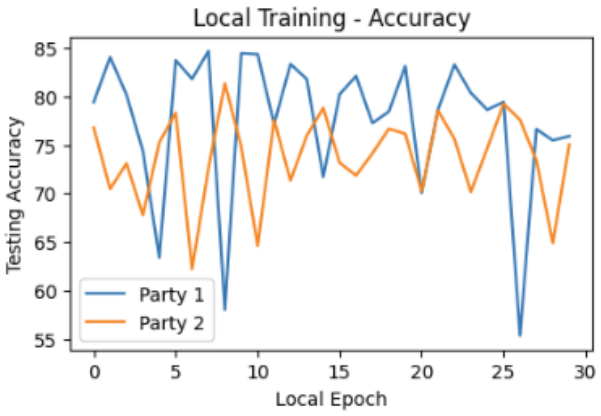

Figure 17 – Local Training (Accuracy)

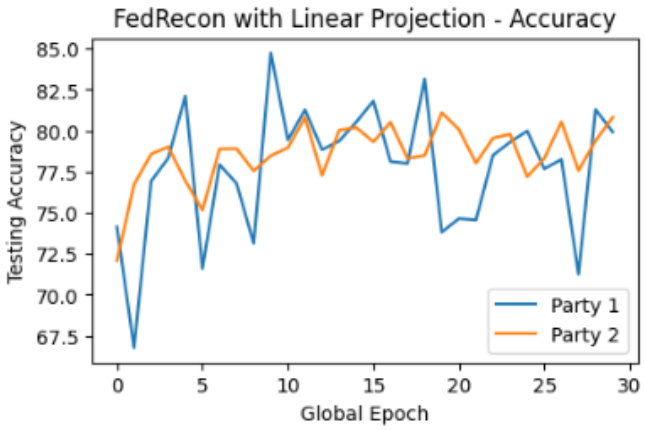

Figure 18 - FedRecon (Accuracy)

To summarize, the effectiveness of this framework in achieving model convergence is significantly influenced by several key factors: feature selection, feature engineering, party selection, and class balancing. We also demonstrated that FedRecon enhanced model performance for parties without target-specific metadata, while ensuring consistent performance for those that do have such metadata. With these findings with FedRecon, we proceed with a detailed comparative analysis with FedAvg that also uses Linear Projection and Local Trainings.

4.5 Comparative Analysis on FedAvg and FedRecon

Table 5 - Comparative Analysis (FedAvg vs FedRecon)

Comparative Analysis of Algorithms utilizing Linear Projection and Local Training

	Average F1 Scores	Average AUC scores	Training Time
Linear Projection + FedAvg	0.5003	0.7670	34 mins approx.
Linear Projection + FedRecon	0.5252	0.7670	20 mins approx.

Compared to FedAvg that also uses linear projections, FedRecon showcased a 2% approx. uplift in F1 scores. We attribute this improvement to the splitting of local and global parameters. We believe model convergences depends heavily on the underlying data distributions and direction of gradient updates, which FedAvg naively handles by making updates in both directions for both parties based on sample sizes.

The implementation of FedRecon / FedAvg with Linear Projection requires local training, which we simulated using a ***sequential*** setup. Our hardware configuration consisted of an RTX 4090 with 16 GB DDR4 RAM and an 8 Core / 16 threads CPU. The training process duration varied based on the number of local epochs selected ranging from 20 to 50 minutes.

In our experiments, we explored a range of batch sizes, and the optimal setting seems to be between 16 to 64. In terms of efficiency, FedRecon required 14 minutes less than FedAvg when the global epoch was set to 30. This increased efficiency can be attributed to FedRecon's approach of separately computing gradients and loss, whereas FedAvg performs backpropagation for the entire model.

The feasibility of FedAvg and FedRecon depends on the regulations in place for the Federated Learning Problem. FedAvg rides on the assumption that each client party is willing to share all model parameters with the server whereas FedRecon relaxes this requirement with the introduction of gradient sharing and data partitioning. Both approaches require parties to have access to label information. In larger-scale scenarios, however, FedRecon is particularly advantageous because it does not require full client participation in each global training round.

We recommend FedRecon since it has a strong emphasis on data protection and leakage prevention, which is the biggest concern for financial institutions. Regarding constraints, FedRecon requires storage of local parameters at each training iteration and requires all parties to conduct local training, which may present challenges associated with limited computational resources at the client level.

4.5 Summary

The following table summarizes and contrasts the key findings and characteristics of the approaches under our study.

Table 6 - Summary

Approach	Performance	Privacy Protection	Public Label	Final Product
Local Learning	Very Good	No	No	A Global Model
Split Learning	Very Good	Medium	No	A Global Model
FedV	Good	Very Good	No	A Global Model
FedAvg	Low	Medium	Yes	Local Models
FedRecon	Low	Good	Yes	Local Models

The "Performance" refers to the model performance measured in the F1 and AUC metrics as detailed in the previous subsection. The "Privacy Protection" is about how much data privacy the approach can help to preserve. For example, in local learning approach, the full dataset is shared to the Aggregator for training a global model, thus, no privacy preservation is possible, and we give a "No" in this approach. On the contrary, in the FedV approach, only the encrypted intermediate computation results are shared from the participating parties to the Aggregator, and all the involved calculations on the intermediate results are performed in an encrypted manner which is opaque to the Aggregator. Theoretically, the Aggregator is unable to recover any useful information on the raw data and the intermediate computation results from the process, therefore, we give this approach a "Very Good" rating in the "Privacy Protection" metric. The FedRecon approach is special in the sense that the participating parties partition the training data into query set and support set, and only share the model weights that are exclusively trained with the query set to the Aggregator for further processing. The support set never leaves the parties in any form including the form of intermediate computation results. For the other approaches, plaintext intermediate computation results are shared to the Aggregator although there is no known efficient way to recover the parties' raw data from the plaintext intermediate results. Therefore, we give a "Good" rating to the FedRecon approach and "Medium" to the Split Learning and FedAvg approaches.

The "Public Label" and "Final Product" are what make our approaches logically fall into two groups. For one group which includes local learning, split learning and FedV, there is only one participating party holding the label information which is considered private to the party. The final

product of this group of approaches is a global model that is collaboratively trained for a better label prediction performance than what this single party alone can do. The other group of approaches, which includes FedAvg and FedReconn, in contrast, assumes that the participating parties share the data labels which are considered public information. The final product for this group is to have all participating parties collaboratively train their own local models for better prediction performance than any parties can do so with their own data individually.

With these characteristics and findings identified, we will proceed with the conclusion and recommendations in the next section.

5. Conclusion and Recommendations

To conclude, Vertical Federated Learning (VFL) presents a promising approach for financial institutions to collaborate and leverage collective data insights while maintaining strict privacy and data regulatory compliance. The application of VFL, however, varies depending on the magnitude of regulatory policies in place for each participating client. For example, an end-to-end centralized model training initiative led by HKMA could be made possible if cross industry data sharing is facilitated among local FIs (financial institutions), credit agencies and other organizations in Hong Kong via an authorized body. In scenarios where the exchange of data and model parameters cannot be assured by an authority, it becomes necessary to explore alternative frameworks for implementing Vertical Federated Learning to preserve the data privacy within the constraints of the regulations. To this end, we conducted an extensive study using a publicly available credit risk dataset. Our research involved experimenting with a range of different approaches and frameworks with the objective of predicting the Probability of Default (PD) for consumer loans.

As summarized in section 4.5, through the experiments conducted in this project, we identified different characteristics of the proposed approaches which may be leveraged to address different scenarios and challenges of Federated Learning in the banking context.

5.1 Split Learning and FedV

For most typical scenarios, a bank would have exclusive access to the label data with which the bank may make business decisions and / or personalize the services and offers for its customers to drive the business growth. Typical examples of events that a bank may want to predict include but certainly are not limited to, (1) whether a given customer would be interested in applying for a travel insurance policy, (2) whether the given customer would be likely in need of an instant loan in the near future or (3) whether the given customer would be interested in subscribing more units of investment fund. Banks typically have their own databases with records of past events which can be treated as the labels for training their own predictive models for customer preference and propensity. They can use these models to personalize the services and offers to their customers in order to improve the likelihood of successful selling. Banks in these typical scenarios are constrained with the limit of their own database. They may wish to partner with non-bank organizations, such as retail stores and travel agencies to collaboratively develop models with greater accuracy and precision to help them grow the business even further. In this case, banks are typically the party who pay for the projects as they can benefit the most from them.

Split Learning and FedV are two suitable approaches to be followed for the joint model development projects of such nature. If the data involved in the training does not contain personally identifiable information (PII), banks and other participants in the joint project may consider using the Split Learning approach as it can offer the best model performance and support the most

flexible model design. The data privacy is protected to a very great extent as again, no single party has all the features and only intermediate computation results are exchanged.

However, if PII is involved, it will be in the bank's best interest to follow the FedV approach for the joint model training to avoid potential data leakage unless the bank also plays the role of Aggregator. If the bank indeed plays the role of Aggregator, the risk of data leakage is lower because only the bank can have access to the intermediate computation results from all participating parties. However, the bank should still be careful and consider the potential risks of leaking their data label through the exploits of some scoring function such as [16]. Compared to leaking personal data, the data label leakage may present a smaller threat to the Bank especially, the data labels are just about whether certain services are subscribed. Nonetheless, in other cases where the data labels are sensitive, such as whether the customer is defaulted in repaying a loan, the bank should consider stepping up the security and adopt the FedV approach.

5.2 FedAvg and FedReconn

In some specific scenarios where the bank and all other participating parties share the common or similar data labels within the same domain and wish to join force in enhancing their own models, FedAvg and FedReconn approaches may be appropriate.

In the context of credit risk assessment, banks in Hong Kong can work with credit reference agencies to collaboratively enhance the credit risk related models. By using FedRecon, each bank can then train its model locally with its proprietary data while simultaneously leveraging the aggregated insights from third party or public data. This collaborative approach can also be extended to other scenarios, such as detecting potential money laundering activities, where banks can share insights derived from public data while keeping their internal data secure.

Framework wise, we recommend FedRecon over FedAvg in these scenarios primarily due to its emphasis on data security. However, both frameworks require careful party selections and some data homogeneity in Federated Learning environments is required to ensure meaningful collaborative learning outcomes.

5.3 The Decision Framework

Based on the properties of different approaches and the discussion above, we created a decision framework to help banks to select an appropriate approach for the Federated Learning project. The decision framework is visualized in Figure 17 below. The decision-making process starts with the most basic question "Is Data Privacy a Concern?". If the data is from the public domain or simply the answer is "No", the most logical decision is to go for local learning as it is most flexible with the potential to produce the best performing model. The next question is whether the data label involved is public information. If the question is "Yes", we may go for the FedRecon approach if

the objective is to improve the parties' local models, or we may go for Split Learning or FedV if the objective is to produce a global model to predict the label. For the decision between Split Learning and FedV, it will depend on whether the Personally Identifiable Information is involved. If the answer is "Yes", FedV will be more appropriate due to its robust functional encryption scheme. Split Learning may be pursed otherwise due to limited exposure for the data leakage risk.

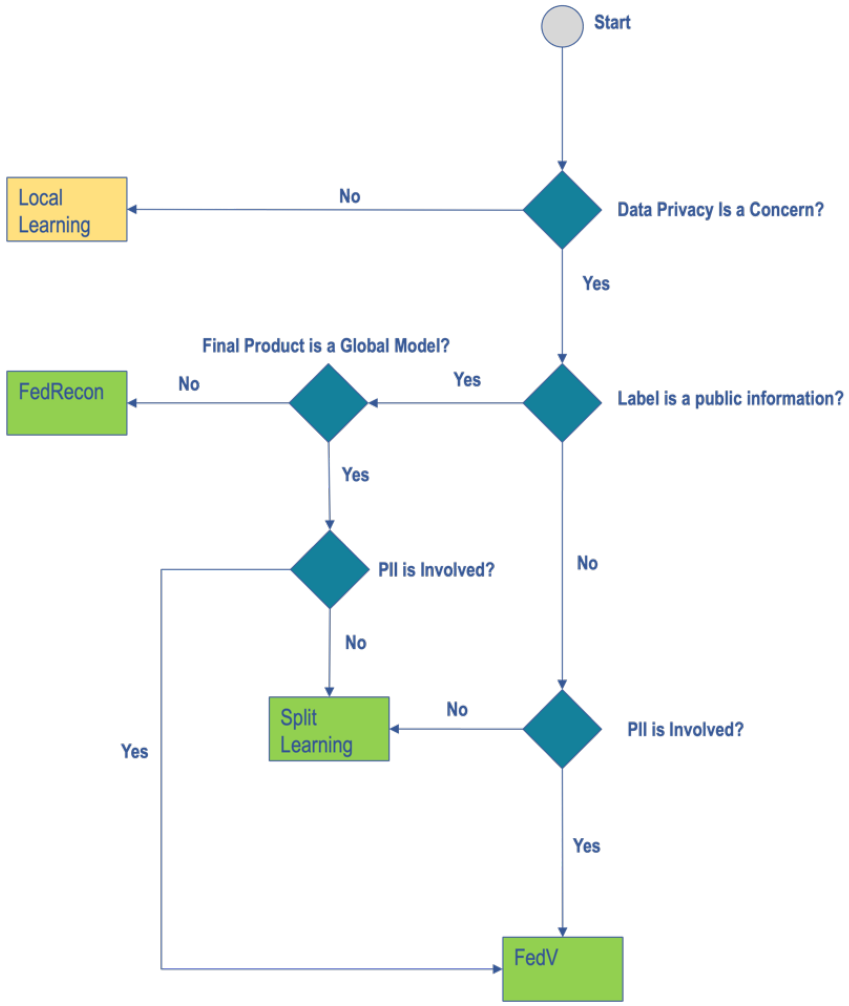

Figure 17 - Decision Framework

5.4 Python Library for Multi-Input Functional Encryption

In this work, we also developed a python library capable of performing inner products under Multi-Input Functional Encryption scheme as another contribution. At the time of writing, we could not find any such library that is publicly available. The library does not only help demonstrate the capabilities of the FedV approach but can also be served as a reference implementation for other MIFE schemes and a tool for further research work.

5.5 Future Work

In this study, we mainly focus on applying different Federated Learning approaches to train neural network models. In real life, banks may want to exploit other types of machine learning models, such as Tree, to improve the interpretability of the models. We may extend our study to cover other types of machine learning models in the future and enhance our decision framework.

Additionally, we observed that the functional encryption did impose considerable overhead in the model training process mainly due to the complexity of the selected encryption schemes. We may also experiment with other encryption schemes to improve the efficiency of the training process.

References

[1] Y. Wang, L. Lin, and J. Chen, Communication-efficient adaptive federated learning, ICML, 2022.

[2] B. McMahan, E. Moore, D. Ramage, S. Hampson, amd B. A. y Arcas, Communication-Efficient Learning of Deep Networks from Decentralized Data, 2016.

[3] K. Singhal, H. Sidahmed, Z. Garrett, S. Wu, K. Rush and S. Prakash, Federated Reconstruction: Partially Local Federated Learning, 2022.

[4] L. Otse, "Credit Risk Dataset," Kaggle, 2023. [Online].

[5] OpenAPI Platform, "TSP List," 2023. [Online]. Available: https://openapi.hkstp.org/banking/en-us/developer-list/. [Accessed: 28- Jul- 2024].

[6] J. Konečný, H.B. McMahan, Y. Felix X., P. Richtárik, A.T. Suresh and D. Bacon, Federated learning: strategies for improving communication efficiency, 2016.

[7] H. Ludwig and N. Baracaldo, Introduction to Federated Learning. In: Ludwig, H., Baracaldo, N. (eds) Federated Learning. Springer, Cham, 2022.

[8] S. Agrawal, B. Libert and D. Stehle, Fully Secure Functional Encryption for Inner Products, from Standard Assumptions. Cryptology ePrint Archive, 2015.

[9] M. Abdalla, F. Bourse and A.D. Caro and D. Pointcheval, Simple Functional Encryption Schemes for Inner Products, Cryptology ePrint Archive, 2015.

[10] M. Abdalla, D. Catalano, D. Fiore and R. Gay and B. Ursu, Multi-Input Functional Encryption for Inner Products: Function-Hiding Realizations and Constructions without Pairings, Cryptology ePrint Archive, 2017.

[11] R. Xu, N. Baracaldo, Y. Zhou, Ali Anwar, James Joshi, and Heiko Ludwig, FedV: Privacy-Preserving Federated Learning over Vertically Partitioned Data. In Proceedings of the 14th ACM Workshop on Artificial Intelligence and Security (AISec '21). Association for Computing Machinery, New York, NY, USA, 181–192, 2021.

[12] O. Regev, On lattices, learning with errors, random linear codes, and cryptography. J. ACM, 2009.

[13] Y. Chen, Quantum Algorithms for Lattice Problems. Cryptology ePrint Archive, 2024.

[14] M. Hhan, T. Yamakawa and A.Yun, Quantum Complexity for Discrete Logarithms and Related Problems, 2023.

[15] P. Vepakomma, O. Gupta, T. Swedish and R. Raskar, Split learning for health: Distributed deep learning without sharing raw patient data, 2018.

[16] O. Li, J. Sun, X. Yang, W. Gao, H. Zhang, J. Xie, V. Smith and C. Wang, Label Leakage and Protection in Two-party Split Learning, ICLR, 2022.