2019 IEEE International Conference on Big Data (Big Data)

Secure and Efficient Federated Transfer Learning

Shreya Sharma*, Chaoping Xing†, Yang Liu‡ and Yan Kang‡
*Department of Electronics Engineering, Indian Institute of Technology (BHU) Varanasi, India
Email: shreyas.cd.ece17@iitbhu.ac.in
†School of Electronics, Information & Electrical Engineering, Shanghai Jiao Tong University, China
School of Physical & Mathematical Sciences, Nanyang Technological University, Singapore
Email: xingcp@ntu.edu.sg
‡Webank, Shenzhen, China
Email: yangliu@webank.com, yangkang@webank.com

***Abstract*—**

Machine Learning models require a vast amount of data for accurate training. In reality, most data is scattered across different organizations and cannot be easily integrated under many legal and practical constraints. Federated Transfer Learning (FTL) was introduced in [1] to improve statistical models under a data federation that allow knowledge to be shared without compromising user privacy, and enable complementary knowledge to be transferred in the network. As a result, a target-domain party can build more flexible and powerful models by leveraging rich labels from a source-domain party. However, the excessive computational overhead of the security protocol involved in this model rendered it impractical.

In this work, we aim towards enhancing the efficiency and security of existing models for practical collaborative training under a data federation by incorporating Secret Sharing (SS). In literature, only the semi-honest model for Federated Transfer Learning has been considered. In this paper, we improve upon the previous solution, and also allow malicious players who can arbitrarily deviate from the protocol in our FTL model. This is much stronger than the semi-honest model where we assume that parties follow the protocol precisely. We do so using the one of the practical MPC protocol called SPDZ, thus our model can be efficiently extended to any number of parties even in the case of a dishonest majority.

In addition, the models evaluated in our setting significantly outperform the previous work, in terms of both runtime and communication cost. A single iteration in our model executes in 0.8 seconds for the semi-honest case and 1.4 seconds for the malicious case for 500 samples, as compared to 35 seconds taken by the previous implementation.

I. Introduction

The application of Artificial Intelligence (AI) is driven by big data availability. To calibrate the performance of real-life complex models such as AlexNet [2] with 60 million parameters and 650,000 neurons, large datasets providing millions of samples are used. AlphaGo [3], was trained on a collection of 29.4 million moves from 160,000 actual games. However, high quality data is not readily available to meet such extensive requirements. Across various organisations, data is present in small quantities (i.e. few samples and labels) and is not heavily supervised (i.e. exists in unlabeled form). Thus, data needed for a particular task might not be present in a single place. Such a scenario can be dealt with by the unification of datasets from different platforms for better statistical modelling. But this approach lacks practicality due to the privacy concerns involved. The datasets available might contain sensitive information such as medical-records or financial data. Strict legislative laws that require explicit user approval before data usage further bound the merging of data. Apart from this, any form of well-supervised (labelled) data might require confidentiality because it constitutes a monetary or competitive advantage for the parties that own it. Thus arises the need for AI models that comply with both security and accuracy.

One solution to the above problem can be Federated Learning [4]. This system enables parties to learn a shared prediction model while keeping all the training data locally stored. A prerequisite for this framework is data being horizontally partitioned (i.e. sharing a common feature space). As an alternative, secure machine learning on vertically partitioned data (i.e. data partitioned in the feature space) has also been studied in depth [5], [6]. Since these approaches require data to either have common samples or common features, they leave majority of the non-overlapping data underutilized.

In this paper, we work on Federated Transfer Learning, where the target-domain party builds a prediction model by leveraging rich labels from a source-domain. This framework provides results for the entire feature and sample space as it doesn't place any restrictions on the distribution of data and thus can find applicability in diverse fields. For instance, many businesses can rely their decisions on the financial activities of different customers and since banks own authentic labels on such financial activities, a collaborative model trained using the datasets of two such organizations can benefit such businesses. In this case, due to involvement of different institutions for training, a small portion of the feature space will overlap, and only a part of the entire user space would intersect. Thus, here FTL becomes an important extension of Federated Learning. Additional security protocols are needed to maintain privacy in the case of FTL as the training involves collaborative calculations using the parameters related to data of both source and target domain. Although our work focusses

978-1-7281-0858-2/19/$31.00 ©2019 IEEE
978-1-7281-0858-2/19/$31.00 ©2019 IEEE

on a two-party case, we discuss the importance of its scalability in the last section of the paper.

A. Our Contribution

Our work takes the FTL model introduced in [1], and:

- Enhances the efficiency by an order of magnitude in the semi-honest security setting by implementing the model using Multi-Party Computation (MPC) as opposed to the previous Homomorphic Encryption (HE) based approach.
- Introduces active security to it, dealing with the case where $m-1$ out of the m parties can deviate from the protocol. We do so using the SPDZ protocol[7], thus our model can be efficiently extended to any number of parties. To the best of our knowledge, ours is the first work to achieve active security for any non-linear deep learning based model for a dishonest majority.
- We present extensive experimental results that underscore the practicality of secure ML even in the active setting. We investigate different aspects related to scalability of the model and also highlights a comparison between the performance in our setting versus the previous work on secure FTL in the semi-honest setting. Our model significantly outperforms this HE-based implementation in terms of both communication cost and runtime.

B. Related Work

Federated Learning can be seen as decentralized machine learning, and thus is closely related to multi-party privacy preserving machine learning. This field has undergone extensive research in recent years due to its vast applicability. CryptoNets [8] based on Leveled Homomorphic Encryption (LHE), enabled encrypted prediction on a server-side model. But the use of LHE results in high computational overhead and since LHE restricts the degree of the polynomial used in the approximation of non-linear activation functions, the model wasn't accurate enough. SecureML [9] focuses on private training and inference of various ML models involving multiple parties. The approach provides security in the semi-honest model and incurred a loss of accuracy due to the crypto-friendly techniques involved. For the evaluation of certain non-linear activation functions like softmax, it proposes an expensive switch between arithmetic secret shares and Garbled circuits. Recent state-of-the-art work in the realm of privacy preservation in ML do so by outsourcing computation to a non-colluding server, and fail to achieve active security in the case of a dishonest majority.

There aren't many instances of work on actively secure ML, as this setting is far less efficient as compared to its semi-honest counter parts. [10] showed promising results for ML training in an actively secure setting, but this framework is only applicable for linear regularized models. [11] is another work in the active setting that shows linear regression and logistic regression using SPDZ can match similar latency as SecureML.

All the frameworks so far do not consider the cases where data sets are small or have weak supervision. To address this scenario, secure Federated Transfer Learning was explored in [1]. It was the first framework enabling federated learning to benefit from transfer learning in a secure way. The work presented experimental results for a model based on HE and provided security in the semi-honest setting, with the major drawback being a lack of efficiency.

II. Problem Definition

Consider $\mathcal{D}_S$ and $\mathcal{D}_T$ are two datasets owned by two different parties and need to be kept private. The label rich source domain dataset can be defined as $\mathcal{D}_S \coloneqq \{(x_i^S, y_i^S)\}_{i=1}^{N_S}$, where $x_i^S \in \mathbb{R}^d$ is the i^{th} sample and $y_i^S \in \{0,1\}$ is the i^{th} label. And the target domain dataset can be defined as $\mathcal{D}_T \coloneqq \{(x_j^T\}_{j=1}^{N_T}$ where $x_j^T \in \mathbb{R}^d$ is the j^{th} sample. We work on the case where there is a co-occurrence of certain samples i.e. $\mathcal{D}_{ST} \coloneqq \{(x_k^{ST}, x_k^{ST})\}_{k=1}^{N_{ST}}$ and, certain samples in $\mathcal{D}_T$ have a label in $\mathcal{D}_S$ i.e. $\mathcal{D}_L \coloneqq \{(x_k^L, y_k^L)\}_{k=1}^{N_L}$. The sample IDs are masked with an encryption scheme which enables the sharing of set of common IDs. We assume that both parties already know the commonly shared IDs.

In this paper, we aim to enable the two parties to build a transfer learning model to predict accurate labels for the target domain dataset while keeping their data private against an adversary. We work on two threat models i.e. *semi-honest* and *malicious*. In the former, an adversary can corrupt one of the two parties, and try to know more information about the private data of the other party than what can be inferred from the output, while following the protocol. In the latter, the adversary can corrupt one of the parties and make it deviate randomly from the protocol specification in order to falsify the output or learn private data of the other party. In the malicious case, we assume that for every iteration the local computations by the adversary on its own data are honest, and it tries to cheat during the interactive calculations on the joint data of the parties. This assumption is reasonable since any MPC protocol guarantees security against an adversary trying to know additional information from the visible messages during protocol execution and not for the case where the protocol starts with false inputs.

III. Preliminaries

(Additive) Secret Sharing protocols rely on splitting every private value involved in the computation (of arithmetic circuits) into additive secret shares. Such a sharing enables linear operations (i.e. addition and scalar multiplication) on the actual values, to be performed locally (without interaction) on the shares in order to obtain the corresponding shares of the output. In contrast, each multiplication of shared values requires a unique multiplication triple [12]. These triples are input-independent and can be generated at any time before the execution of the protocol. Thus these protocols are executed in two phases, the computationally expensive offline phase and a relatively cheap (information-theoretic) online phase. The offline phase is run to generate raw material, which is consumed later by the online phase.

We implement the secret sharing component of our protocol in the ABY framework [13] for the semi-honest setting, and in the SPDZ framework [14] for the malicious setting. In this section, we summarize these two frameworks.

A. ABY

ABY is a mixed-protocol framework that combines various secret sharing schemes in a two-party setting, namely Arithmetic, Boolean and Yao Sharing. Our work is based on Arithmetic Sharing i.e. the values are additively shared in a ring $\mathbb{Z}_{2^l}$, in a semi-honest threat model. The offline phase in this framework involves generation of multiplication triples using Oblivious Transfer (OT). An efficient way to perform many OTs is to extend a small number of expensive baseOTs (based on asymmetric cryptographic operations) using OTExtension (based on much faster symmetric cryptographic primitives) in a constant number of rounds [15]. The online protocol for arithmetic sharing in ABY is described in Figure 1.

Fig. 1: $\Pi^{\mathsf{ABY}}_{\mathsf{Online}}$ - ABY Online Protocol

Input: To input a value x, Party P_i chooses $r \in \mathbb{Z}_{2^l}$, sets its share $\langle x \rangle_i = x - r$ and sends r to P_{1-i}, who sets $\langle x \rangle_{1-i} = r$.
Add: $\langle z \rangle = \langle x \rangle + \langle y \rangle$: P_i locally computes $\langle z \rangle_i = \langle x \rangle_i + \langle y \rangle_i$.
Multiply: $\langle z \rangle = \langle x \rangle \cdot \langle y \rangle$: a multiplication triple generated of the form $\langle c \rangle = \langle a \rangle \cdot \langle b \rangle$ during offline phase is taken. P_i sets $\langle \epsilon \rangle_i = \langle x \rangle_i - \langle a \rangle_i$ and $\langle \rho \rangle_i = \langle y \rangle_i - \langle b \rangle_i$. Parties output ϵ and ρ. P_i sets $\langle z \rangle_i = i \cdot \epsilon \cdot \rho + \rho \cdot \langle a \rangle_i + \epsilon \cdot \langle b \rangle_i + \langle c \rangle_i$
Output: To output a value x to P_i, P_{1-i} sends its share $\langle x \rangle_{1-i}$ to P_i who computes $x = \langle x \rangle_0 + \langle x \rangle_1$.

B. SPDZ

SPDZ is a family of Multi-Party Computation (MPC) protocols for arbitrary number of parties, providing active security with information theoretic MACs, in the case where majority of the parties are corrupted. More specifically, every party P_i has an additive share of the fixed MAC key i.e. $\alpha_i \in \mathbb{F}_p$ such that $\alpha = \alpha_1 + ... + \alpha_n$. A data-item is $\langle \cdot \rangle$-shared, if every Party P_i has a tuple $(a_i, \gamma(a)_i)$ such that a_i is an additive share of a i.e. $a = a_1 + ... a_n$ and $\gamma(a)_i$ is an additive share of the corresponding MAC $\gamma(a)$ i.e. $\gamma(a) = \alpha a$ and $\gamma(a) = \gamma(a)_1 + .. + \gamma(a)_n$. The online-phase of the SPDZ protocol is given in Figure 2.

IV. Secure FTL Interface

In this section, we explain the FTL model used for our work and then present an algorithm to train this model using Secret Sharing.

We assume that the parties S and T have produced a hidden representation of their data using neural networks Net^T and Net^S i.e. $u_i^S = Net^S(x_i^S)$ and $u_i^T = Net^T(x_i^T)$, where $u^S \in \mathbb{R}^{N_S \times d}$ and $u^T \in \mathbb{R}^{N_T \times d}$, d is the dimension of hidden layer. In order to generate labels for target domain the following translator function from [16] is used:

$$\psi(u_j^T) = \frac{1}{N_S} \sum_i^{N_S} y_i^S \cdot u_i^S \cdot (u_j^T)'$$

For simplification, translator function is seen to be of the form, $\psi(u_j^T) = \Lambda^S \mathcal{C}(u_j^T)$, where $\Lambda^S = \frac{1}{N_S} \sum_i^{N_S} y_i^S \cdot u_i^S$ and $\mathcal{C}(u_j^T) = (u_j^T)'$.

Then for training purpose, we follow:

$$\underset{\omega^S, \omega^T}{\operatorname{argmin}} \mathcal{L}_1 = \sum_{i=1}^{N_L} \ell_1(y_i^S, \psi(u_i^T))$$

Fig. 2: $\Pi^{\mathsf{SPDZ}}_{\mathsf{Online}}$ - SPDZ Online Protocol

The set P is the complete set of parties.
Initialise: The parties call preprocessing functionality to obtain enough multiplication triples $(\langle a \rangle, \langle b \rangle, \langle c \rangle)$ and input mask values $(r_j, \langle r_j \rangle)$ according to the function being evaluated. If the functionality aborts, the parties output $\perp$ and abort.
Input: To input x_j, party $P_j \in P$ takes a mask value $(r_j, \langle r_j \rangle)$, then:

1) Broadcasts $\Delta \leftarrow x_j - r_j$
2) Parties compute $\langle x_j \rangle \leftarrow \langle r_j \rangle + \Delta$

Add: On input $(\langle x \rangle, \langle y \rangle)$, locally compute $\langle x + y \rangle \leftarrow \langle x \rangle + \langle y \rangle$
Multiply: On input $(\langle x \rangle, \langle y \rangle)$, the parties:

1) Take a multiplication triple $(\langle a \rangle, \langle b \rangle, \langle c \rangle)$, compute $\langle \epsilon \rangle \leftarrow \langle x \rangle - \langle a \rangle$ and $\langle \rho \rangle \leftarrow \langle y \rangle - \langle b \rangle$ and partially open them to obtain ϵ and ρ.
   Partially opening a share involves each party sending its own share of the value to every other party and computing the sum of all the shares available to it, while the corresponding MAC value $\gamma(x_i)$ is kept secret.
2) Set $\langle z \rangle \leftarrow \epsilon \cdot \rho + \epsilon \cdot \langle a \rangle + \rho \cdot \langle b \rangle + \langle c \rangle$.

Output: To output a share $\langle x \rangle$:

1) Check all partially opened values since the last batched MAC-check, as follows:
   - The parties have ids id_1, ... id_k corresponding to opened values x_1, .. x_k
   - Parties agree on a random vector $\mathbf{r} \leftarrow \mathcal{F}_{\mathrm{Rand}}\left(\mathbb{F}_q^k\right)$
   - Party P_i computes $c \leftarrow \sum_{j=1}^k r_j \cdot x_j$ and $\gamma(c)_i \leftarrow \sum_{j=1}^k r_j \cdot \gamma\left(x_j\right)_i$
   - Parties run batched MAC-check on c, where party P_i inputs c and $\gamma(c)_i$.
2) If the MAC-check fails, output $\perp$ and abort.
3) Open each party P_i's input sent to every other party P_j, to compute $x \leftarrow \sum_{i \in P} x_i$. Run MAC-check with party P_i's input x and $\gamma(x_i)$, to verify $\langle y \rangle$. In case this check fails, output $\perp$ and abort; otherwise output x.

In the above equation, ω^S and ω^T are training parameters of Net^S and Net^T respectively. If Net^S has l_S layers, $\omega^S = \{\omega_l^S\}_{l=1}^{l_S}$, similarly if Net^T has l_T layers, $\omega^T = \{\omega_l^T\}_{l=1}^{l_T}$, where ω_l^S and ω_l^T are training parameters for the l^{th} layer. ℓ_1 is loss function used i.e. $\ell_1(y,\psi) = \log(1+\exp(-y\psi))$. In order to achieve feature transfer learning in federated learning setting, we follow:

$$\underset{\omega^S,\omega^T}{\operatorname{argmin}} \mathcal{L}_2 = -\sum_i^{N_{ST}} \ell_2\left(u_i^S, u_i^T\right),$$

where ℓ_2 is alignment loss, i.e. $\ell_2\left(u_i^S, u_i^T\right) = \kappa u_i^S \left(u_i^T\right)'$ and $\kappa = -1$.

By combining the two loss equations with regularization terms, the objective of training becomes:

$$\underset{\omega^S,\omega^T}{\operatorname{argmin}} \mathcal{L} = \mathcal{L}_1 + \gamma\mathcal{L}_2 + \frac{\lambda}{2}\left(\mathcal{L}_3^S + \mathcal{L}_3^T\right), \tag{1}$$

where the weight parameters are given by λ and γ, and the regularization terms are given as, $\mathcal{L}_3^S = \sum_l^{l_S} \left\|\omega_l^S\right\|_F^2$ and $\mathcal{L}_3^T = \sum_l^{l_T} \left\|\omega_l^T\right\|_F^2$.

The relation followed for back-propagation:

$$\frac{\partial\mathcal{L}}{\partial\omega_l^i} = \frac{\partial\mathcal{L}_1}{\partial\omega_l^i} + \gamma\frac{\partial\mathcal{L}_2}{\partial\omega_l^i} + \lambda\omega_l^i. \tag{2}$$

In order to prevent leakage of any data for both parties A and B, we use secret sharing to collaboratively compute Eq. 1 and Eq. 2. For better suitability to the framework used, we use second order Taylor approximation for the ℓ_1 function i.e.

$$\ell_1(y,\psi) \approx \ell_1(y,0) + \frac{1}{2}\mathcal{T}_1(y)\psi + \frac{1}{8}\mathcal{T}_2(y)\psi^2. \tag{3}$$

where, $\mathcal{T}_1(y) = \left.\frac{\partial\ell_1}{\partial\psi}\right|_{\psi=0}$, $\mathcal{T}_2(y) = \left.\frac{\partial^2\ell_1}{\partial\psi^2}\right|_{\psi=0}$.

$$\frac{\partial\ell}{\partial\psi} = \frac{1}{2}\mathcal{T}_1(y) + \frac{1}{4}\mathcal{T}_2(y)\psi \tag{4}$$

For logistic loss, $\mathcal{T}_1(y) = y$, $\mathcal{T}_2(y) = y^2$. This approximation helps avoid the use of expensive techniques for division by a secret value, as required in the calculation of $(\exp(-y\psi))$ for ℓ_1.

Hence the final loss and gradient equations become:

$$\begin{aligned}\mathcal{L} = &\sum_i^{N_L}\left(\ell_1\left(y_i^S,0\right) + \frac{1}{2}\mathcal{T}_1\left(y_i^S\right)\Lambda^S\mathcal{C}\left(u_i^T\right)\right.\\ &\left.+\frac{1}{8}\mathcal{T}_2\left(y_i^S\right)\Lambda^S\mathcal{C}\left(u_i^T\right)\left(\Lambda^S\mathcal{C}\left(u_i^T\right)\right)\right)\\ &+\gamma\sum_i^{N_{ST}}\left(\ell_2^T(u_i^T) + \ell_2^S\left(u_i^S\right) + \kappa u_i^S\left(u_i^T\right)'\right)\\ &+\frac{\lambda}{2}\mathcal{L}_3^S + \frac{\lambda}{2}\mathcal{L}_3^T\end{aligned} \tag{5}$$

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\omega_l^T} = &\sum_i^{N_L}\left(\frac{1}{4}\mathcal{T}_2\left(y_i^S\right)\Lambda^S\mathcal{C}\left(u_i^T\right)\left(\Lambda^S\frac{\partial\mathcal{C}\left(u_i^T\right)}{\partial\omega_l^T}\right)\right.\\ &\left.+\frac{1}{2}\mathcal{T}_1\left(y_i^S\right)\Lambda^S\frac{\partial\mathcal{C}\left(u_i^T\right)}{\partial\omega_l^T}\right)\\ &+\sum_i^{N_{ST}}\left(\gamma\kappa u_i^S\frac{\partial u_i^T}{\partial\omega_l^T} + \gamma\frac{\partial\ell_2^T\left(u_i^T\right)}{\partial\omega_l^T}\right) + \lambda\omega_l^T\end{aligned} \tag{6}$$

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\omega_l^S} = &\sum_i^{N_L}\left(\frac{1}{4}\mathcal{T}_2\left(y_i^S\right)\Lambda^S\mathcal{C}(u_i^T)\left(\frac{\partial\Lambda^S}{\partial\omega_l^S}\mathcal{C}\left(u_i^T\right)\right)\right.\\ &\left.+\frac{1}{2}\mathcal{T}_1\left(y_i^S\right)\mathcal{C}\left(u_i^T\right)\frac{\partial\Lambda^S}{\partial\omega_l^S}\right)\\ &+\gamma\sum_i^{N_{ST}}\left(\kappa u_i^T\frac{\partial u_i^S}{\partial\omega_l^S} + \frac{\partial\ell_2^S\left(u_i^S\right)}{\partial\omega_l^S}\right) + \lambda\omega_l^S\end{aligned} \tag{7}$$

The components of the Eq. 5, 6, 7 can be divided into three categories according to the ones that can be locally computed by S and T, and the ones that are to be computed on the joint data of both parties.

For Eq. 5:

$$\begin{aligned}\mathcal{L}_1^{ST} = &\sum_i^{N_L}\frac{1}{8}\mathcal{T}_2\left(y_i^S\right)\Lambda^S\mathcal{C}\left(u_i^T\right)\left(\Lambda^S\mathcal{C}\left(u_i^T\right)\right)\\ &+\frac{1}{2}\mathcal{T}_1\left(y_i^S\right)\Lambda^S\mathcal{C}\left(u_i^T\right) + \sum_i^{N_{ST}}\kappa u_i^S\left(u_i^T\right)'\end{aligned} \tag{8}$$

$$\mathcal{L}_1^S = \sum_i^{N_L}\ell_1\left(y_i^S,0\right) + \sum_i^{N_{ST}}\ell_2^S\left(u_i^S\right) + \frac{\lambda}{2}\mathcal{L}_3^S \tag{9}$$

$$\mathcal{L}_1^T = \gamma\sum_i^{N_{ST}}\ell_2^T(u_i^T) + \frac{\lambda}{2}\mathcal{L}_3^T \tag{10}$$

For Eq. 6:

$$\begin{aligned}\mathcal{L}_2^{ST} = &\sum_i^{N_L}\frac{1}{4}\mathcal{T}_2\left(y_i^S\right)\Lambda^S\mathcal{C}\left(u_i^T\right)\left(\Lambda^S\frac{\partial\mathcal{C}\left(u_i^T\right)}{\partial\omega_l^T}\right)+\\ &\frac{1}{2}\mathcal{T}_1\left(y_i^S\right)\Lambda^S\frac{\partial\mathcal{C}\left(u_i^T\right)}{\partial\omega_l^T} + \sum_i^{N_{ST}}\left(\gamma\kappa u_i^S\frac{\partial u_i^T}{\partial\omega_l^T}\right)\end{aligned} \tag{11}$$

$$\mathcal{L}_2^T = \sum_i^{N_{ST}}\gamma\frac{\partial\ell_2^T\left(u_i^T\right)}{\partial\omega_l^T}) + \lambda\omega_l^T \tag{12}$$

For Eq. 7:

$$\begin{aligned}\mathcal{L}_3^{ST} = &\sum_i^{N_L}\left(\frac{1}{4}\mathcal{T}_2\left(y_i^S\right)\Lambda^S\mathcal{C}(u_i^T)\left(\frac{\partial\Lambda^S}{\partial\omega_l^S}\mathcal{C}\left(u_i^T\right)\right)\right.\\ &\left.+\frac{1}{2}\mathcal{T}_1\left(y_i^S\right)\mathcal{C}\left(u_i^T\right)\frac{\partial\Lambda^S}{\partial\omega_l^S}\right) + \gamma\sum_i^{N_{ST}}\kappa u_i^T\frac{\partial u_i^S}{\partial\omega_l^S}\end{aligned} \tag{13}$$

$$\mathcal{L}_3^S = \gamma\sum_i^{N_{ST}}\left(\frac{\partial\ell_2^S\left(u_i^S\right)}{\partial\omega_l^S}\right) + \lambda\omega_l^S \tag{14}$$

Using the notations stated above, we construct Algorithm 1 to execute the FTL training, where $\Pi^{\mathsf{SS}}_{\mathsf{Online}}$ can be $\Pi^{\mathsf{ABY}}_{\mathsf{Online}}$ or $\Pi^{\mathsf{SPDZ}}_{\mathsf{Online}}$. Once the model has been trained, party T can use it to obtain predictions for its data samples. This would involve the parties computing $\psi(x^T)$ collaboratively, and then S sending the predicted label for this entry by the federated model to T.

Although the collaborative computations are secured by the secret-sharing schemes we use, our protocol does involve revealing some values after each iteration, which would have compromised privacy. To combat any corrupt move based on the information attained from this part of the training, we reveal whether $\mathcal{L}_{prev} - \mathcal{L} < \epsilon$ follows or not, rather than revealing the value of $\mathcal{L}$ itself. This comparison can performed in a secure way on the shares itself in both our frameworks, without having to reveal the actual value. Moreover, we reveal the respective gradients of the parties only to them, thereby ensuring privacy.

Theorem 1: *Algorithm 1 using $\Pi^{\mathsf{ABY}}_{\mathsf{Online}}$ is information theoretically secure against a semi-honest adversary.*

Proof ABY guarantees that no additional information is revealed except for the outputs. The loss function being revealed after each iteration is masked and the gradients of each party are only revealed to that particular party, thereby maintaining privacy.

Theorem 2: *Algorithm 1 using $\Pi^{\mathsf{SPDZ}}_{\mathsf{Online}}$ for interactive calculation is information theoretically secure against a malicious adversary.*

Proof Given that the SPDZ protocol is secure against active adversaries in the two-party setting, the proof for active security follows exactly like that of Theorem 1.

V. Experiments

A. Setup

We run all experiments by simulating both parties on a single Intel i5 machine with 16 GB memory. The MP-SPDZ Library [14] is used to implement actively secure FTL training. The computation is conducted in a 64-bit prime field with statistical security parameter $\sigma = 40$. The offline phase of the protocol is based on LowGear version of Overdrive [17], which is the most efficient preprocessing protocol for two parties. The online version is based on SPDZ-2 [18]. The semi-honest version of FTL training based on Secret Sharing is implemented using ABY Framework[19]. The ring size was chosen to be 64 bits i.e. $l = 64$, symmetrical security parameter to be $\kappa = 128$ and $\sigma = 40$. Apart from this, the encrypted version introduced in [1] is emulated using FATE Framework [20].

B. Model Specifications

For our experiments related to scalability, we work on the Kaggle's Default-of-Credit-Card-Clients [21] is used. This dataset contains credit card records of the users with the user's default payment as labels. The dataset offers 33 features and 30,000 samples after applying one-hot encoding to categorical features. To simulate the federation setting, the dataset is split in both samples and feature space. Party S, simulating the source domain, is given all the labels. Each sample is assigned to party T or S or both, wherein the co-occurring samples are termed as overlapping samples. For realistic experimentation we derive from our bank-business company example in the first section, the demographic features such as age, education etc. are provided to party T emulating business company, and features relating to financial activity such as bill balance data, six months of payment etc. to party S emulating the bank.

Input : learning rate η, weight parameter γ, λ, max iteration m, tolerance ϵ
Output : Model parameters ω^S, ω^T

S, T initialize ω^S, ω^T
$iter = 0$;
while *iter* $\leq m$ **do**
  S do:
  $u_i^S \leftarrow Net^S(\omega^S, x_i^S)$ for $i \in \mathcal{D}_S$;
  T do:
  $u_i^T \leftarrow Net^T(\omega^T, x_i^T)$ for $i \in \mathcal{D}_T$;
  S, T do:
  Obtain $\langle \mathcal{L}_1^{ST} \rangle_k$ using $\Pi^{\mathsf{SS}}_{\mathsf{Online}}$, where k = {S, T} ;
  S, T do:
  Set $\langle \mathcal{L} \rangle_k = \mathcal{L}_1^k + \langle \mathcal{L}_1^{ST} \rangle_k$, where k = {S, T} ;
  for $l \leftarrow 0$ **to** l_S **by** 1 **do**
    S, T do:
    Obtain $\langle \mathcal{L}_2^{ST} \rangle_k$ using $\Pi^{\mathsf{SS}}_{\mathsf{Online}}$, where k = {S, T} ;
    S, T do:
    Set $\left\langle \frac{\partial \mathcal{L}}{\partial \omega_l^T} \right\rangle_S = \langle \mathcal{L}_2^{ST} \rangle_S$ and $\left\langle \frac{\partial \mathcal{L}}{\partial \omega_l^T} \right\rangle_T = \langle \mathcal{L}_2^{ST} \rangle_T + \mathcal{L}_2^T$. Output the value $\frac{\partial \mathcal{L}}{\partial \omega_l^T}$ to T ;
    T do:
    Update ω_l^T ;
  end
  for $l \leftarrow 0$ **to** l_T **by** 1 **do**
    S, T do:
    Obtain $\langle \mathcal{L}_3^{ST} \rangle_k$ using $\Pi^{\mathsf{SS}}_{\mathsf{Online}}$, where k = {S, T} ;
    S, T do:
    Set $\left\langle \frac{\partial \mathcal{L}}{\partial \omega_l^S} \right\rangle_T = \langle \mathcal{L}_3^{ST} \rangle_S$ and $\left\langle \frac{\partial \mathcal{L}}{\partial \omega_l^S} \right\rangle_S = \langle \mathcal{L}_3^{ST} \rangle_S + \mathcal{L}_3^S$. Output the value $\frac{\partial \mathcal{L}}{\partial \omega_l^S}$ to S ;
    T do:
    Update ω_l^S ;
  end
  Output the binary value $\mathcal{L}_{diff} = (\mathcal{L}_{prev} - \mathcal{L} < \epsilon)$ using $\Pi^{\mathsf{SS}}_{\mathsf{Online}}$;
  if $\mathcal{L}_{diff} == 1$ **then**
    Stop the protocol ;
  else
    Set $\langle \mathcal{L}_{prev} \rangle = \langle \mathcal{L} \rangle$;
    $iter \leftarrow iter + 1$;
  end
end

Algorithm 1: FTL Training

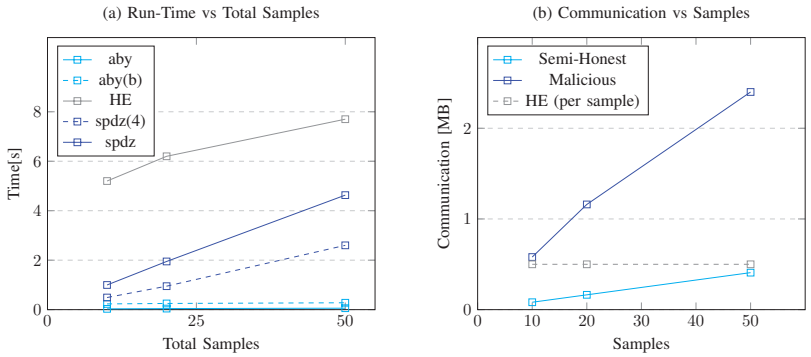

Fig. 3: Comparison in Performance

TABLE I: Online-Phase runtime for varying sample size (Malicious)

Process	Time[ms]				
	10	20	50	100	500
Initialize	5.3	8.7	19	44	205
Computation	6.2	6.9	16	29	332
Reveal	15	22	36	53	292

TABLE II: Runtime for varrying sample size (Semi-Honest)

Phase	Process	Time[ms]				
		10	20	50	100	500
Setup	BaseOT	190.94	190.1	194.7	201.4	196.3
Offline	OTExtension	38.18	55.7	106.1	182.4	780.1
Online	Computation	4.1	7.8	21.55	44.3	220.6

For FTL Training, tests were conducted on NUS-WIDE dataset [22], which consists of Flickr images with 634 low-level image features and their associated 1000 tag features and 81 ground truth labels. One-vs-all classification problem is considered with a data federation formed between party S and party T, where S has text tag features and image labels, and T has low-level image features. For training FTL, the three most frequently occurring labels in the NUS-WIDE dataset are picked i.e., water, person and sky. The training iterates until convergence or reaches the maximum iteration, i.e., 50, where $\gamma = 0.05$ and $\lambda = 0.005$. The local model we use for evaluation is a single layer stacked auto-encoder with domain invariant hidden layer representation $d = 32$.

C. Results and Analysis

Figure 3 compares the performance of different settings discussed for a single iteration, in terms of runtime and communication cost. Here *aby* and *aby(b)* stand for the runtime in semi-honest setting excluding and including baseOTs. Figure 4 further highlight results pertaining to the online SPDZ-based evaluation for a single iteration. All experiments were run on a single thread. Table III highlights the accuracy results comparing SS-based FTL (SST) with the ones of HE-based FTL with Taylor loss (TLT) and FTL with logistic loss (TLL) in [1]. Experimental results can be summarized as:

COMPARISON WITH HE-BASED IMPLEMENTATION:

– For 500 samples, the HE-based model takes around 35 seconds to evaluate where as our model takes 1.4 seconds in the malicious case and 0.8 seconds in the semi-honest case (excluding offline phase).
– Even after including the time for preprocessing, our model outperforms the HE-based model, as evident in Figure 3.(a)
– In contrast to the online communication cost for different sample sizes have been highlighted in Figure 3.(b), the HE-based model has a communication cost of about 0.50 MB per sample.
– For the offline phase in malicious setting, generation of each triple requires sending 13.71 kbit of data, where using a single thread 8856 triples can be generated per second. The performance can be drastically improved by increasing the number of threads. The performance with 4 threads is highlighted in Figure 3.(a) labelled as *spdz(4)*.
– The offline phase of the semi-honest version sends $(\ell + 1)(\kappa + \ell)/2$ to generate an ℓ bit multiplication triple.
– SS-based schemes can attain plain-text level accuracy whereas HE-based scheme incur a loss in accuracy, thus our model shows more accurate results on the NUS-WIDE dataset, as evident from Table III.

SECRET SHARING BASED EVALUATION:

– Table I and II highlight that the runtime increases linearly with the increase in samples, since the number of inner products involved in the online phase is proportional to $(aN_{ST} + bN_L)$, where a and b are two constants.
– Revealing outputs is a bottleneck for small sample size in case of the SPDZ-based online evaluation (Table I). Since all the online communication stems from the multiplication gates evaluated, and the loss and gradient values revealed; for small sample sizes both the costs are comparable but as on increasing sample size the cost of evaluating multiplication gates shoots up and surpasses the cost of revealing.
– Figure 4.(a) suggests that on increasing features and overlapping samples, rate of increase in runtime becomes narrow i.e. training time will converge if we keep increasing features or overlapping samples. Figure 4.(b) highlights the quadratic growth in runtime of with respect to hidden layer dimension.

SCALABLITY SPDZ:

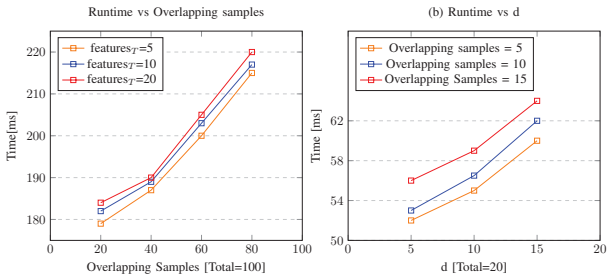

Fig. 4: Scalability in Malicious Setting

– We focus on a two-party case which is a highly favourable setting since instances of a two-party interaction are extensively available on the Internet in the form of client-server based protocols. Since this model requires domain-independent common representation layer, it is only applicable to a subset of transfer mechanisms. But the techniques use by us are flexible enough to implement any other model, and can be efficiently extended to arbitrary number of parties while guaranteeing strong security.
– Preprocessing throughput can further be increased upto 100000 triples per second by increasing the number of threads used to 16.
– Merging communication for multiple operations in a single round is the major reason behind the improvements achieved by this setting. Vectorizing more operations and increasing threads can lead to improvements in performance.

Tasks	N_L	SST	TLT	TLL
water vs. others	100	$\mathbf{0.698} \pm 0.011$	0.692 ± 0.062	0.691 ± 0.060
water vs. others	200	$\mathbf{0.707} \pm 0.013$	0.702 ± 0.010	0.701 ± 0.007
person vs. others	100	$\mathbf{0.703} \pm 0.015$	0.697 ± 0.010	0.697 ± 0.020
person vs. others	200	$\mathbf{0.735} \pm 0.004$	0.733 ± 0.009	$\mathbf{0.735} \pm 0.010$
sky vs. others	100	0.708 ± 0.015	0.700 ± 0.022	$\mathbf{0.713} \pm 0.006$
sky vs. others	200	$\mathbf{0.724} \pm 0.014$	0.718 ± 0.033	0.718 ± 0.024

TABLE III: Comparison of weighted F1 scores.

As compared to other privacy preserving work on ML, secure FTL training highly benefits due to local evaluation of the neural network, as:

1) Due to the restrictions on division and exponentiation, many MPC-based solutions require approximation of the non-linear activation function for each layer of the model in each iteration, leading to loss of accuracy. For instance, SecureML incurs a 1% loss in the evaluation of a 2 layer Fully-Connected Neural Network with 128 neurons in each layer. Since in our case, the model evaluation is local and approximations are made for evaluation of a comparatively low depth circuit once after each iteration, higher level of accuracy is observed.
2) ML platforms like Tensorflow can be used for the evaluation. In contrast to this, most MPC-based solutions for secure self learning models, including use of MP-SPDZ Framework, are not compatible with such ML platforms.
3) Security needs to be adapted for low depth circuits of gradients and loss calculation, yielding highly practical implementations even with malicious security.

VI. Conclusion

In this paper, we establish the practicality and scalability of secure FTL for both semi-honest and malicious setting. Our techniques for training bring multifold improvement in the runtime and communication cost as compared to the previous on HE-based approach. Given the lack of actively secure machine learning protocols, we hope this paper would pave the way for future works that guarantee strong security.

A. Future Work

For the actively secure case, in contrast to our work over fields i.e. modulo prime, another direction can be an implementation over rings of the form 2^k using SPDZ-2k protocol [23]. Such an setting can leverage the efficiency of native operations in a 32-bit/64-bit standard CPU to show efficient results.

While the security we introduce for collaborative calculation of gradients will suffice for many real-life scenarios, some applications might require a stronger security guarantee. Since trained model parameters are a function of training data, in some cases they can reveal private information about the datasets involved in training [24]. For security against such attacks, our techniques can be merged with secure aggregation techniques introduced in [25]. As secure aggregation involves revealing the aggregation of trained parameters from multiple parties for updation of a shared global model, SPDZ remains a sound option for a FTL model involving more than two parties and secure aggregation, because of its efficient scalability. The online phase of SPDZ scales linearly as the number of participants are increased, thus leading to an even more robust yet practical system. Apart from this, guarding of revealed gradients in the current model with differential privacy [26] can be a considered as a solution.

References

[1] Y. Liu, T. Chen, and Q. Yang, "Secure federated transfer learning," *CoRR*, vol. abs/1812.03337, 2018, http://arxiv.org/abs/1812.03337.
[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., 2012, pp. 1097–1105.
[3] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–503, 2016, http://www.nature.com/nature/journal/v529/n7587/full/nature16961.html.
[4] H. B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas, "Federated learning of deep networks using model averaging," *ArXiv*, vol. abs/1602.05629, 2016.
[5] A. Gascón, P. Schoppmann, B. Balle, M. Raykova, J. Doerner, S. Zahur, and D. Evans, "Secure linear regression on vertically partitioned datasets," *IACR Cryptology ePrint Archive*, vol. 2016, p. 892, 2016.
[6] R. Nock, S. Hardy, W. Henecka, H. Ivey-Law, G. Patrini, G. Smith, and B. Thorne, "Entity resolution and federated learning get a federated resolution," *CoRR*, vol. abs/1803.04035, 2018. [Online]. Available: http://arxiv.org/abs/1803.04035

[7] I. Damgård, V. Pastro, N. Smart, and S. Zakarias, "Multiparty Computation from Somewhat Homomorphic Encryption," in *Advances in Cryptology – CRYPTO 2012*, R. Safavi-Naini and R. Canetti, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 643–662.
[8] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, , and J. Wernsing, "CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ser. ICML'16. JMLR.org, 2016, pp. 201–210.
[9] P. Mohassel and Y. Zhang, "SecureML: A System for Scalable Privacy-Preserving Machine Learning," *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 19–38, May 2017.
[10] W. Zheng, R. A. Popa, J. E. Gonzalez, and I. Stoica, "Helen: Maliciously Secure Coopetitive Learning for Linear Models," 2019.
[11] V. Chen, V. Pastro, and M. Raykova, "Secure Computation for Machine Learning With SPDZ," 2019.
[12] D. Beaver, "Efficient multiparty protocols using circuit randomization," in *Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology*, ser. CRYPTO '91. London, UK, UK: Springer-Verlag, 1992, pp. 420–432.
[13] D. Demmler, T. Schneider, and M. Zohner, "ABY - A Framework for Efficient Mixed-Protocol Secure Two-Party Computation," in *NDSS*, 2015.
[14] N1 Analytics. MP-SPDZ - Versatile framework for multi-party computation. https://github.com/n1analytics/MP-SPDZ.
[15] G. Asharov, Y. Lindell, T. Schneider, and M. Zohner, "More efficient oblivious transfer and extensions for faster secure computation," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, ser. CCS '13. New York, NY, USA: ACM, 2013, pp. 535–548. [Online]. Available: http://doi.acm.org/10.1145/2508859.2516738
[16] X. Shu, G.-J. Qi, J. Tang, and J. Wang, "Weakly-shared deep transfer networks for heterogeneous-domain knowledge propagation," 10 2015, pp. 35–44.
[17] M. Keller, V. Pastro, and D. Rotaru, "Overdrive: Making SPDZ Great Again," in *Advances in Cryptology – EUROCRYPT 2018*, J. B. Nielsen and V. Rijmen, Eds. Cham: Springer International Publishing, 2018, pp. 158–189.
[18] I. Damgård, M. Keller, E. Larraia, V. Pastro, P. Scholl, and N. P. Smart, "Practical Covertly Secure MPC for Dishonest Majority – Or: Breaking the SPDZ Limits," in *Computer Security – ESORICS 2013*, J. Crampton, S. Jajodia, and K. Mayes, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 1–18.
[19] encryptogroup. ABY - A Framework for Efficient Mixed-protocol Secure Two-party Computation. https://github.com/encryptogroup/ABY.
[20] WeBank. FATE - Federated AI Technology Enabler. https://github.com/WeBankFinTech/FATE.
[21] Kaggle. (2019) Default of credit card clients dataset. https://www.kaggle.com/uciml/default-of-credit-card-clients-dataset.
[22] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y.-T. Zheng, "NUS-WIDE: a real-world web image database from national university of singapore," in *Proc. of ACM Conf. on Image and Video Retrieval (CIVR'09)*, 2009.
[23] R. Cramer, I. Damgård, D. Escudero, P. Scholl, and C. Xing, "SPDZ2k: efficient MPC mod 2k for dishonest majority," in *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part II*, 2018, pp. 769–798.
[24] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE Symposium on Security and Privacy (SP)*, May 2017, pp. 3–18.
[25] A. Segal, A. Marcedone, B. Kreuter, D. Ramage, H. B. McMahan, K. Seth, K. Bonawitz, S. Patel, and V. Ivanov, "Practical secure aggregation for privacy-preserving machine learning," in *CCS*, 2017.
[26] A. Rajkumar and S. Agarwal, "A differentially private stochastic gradient descent algorithm for multiparty classification," in *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, N. D. Lawrence and M. Girolami, Eds., vol. 22. La Palma, Canary Islands: PMLR, 2012, pp. 933–941.