Privacy-Enhanced Knowledge Transfer with Collaborative Split Learning over Teacher Ensembles

Ziyao Liu
Nanyang Technological University
Singapore
ziyao002@e.ntu.edu.sg

Jiale Guo
Nanyang Technological University
Singapore
jiale.guo@ntu.edu.sg

Mengmeng Yang
Data61, CSIRO
Australia
mengmeng.yang@data61.csiro.au

Wenzhuo Yang
Nanyang Technological University
Singapore
wenzhuo001@e.ntu.edu.sg

Jiani Fan
Nanyang Technological University
Singapore
jiani001@e.ntu.edu.sg

Kwok-Yan Lam
Nanyang Technological University
Singapore
kwokyan.lam@ntu.edu.sg

ABSTRACT

Knowledge Transfer has received much attention for its ability to transfer knowledge, rather than data, from one application task to another. In order to comply with the stringent data privacy regulations, privacy-preserving knowledge transfer is highly desirable. The Private Aggregation of Teacher Ensembles (PATE) scheme is one promising approach to address this privacy concern while supporting knowledge transfer from an ensemble of "teacher" models to a "student" model under the coordination of an aggregator. To further protect the data privacy of the student node, the privacy-enhanced version of PATE makes use of cryptographic techniques at the expense of heavy computation overheads at the teacher nodes. However, this inevitably hinders the adoption of knowledge transfer due to the highly disparate computational capability of teachers. Besides, in real-life systems, participating teachers may drop out of the system at any time, which causes new security risks for adopted cryptographic building blocks. Thus, it is desirable to devise privacy-enhanced knowledge transfer that can run on teacher nodes with relatively fewer computational resources and can preserve privacy with dropped teacher nodes. In this connection, we propose a dropout-resilient and privacy-enhanced knowledge transfer scheme, Collaborative Split learning over Teacher Ensembles (CSTE), that supports the participating teacher nodes to train and infer their local models using split learning. CSTE not only allows the compute-intensive processing to be performed at a split learning server, but also protects the data privacy of teacher nodes from collusion between the student node and aggregator. Experimental results showed that CSTE achieves significant efficiency improvement from existing schemes.

CCS CONCEPTS

• **Security and privacy → Security protocols**.

SecTL '23, July 10–14, 2023, Melbourne, VIC, Australia

ACM ISBN 979-8-4007-0181-8/23/07.
https://doi.org/10.1145/3591197.3591303

KEYWORDS

Knowledge transfer, privacy-preservation, collaborative split learning

ACM Reference Format:
Ziyao Liu, Jiale Guo, Mengmeng Yang, Wenzhuo Yang, Jiani Fan, and Kwok-Yan Lam. 2023. Privacy-Enhanced Knowledge Transfer with Collaborative Split Learning over Teacher Ensembles. In *Proceedings of the 2023 Secure and Trustworthy Deep Learning Systems Workshop (SecTL '23), July 10–14, 2023, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3591197.3591303

1 INTRODUCTION

With the increasing concerns for personal data privacy protection, knowledge transfer has received much attention of the academic community for its ability to transfer knowledge, rather than data, from one application task to another. Knowledge transfer is especially suitable for supporting the machine learning based intelligent processing of multiple organizations due to, firstly, data sovereignty regulations and, secondly, the disparate computing capacities with a varying scale of operations. The teacher-student scheme is a common way to realize knowledge transfer where an organization unit (the student) may learn its model parameters from knowledge of other organization units (the teachers) who have trained its model from their own corpus, hence knowledge of the teachers can be transferred to the student without sharing access to the data.

In order to comply with the stringent data privacy regulations, privacy-preserving knowledge transfer (PPKT) is highly desirable. PPKT allows a student, say a small organization unit with a locally trained model based on the limited data resource of its local base, to learn and fuse the knowledge of the trained models of several teachers, say partnering organization units with models trained using the huge data bases, while complying with the data privacy and data sovereignty regulations of the teachers. As a result, the student is able to absorb the knowledge from teachers to improve the generality of its model, and thus the performance of its application operations.

One promising approach to achieve PPKT is *Private Aggregation of Teacher Ensembles* (PATE) [32] which is designed to transfer the knowledge from an ensemble of "teacher" models to a "student" model. In PATE-based PPKT applications, as illustrated in Figure 1, the student, who is faced with the challenge of insufficient labeled data for supervised model training, poses a query to the teachers

for the label of its training data. The teachers in turn compute the predicted labels using their own well-trained models, and return the prediction results as their votes. Thereafter, the teachers jointly compute the label for the student with the most votes through the coordination of an aggregator, who then sends that answer to the student. Being able to learn from the teachers through this process, the student will be able to obtain sufficient labeled data, instead of labeling them locally, to improve the accuracy of its local model.

Subsequent works based on PATE adopted various differential privacy (DP) schemes [9, 33, 41] to achieve better privacy protection for teachers' models. Other works adopted different query methods [42, 47], different aggregation protocols [41, 45], or integrated with data synthesizing techniques, such as generative adversarial network (GAN) [2, 25, 29, 46] and auto-encoder [14, 43, 45], in order to improve the model accuracy. However, none of these PATE-based schemes protect the student's data privacy. Instead, it only provides a weak privacy guarantee on teachers' models by assuming that the aggregator is an honest participant. In this connection, cryptographic techniques such as homomorphic encryption (HE) [20, 26] and secure multiple-party computation (MPC) [17] have been proposed to enhance the privacy of PATE.

For instance, HE is used in CaPC [16] to protect the privacy of the student's query by encrypting the query before sending it to the teacher such that teachers can make their predictions over ciphertext using HE operations. In SPEED [35], HE is adopted to protect the privacy of teachers' predictions during vote aggregation by the aggregator. Secure two-party computation techniques such as garbled circuits (GC) [44] and secret sharing [17] are applied in CaPC [16] and PRICURE [24] to enable privacy-preserving voting and prediction. Nevertheless, the aforementioned works vary in their privacy assumptions. PRICURE leverages two assistant servers. SPEED assumes queries are public. Besides, both of the aforementioned works are secure against only non-colluding adversaries, which are impractical for real-life applications. A simple case is that the aggregator can collude with the student to access both teacher's predictions and student's queries, and thus able to launch a membership inference attack [37] to invade teachers' data privacy.

Albeit addressing the privacy concerns, the privacy-enhanced version of PATE uses homomorphic encryption at the expense of heavy computation overheads at the teacher nodes, which could hinder the adoption of privacy-enhanced PATE. For example, teachers in CaPC have to evaluate the whole ML model over ciphertext using FHE, which causes unaffordable computation costs for complex ML algorithms. Such a task is overwhelmingly difficult even over plaintext when training a large-scale neural network (NN) on nodes with fewer computing capabilities. Moreover, all the aforementioned schemes were based on security models that did not consider the possibility of dropped users potentially caused by insider fraud, which is an important consideration in FinTech. For example, CaPC adopts 2-party computation techniques between the student and teachers, which requires the student to be online during the whole protocol execution [7]. (see Table 1 for a comparison between the protocols as mentioned above).

In this paper, we proposed to integrate split learning (SL) [23, 40] with a set of cryptographic tools in order to reduce the heavy computation overheads at the teacher nodes while addressing the privacy concerns. In specific, SL is proposed to address the shortage of computational resources while preserving data privacy, and has been widely adopted in many resource-constrained and privacy-concerned areas [3, 4, 38]. In split learning, the NN model is split into three portions where the small-scale first and last portion are stored on the client-side while the large-scale middle portion is stored on the server-side. The SL training is done by a sequence of training processes where the output of different portions is transmitted between the client (Alice) and the server (Bob) (see Figure 2 for an illustrative SL scheme). Therefore, SL enables NN training for clients having low computational power where the client trains only the first and last few layers of the split NN model. Note that the transmitted intermediate results between the client and the SL server leak only negligible information as it is challenging to infer any information from them if the NN is split appropriately (see [23] for the detailed discussion on security). Furthermore, we consider a setting with both dropped users and semi-honest behaviors, i.e., users that do not deviate from the protocol but try to infer the honest parties' information. To summarize, our contributions are the following:

- We propose CSTE, a collaborative split learning scheme for privacy-preserving knowledge transfer based on teacher ensembles, that can run on teacher nodes with less computing capabilities.
- CSTE improves the robustness of previous works. Specifically, CSTE not only additionally protects the data privacy of teacher nodes from collusion between the student node and the aggregator, but also provides resilience to dropped users.
- Experimental results showed that CSTE achieves a significant efficiency improvement from the state-of-the-art privacy-enhanced PATE-based scheme.

The rest of the paper is organized as follows. In Section 2, we provide a brief introduction to the building blocks. Then we describe our proposed CSTE protocol with the security and privacy analysis in Section 3. We present our experimental evaluation in Section 4, and give conclusions in Section 5.

2 PRELIMINARIES

2.1 Split Learning

Split learning (SL) [23, 40] is proposed to address the shortage of computational resources and labeled data, and has been widely adopted in many resource-constrained and privacy-concerned areas [3, 4, 38]. SL requires collaboration among a set of untrusted parties, and thus privacy is also one of the concerns for SL. In split learning, the neural network (NN) model is split into two or three portions where the small-scale portions are stored on the client-side while the large-scale portion is stored on the server-side. The SL training is done by a sequence of training processes between the client (Alice) and the server (Bob) (see Figure 3 and Figure 2 for the illustrative SL scheme). Therefore, SL enables NN training with clients having low computational power since the client trains only a few layers of the split NN model.

As shown in Figure 3, Alice sends the label of its data to Bob for training purposes. The NN model is split into two parts: the first few layers are stored on the Alice-side, and the rest of the layers

Table 1: Comparison between PATE[32], SPEED[35], PRICURE[24], CaPC[16] and CSTE. Note that all the protocols listed in this table are based on the PATE teacher-student style. In specific, collusion means that the adversary can corrupt either the server or aggregator, the student, a set of teachers, or both. Query privacy is whether the student's queries are public or not. Prediction privacy is whether others can access the predictions over plaintext made by the teachers. The computation and communication overheads are evaluated from the view of the client.

Protocol	PATE[32]	SPEED[35]	PRICURE[24]	CaPC[16]	CSTE
Security against collusion	No	No	No	No	Yes
Query privacy	No	No	Yes	Yes	Yes
Prediction privacy	No	Yes	No	Yes	Yes
Computation overheads	Low	Low	Low	High	Medium
Communication overheads	Low	Low	Low	High	Medium
Dropout-resilience	No	No	Yes	No	Yes

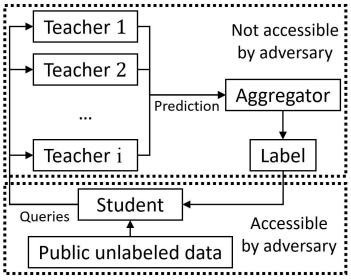


Figure 1: Private Aggregation of Teacher Ensembles (PATE)

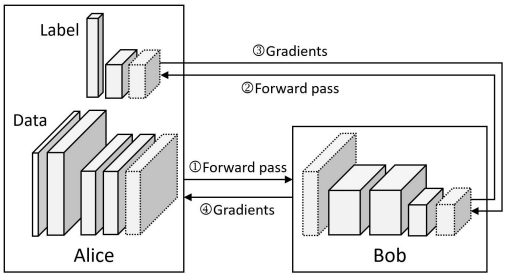


Figure 2: SL training without label sharing

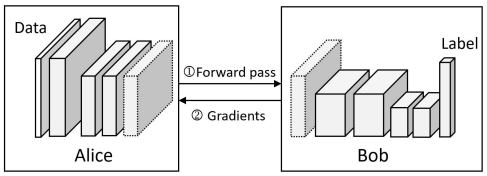


Figure 3: SL training with label propagation.

are stored on the Bob-side. In Figure 2, Alice keeps both data and label. This requires the NN model to be split into three parts: the first and last few layers are stored on the Alice-side, and the rest of the middle layers are stored on the Bob-side. Although Alice's privacy is further protected without label propagation, one more round of communication is needed, which increases the overall overheads. In CSTE, we use the SL scheme without label sharing to fully protect the privacy of SL clients.

Privacy guarantees. As mentioned earlier, the transmitted intermediate results between the client and the server leak only negligible information as it is challenging to infer any information from them if the SL NN is split appropriately. Next, we describe the intuition adapted from [23] to address such privacy concerns. Let Alice's layer M be a fully connected layer containing N outputs to be transmitted to Bob, then layer M has at least $N!$ functionally equivalent configurations. To infer the data information, Bob has to go through all $N! > (\frac{N}{2})^N > e^N$ possible configurations. For example, if the fully connected layer M contains 128 neurons and each configuration could be tested in 0.01 second, it would take Bob more than 10^{206} years to figure out parameters used by Alice. In this case, if the SL NN model is appropriately split, data privacy can be guaranteed.

2.2 Fully Homomorphic Encryption

Fully homomorphic encryption (FHE) [20] allows a worker to receive encrypted data and perform arbitrarily chosen computations on that data while it remains encrypted, without having the secret key.

Definition 1 (Fully homomorphic encryption scheme). *Let (P, C, K, E, D) be an encryption scheme where P and C are the plaintext and ciphertext spaces respectively. E is the encryption algorithm while D is the corresponding decryption algorithm. The encryption key k which is either a secret key (in the secret key cryptosystem) or a public key (in the public-key cryptosystem) is sampled from the key space K. In specific, the encryption algorithm E is a map from the ring $(P, \oplus_p, \otimes_p)$ to the ring $(C, \oplus_c, \otimes_c)$ such that $E_k : P \rightarrow C$. If for all $a, b \in P$ and $k \in K$, we have*

$$E_k(a) \oplus_c E_k(b) = E_k(a \oplus_p b)$$

$$E_k(a) \otimes_c E_k(b) = E_k(a \otimes_p b)$$

The encryption scheme is said to be fully homomorphic.

However, FHE involves impractical computation overheads to support any depth of function (the number of multiplications involved to compute the function) using expensive bootstrapping techniques. Thus, deploying FHE on resource-constrained devices for deep neural network prediction is not practical so far. Luckily, if the number of multiplications on the ciphertext is bounded, a weaker version, i.e., leveled FHE [11], can be used to achieve very

efficient computation over ciphertext by setting proper security parameters. Note that the lower the bound, the more efficient computation over ciphertext. In CSTE, we use the leveled homomorphic encryption scheme CKKS [15] implemented in SEAL library [34] to perform the neural network prediction. The packing technique Single-Instruction-Multiple-Data (SIMD) operation is adopted to improve the overall throughput of computation over ciphertext. Supplementary details of CKKS scheme can be found in the appendix.

2.3 Shamir Secret Sharing

A t-out-of-n or (t, n) Shamir secret sharing (SSS) scheme [36] divides a secret S into n data pieces called shares such that the secret S can be reconstructed by any combination of t shares. Besides, there is no information leakage of the secret S by any set of shares of which the number is less than t. Shamir scheme is widely used to handle dropped users in privacy-preserving aggregations [6, 10, 22]. In specific, for a standard (t, n) Shamir secret sharing scheme, the secret S and shares $S_1, \ldots, S_n$ are elements over a finite field $\mathbb{Z}_p$ for some prime p where $0 < t \leq n < p$ and $s < p$. The intuitive idea of Shamir's secret sharing scheme is to hide the secret in a polynomial $f(x)$ that $S = f(0)$. Since a line can be defined by 2 points, a parabola can be defined by 3 points, a cubic curve can be defined by 4 points, and so forth. A polynomial of a degree of t can be defined by $t - 1$ points, and thus can be used to construct a (t, n) secret sharing scheme. Assume that there is a party u_s that holds a secret S and wishes to secretly share S to a set of n parties $\{u_1, \ldots, u_n\}$. The Shamir secret sharing scheme works as follows:

- Preparation: The secret holder u_s randomly samples $t - 1$ coefficients $a_1, \ldots, a_{t-1}$ from $\mathbb{Z}_P$ to construct a polynomial with $t - 1$ degree $f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \cdots + a_{t-1} x^{t-1} \bmod p$.
- Secret sharing: The secret holder u_s randomly selects n points $x_1, \ldots, x_n$ to compute the apir $\{x_i, f(x_i)\}$ for $i \in [1, n]$, and sends $\{x_i, f(x_i)\}$ to the party u_i.
- Secret reconstruction: The secret holder u_s collects a set of $\{x_i, f(x_i)\}$ of which the number is greater than the threshold t to calculate the coefficients $a_0, \ldots, a_{t-1}$ of the polynomial $f(x)$. Such calculation follows Lagrange interpolation that the constant term a_0 would be the secret S. A more efficient method to directly reconstruct the secret is to compute $S = a_0 = \sum_{j=0}^{t} f(x_j) \prod_{m=0, m\neq j}^{t} \frac{x_m}{x_m - x_j} \bmod p$.

In CSTE, the noised votes, i.e., teachers' predicted labels, are secretly shared to provide both the privacy guarantee and dropout resilience. For the simplicity of notation, when the context is clear, we abuse the notation $SS(x)$ to denote that x is secretly shared using (t, n) Shamir scheme.

2.4 Secure Multiple Party Computation

Secure Multiple Party Computation (MPC) [17] allows several parties to jointly evaluate a function based on their secretly shared data, e.g., the Shamir shares as introduced in Section 2.3, while preserving the data privacy of each party. In CSTE, MPC is only used for all the teachers to jointly compute the label with the most votes. In specific, such MPC protocol involves secure arithmetic operations and Argmax constructed by secure comparison. We will describe its standard construction in the appendix.

2.5 Differential Privacy

Differential privacy is a provable privacy definition, which is proposed by Dwork et al.[18]. It provides strong privacy that a single data record in or out of the dataset makes no big difference to the final statistics. The formal definition is shown as follows.

Definition 2 ((ϵ, δ)-Differential Privacy). *A random algorithm $\mathcal{M}$ is (ϵ, δ)-differentially private iff for every set of outputs $O \subseteq \mathcal{M}(\mathcal{D})$ and for all neighbouring data set D and $D' \in \mathcal{D}$, we have*

$$Pr[\mathcal{M}(D) \in O] \leq e^{\epsilon} Pr[\mathcal{M}(D') \in O] + \delta. \quad (1)$$

The definition can be achieved by adding Gaussian noise with calibrated variance (See the appendix for more details).

3 THE CSTE PROTOCOL

3.1 Problem Description

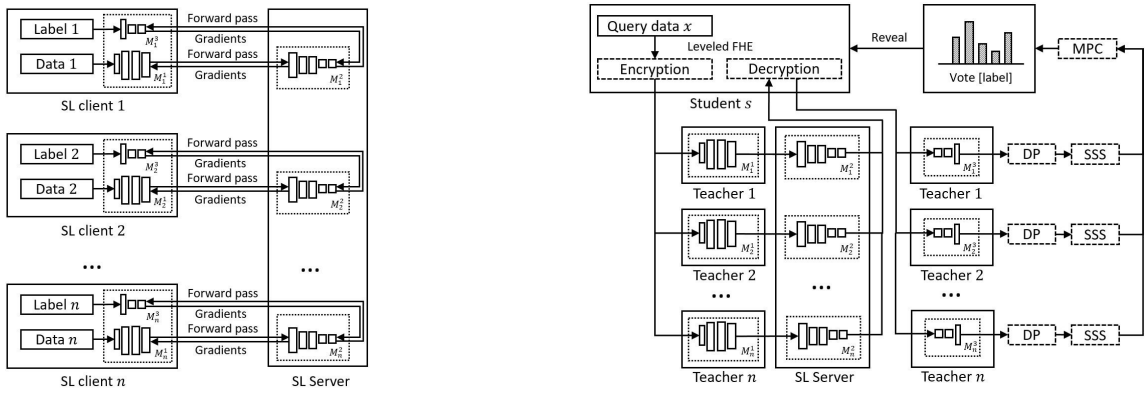
The participants of CSTE are divided into two classes: a central server $\mathcal{S}$ and a set of n resource-constrained clients $\mathcal{U} = \{u_1, \ldots, u_n\}$. Each client u_i holds a private dataset $\mathcal{D}_i = \{x_i, y_i\}$ where vector x_i is the data and vector y_i is the corresponding label, and wishes to:

- Fit a predictive ML model $\mathcal{M}_i$ to its dataset $\mathcal{D}_i$.
- Improve the utility of its model $\mathcal{M}_i$ via collaboration with other clients following the PATE teacher-student style. In specific, the students or the querying party, say u_s, poses queries in the form of data samples x and all the other participants (teachers) $\mathcal{U} \setminus \{u_s\}$ jointly compute to provide the predictions on the x as $\hat{y}$ to the student u_s, which can then be used to further train the student's model $\mathcal{M}_s$ to improve its utility.
- Preserve the privacy of the dataset $\mathcal{D}_i$, the student's queries x, trained model $\mathcal{M}_i$, and the teacher's prediction as the output of the model $\mathcal{M}_i$.
- Achieve the goals mentioned above with affordable overheads.

Threat Model. We assume an adversary that corrupts either the server, the student, and a set of teachers or both. For instance, the server and the student can collude to infer the teacher's information. The worst case is that both the server, the student, and a set of teachers collude to invade the rest of the teachers' privacy using the membership inference attack. CSTE works in a semi-honest threat model where the adversary does not deviate from the protocol but tries to infer the honest parties' information.

3.2 High-level Overview

Intuitively, the CSTE protocol adopts a semi-honest central SL server and uses split learning to reduce the client-side computation overheads. Therefore, CSTE includes two phases, as shown in Figure 4, which consists of (i) SL training, and (ii) teacher-student knowledge transfer. Note that each SL client u_i's model $\mathcal{M}_i$ is split into three consecutive parts $\mathcal{M}_i^1$, $\mathcal{M}_i^2$, and $\mathcal{M}_i^3$ that $\mathcal{M}_i^1$ and $\mathcal{M}_i^3$ are stored on the SL client i, $\mathcal{M}_i^2$ is stored on the SL server.



(a) SL training

(b) Teacher-student knowledge transfer

Figure 4: Collaborative Split Learning (CSTE) Protocol

During the SL training phase, each client u_i trains the model $\mathcal{M}_i$ using its own dataset $\mathcal{D}_i$ respectively. During the phase of teacher-student knowledge transfer, some SL clients in the SL training phase perform as teachers to transfer their knowledge to some other SL clients that query for the labels as students. The key features of CSTE are summarized as follows:

- To protect the student's data privacy, the student's queries are encrypted such that teachers make their predictions over ciphertext using HE operations.
- To reduce the computation overheads of teachers' prediction, the SL scheme is adopted.
- To improve the differential privacy considering dropped users, adapted DP noises are added to teachers' predictions before voting.
- To handle the dropped teachers and avoid the information leakage of the voting results, SSS and MPC are used to securely compute the final label as the answer to the student's query.

3.3 Detailed Description

Our protocol enables a set of n clients $\mathcal{U} = \{u_1, \ldots, u_n\}$ to collaboratively improve the performance of their ML models under the coordination of the central server. We focus on the m-class classification task. The proposed CSTE protocol consists of the following steps:

(1) The clients make SL training respectively cooperating with the SL server.
  (a) Each client u_i and the server initiate the seperate SL models of $\mathcal{M}_i$, i.e., $\mathcal{M}_i^1$, $\mathcal{M}_i^2$, and $\mathcal{M}_i^3$.
  (b) Each client u_i uses the dataset $\mathcal{D}_i$ to train its model $\mathcal{M}_i$ in cooperation with the server following SL scheme.
(2) The teachers make the predictions on student's query respectively.
  (a) The student u_s generates the key-pair (pk, sk), encrypts its query x using pk, and then sends $\text{Enc}(x)$ and the public key pk to the teachers via the SL server.
  (b) Each teacher i takes $\text{Enc}(x)$ as input and computes the output of $\mathcal{M}_i^1$ as $\text{Enc}(x_i')$ using FHE operations, then sends $\text{Enc}(x_i')$ to the SL server.
  (c) The SL server takes $\text{Enc}(x_i')$ as inputs and computes the output of $\mathcal{M}_i^2$ as $\text{Enc}(x_i'')$ using FHE operations, then sends $\text{Enc}(x_i'')$ to the student u_s.
  (d) The student u_s decrypts $\text{Enc}(x_i'')$ using its secret key sk, and then sends x_i'' to the teacher i via the SL server.
  (e) Each teacher i takes x_i'' as inputs and computes the output of $\mathcal{M}_i^3$ as a vector $z_i = [z_1, z_2, \ldots, z_m]$.
(3) The teachers jointly vote on the label in a privacy-preserving manner.
  (a) Each teacher i adds DP noise to z_i to get the vector $\hat{z}_i = [\hat{z}_1, \hat{z}_2, \ldots, \hat{z}_m] = z_i + \text{DP noise}$. Given teachers' number n, assume there are at least $p_l n, 0 < p_l \leq 1$ teachers online for each query, we add noise vector $\boldsymbol{\eta}$ as the DP noise with each element $\eta_i \sim \mathcal{N}(0, \frac{1}{p_l n}\sigma^2), i = 1, \cdots, m$ to z_i, which ensures the aggregated result is differentially private.
  (b) Each teacher i shares $\hat{z}_i$ to the rest of teachers $\mathcal{U} \setminus \{u_s, u_i\}$ using SSS, denoted as $SS(\hat{z}_i, t)$ where t is the threshold.
  (c) All the teachers jointly compute the voting histogram $SS(\hat{z}_1^*, \hat{z}_2^*, \ldots, \hat{z}_m^*) = \sum SS(\hat{z}_i)$ using MPC.
  (d) All the teachers jointly compute the label with the most votes $SS(\hat{y}) = SS(\text{argmax}_k(\hat{z}_k))$ using MPC (the detailed MPC protocol to evaluate argmax function is described in the appendix). After that, each teacher sends the share of $\hat{y}$ to the server.
  (e) After receiving threshold t shares of $\hat{y}$ from teachers, the server reconstructs $\hat{y}$ and sends it to the student.
(4) The student uses query-answer pairs $(x, \hat{y})$ plus $\mathcal{D}_i$ to retrain its model $\mathcal{M}_s$.

3.4 Security and Privacy Analysis

Security Analysis. For the SL training phase (similar to the SL prediction in the teacher-student knowledge transfer phase), since each client u_i trains the model $\mathcal{M}_i$ using its own dataset $\mathcal{D}_i$ respectively, according to the security property of SL, the SL server learns

nothing from the protocol execution beyond the intermediate results and $\mathcal{M}_i^2$. Thus the privacy of the dataset $\mathcal{D}_i$ can be guaranteed. For the teacher-student knowledge transfer phase, CSTE is secure against a semi-honest adversary that corrupts either the server, the student, and a set of teachers or both. The proof can be intuitively derived based on the security of the underlay building blocks, including fully homomorphic encryption, Shamir secret sharing, and secure multiple-party computation (see Section 2 for the details).

Differential Privacy Analysis. To prevent the student from inferring the teachers' sensitive information by observing the predictions, we follow the improved PATE protocol [33] that adds Gaussian noise to the label counts of the histogram with a sensitivity of 1 before outputting the final vote result. Instead of adding noise to the histogram directly, we let each teacher add part of the noise separately and pass the votes through secret sharing to the aggregator, which relaxed the security assumption that the aggregator does not need to be fully trusted. Specifically, to add the amount of noise with variance σ^2 to final aggregation, we let each teacher sample and add noise $\eta \sim \mathcal{N}(0, \frac{1}{p_l n}\sigma^2)$. Then the aggregation of the noise $\sum_{i=1}^{p_l n} \eta_i \sim \mathcal{N}(0, \sigma^2)$, which equivalently adding noise with variance σ^2 to the histogram direclty. Therefore, the differential privacy analysis of improved PATE [33] applies to our protocol. Specifically, they utilize moments accountant and RDP techniques to provide a tighter estimation of the privacy loss and further improved the bound by considering the consensus of teacher models. More details can be found in Appendix 1.5.

4 EXPERIMENTS

System setting. Our prototype is tested over two Linux workstations with an Intel Xeon E5-2603v4 CPU (1.70GHz) and 64 GB of RAM, running CentOS 7, with a 0.207 ms average network latency and 1.25 GB/s average bandwidth. The central (SL) server runs in multi-thread on one workstation, while the clients are running in parallel in the other workstation. In our experiments, original data is represented in fixed-point with 32 bits, including 6 bits for the integer part and 26 bits for the fractional part. We use leveled HE scheme CKKS implemented in SEAL library [34] with 128-bit security level for neural network inference, and standard Shamir secret sharing implemented in MP-SPDZ [27] for generic multi-party computation including sum and argmax for voting. Detailed settings regarding cryptographic primitives are given in the additional experimental results.

Collaboration setting. To evaluate our proposed scheme, we first train the local model $\mathcal{M}_i$ for each client u_i using its local dataset $\mathcal{D}_i$. Then by executing CSTE, students pose their queries to teachers for the answers as labels. After that, students retrain their local models using $\mathcal{D}_i$ plus new samples, i.e., query-answer pairs. The difference in accuracies are the improvements brought by the newly labeled data, and thus CSTE.

To simulate the environment for an n-client collaborative learning that each client keeps a local model $\mathcal{M}_i$ trained using its local dataset $\mathcal{D}_i$, we uniformly sample n disjoint dataset from the training set to create $\mathcal{D}_i$ without replacement, and the total number of clients is set to be 300. Therefore, for MNIST [28] with 60000 training samples, each client has 200 samples for its local training. For SVHN [31] with 73257 basic training set plus 531131 additional set, each client has 2014 samples for its local training. We fix $\epsilon = 0.5$ and $\epsilon = 4$ for MNIST and SVHN respectively ($p_l = 0.7$ and $p_r = 1$). For the evaluation on MNIST and SVHN, each student has 600 and 3000 queries, respectively.

To accommodate the small size of the local datasets, we adopt LeNet5, ResNet8 for MNIST, and VGG7, ResNet10 for SVHN. LeNet5 consists of 2 convolutional layers, 2 max-pooling layers, and 3 fully connected layers. ResNet8 consists of 1 convolutional layer, followed by 3 basic blocks with 2 convolutional layers in each block, and 1 fully connected layer at the end. ResNet10 has a similar architecture as ResNet8, but it includes an extra block with 2 convolutional layers before the fully connected layer. VGG7 has 7 functional layers, including 6 convolutional layers and 1 fully-connected layer.

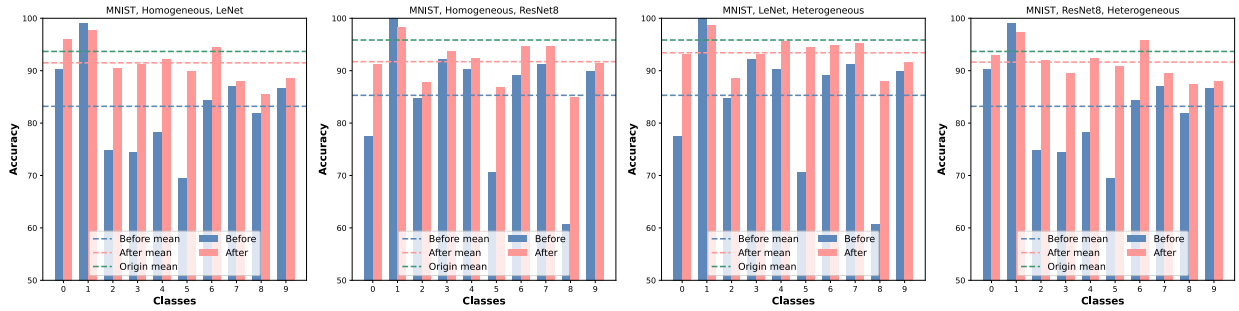
To simulate the heterogeneity as the nature of distributed machine learning systems, we require two sets of $\frac{n}{2}$ clients to keep two different neural networks. Specifically, for MNIST, 150 clients use LeNet, and 150 clients use ResNet8. For SVHN, 150 clients use VGG7 and 150 clients use ResNet10. Besides, one model of each neural network architecture is used per student for MNIST and SVHN. Thus, the accuracy improvement in the heterogeneous setting is the average value of two students.

Split learning setting. We follow the standard split learning scheme without label sharing (see Figure 2). For each neural network architecture used for evaluation, we split it into three portions. The first portion is the first convolutional layer, and the last portion is the last fully connected layer. The second portion consists of the rest of the layers. The first and last portions of split NN are stored in the workstation that simulates that clients are running in parallel. The second portion is stored in the SL server. During the execution of CSTE, teachers compute only the first and last portion, and thus significantly reduce their computational costs (see Figure 4). For example, the first portion of LeNet is its first convolutional layer. The second portion consists of the second convolutional layer and two fully connected layers. The last portion is the last fully connected layer of LeNet.

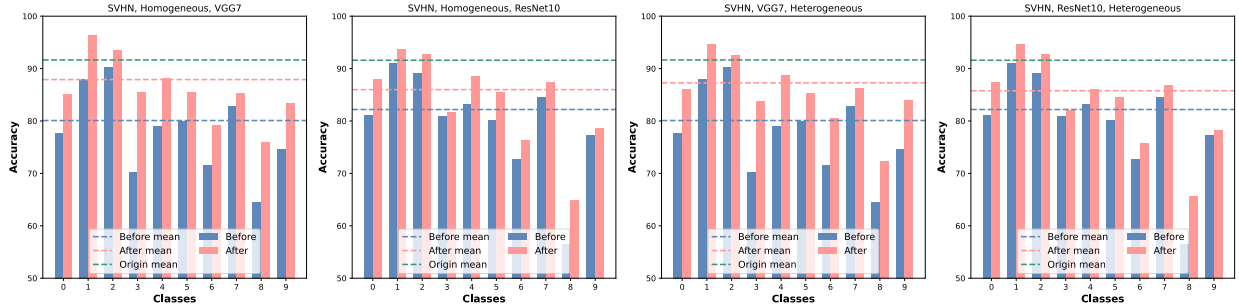
Dropout setting. For an n-client system, we consider there are at least $p_l n$ teachers and at most $p_r n$ teachers that participate in the execution of CSTE. Such setting achieves an (ϵ, δ)-differential privacy by adding noise $\eta_i \sim \mathcal{N}(0, \frac{1}{p_l n} C^2 \sigma^2)$ to teacher's prediction, where $\sigma = \sqrt{2log(\frac{1.25 p_r}{\delta})}/log(1 + \frac{e^\epsilon - 1}{p_r})$ (see Section 3.4). Standard (t, n) Shamir secret sharing scheme is used to handle dropped users where t is set to be $0.3n$. We evaluate the student's model accuracy under different differential privacy budget which depends on the number of clients n and dropout rate $p_l n$ and $p_r n$.

4.1 Model Performance Analysis

First, we investigate the student's model performance improvement by using CSTE for collaborative learning. Under fixed differential privacy budget and user dropout rate, we evaluate the accuracy improvement of each class and their mean accuracies on MNIST and SVHN datasets in both homogeneous and heterogeneous settings. From Figure 5(a) and Figure 7(d), we can observe a consistent improvement in both per-class accuracies and mean accuracies. Figure 6 describe the relationship between the student's model accuracy and differential privacy budget regarding the number of clients and dropout rate. We can observe that a higher dropout rate requires



(a) Improve student's model accuracy on MNIST using CSTE. We can observe an accuracy improvement of 8.30%, 6.42%, 8.44% and 8.12% for (LeNet, homogeneous), (ResNet8, homogeneous), (LeNet, heterogeneous) and (ResNet8, heterogeneous) setting respectively.



(b) Improve student's model accuracy on SVHN using CSTE. We can observe an accuracy improvement of 7.81%, 3.80%, 7.19% and 3.58% for (VGG7, homogeneous), (ResNet10, homogeneous), (VGG7, heterogeneous) and (ResNet10, heterogeneous) setting respectively.

Figure 5: Improve student's model accuracy using CSTE. Blue and orange dashed lines represent the mean accuracy of the student's model before and after CSTE. The green dashed line represents the mean accuracy of the student's model retrained using the same queries with ground-truth labels.

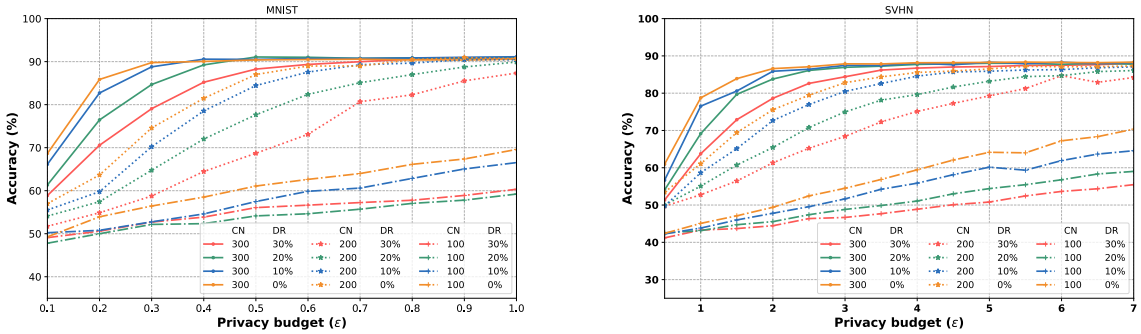


Figure 6: Student's model accuracy under different differential privacy budget. CN is the client number including student, DR is the dropout rate of teachers.

a larger budget to achieve the same accuracy. With more number of clients, higher accuracy can be achieved at a smaller differential privacy budget.

4.2 Efficiency Analysis

Next, we proceed to the efficiency evaluation. Since both CaPC and CSTE consist of HE based privacy-preserving prediction on student's query, on which the whole protocol execution runtime depends heavily, we give the time evaluation on both prediction and whole protocol execution in Table 2 and Table 3 respectively.

Table 2: Computation time of prediction per query. This table gives the time spent on making prediction on one student's query. PATE, CaPC, CaPC-SIMD and CSTE-single thread (s-t) are evaluated using single thread. CSTE-multiple thread (m-t) is evaluated using multiple threads. CSTE-Teacher (t) runs in single thread evaluating the computation time of the first portion of split NN (performed by individual teacher). Note that since CaPC adopts HE-GC hybrid scheme, we also apply the SIMD packing technique to CaPC to accelerate its HE part for a fair comparison.

	CNet-R	LeNet5	ResNet8	VGG7	ResNet10
PATE	0.49±0.20ms	0.56±0.20ms	9.33±5.76ms	19.38±7.86ms	17.27±5.85ms
CaPC	1.18K±9.51s	3.2K±44.83s	0.33M±4.56Ks	0.2M±2.77Ks	1.01M±14.16Ks
CaPC-SIMD	8.88±0.10s	56.22±0.67s	1.96K±23.76s	1.31K±15.63s	2.94K±35.03s
CSTE (s-t)	0.72±0.01s	2.03±0.01s	760.57±10.35s	465.39±6.34s	3.00K±179.04s
CSTE (m-t)	0.35±0.01s	0.33±0.01s	67.56±0.92s	39.92±0.54s	255.17±15.20s
CSTE (t)	0.04±0.01s	0.18±0.01s	5.29±0.03s	12.46±0.12s	16.65±0.72s

Table 3: Wall-time of the whole protocol execution per query. This table gives the wall-time in second spent on the whole protocol execution, i.e., from the student posing the query to the student obtaining the answer. The number of teachers is set to be 30. The student and teachers are running in single thread, and the SL server is running in multiple threads. SIMD is applied to both CaPC and CSTE. * denotes that we only run several key layers and the values are made by projections.

	CNet-R(s)	LeNet5(s)	ResNet8(s)	VGG7(s)	ResNet10(s)	VGG19(s)	ResNet50(s)
PATE	0.11±0.03	0.06±0.01	0.20±0.03	0.23± 0.03	0.32±0.06	0.70±0.03	0.90±0.04
CaPC	17.48±0.10	64.82±0.68	1.97K±23.77	1.32K±15.64	2.96K±35.04	23.29K*	33.05K*
CSTE	23.82±0.01	24.12±0.01	95.98±0.95	74.82±0.65	0.29K±15.84	102.98K*	6.1M*

Table 4: Performance evaluation on homomorphic operations of CKKS scheme. Plaintext is denoted by pt and ciphertext is denoted by ct. The poly modulus degree is selected according to the SEAL's guidance with respect to multiplicative depth (Mult Depth), bit precision (6 bits for the integer part and 26 bits for the fractional part) and security level (128-bit security). Adds, Mults, and Square refer to homomorphic addition, multiplication and square operations. Note that because of the necessary of truncation after each multiplication, the time of a single multiplication includes both the ciphertext-ciphertext multiplication, relinear and rescale operations (see [15, 34] for the details).

	Mult Depth	Poly Modulus	ct_ct_Mult	ct_ct_Square	ct_pt_Mult	ct_ct_Add
CaPC	4	8192	16652μs	16157μs	3284μs	158μs
CryptoNet	8	16384	81639μs	79999μs	12614μs	562μs
LeNet	11	16384	129242μs	126681μs	17027μs	757μs
ResNet8	23	32768	1023506μs	1014767μs	75848μs	3406μs
VGG7	22	32768	915452μs	905863μs	72228μs	3229μs
ResNet10	29	32768	1443455μs	1431040μs	98714μs	4285μs

Since SIMD is adopted in the experiments, we evaluate the efficiency of different models with respect to throughput. From Table 2, we can observe that CSTE achieves a significant efficiency improvement upon CaPC [16]. By using multiple threads (12 in our experiments), CSTE further reduces the time consumption of prediction by around 10 times. Note that multiple-thread strategy does not much affect CaPC, because most of its overhead comes from evaluating non-linear activation functions using a garbled circuit which is constrained by the communication bottleneck. By combining with a split learning scheme, the individual teacher evaluates only the first few layers of split NN, which results in more reasonable time consumption. Here CNet-R (CryptoNet-ReLU) [8] is the NN architecture evaluated in CaPC paper [16], which is a variant of CryptoNet [21] consisting of two convolutional layers and two fully connected layers.

In Table 3, we see that for simple neural networks such as CryptoNet and LeNet5, CSTE's efficiency improvement on the whole protocol execution is not that much compared to that on prediction given in Table 2. The time increment is caused by the (t, n) Shamir scheme used in CSTE to handle dropped users, which is an unavoidable trade-off if dropped users are taken into account. The effect of using the Shamir scheme is diminished for large neural networks such as ResNet8, VGG7, and ResNet10. However, we should note that although CaPC and CSTE use the same leveled FHE scheme, different parameters are set according to the different requirements of the number of multiplications. When the neural network size

Table 5: Performance evaluation on MPC operations in CaPC and CSTE. Note that CaPC adopts Garbled Circuit (GC) to perform MPC for ReLU and Argmax, and CSTE adopts Shamir secret sharing scheme (SSS) to perform MPC for Argmax. The size of the matrix to be used to evaluate ReLU is set to be 32×32, and the number of classes to be used to evaluate Argmax is set to be 10, which keeps the consistency with the setting in both MNIST, SVHN, and CIFAR10. The data is represented using a 32-bit fixed-point number (6 bits for the integer part and 26 bits for the fractional part), and security parameters of GC and SSS are set to be 80-bit, and thus a 128-bit plaintext space.

Scheme	Function	Number of clients					
		5	10	15	20	25	30
CaPC (GC)	ReLU	42.30s	86.2s	126.05s	173.52s	214.23s	261.26s
	ArgMax	0.84s	1.57s	2.69s	3.34s	3.87s	4.98s
CSTE (SSS)	ArgMax	0.09s	0.51s	2.11s	4.94s	12.28s	23.63s

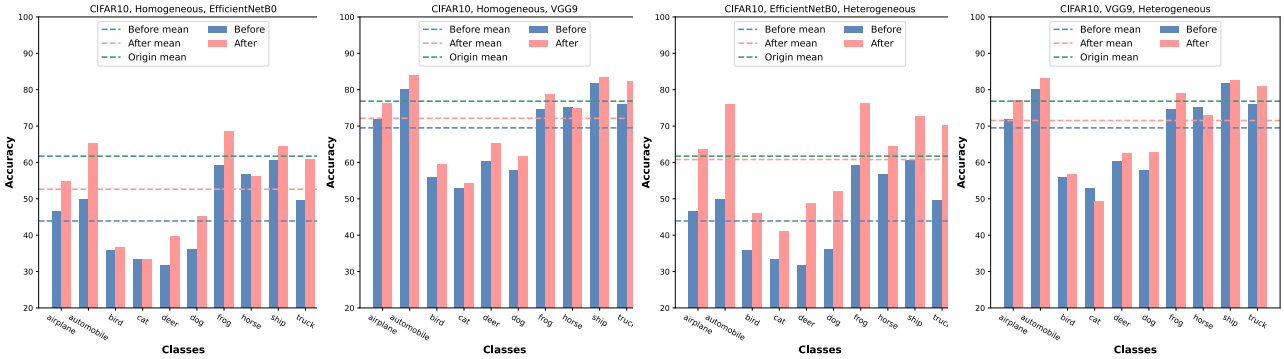


Figure 7: Improve student's model accuracy using CSTE on CIFAR10. We can observe an accuracy improvement of 8.74%, 2.62%, 16.9% and 2.03% for (EfficientNet, homogeneous), (VGG9, homogeneous), (EfficientNet, heterogeneous) and (VGG9, heterogeneous) setting respectively. Blue and orange dashed lines represent the mean accuracy of the student's model before and after CSTE. The green dashed line represents the mean accuracy of the student's model retrained using the same queries with ground-truth labels.

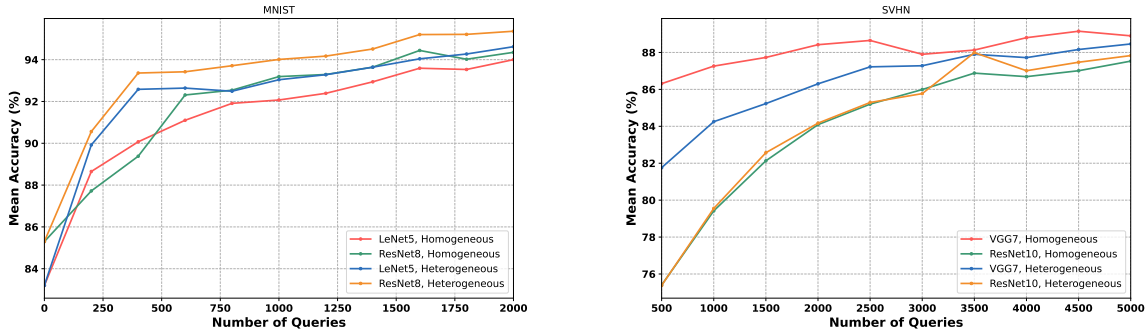


Figure 8: Student's model mean accuracy under different numbers of queries. We can observe a consistent improvement with an increasing number of queries on different neural networks (LeNet5, ResNet8, VGG7, and ReNet10) in both homogeneous and heterogeneous settings.

is not too large, evaluating linear function using HE in CaPC and CSTE results in similar efficiency. Thus, the majority of efficiency improvement of CSTE comes from the throughput of computing comparison (for ReLU and max-pooling) using HE is much faster than using GC. For example, the time consumption of computing comparison on a 32-bit fixed-point number in ResNet10 using HE

and GC are 0.22ms and 8.74ms, respectively. For very deep neural networks such as VGG19 and ResNet50, the time consumption of HE-based operations in CSTE increases to about 250 times upon that in CaPC. In this case, the overall efficiency largely depends on the neural network architecture regarding the ratio of the number of comparisons to arithmetic operations. From Table 3, we can observe CSTE takes more time than CaPC for VGG19 and ResNet50. However, GC used in CaPC requires the student to be online during the whole prediction, which is impractical for real-life applications, and thus HE based prediction is more promising. Evaluating very deep neural networks in an outsourcing and privacy-preserving manner remains an open problem for both the cryptography and machine learning communities.

Additional experimental results. We also evaluate the accuracy improvement of each class and the mean accuracies of all classes on CIFAR10 datasets using EfficientNetB0 [39] and VGG9, respectively. The total number of clients is set to be 30, while each client maintains a disjoint dataset sampled from the training set of the CIFAR10 database. In this experiment, the sizes of clients' local datasets are different. In specific, the clients are uniformly divided into three groups of each the size of the training dataset is set to be 1000, 1500, and 2500, respectively, for three groups. The experimental results are shown in figure 7. Note that the student keeping EfficientNet has 1000 training samples before CSTE, and another student keeping VGG9 has 2500 training samples before CSTE. Both students have 3000 queries, respectively. Performance evaluation on cryptographic tools, i.e., HE, GC, and SSS, of related schemes are given in Table 4 and Table 5.

5 CONCLUSIONS

We proposed CSTE, a collaborative split learning scheme for privacy-preserving knowledge transfer suitable for FinTech applications, that can run on teacher nodes with less demand for computing capabilities. CSTE not only protects the data privacy of teacher nodes from collusion between student node and aggregator, but also resilient to dropped users. Experimental results showed that CSTE achieves a significant efficiency improvement from previous works.

ACKNOWLEDGMENTS

This research is supported by the National Research Foundation, Singapore under its Strategic Capability Research Centres Funding Initiative. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore.

REFERENCES

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 308–318.

[2] Martín Abadi, Ulfar Erlingsson, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Nicolas Papernot, Kunal Talwar, and Li Zhang. 2017. On the protection of private information in machine learning systems: Two recent approches. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*. IEEE, 1–6.

[3] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. 2017. Designing Neural Network Architectures using Reinforcement Learning. In *5th International Conference on Learning Representations, ICLR*.

[4] Bowen Baker, Otkrist Gupta, Ramesh Raskar, and Nikhil Naik. 2018. Accelerating Neural Architecture Search using Performance Prediction. In *6th International Conference on Learning Representations, ICLR*.

[5] Donald Beaver. 1991. Efficient multiparty protocols using circuit randomization. In *Annual International Cryptology Conference*. Springer, 420–432.

[6] James Henry Bell, Kallista A Bonawitz, Adrià Gascón, Tancrède Lepoint, and Mariana Raykova. 2020. Secure single-server aggregation with (poly) logarithmic overhead. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. 1253–1269.

[7] Fabian Boemer, Rosario Cammarota, Daniel Demmler, Thomas Schneider, and Hossein Yalame. 2020. MP2ML: a mixed-protocol machine learning framework for private inference. In *Proceedings of the 15th International Conference on Availability, Reliability and Security*. 1–10.

[8] Fabian Boemer, Anamaria Costache, Rosario Cammarota, and Casimir Wierzynski. 2019. ngraph-he2: A high-throughput framework for neural network inference on encrypted data. In *Proceedings of the 7th ACM Workshop on Encrypted Computing & Applied Homomorphic Cryptography*. 45–56.

[9] Franziska Boenisch, Christopher Mühl, Roy Rinberg, Jannis Ihrig, and Adam Dziedzic. 2023. Individualized PATE: Differentially Private Machine Learning with Individual Privacy Guarantees. *Proc. Priv. Enhancing Technol.* 2023, 1 (2023).

[10] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 1175–1191.

[11] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. 2014. (Leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)* 6, 3 (2014), 1–36.

[12] Octavian Catrina and Sebastiaan De Hoogh. 2010. Improved primitives for secure multiparty integer computation. In *International Conference on Security and Cryptography for Networks*. Springer, 182–199.

[13] Melissa Chase, Hao Chen, Jintai Ding, Shafi Goldwasser, Sergey Gorbunov, Jeffrey Hoffstein, Kristin Lauter, Satya Lokam, Dustin Moody, Travis Morrison, et al. 2017. Security of homomorphic encryption. *HomomorphicEncryption. org, Redmond WA, Tech. Rep* (2017).

[14] Qingrong Chen, Chong Xiang, Minhui Xue, Bo Li, Nikita Borisov, Dali Kaarfar, and Haojin Zhu. 2018. Differentially private data generative models. *arXiv preprint arXiv:1812.02274* (2018).

[15] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. 2017. Homomorphic encryption for arithmetic of approximate numbers. In *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 409–437.

[16] Christopher A. Choquette-Choo, Natalie Dullerud, Adam Dziedzic, Yunxiang Zhang, Somesh Jha, Nicolas Papernot, and Xiao Wang. 2021. CaPC Learning: Confidential and Private Collaborative Learning. *9th International Conference on Learning Representations, ICLR* (2021).

[17] R. Cramer, I.B. Damgård, and J.B. Nielsen. 2015. *Secure Multiparty Computation*. Cambridge University Press.

[18] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*. Springer, 265–284.

[19] Cynthia Dwork, Aaron Roth, et al. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science* 9, 3-4 (2014), 211–407.

[20] Craig Gentry. 2009. Fully homomorphic encryption using ideal lattices. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*. 169–178.

[21] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. 2016. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *International Conference on Machine Learning*. PMLR, 201–210.

[22] Jiale Guo, Ziyao Liu, Kwok-Yan Lam, Jun Zhao, Yiqiang Chen, and Chaoping Xing. 2020. Secure Weighted Aggregation in Federated Learning. *arXiv preprint arXiv:2010.08730* (2020).

[23] Otkrist Gupta and Ramesh Raskar. 2018. Distributed learning of deep neural network over multiple agents. *Journal of Network and Computer Applications* 116 (2018), 1–8.

[24] Ismat Jarin and Birhanu Eshete. 2021. PRICURE: Privacy-Preserving Collaborative Inference in a Multi-Party Setting. In *Proceedings of the 2021 ACM Workshop on Security and Privacy Analytics*. 25–35.

[25] James Jordon, Jinsung Yoon, and Mihaela Van Der Schaar. 2018. PATE-GAN: Generating synthetic data with differential privacy guarantees. In *International Conference on Learning Representations*.

[26] Nikola Jovanovic, Marc Fischer, Samuel Steffen, and Martin Vechev. 2022. Private and Reliable Neural Network Inference. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. 1663–1677.

[27] Marcel Keller. 2020. MP-SPDZ: A Versatile Framework for Multi-Party Computation. Cryptology ePrint Archive, Report 2020/521. https://eprint.iacr.org/2020/521.

[28] Yann LeCun, Corinna Cortes, and CJ Burges. 2010. MNIST handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist* 2 (2010).
[29] Yunhui Long, Suxin Lin, Zhuolin Yang, Carl A Gunter, and Bo Li. 2019. Scalable differentially private generative student model via pate. *arXiv preprint arXiv:1906.09338* (2019).
[30] Ilya Mironov. 2017. Rényi differential privacy. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*. IEEE, 263–275.
[31] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. 2011. Reading digits in natural images with unsupervised feature learning. *Advances in Neural Information Processing Systems (NIPS)* (2011).
[32] Nicolas Papernot, Martín Abadi, Ulfar Erlingsson, Ian Goodfellow, and Kunal Talwar. 2017. Semi-supervised knowledge transfer for deep learning from private training data. *5th International Conference on Learning Representations, ICLR* (2017).
[33] Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Ulfar Erlingsson. 2018. Scalable Private Learning with PATE. *6th International Conference on Learning Representations, ICLR* (2018).
[34] SEAL 2019. Microsoft SEAL (release 3.4). https://github.com/Microsoft/SEAL. Microsoft Research, Redmond, WA..
[35] Arnaud Grivet Sébert, Rafael Pinot, Martin Zuber, Cédric Gouy-Pailler, and Renaud Sirdey. 2021. SPEED: secure, PrivatE, and efficient deep learning. *Machine Learning* (2021), 1–20.
[36] Adi Shamir. 1979. How to share a secret. *Commun. ACM* 22, 11 (1979), 612–613.
[37] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 3–18.
[38] Abhishek Singh, Ayush Chopra, Vivek Sharma, Ethan Garza, Emily Zhang, Praneeth Vepakomma, and Ramesh Raskar. 2021. DISCO: Dynamic and Invariant Sensitive Channel Obfuscation for deep neural networks. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR*. IEEE.
[39] Mingxing Tan and Quoc Le. 2019. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*. PMLR, 6105–6114.
[40] Praneeth Vepakomma, Otkrist Gupta, Tristan Swedish, and Ramesh Raskar. 2018. Split learning for health: Distributed deep learning without sharing raw patient data. *arXiv preprint arXiv:1812.00564* (2018).
[41] Boxin Wang, Fan Wu, Yunhui Long, Luka Rimanic, Ce Zhang, and Bo Li. 2021. DataLens: Scalable Privacy Preserving Training via Gradient Compression and Aggregation. *arXiv preprint arXiv:2103.11109* (2021).
[42] Ji Wang, Weidong Bao, Lichao Sun, Xiaomin Zhu, Bokai Cao, and S Yu Philip. 2019. Private model compression via knowledge distillation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 1190–1197.
[43] Chao-Han Huck Yang, Sabato Marco Siniscalchi, and Chin-Hui Lee. 2021. PATE-AAE: Incorporating Adversarial Autoencoder into Private Aggregation of Teacher Ensembles for Spoken Command Classification. *arXiv preprint arXiv:2104.01271* (2021).
[44] Andrew Chi-Chih Yao. 1986. How to generate and exchange secrets. In *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*. IEEE, 162–167.
[45] Hong-Jun Yoon, Hilda Klasky, Christopher Stanley, Blair Christian, Georgia Tourassi, Eric B Durbin, Xiao-Cheng Wu, Antoinette Stroup, Jennifer Doherty, Linda Coyle, et al. 2021. *Privacy-Preserving Knowledge Transfer with Bootstrap Aggregation of Teacher Ensembles*. Technical Report. Oak Ridge National Lab.(ORNL), Oak Ridge, TN (United States).
[46] Qiuchen Zhang, Jing Ma, Jian Lou, Li Xiong, and Xiaoqian Jiang. 2020. Towards Training Robust Private Aggregation of Teacher Ensembles Under Noisy Labels. In *2020 IEEE International Conference on Big Data (Big Data)*. IEEE, 1103–1110.
[47] Yuqing Zhu, Xiang Yu, Manmohan Chandraker, and Yu-Xiang Wang. 2020. Private-knn: Practical differential privacy for computer vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11854–11862.

A CKKS SCHEME

CKKS allows homomorphic operations over approximate numbers.In general, CKKS involves four key parameters including (i) polynomial modulus degree N, (ii) coefficient modulus vector L, (iii) bit precision i_s and f_s, and (iv) security level λ. A rough guideline of how to choose CKKS parameters is given below:

(1) Select the security level λ According to the recommended parameters from the security white paper [13], or SEAL's table of recommended parameters [34].
(2) Set the length of coefficient modulus vector L according to the multiplicative depth, which is the largest number of multiplications needed to evaluate the target function.
(3) Choose the bit precision i_s for integer part and f_s for the fractional part.
(4) Construct the coefficient modulus vector according to the selected length of coefficient modulus vector and bit precision such that $L = [i_s + f_s, f_s, \ldots, i_s + f_s]$ where the number of f_s equals to the multiplicative depth.
(5) Set the polynomial modulus degree N to the proper value based on $\sum L$ and security parameters recommended in [13] and [34] (see the table as below).

N	1024	2048	4096	8192	16384	32768
$\sum L$	27	54	109	218	438	881

Note that the largest N supported by SEAL is 32768. By setting $N = 32768$, $i_s = 6$, $f_s = 26$ (about 8 bits in base 10), and $\lambda = 128$, the length of coefficient modulus vector L cannot be larger then 31. This means neural networks of which the number of multiplication depth is greater than 31 cannot be evaluated.

B SECURE MULTIPLE PARTY COMPUTATION

Secure Multiple Party Computation (MPC) [17] allows several parties to jointly evaluate a function based on their secretly shared data, e.g., the shares as introduced in Section 2.3, while preserving the data privacy of each party. In CSTE, each client performs its SL training respectively, MPC is only used for all the teachers jointly compute the label with the most votes.

B.1 Secure Arithmetic Operations over Shamir Shares

A secret sharing scheme is homomorphic if it enables computing new valid shares from existing shares. More formally, let x and y be two values and $[x] = [x_1, \ldots, x_n]$[1] and $[y] = [y_1, \ldots, y_n]$ be the shares, a secret sharing scheme is $(\oplus, \otimes)$-homomorphic if the new shares $[(x_1 \otimes y_1), \ldots, (x_n \otimes y_n)]$ uniquely determine the value $x \oplus y$. Once the Shamir shares have been distributed, all parties can perform operations with the shares using the homomorphic property of the secret sharing scheme. Next, we give the secure computation protocols for addition and multiplication between two Shamir shares in Algorithm 1 and Algorithm 2 respectively.

Algorithm 1 Protocol for adding two Shamir shares for the party u_k: $[z] \leftarrow Add([x], [y])$

Input: Two Shamir shares x_k and y_k.
Output: The share z_k that represents the sum of $[x]$ and $[y]$.

Round 1: Compute $z_k = x_k + y_k$.

The value β_k used in Algorithm 2 can be precomputed using the following equation:

$$\beta_k = b_k(0) = \prod_{j=1, j\neq k}^{t} \frac{-a_j}{a_k - a_j} \tag{2}$$

[1]The bracket here "[]" means the value is secretly shared and it is different from the notation used in the main body where we use $SS(x)$ to avoid confusion of the vector representation

Algorithm 2 Protocol for multiplying two Shamir shares for the party u_k: $[z] \leftarrow Mul([x], [y])$

Input: Two Shamir shares x_k and y_k, precomputed value β_k.
Output: The share z_k that represents the value of $[x][y]$.

Round 1:
$w_k = x_k y_k \beta_i$.
Share w_k to $w_{k_1}, \ldots, w_{k_n}$ using the same Shamir sheme, i.e., the same polynomial.
Send the share w_{k_l} to the party u_l where $k \neq l$.
Round 2:
Receive share w_{j_k} from other parties where $k \neq j$.
Compute $z_k = w_{k_k} + \sum_{j=1, j\neq k}^{n} w_{j_i}$

Here a_j, a_k are the coefficients of the polynomial used for Shamir secret sharing. Note that the technique in Algorithm 2 called degree reduction can be used only for the Shamir scheme that the threshold $t < \frac{n}{3}$. For larger threshold $t < \frac{n}{2}$, multiplication between Shamir shares can be done leveraging Beaver triples [5]. A Beaver triple consists of three secretly shared values $[a]$, $[b]$ and $[c]$ such that $c = ab$. Given $[x]$ and $[y]$, all parties can follow the protocol in Algorithm 3 to calculate $[z]$ such that $z = xy$.

Algorithm 3 Multiplication based on Beaver triple

Input: Two Shamir shares $[x]$ and $[y]$, precomputed Beaver triple $[a]$, $[b]$ and $[c]$ that $c = ab$.
Output: The Shamir shares $[z]$ that represents the value of $[x][y]$.

1: Each party u_i publishes $x_i - a_i$ and $y_i - b_i$
2: Each party u_i compute $x - a$ and $y - b$
3: u_1 sets its share $z_1 = c_1 + (x-a)\cdot b_1 + (y-b)\cdot a_1 + (x-a)\cdot(y-b)$, u_j sets its share $z_j = c_j + (x-a)\cdot b_j + (y-b)\cdot a_j$ for $j \neq 1$
4: Return $[z]$

B.2 Secure Argmax

The MPC protocol in the CSTE scheme involves only one non-arithmetic operation, i.e., the comparison, which can then be used to construct $\text{argmax}_k f(k)$ protocol, i.e., computing k that $f(k)$ is the maximum value for all possible k. We adopt the comparison protocol in [12] as a blackbox and denote it as $[z] \leftarrow Comp([x], [y])$ such that $[z] = [x \geq y]$. Then the MPC protocol for secure Argmax can be constructed as given in Algorithm 4.

Algorithm 4 Protocol for Argmax function: $[i_m] \leftarrow Argmax(\{[c_1], \ldots, [c_k]\})$

Input: A vector of shares $[c_i]$ for $i = 1, \ldots, n$. $[c_m] = [c_1]$ and $[i_m] = [1]$.
Output: The shares of index $[i_m]$ at which c_{i_m} is the maximum value.

for j=2,…,k **do**
  $[w] = Comp([c_j], [c_m])$
  $[c_m] = Add([c_m], Mul([w], Add([c_j], [-c_m])))$
  $[i_m] = Add([i_m], Mul([w], Add([j], [-i_m])))$
end for
Return $[i_m]$

C DIFFERENTIAL PRIVACY

Definition 3 (The Gaussian Mechanism). *Given any function $f : \mathcal{D} \to \mathbb{R}$, the Gaussian mechanism is defined as:*

$$\mathcal{M}(x, f(\cdot), \epsilon) = f(x) + (\eta_1, \cdots, \eta_m) \tag{3}$$

where η_i are i.i.d. random variables drawn from $\mathcal{N}(0, S_f^2\sigma^2)$.

The application of the Gaussian mechanism to function f with sensitivity S_f satisfies (ϵ, δ)-differential privacy. Theorem C.1 shows the details.

Theorem C.1 ([19]). *Given any $0 < \epsilon < 1$. For $c^2 > 2ln(1.25/\delta)$, the Gaussian Mechansim with parameter $\delta \geq cS_f/\epsilon$ is (ϵ, δ)-differentially private.*

Differential privacy has several composition properties which can be used for more complicated privacy algorithm analysis. The basic sequential composition is shown as follows.

Theorem C.2 (Sequential Composition [19]). *Suppose that a method includes m independent randomized functions $\mathcal{M} = \{\mathcal{M}_1, \mathcal{M}_2, ..., \mathcal{M}_m\}$, each $\mathcal{M}_i$ provides (ϵ_i, δ_i)-differential privacy guarantee. If these functions are performed on the same dataset sequentially, then $\mathcal{M}$ is $(\sum_{i=1}^{m} \epsilon_i, \sum_{i=1}^{m} \delta_i)$-differentially private.*

The general sequential composition theorem considers the biggest value of privacy loss. The bound is not tight. To provide a tighter bound, Dwork et al. consider the expectation of the loss, which results in the advanced composition shown as follows.

Theorem C.3 (Advanced Composition [19]). *For all $\epsilon, \delta, \delta' \geq 0$, the class of (ϵ, δ)-differentially private mechansims satisfies $(\epsilon', m\delta + \delta')$-differential privacy under m-fold adaptive compsoition for:*

$$\epsilon' = \epsilon\sqrt{2m\log(1/\delta')} + m\epsilon(e^{\epsilon} - 1)$$

Instead of increasing linearly with the number of queries m, the privacy loss increases linearly with $\sqrt{m}$. The advanced composition has a significant advantage, especially when m is large. Later, Abadi et al. [1] proposed the concept of moments accountant to track privacy loss, which reduces the upper bound further. The general idea is that instead of only considering the expectation, they explore the higher-order moments to bound the tail of the privacy loss variable.

Definition 4 (Moments Accountant [1]). *Given a random mechanism $\mathcal{M}$, auxiliary input aux, and neighbored datasets $d, d' \in \mathcal{D}$, the moment account is defined as:*

$$\alpha(\lambda) \triangleq \max_{aux,d,d'} \alpha_{\mathcal{M}}(\lambda; aux, d, d')$$

where $\alpha_{\mathcal{M}}(\lambda; aux, d, d')$ is the λ^{th} moment, which is defined as:

$$\alpha_{\mathcal{M}}(\lambda; aux, d, d') \triangleq logE_{o\sim\mathcal{M}(aux,d)}[exp(\lambda c(o; \mathcal{M}, aux, d, d'))]$$

(ϵ, δ)-differential privacy offers smaller cumulative loss under composition. However, the application of an advanced composition leads to a wide selection of possibilities for $(\epsilon(\delta), \delta)$-diferentially

private guarantees, which implicitly comes with an $\epsilon - \delta$ trade-off. Finding an optimal value may be non-trivial. To solve such a problem, Renyi differential privacy was proposed.

Definition 5 ((α, ϵ)-RDP [30]). *A randomized mechanism $\mathcal{M}: \mathbb{D} \rightarrow \mathbb{R}$ is said to have ϵ-Renyi differential privacy of order α, or (α, ϵ)-RDP for short, if for any adjacent $D, D' \in \mathbb{D}$ it holds that*

$$D_\alpha(\mathcal{M}(D)||\mathcal{M}(D')) \leq \epsilon$$

where $D_\alpha(\mathcal{M}(D)||\mathcal{M}(D'))$ is the Renyi Divergence of order $\alpha > 1$, specifically,

$$D_\alpha(\mathcal{M}(D)||\mathcal{M}(D')) = \frac{1}{\alpha - 1} \log E_{x \sim \mathcal{M}(D)}[(\frac{Pr[\mathcal{M}(D) = x]}{Pr[\mathcal{M}(D') = x]})^{\alpha - 1}] \tag{4}$$

The α does not change, and the composition rule of RDP is straightforward, which is shown as follows.

Theorem C.4 (Composition). *Let $\mathcal{M}$ be a sequence of adaptive mechanisms $\mathcal{M}_1, \cdot, \mathcal{M}_m$, where $\mathcal{M}_i, i \in [m]$ guarantees (α, ϵ_i)-RDP, then $\mathcal{M}$ satisfies $(\alpha, \sum_{i=1}^{m} \epsilon_i)$-RDP.*

The definitaion of ϵ-differential privacy coincides with (∞, ϵ)-RDP, where (∞, ϵ)-RDP implies (α, ϵ)-RDP for all finite α. In turn, and (α, ϵ)-RDP implies $(\epsilon_\delta, \delta)$-differential privacy for any $\delta > 0$. The following proposition shows the details.

Proposition C.5 (From RDP to (ϵ, δ)-DP). *If $\mathcal{M}$ is an (α, ϵ)-RDP mechanism, it also satisfies $(\epsilon + \frac{log(1/\delta)}{\alpha - 1}, \delta)$-differential privacy for any $0 < \delta < 1$.*

The moments' accountant and the RDP accountant are basically equivalent and lead to identical outcomes when translated to the (ϵ, δ)-DP language.

The improved PATE [33] modified the aggregation mechanism of PATE [32] by adding Gaussian noise instead of Laplace noise. Utilizing moments accounts and RDP techniques, they provide a much tighter bound for the privacy analysis and further improve the bound by considering the consensus of teacher models. Specifically, instead of answering all the queries from the student, the aggregator only answers queries with higher consensus.