Practical One-Shot Federated Learning for Cross-Silo Setting

Qinbin Li[1] , **Bingsheng He**[1] , **Dawn Song**[2]
[1]National University of Singapore
[2]University of California, Berkeley
{qinbin, hebs}@comp.nus.edu.sg, dawnsong@cs.berkeley.edu

Abstract

Federated learning enables multiple parties to collaboratively learn a model without exchanging their data. While most existing federated learning algorithms need many rounds to converge, one-shot federated learning (i.e., federated learning with a single communication round) is a promising approach to make federated learning applicable in cross-silo setting in practice. However, existing one-shot algorithms only support specific models and do not provide any privacy guarantees, which significantly limit the applications in practice. In this paper, we propose a practical one-shot federated learning algorithm named FedKT. By utilizing the knowledge transfer technique, FedKT can be applied to any classification models and can flexibly achieve differential privacy guarantees. Our experiments on various tasks show that FedKT can significantly outperform the other state-of-the-art federated learning algorithms with a single communication round.

1 Introduction

While the size of training data can influence the machine learning model quality a lot, the data are often dispersed over different parties in reality. Due to regulations on data privacy, the data cannot be centralized to a single party for training. A popular solution is federated learning [Kairouz *et al.*, 2019; Li *et al.*, 2019; Yang *et al.*, 2019], which enables multiple parties to collaboratively learn a model without exchanging their local data.

A typical and widely used federated learning algorithm is FedAvg [McMahan and others, 2016]. Its training is an iterative process with four steps in each iteration. First, the server sends the global model to the selected parties. Second, each of the selected parties updates its model with their local data. Third, the updated models are sent to the server. Last, the server averages all the received models to update the global model. There are many variants of FedAvg [Li *et al.*, 2020b; Karimireddy and others, 2020; Lin *et al.*, 2020], which have similar frameworks to FedAvg.

The above iterative algorithms are mainly designed for the cross-device setting, where the parties are mobile devices. In such a setting, the server is usually managed by the federated learning service provider (e.g., Google) and the parties are the users that are willing to improve their service quality (e.g., Google Keyboard users). The server can sample part of devices to conduct federated learning in each round and there are always users available. However, in cross-silo settings, where the parties are usually organizations, approaches like FedAvg may not work in practice due to the following reasons. First, the algorithm requires parties to participate multi-round training, which is not practical in some scenarios such as model markets [Vartak and others, 2016; Baylor and others, 2017]. Second, federated learning across rounds may suffer from attacks (e.g., inference attacks [Shokri *et al.*, 2017]) from curious parties. Last, it is hard to find a fair and trusted server to lead the training process.

A promising solution is one-shot federated learning (i.e., federated learning with only a single communication round). With one-shot federated learning, the parties can simply sell or upload their local models to a model market (or a model management platform) [Vartak and others, 2016; Baylor and others, 2017]. Then, a buyer or the market can use these models collectively to learn a final model, which is very suitable for cross-silo settings. Such a process largely reduces the multi-round requirements on the stability of the parties. It is natural that sellers put their models into the model market (in return for incentives and benefits, whose design is interesting but out of scope of this paper). Usually, sellers are not engaged in purchasing and consuming the models, and one-shot federated learning algorithms are a must here.

There have been several studies on one-shot federated learning [Guha *et al.*, 2019; Zhou *et al.*, 2020; Yurochkin *et al.*, 2019]. However, existing one-shot federated learning studies have the following obvious shortcomings. First, they usually are specially designed for a special model architecture (i.e., support vector machines or multi-layer perceptrons), which significantly limit the applications in the real-world. Second, they do not provide any privacy guarantees. This is important in the model market scenario, since the models may be sold to anyone including attackers.

In this paper, we propose a new one-shot federated learning algorithm named FedKT (Federated learning via Knowledge Transfer). Inspired by the success of the usage of unlabelled public data in many studies [Papernot *et al.*, 2017; Papernot and others, 2018; Chang *et al.*, 2019; Lin *et al.*, 2020], which often exists such as text and images and can be obtained by public repositories or synthetic data generator or various data markets , we design a two-tier knowledge transfer framework to achieve effective and private one-shot federated learning. As such, unlike most existing studies that only work on either differentiable models (e.g., neural networks [McMahan and others, 2016]) or non-differentiable models (e.g., decision trees [Li *et al.*, 2020a]), FedKT is able to learn any classification model. Moreover, we develop differentially private versions and theoretically analyze the privacy loss of FedKT in order to provide different differential privacy guarantees. Our experiments on various tasks and models show that FedKT significantly outperforms the other state-of-the-art federated learning algorithms with a single communication round.

Our main contributions are as follows.

- Based on the knowledge transfer approach, we propose a new federated learning algorithm named FedKT. To the best of our knowledge, FedKT is the first one-shot federated learning algorithm which can be applied to any classification models.
- We consider comprehensive privacy requirements and show that FedKT is easy to achieve both example-level and party-level differential privacy and theoretically analyze the bound of its privacy cost.
- We conduct experiments on various models and tasks and show that FedKT can achieve much better accuracy compared with the other federated learning algorithms with a single communication round.

2 Background and Related Work

2.1 Knowledge Transfer

Knowledge transfer (KT) has been successfully used in previous studies [Hinton *et al.*, 2015; Papernot *et al.*, 2017; Papernot and others, 2018]. Through knowledge transfer, an ensemble of models can be compressed into a single model. A typical example is the PATE (Private Aggregation of Teacher Ensembles) [Papernot *et al.*, 2017] framework. In this framework, PATE first divides the original dataset into multiple disjoint subsets. A teacher model is trained separately on each subset. Then, the max voting method is used to make predictions on the public unlabelled datasets with the teacher ensemble, i.e., choosing the majority class among the teachers as the label. Last, a student model is trained on the public dataset. A good feature of PATE is that it can easily satisfy differential privacy guarantees by adding noises to the vote counts. Moreover, PATE can be applied to any classification model regardless of the training algorithm. However, PATE is not designed for federated learning.

2.2 Federated Learning with Knowledge Transfer

There have been several studies [Li and Wang, 2019; Chang *et al.*, 2019; He *et al.*, 2020; Lin *et al.*, 2020; Zhu *et al.*, 2020] using knowledge transfer in federated learning. [Li and Wang, 2019] needs a public labeled dataset to conduct initial transfer learning, while FedKT only needs a public unlabeled dataset. [Chang *et al.*, 2019] and [He *et al.*, 2020] have different objectives from FedKT. Specifically, [Chang *et al.*, 2019] designs a robust federated learning algorithm to protect against poisoning attacks. [He *et al.*, 2020] considers cross-device setting with limited computation resources and uses group knowledge transfer to reduce the overload of each edge device. [Lin *et al.*, 2020] has a similar setting with FedKT. They use a public dataset to improve the global model in the server side. As we will show in the experiment, FedKT has a much better accuracy than [Lin *et al.*, 2020] with a single round.

All the above studies conduct in an iterative way, which require many communication rounds to converge and cannot be applied in the model market scenario. Moreover, all existing studies transfer the prediction vectors (i.g., logits) on the public dataset between clients and the server. As we will show in Section 3, FedKT transfers the voting counts and can easily satisfy differential privacy guarantees with a tight theoretical bound on the privacy loss.

We notice that there is a contemporary work [Zhu *et al.*, 2020] which also utilizes PATE in federated learning. While they simply extend PATE to a federated setting, we design a two-tier PATE structure and provide more flexible differential privacy guarantees.

2.3 One-Shot Federated Learning

There have been several studies [Yurochkin *et al.*, 2019; Guha *et al.*, 2019; Zhou *et al.*, 2020; Kasturi *et al.*, 2020] on one-shot federated learning. Instead of simply averaging all the model weights in FedAvg, [Yurochkin *et al.*, 2019] propose PFNM by adopting a Bayesian nonparametric model to aggregate the local models when they are multilayer perceptrons (MLPs). Their method shows a good performance in a single communication round and can also be applied in multiple communication rounds. [Guha *et al.*, 2019] propose an one-shot federated learning algorithm to train support vector machines (SVMs) in both supervised and semi-supervised settings. [Zhou *et al.*, 2020] and [Kasturi *et al.*, 2020] transfer the synthetic data or data distribution to the server, which trains the final model using generated dataset. Such data sharing approaches do not fit the mainstream model sharing schemes. Moreover, all existing one-shot federated learning studies do not provide privacy guarantees, which is important especially in the model market scenario where the models are sold and may be bought by anyone including attackers.

2.4 Differential Privacy

Differential privacy [Dwork *et al.*, 2014] is a popular standard of privacy protection. It guarantees that the probability of producing a given output does not depend much on whether a particular data record is included in the input dataset or not. It has been widely used to protect the machine learning models [Abadi *et al.*, 2016; Choquette-Choo *et al.*, 2021].

Definition 1. ((ε, δ)-Differential Privacy) Let $\mathcal{M} \colon \mathcal{D} \to \mathcal{R}$ be a randomized mechanism with domain $\mathcal{D}$ and range R. M satisifes (ϵ, δ)-differential privacy if for any two adjacent inputs $d, d' \in \mathcal{D}$ and any subset of outputs $S \subseteq \mathcal{R}$ it holds that:

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

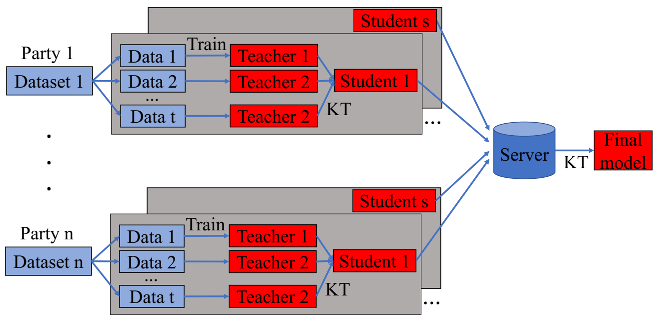

Figure 1: The framework of FedKT.

The moments accountant method [Abadi *et al.*, 2016] is a state-of-the-art approach to track the privacy loss. We briefly introduce the key concept, and refer readers to the previous paper [Abadi *et al.*, 2016] for more details.

Definition 2. (Privacy Loss) Let $\mathcal{M}\colon \mathcal{D} \to \mathcal{R}$ be a randomized mechanism. Let aux denote an auxiliary input. For two adjacent inputs $d, d' \in \mathcal{D}$, an outcome $o \in \mathcal{R}$, the privacy loss at o is defined as:

$$c(o; \mathcal{M}, \mathsf{aux}, d, d') \triangleq \log \frac{\Pr[\mathcal{M}(\mathsf{aux}, d) = o]}{\Pr[\mathcal{M}(\mathsf{aux}, d') = o]}. \qquad (2)$$

Definition 3. (Moments Accountant) Let $\mathcal{M}\colon \mathcal{D} \to \mathcal{R}$ be a randomized mechanism. Let aux denote an auxiliary input. For two adjacent inputs d, d', the moments accountant is defined as:

$$\alpha_{\mathcal{M}}(\lambda) \triangleq \max_{\mathsf{aux}, d, d'} \alpha_{\mathcal{M}}(\lambda; \mathsf{aux}, d, d') \qquad (3)$$

where $\alpha_{\mathcal{M}}(\lambda; \mathsf{aux}, d, d') \triangleq \log \mathbb{E}_o[\exp(\lambda c(o; \mathcal{M}, \mathsf{aux}, d, d'))]$ is the log of moment generating function.

The moments have good composability and can be easily converted to (ε, δ)-differential privacy [Abadi *et al.*, 2016].

Party-level differential privacy. In addition to the standard example-level differential privacy, party-level differential privacy [Geyer *et al.*, 2017; McMahan *et al.*, 2018] is more strict and attractive in the federated setting. Instead of aiming to protect a single record, party-level differential privacy ensures that the model does not reveal whether a party participated in federated learning or not.

Definition 4. (Party-adjacent Datasets) Let d and d' be two datasets of training examples, where each example is associated with a party. Then, d and d' are party-adjacent if d' can be formed by changing the examples associated with a single party from d.

3 Our Approach

Problem statement. Suppose there are n parties $P_1, .., P_n$. We use $\mathcal{D}^i$ to denote the dataset of P_i. With the help of a central server and a public unlabelled dataset $\mathcal{D}_{aux}$, our objective is to build a machine learning model over the datasets $\bigcup_{i \in [n]} \mathcal{D}^i$ without exchanging the raw data. Moreover, the learning process should be able to support three different privacy level settings: (1) $L0$: like most federated learning studies [McMahan and others, 2016; Yurochkin *et al.*, 2019; Lin *et al.*, 2020], the possible inference attacks [Shokri *et al.*, 2017; Fredrikson *et al.*, 2015] on the models are not considered. At $L0$, we do not enforce any privacy mechanism on the model. (2) $L1$ (server-noise): in the case where the final model has to be sent back to the parties or even published, it should satisfy differential privacy guarantees to protect against potential inference attacks. (3) $L2$ (party-noise): in the case where the server is curious and all the models transferred from the parties to the server during training should satisfy differential privacy guarantees. These three levels have their own application scenarios in practice. For example, in the model market, $L0$ is for a model market within an enterprise where different departments can share model but not the data due to data privacy regulations, and departments are trusted. $L1$ is similar to $L0$ but additionally the final model will be published in the market. $L2$ is for the public market where sellers need to protect their own data privacy.

The overall framework. The framework of FedKT is shown in Figure 1. It is designed to be a simple and practical one-shot algorithm for cross-silo settings. Specifically, FedKT adopts a two-tier knowledge transfer structure. On the party side, each party uses knowledge transfer to learn student models and sends them to the server. On the server side, the server takes the received models as teachers to learn a final model using knowledge transfer again. The final model is sent back to the parties and used for predictions. Two techniques, multi-partitioning and consistent voting, are proposed to improve the performance.

Learning student models on the parties (multi-partitioning). Locally, each party has to create s ($s \geq 1$) partitions and learn a student model on each partition. Each partition handles the entire local dataset. Since the operations in each partition are similar, here we describe the process in one partition for ease of presentation. Inside a partition, the local dataset is divided into t disjoint subsets. We train a teacher model separately on each subset, denoted as $T_1, ..., T_t$. Then, the ensemble of teacher models is used to make predictions on the public dataset $\mathcal{D}_{aux}$. For an example $\mathbf{x} \in \mathcal{D}_{aux}$, the *vote count* of class m is the number of teachers that predicts m, i.e., $v_m(\mathbf{x}) = |\{i : i \in [t], T_i(\mathbf{x}) = m\}|$. The prediction result of the ensemble is the class that has the maximum vote counts, i.e., $f(\mathbf{x}) = \arg\max_m v_m(\mathbf{x})$. Then, we use the public dataset $\mathcal{D}_{aux}$ with the predicted labels to train a student model. For each partition, we get a student model with the above steps. After all the student models are trained, the parties send their student models to the server for further processing.

Learning the final model on the server. Suppose the student models of party i are denoted as $U_1^i, ..., U_s^i$ ($i \in [n]$). After receiving all the student models, like the steps on the party side, the server can use these student models as an ensemble to make predictions on the public dataset $\mathcal{D}_{aux}$. The public dataset with the predicted labels is used to train the final model.

Consistent voting. Here we introduce a technique named *consistent voting* for computing the vote counts of each class.

Algorithm 1: The FedKT algorithm

```
Input: local datasets $\mathcal{D}^1, ..., \mathcal{D}^n$, number of partitions
    $s$ in each party, number of subsets $t$ in each
    partition, number of classes $u$, public dataset
    $\mathcal{D}_{aux}$, privacy parameter $\gamma$, privacy level $l$
Output: The final model $F$.
1  for $i = 1, ..., n$ do
       /* Conduct on party $P_i$ */
2      Create $s$ partitions (i.e., $D^i_1, ..., D^i_s$) on dataset $\mathcal{D}^i$
         such that $D^i_j = \bigcup_{k \in [t]} D^i_{j,k}$ for all $j \in [s]$, where
         $D^i_{j,k}$ is a subset.
3      for $j = 1, ..., s$ do
4          for $k = 1, ..., t$ do
5              Train a teacher model $T^i_{j,k}$ on subset $D^i_{j,k}$.
6          for all $\mathbf{x} \in \mathcal{D}_{aux}$ do
7              for all $m \in [u]$ do
8                  $v_m(\mathbf{x}) \leftarrow |\{k : k \in [t], T^i_{j,k}(\mathbf{x}) = m\}|$
9                  if $l == L2$ then
10                     $v_m(\mathbf{x}) \leftarrow v_m(\mathbf{x}) + Lap(1/\gamma)$
11             $f(\mathbf{x}) = \arg\max_m v_m(\mathbf{x})$
12         Train a student model $U^i_j$ on dataset
             $\{(\mathbf{x}, f(\mathbf{x}))\}_{\mathbf{x} \in \mathcal{D}_{aux}}$.
13     Send the student models $\{U^i_j : j \in [s]\}$ to server.
   /* Conduct on the server */
14 for all $\mathbf{x} \in \mathcal{D}_{aux}$ do
15     for all $i \in [n]$ do
16         for all $m \in [u]$ do
17             $v^i_m(\mathbf{x}) \leftarrow |\{k : k \in [s], U^i_k(\mathbf{x}) = m\}|$
18     for all $m \in [u]$ do
19         $v_m(\mathbf{x}) \leftarrow s \cdot |\{i : i \in [n], v^i_m(\mathbf{x}) = s\}|$
20         if $l == L1$ then
21             $v_m(\mathbf{x}) \leftarrow v_m(\mathbf{x}) + Lap(1/\gamma)$
22     $f(\mathbf{x}) = \arg\max_m v_m(\mathbf{x})$
23 Train the final model $F$ on dataset $\{(\mathbf{x}, f(\mathbf{x}))\}_{\mathbf{x} \in \mathcal{D}_{aux}}$.
```

If the student models of a party make the same prediction on an example, we take their predictions into account. Otherwise, the party is not confident at predicting this example and thus we ignore the predictions of its student models. Formally, given an example $\mathbf{x} \in \mathcal{D}_{aux}$, we first compute the vote count of class m on the student models of party i as $v^i_m(\mathbf{x}) = |\{k : k \in [s], U^i_k(\mathbf{x}) = m\}|$. Next, with consistent voting, the final vote count of class m on all parties is computed as $v_m(\mathbf{x}) = s \cdot |\{i : i \in [n], v^i_m(\mathbf{x}) = s\}|$.

Differentially private versions of FedKT. FedKT can easily satisfy differential privacy guarantees by providing differentially private prediction results to the query dataset. Given the privacy parameter γ, we can add noises to the vote count histogram such that $f(\mathbf{x}) = \arg\max_m\{v_m(\mathbf{x}) + Lap(\frac{1}{\gamma})\}$, where $Lap(\frac{1}{\gamma})$ is the noises generated from Laplace distribution with location 0 and scale $\frac{1}{\gamma}$. Note that we do not need to add noises on both the parties and the server. For the $L1$ setting, we only need to add noises on the server side. The parties can train and send non-differentially private student models to the server. For the $L2$ setting, we only need to add noises on the party side so that the student models are differentially private. Then, the final model naturally satisfies differential privacy guarantees. More analysis on privacy loss will be presented in Section 4. Algorithm 1 shows the whole training process of FedKT. In the algorithm, for each party (i.e., Line 1) and its each partition (i.e., Line 3), we train a student model using knowledge transfer (i.e., Lines 4-12). The student models are sent to the server. Then, the server trains the final model using knowledge transfer again (i.e., Lines 14-23). For different privacy level settings, we have the corresponding noises injection operations on the server side (i.e., Lines 20-21) or the party side (i.e., Lines 9-10).

Communication/Computation overhead of FedKT. Suppose the size of each model is M. Then, the total communication size of FedKT is nsM for sending the student models to the server. Suppose the number of communication rounds in FedAvg is r and all the parties participate in the training in every iteration. Then the total communication size of FedAvg is $2nMr$ including the server sends the global model to the parties [1] and the parties send the local models to the server. Thus, when $r > \frac{s}{2}$, the communication cost of FedAvg is higher than FedKT. This value can be quite small, e.g., $r = 2$ if we set $s = 2$. Moreover, when $s = 2$, FedAvg has the same communication cost with FedKT in the first round. The computation overhead of FedKT is usually larger than FedAvg in each round since FedKT needs to train multiple teacher and student models. However, the computation overhead is acceptable in the cross-silo setting, where the parties (e.g., companies, data centers) usually have a relatively large computation power.

4 Data-Dependent Privacy Analysis of FedKT

In this section, we use the moments accountant method [Abadi *et al.*, 2016] to track the privacy loss in the training process. For $L1$ setting, we mainly consider the party-level differential privacy [Geyer *et al.*, 2017; McMahan *et al.*, 2018], which is more attractive in the federated setting. Instead of aiming to protect a single record, party-level differential privacy ensures that the learned model does not reveal whether a party participated in federated learning or not. For $L2$ setting, we mainly consider the example-level differential privacy since it is not practical to require each local model to satisfy party-level differential privacy. For proofs of the theorems in this section, please refer to Section A of Appendix.

FedKT-L1. Considering we change the whole dataset of a party, then at most s student models will be influenced. Thus, on the server side, the sensitivity of the vote count histogram is $2s$ (i.e., the vote count of a class increases by s and the vote count of another class decreases by s). According to the Laplace mechanism, we have the following theorem.

[1] It also happens in the first round to ensure all parties have the same initialized model [McMahan and others, 2016].

Theorem 1. Let $\mathcal{M}$ be the f function executed on the server side. Given the number of partitions s and the privacy parameter γ, $\mathcal{M}$ satisfies $(2s\gamma, 0)$ party-level differential privacy.

Similar with [Papernot *et al.*, 2017], we conduct a data-dependent privacy analysis for FedKT with the moments accountant method.

Theorem 2. Let $\mathcal{M}$ be $(2s\gamma, 0)$ party-level differentially private. Let $q \geq \Pr[\mathcal{M}(d) \neq o^*]$ for some outcome o^*. Let $l, \gamma \geq 0$ and $q < \frac{e^{2s\gamma}-1}{e^{4s\gamma}-1}$. For any aux and any two party-adjacent datasets d and d', $\mathcal{M}$ satisfies

$$\alpha_{\mathcal{M}}(l; \mathsf{aux}, d, d') \leq \min(\log((1-q)\Big(\frac{1-q}{1-e^{2s\gamma}q}\Big)^l + qe^{2s\gamma l}),\\ 2s^2\gamma^2 l(l+1)).$$

Here $\Pr[\mathcal{M}(d) \neq o^*]$ can be bounded by Lemma 4 of [Papernot *et al.*, 2017].

With Theorem 2, we can track the privacy loss of each query [Abadi *et al.*, 2016].

FedKT-L2. For each partition on the party side, we add Laplace noises to the vote counts, which is same with the PATE approach. Thus, we have the following theorem.

Theorem 3. Let $\mathcal{M}$ be the f function executed on each partition of a party. Let $q \geq \Pr[\mathcal{M}(d) \neq o^*]$ for some outcome o^*. Let $l, \gamma \geq 0$ and $q < \frac{e^{2\gamma}-1}{e^{4\gamma}-1}$. For any aux and any two adjacent datasets d and d', $\mathcal{M}$ satisfies

$$\alpha_{\mathcal{M}}(l; \mathsf{aux}, d, d') \leq \min(\log((1-q)\Big(\frac{1-q}{1-e^{2\gamma}q}\Big)^l + qe^{2\gamma l}),\\ 2\gamma^2 l(l+1)).$$

After computing the privacy loss of each party, we can use the parallel composition to compute the privacy loss of the final model.

Theorem 4. Suppose the student models of party P_i satisfy (ε_i, δ)-differential privacy. Then, the final model satisfies $(\max_i \varepsilon_i, \delta)$-differential privacy.

Note that the above privacy analysis is data-dependent. Thus, the final privacy budget is also data-dependent and may have potential privacy breaches if we publish the budget. Like previous studies [Papernot *et al.*, 2017; Jordon *et al.*, 2019], we report the data-dependent privacy budgets in the experiments. As future work, we plan to use the smooth sensitivity algorithm [Nissim *et al.*, 2007] to add noises to the privacy losses. Also, we may get a tighter bound of the privacy loss if adopting the Gaussian noises [Papernot and others, 2018].

5 Experiments

To evaluate FedKT, we conduct experiments on four public datasets: (1) A random forest on *Adult* dataset. The number of trees is set to 100 and the maximum tree depth is set to 6. (2) A gradient boosting decision tree (GBDT) model on *cod-rna* dataset. The maximum tree depth is set to 6. (3) A multilayer perceptron (MLP) with two hidden layers on *MNIST* dataset. Each hidden layer has 100 units using ReLu activations. (4) A CNN on extended *SVHN* dataset. The CNN has two 5x5 convolution layers followed with 2x2 max pooling (the first with 6 channels and the second with 16 channels), two fully connected layers with ReLu activation (the first with 120 units and the second with 84 units), and a final softmax output layer. For the first two datasets, we split the original dataset at random into train/test/public sets with a 75%/12.5%/12.5% proportion. For MNIST and SVHN, we use one half of the original test dataset as the public dataset and the remaining as the final test dataset. Like many existing studies [Yurochkin *et al.*, 2019; Lin *et al.*, 2020; Li *et al.*, 2021], we use the Dirichlet distribution to simulate the heterogeneous data partition among the parties. Suppose there are n parties. We sample $p_k \sim Dir_n(\beta)$ and allocate a $p_{k,j}$ proportion of the instances of class k to party j, where $Dir(\beta)$ is the Dirichlet distribution with a concentration parameter β (0.5 by default). By default, we set the number of parties to 50 for Adult and cod-rna and to 10 for MNIST and SVHN. We set s to 2 and t to 5 by default for all datasets. For more details in the experimental settings and study of the hyper-parameters, please refer to Section B.1 and Section B.2 of Appendix. The code is publicly available [2].

We compare FedKT with eight baselines: (1) SOLO: each party trains its model locally and does not participate in federated learning. (2) FedAvg [McMahan and others, 2016]; (3) FedProx [Li *et al.*, 2020b]; (4) SCAFFOLD [Karimireddy and others, 2020]; (5) FedDF [Lin *et al.*, 2020]; (6) PNFM [Yurochkin *et al.*, 2019]; (7) PATE [Papernot *et al.*, 2017]: we use the PATE framework to train a final model on all data in a centralized setting (i.e., only a single party with the whole dataset) without adding noises. This method defines an upper bound of learning a final model using knowledge transfer with public unlabelled data. (8) XGBoost [Chen and Guestrin, 2016]: the XGBoost algorithm for the GBDT model on the whole dataset in a centralized setting. This method defines an upper bound of learning the GBDT model. Here approaches (2)-(5) are popular or state-of-the-art federated learning algorithms and approach (6) is an one-shot algorithm. Moreover, same as FedKT, approaches (5) and (7) also utilize the unlabeled public dataset.

5.1 Effectiveness

Table 1 shows the accuracy of FedKT[3] compared with the other baselines. For fair comparison and practical usage in model markets, we run all approaches for a single communication round. For SOLO, we report the average accuracy of the parties. From this table, we have the following observations. First, except for FedKT, the other federated learning algorithms can only learn specific models. FedKT is able to learn all the studied models including trees and neural networks. Second, FedKT can achieve much better performance than the other federated learning algorithms running with a single round. FedKT can achieve about 6.5% higher accu-

[2]https://github.com/QinbinLi/FedKT

[3]For simplicity, we use FedKT to denote FedKT-L0, unless specified otherwise.

Datasets	FedKT	SOLO	FedAvg	FedProx	SCAFFOLD	FedDF	PNFM	PATE	XGBOOST
Adult	**82.2%** ± 0.6%	68.6%	\					83.5%	\
cod-rna	**88.3%** ± 0.6%	65.0%						91.1%	91.2%
MNIST	**90.5%** ± 0.3%	69.0%	62.8%	44.3%	51.7%	83.8%	65.9%	92.7%	\
SVHN	**83.2%** ± 0.4%	62.8%	26.8%	20.1%	16.2%	77.2%	\	86.6%	\

Table 1: The test accuracy comparison between FedKT and the other baselines in a single round.

datasets	FedKT-L1					FedKT-L2				
	γ	#queries	ε	acc	non-private acc	γ	#queries	ε	acc	non-private acc
Adult	0.04	0.5%	2.56	76.8%	82.2%	0.04	0.5%	2.59	77.6%	82.4%
	0.04	1.0%	4.73	80.2%		0.04	1.0%	3.72	78.6%	
cod-rna	0.06	0.50%	5.48	82.6%	88.0%	0.05	1.0%	4.51	82.7%	89.7%
	0.1	0.5%	6.89	84.7%		0.05	2.0%	9.78	84.7%	

Table 2: The privacy loss and test accuracy of FedKT-L1 and FedKT-L2 given different γ and number of queries. L0 acc is the test accuracy of FedKT-L0. The failure probability δ is set to 10^{-5}.

racy than FedDF, which also utilizes the public dataset. Third, although PNFM is an one-shot algorithm specially designed for MLPs, FedKT outperforms PNFM about 25% accuracy on MNIST. Last, the accuracy of FedKT is close to PATE and XGBoost, which means our design has little accuracy loss compared with the centralized setting. The gap between FedKT and the upper bound is very small.

5.2 Privacy

We run FedKT with different γ and number of queries. When running FedKT-L1, we tune the percentage of number of queries on the server side. When running FedKT-L2, on the contrary, we tune the percentage of number of queries on the party side. The selected results on Adult and cod-rna are reported in Table 2. While differentially private FedKT does not need any knowledge on the model architecture, the accuracy is still comparable to the non-private version given a privacy budget less than 10. For more results, please refer to Section B.4 of Appendix.

5.3 Size of the Public Dataset

We show the performance of FedKT with different size of public dataset. The results are shown in Table 3. We can observe that FedKT is stable even though reducing the size of the public dataset. The accuracy decreases no more than 1% and 2% using only 20% of the public dataset on cod-rna and MNIST (i.e., 1807 examples on cod-rna and 1000 examples on MNIST), respectively. Moreover, the accuracy is almost unchanged on Adult.

5.4 Extend to Multiple Rounds

While FedKT is an one-shot algorithm, it is still applicable in the scenarios where multiple rounds are allowed. FedKT can be used as an initialization step to learn a global model in the first round. Then the parties can use the global model to conduct iterative federated learning algorithms. By combining FedKT with FedProx (denoted as FedKT-Prox), FedKT-Prox is much more communication-efficient than the other algorithms. FedKT-Prox needs about 11 rounds to achieve 87% accuracy, while the other approaches need at least 32 rounds. For more details, please refer to Section B.3 of Appendix.

Datasets	Portion of the public dataset used in training				
	20%	40%	60%	80%	100%
Adult	82.1%	82.1%	82.1%	82.3%	82.2%
cod-rna	87.3%	87.6%	87.8%	87.8%	88.3%
MNIST	89.3%	89.7%	90.2%	90.3%	90.5%
SVHN	80.1%	81.3%	82.1%	82.9%	83.2%

Table 3: The test accuracy of FedKT with different size of the public dataset. We vary the portion of the public dataset used in the training from 20% to 100%.

6 Conclusions

Motivated by the rigid multi-round training of current federated learning algorithms and emerging applications like model markets, we propose FedKT, a one-shot federated learning algorithm for the cross-silo setting. Our experiments show that FedKT can learn different models with a much better accuracy then the other state-of-the-art algorithms with a single communication round. Moreover, the accuracy of differentially private FedKT is comparable to the non-differentially private version with a modest privacy budget. Overall, FedKT is a practical one-shot solution for model-based sharing in cross-silo federated learning.

Acknowledgements

This research is supported by the National Research Foundation, Singapore under its AI Singapore Programme (AISG Award No: AISG2-RP-2020-018). Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not reflect the views of National Research Foundation, Singapore.

References

[Abadi *et al.*, 2016] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *ACM CCS*, 2016.

[Baylor and others, 2017] Denis Baylor et al. Tfx: A tensorflow-based production-scale machine learning platform. In *ACM SIGKDD*, 2017.

[Chang *et al.*, 2019] Hongyan Chang, Virat Shejwalkar, Reza Shokri, and Amir Houmansadr. Cronus: Robust and heterogeneous collaborative learning with black-box knowledge transfer. *arXiv:1912.11279*, 2019.

[Chen and Guestrin, 2016] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *ACM SIGKDD*, 2016.

[Choquette-Choo *et al.*, 2021] Christopher A. Choquette-Choo, Natalie Dullerud, Adam Dziedzic, Yunxiang Zhang, Somesh Jha, Nicolas Papernot, and Xiao Wang. Capc learning: Confidential and private collaborative learning. In *ICLR*, 2021.

[Dwork *et al.*, 2014] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 2014.

[Fredrikson *et al.*, 2015] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *ACM CCS*, 2015.

[Geyer *et al.*, 2017] Robin C Geyer, Tassilo Klein, and Moin Nabi. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557*, 2017.

[Guha *et al.*, 2019] Neel Guha, Ameet Talwlkar, and Virginia Smith. One-shot federated learning. *arXiv preprint arXiv:1902.11175*, 2019.

[He *et al.*, 2020] Chaoyang He, Murali Annavaram, and Salman Avestimehr. Group knowledge transfer: Federated learning of large cnns at the edge. *NeurIPS*, 33, 2020.

[Hinton *et al.*, 2015] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[Jordon *et al.*, 2019] James Jordon, Jinsung Yoon, and Mihaela van der Schaar. Differentially private bagging: Improved utility and cheaper privacy than subsample-and-aggregate. In *NeurIPS*, 2019.

[Kairouz *et al.*, 2019] Peter Kairouz, H Brendan McMahan, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

[Karimireddy and others, 2020] Sai Praneeth Karimireddy et al. Scaffold: Stochastic controlled averaging for on-device federated learning. In *ICML*. PMLR, 2020.

[Kasturi *et al.*, 2020] Anirudh Kasturi, Anish Reddy Ellore, and Chittaranjan Hota. Fusion learning: A one shot federated learning. In *ICCS*. Springer, 2020.

[Li and Wang, 2019] Daliang Li and Junpu Wang. Fedmd: Heterogenous federated learning via model distillation. *arXiv preprint arXiv:1910.03581*, 2019.

[Li *et al.*, 2019] Qinbin Li, Zeyi Wen, Zhaomin Wu, Sixu Hu, Naibo Wang, Yuan Li, Xu Liu, and Bingsheng He. A survey on federated learning systems: vision, hype and reality for data privacy and protection. *arXiv preprint arXiv:1907.09693*, 2019.

[Li *et al.*, 2020a] Qinbin Li, Zeyi Wen, and Bingsheng He. Practical federated gradient boosting decision trees. In *AAAI*, 2020.

[Li *et al.*, 2020b] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. In *MLSys*, 2020.

[Li *et al.*, 2021] Qinbin Li, Bingsheng He, and Dawn Song. Model-contrastive federated learning. In *CVPR*, 2021.

[Lin *et al.*, 2020] Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. *NeurIPS*, 33, 2020.

[McMahan and others, 2016] H Brendan McMahan et al. Communication-efficient learning of deep networks from decentralized data. *arXiv:1602.05629*, 2016.

[McMahan *et al.*, 2018] H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. In *ICLR*, 2018.

[Nissim *et al.*, 2007] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth sensitivity and sampling in private data analysis. In *ACM STOC*, 2007.

[Papernot and others, 2018] Nicolas Papernot et al. Scalable private learning with pate. In *ICLR*, 2018.

[Papernot *et al.*, 2017] Nicolas Papernot, Martín Abadi, Úlfar Erlingsson, Ian Goodfellow, and Kunal Talwar. Semi-supervised knowledge transfer for deep learning from private training data. In *ICLR*, 2017.

[Shokri *et al.*, 2017] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *IEEE SP*, 2017.

[Vartak and others, 2016] Manasi Vartak et al. Modeldb: a system for machine learning model management. In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics*, 2016.

[Yang *et al.*, 2019] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM TIST*, 2019.

[Yurochkin *et al.*, 2019] Mikhail Yurochkin, Mayank Agarwal, Soumya Ghosh, Kristjan Greenewald, Nghia Hoang, and Yasaman Khazaeni. Bayesian nonparametric federated learning of neural networks. In *ICML*. PMLR, 2019.

[Zhou *et al.*, 2020] Yanlin Zhou, George Pu, Xiyao Ma, Xiaolin Li, and Dapeng Wu. Distilled one-shot federated learning. *arXiv preprint arXiv:2009.07999*, 2020.

[Zhu *et al.*, 2020] Yuqing Zhu, Xiang Yu, Yi-Hsuan Tsai, Francesco Pittaluga, Masoud Faraki, Yu-Xiang Wang, et al. Voting-based approaches for differentially private federated learning. *arXiv preprint arXiv:2010.04851*, 2020.