CaPriDe Learning: Confidential and Private Decentralized Learning based on Encryption-friendly Distillation Loss

Nurbek Tastan Karthik Nandakumar
Mohamed bin Zayed University of Artificial Intelligence, UAE
{nurbek.tastan,karthik.nandakumar}@mbzuai.ac.ae

Abstract

Large volumes of data required to train accurate deep neural networks (DNNs) are seldom available with any single entity. Often, privacy concerns prevent entities from sharing data with each other or with a third-party learning service provider. While cross-silo federated learning (FL) allows collaborative learning of large DNNs without sharing the data itself, most existing cross-silo FL algorithms have an unacceptable utility-privacy trade-off. In this work, we propose a framework called ***C****onfidential* ***a****nd* ***Pri****vate* ***De****centralized (CaPriDe) learning, which optimally leverages the power of fully homomorphic encryption (FHE) to enable collaborative learning without compromising on the confidentiality and privacy of data. In CapridDe learning, participating entities release their private data in an encrypted form allowing other participants to perform inference in the encrypted domain. The crux of CaPriDe learning is mutual knowledge distillation between multiple local models through a novel distillation loss, which is an approximation of the Kullback-Leibler (KL) divergence between the local predictions and encrypted inferences of other participants on the same data that can be computed in the encrypted domain. Extensive experiments on three datasets show that CaPriDe learning can improve the accuracy of local models without any central coordination, provide strong guarantees of data confidentiality and privacy, and has the ability to handle statistical heterogeneity. Constraints on the model architecture (arising from the need to be FHE-friendly), limited scalability, and computational complexity of encrypted domain inference are the main limitations of the proposed approach. The code can be found at https://github.com/tnurbek/capride-learning.*

1. Introduction

Rapid strides have been made in machine learning (ML) (in particular, deep learning) over the past decade. However, in many important application domains such as healthcare and finance, the absence of large, centralized datasets is a significant obstacle to realizing the full benefits of deep learning algorithms. Data in these applications often resides in silos and is governed by strict regulations (e.g., HIPAA, GDPR, etc.) because of its privacy sensitive nature [22]. Competing business interests of data owners and the lack of appropriate incentives for data sharing further accentuate the problem. To overcome these issues, there is a need for collaborative learning algorithms that ideally satisfy the following requirements [15]: (i) ***confidentiality*** - no sharing of raw data, (ii) ***privacy*** - minimal leakage of information via the knowledge exchange mechanism (e.g., gradients or predictions), (iii) ***utility*** - gain in accuracy (over the individual models) resulting from the collaboration, even in the presence of statistical heterogeneity, (iv) ***efficiency*** - minimize computational complexity and communication burden, (v) ***robustness*** - handle unintentional failures and attacks emanating from malicious entities, and (vi) ***fairness*** - utility should be proportional to the individual contributions.

Federated learning (FL) [15] is a special case of collaborative learning, which works under the orchestration of a central server. FL allows multiple entities to collaboratively solve a ML problem by sharing of focused updates (e.g., gradients), instead of raw data. Specifically, cross-silo FL (typically between 2-100 participants) has been touted as a promising solution to address the data fragmentation problem in finance [8] and healthcare [16, 23]. Most prior cross-silo FL algorithms assume that all the parties are collectively training a *single model with a common architecture*, which is too restrictive in practice. Furthermore, knowledge exchange usually happens through *sharing of gradients or model parameters*. Recent gradient inversion attacks [9, 14] demonstrate that it is indeed possible to recover high fidelity information from individual gradient updates, thus violating the privacy requirement. While differential privacy (DP) [24], secure multi-party computation (MPC) [26], and trusted execution environment (TEE) [21] have been proposed as potential remedies to safeguard privacy in FL, none of the existing solutions offer an acceptable privacy-utility trade-off with reasonable efficiency. As noted in [5], sharing of high dimensional gradients/model parameters is a fundamental privacy limitation of standard FL methods, which cannot be addressed by simply wrapping around a DP, MPC, or TEE solution.

Confidential and private collaborative learning (CaPC) [7] is the only method that claims to achieve both confidentiality and privacy in a collaborative setting. In CaPC learning, each participant is able to query other parties via a private inference protocol based on 2-party computation (2PC). However, the answering parties cannot directly reveal the inference logits to the querying party because it leaks information about their local models (e.g., through model extraction attacks). To circumvent this problem, differential privacy, private aggregation of teacher ensembles, and secure argmax computation through a central privacy guardian (PG) are employed to output only the predicted label to the querying party. A drawback of the CaPC approach is that it achieves all privacy guarantees using the help of a semi-trusted PG. Moreover, the use of differential privacy reduces the performance of FL algorithms, especially if strong privacy guarantees are required. Finally, the use of MPC requires multiple rounds of communication between the parties for each query. All other approaches that have been proposed to achieve knowledge transfer through distillation [18] require a non-private labeled/unlabeled/synthetic dataset that can be shared among participants.

In this work, we propose a new protocol called **C**onfidential **a**nd **Pri**vate **De**centralized (CaPriDe) learning, where participants learn from each other in a collaborative setting while preserving their confidentiality and privacy (see Figure 1). Unlike the 2PC protocols used in CaPC, we leverage fully homomorphic encryption (FHE) to enable participants to publish their encrypted data. Since the published data is encrypted using the data owner's public key and only the owner can decrypt the data (using the secret key), confidentiality is preserved. However, the CaPriDe framework allows other collaborators to perform encrypted inference on the published (encrypted) data by applying their own local model. Mutual knowledge distillation (KD) [13] between the data owner's local model and the local models of the collaborators is used to transfer knowledge between models and ensure a collaborative gain. KD is typically achieved through minimizing the distillation loss between the student and teacher responses (logits). Since the collaborators in CaPriDe learning make predictions on encrypted data, a loss function that can be computed without any decryption is required. Hence, we propose a new distillation loss, which is an approximation of the KL divergence that can be securely computed. Only an encrypted value of the distillation loss aggregated over an entire batch is sent back to the data owner, which ensures strong privacy. Data owners can decrypt this aggregate loss value and use it for updating their local models. Our contributions are:

- We introduce the CaPriDe learning framework, which exploits FHE-based encrypted inference and knowledge distillation to achieve confidential and private collaborative learning without any central orchestration and any need for non-private shared data.
- To enable CaPriDe learning, we propose an encryption-friendly distillation loss that estimates the *approximate* KL divergence between two model predictions and design a protocol to securely compute this loss in the encrypted domain.
- We conduct extensive experiments to show that CaPriDe learning enables participants to achieve collaborative gain, even in the non-iid setting. To prove feasibility, we implement encrypted inference using the Tile Tensors library with a FHE-friendly model.

2. Background

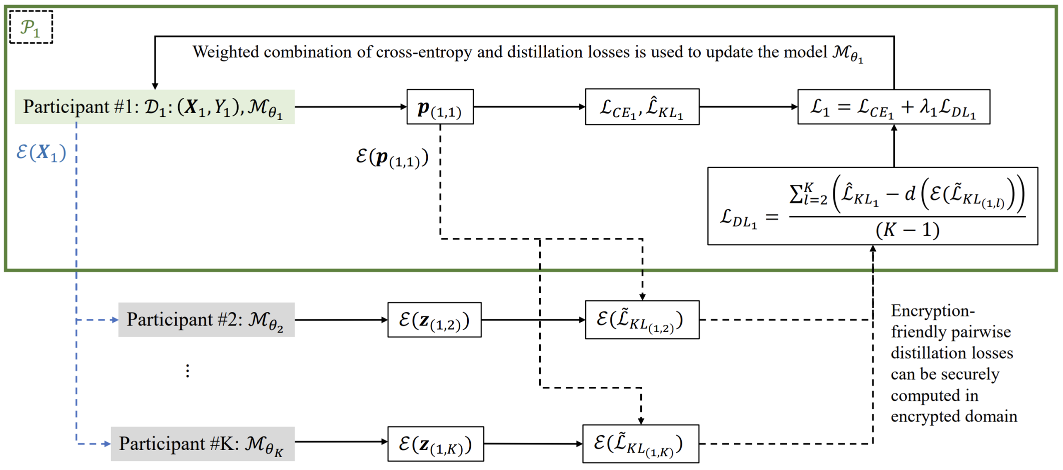
Federated learning (FL) was first proposed in [20] as a distributed ML algorithm that does not use user data directly. Subsequent works attempted to address various challenges in FL, including privacy [7, 19], communication efficiency [12], convergence, catastrophic forgetting [25], and secure aggregation [3]. Though several privacy-enhancing solutions have been proposed, most of them fail to provide a satisfactory balance between accuracy and privacy. Only recently, some promising solutions such as CaPC have been proposed [7]. These advancements have been made by jettisoning the conventional gradient sharing approach and borrowing ideas from the knowledge distillation literature [13]. Specifically, it has been shown that it is possible for an ML model to benefit from predictions made by other models, by aligning the local predictions with those obtained from external parties. Deep mutual learning (DML) [27] is an important work in this direction, where models learn collaboratively and teach each other throughout the training process. Compared to the distillation by a pre-defined static teacher network, DML achieves better performance, and mutually learned models outperform independently trained models irrespective of model capacity. Knowledge Distillation via Collaborative Learning (KDCL) [11], Cronus [5], and Ensemble Distillation [18] also follow a similar idea.

2.1. Fully Homomorphic Encryption

Let $\mathcal{E}(x)$ denote the encrypted value of x using the public key κ. The encryption scheme is called homomorphic over an operation $\oplus$ if it supports the following equation:

$$\mathcal{E}(x_1 \oplus x_2) = \mathcal{E}(x_1) \otimes \mathcal{E}(x_2), \forall x_1, x_2$$

where operators $\otimes$ and $\oplus$ need not be the same. The scheme is fully homomorphic if for some arbitrary function



$\mathcal{P}_1$
Weighted combination of cross-entropy and distillation losses is used to update the model $\mathcal{M}_{\theta_1}$
Participant #1: $\mathcal{D}_1$: $(\boldsymbol{X}_1, Y_1)$, $\mathcal{M}_{\theta_1}$
$\boldsymbol{p}_{(1,1)}$
$\mathcal{L}_{CE_1}, \hat{\mathcal{L}}_{KL_1}$
$\mathcal{L}_1 = \mathcal{L}_{CE_1} + \lambda_1 \mathcal{L}_{DL_1}$
$\mathcal{E}(\boldsymbol{X}_1)$
$\mathcal{E}(\boldsymbol{p}_{(1,1)})$
$\mathcal{L}_{DL_1} = \frac{\sum_{l=2}^{K}\left(\hat{\mathcal{L}}_{KL_1} - d\left(\mathcal{E}(\tilde{\mathcal{L}}_{KL_{(1,l)}})\right)\right)}{(K-1)}$
Participant #2: $\mathcal{M}_{\theta_2}$
$\mathcal{E}(\boldsymbol{z}_{(1,2)})$
$\mathcal{E}(\tilde{\mathcal{L}}_{KL_{(1,2)}})$
Participant #K: $\mathcal{M}_{\theta_K}$
$\mathcal{E}(\boldsymbol{z}_{(1,K)})$
$\mathcal{E}(\tilde{\mathcal{L}}_{KL_{(1,K)}})$
Encryption-friendly pairwise distillation losses can be securely computed in encrypted domain

Figure 1. Illustration of confidential and private decentralized (CaPriDe) learning framework for K participants, showing the learning process for only participant $\mathcal{P}_1$. Here, $\mathcal{D}_1 = (\boldsymbol{X}_1, Y_1) = \{\boldsymbol{x}_{j,1}, y_{j,1}\}_{j=1}^{N_1}$ is the local data of $\mathcal{P}_1$, $\mathcal{E}(\boldsymbol{X}_1)$ denotes the collection of encrypted unlabeled samples of $\mathcal{P}_1$, and $\boldsymbol{p}_{(1,l)}$ and $\boldsymbol{z}_{(1,l)}$ are the prediction probabilities and logits obtained by applying model $\mathcal{M}_{\theta_l}$ of participant $\mathcal{P}_l$ to $\boldsymbol{X}_1$. Black dashed line represents exchange of encrypted data between participants in each round, and blue dashed line denotes a single transfer at the beginning of the protocol. $\mathcal{L}_{CE}$ and $\mathcal{L}_{DL}$ represent the cross-entropy and distillation losses, respectively.

$f(x_1, x_2, ..., x_n)$, there exists a function g and decryption method d such that:

$$f(x_1, x_2, ..., x_n) = d(g(\mathcal{E}(x_1), \mathcal{E}(x_2), ..., \mathcal{E}(x_n))),$$

where d uses secret key $\tilde{\kappa}$ to decrypt the given ciphertext value. Gentry [10] first constructed somewhat homomorphic encryption (SWHE) and later developed bootstrapping of ciphertexts to reduce noise. Among the various FHE schemes such as BGV, BFV, and CKKS [6], only CKKS works with real numbers and approximate computations.

2.2. Preliminaries

Typically, a supervised classifier $\mathcal{M}$ is mapping from the input space $\boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^n$ to the label space $y \in \mathcal{Y} = \{1, 2, \cdots, M\}$, where M is the number of classes. For convenience, we assume that $\mathcal{M} : \mathcal{X} \to \mathcal{Z}$, where $\mathcal{Z}$ denotes the logits space ($\mathcal{Z} \subset \mathbb{R}^M$). In other words, $\mathcal{M}$ maps the input $\boldsymbol{x}$ to a M-dimensional logits vector $\boldsymbol{z} = [z^1, z^2, \cdots, z^M]$. A softmax function ($\sigma$) can be applied to the logits vector $\boldsymbol{z}$ to obtain the probability distribution $\boldsymbol{p} = \sigma(\boldsymbol{z}) = [p^1, p^2, \cdots, p^M]$ over M labels, i.e.,

$$p^m = \frac{\exp(z^m/T)}{C}, \tag{1}$$

where T is the softmax temperature parameter and $C = \sum_{i=1}^{M} \exp(z^i/T)$ is a normalization constant that ensures $\sum_{m=1}^{M} p^m = 1$. Here, p^m denotes the probability that input $\boldsymbol{x}$ belongs to class $m, m = 1, 2, \cdots, M$. Given two V-dimensional vectors $\boldsymbol{v}_1, \boldsymbol{v}_2 \in \mathbb{R}^V$, let $(\boldsymbol{v}_1 \cdot \boldsymbol{v}_2)$ denote the dot product, $(\boldsymbol{v}_1 \odot \boldsymbol{v}_2)$ denote the element-wise product, and $f(\boldsymbol{v}_1)$ represent the vector obtained by applying the function f to each element in $\boldsymbol{v}_1$. The cross entropy loss of model $\mathcal{M}$ on a sample $(\boldsymbol{x}, y)$ is defined as:

$$L_{CE}(\boldsymbol{p}, y) = -\boldsymbol{1}_y \cdot log(\boldsymbol{p}),$$

where $\boldsymbol{p} = \sigma(\mathcal{M}(\boldsymbol{x}))$ and $\boldsymbol{1}_y$ is an indicator vector whose y^{th} element is 1 and other elements are 0. The KL divergence between two distributions $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ is given by:

$$D_{KL}(\boldsymbol{p}_2 || \boldsymbol{p}_1) = \sum_{m=1}^{M} p_2^m \log \frac{p_2^m}{p_1^m} = (\boldsymbol{p}_2 \cdot \log \boldsymbol{p}_2) - (\boldsymbol{p}_2 \cdot \log \boldsymbol{p}_1). \tag{2}$$

The L2 distance between two logit vectors $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$ is:

$$D_{L2}(\boldsymbol{z}_2, \boldsymbol{z}_1) = \sum_{m=1}^{M} (z_2^m - z_1^m)^2 = (\boldsymbol{z}_2 - \boldsymbol{z}_1) \cdot (\boldsymbol{z}_2 - \boldsymbol{z}_1).$$

3. Proposed Framework

3.1. Problem Statement

Suppose that there are K participants $\mathcal{P}_1, \mathcal{P}_2, \cdots, \mathcal{P}_K$. Each party $\mathcal{P}_k, k = 1, 2, \cdots, K$ has its own local training dataset $\mathcal{D}_k = \{\boldsymbol{x}_{j,k}, y_{j,k}\}_{j=1}^{N_k}$ of size N_k using which it learns its own local ML model $\mathcal{M}_{\theta_k}$, where $\mathcal{M}$ denotes the architecture[1] and θ represents the parameter set. We also assume that each party $\mathcal{P}_k$ has its own validation set $\tilde{\mathcal{D}}_k = \{\tilde{\boldsymbol{x}}_{j,k}, \tilde{y}_{j,k}\}_{j=1}^{\tilde{N}_k}$ of size $\tilde{N}_k$ and the validation accuracy of model $\mathcal{M}_{\theta_k}$ on $\tilde{\mathcal{D}}_k$ is $\mathcal{A}_k$. The goal of each participant is to collaborate with other parties in order to ***improve***

[1]For simplicity of notation, we use the same symbol $\mathcal{M}$ to denote the model architecture of all the K participants. The proposed framework does *not* require all the parties to have the same model architecture.

the accuracy of its local model, i.e., obtain a better model $\mathcal{M}_{\theta_k^*}$ that has a higher validation accuracy $\mathcal{A}_k^*$ on $\tilde{\mathcal{D}}_k$ compared to $\mathcal{A}_k$, without compromising on the confidentiality and privacy of its local data $\mathcal{D}_k$. In other words, no participant $\mathcal{P}_l, l \neq k$ can access $\mathcal{D}_k$ in its plaintext form or learn any private information about $\mathcal{D}_k$ such as the $\boldsymbol{x}_{j,k}$ and $y_{j,k}$ values. The utility of the framework is measured as the average collaboration gain, i.e., $U = \frac{1}{K}\sum_{k=1}^{K}(\mathcal{A}_k^* - \mathcal{A}_k)$.

3.2. CaPriDe Learning Framework Description

Let $\mathcal{E}(\boldsymbol{X}_k) = \{\mathcal{E}(\boldsymbol{x}_{j,k})\}_{j=1}^{N_k}$ be the collection of encrypted input samples of participant $\mathcal{P}_k$. Here, the encryption is based on a FHE scheme with public key κ_k of $\mathcal{P}_k$. In CaPriDe learning, each party initiates the collaboration by publishing $\mathcal{E}(\boldsymbol{X}_k)$ to the other participants. We now describe how the learning proceeds at participant $\mathcal{P}_k$. Let $\boldsymbol{z}_{j,(k,l)}$ and $\boldsymbol{p}_{j,(k,l)}$ denote the logits vector and probability distribution obtained when model $\mathcal{M}_{\theta_l}$ of $\mathcal{P}_l$ is applied on the input sample $\boldsymbol{x}_{j,k}$, i.e., the j^{th} training sample of $\mathcal{P}_k$. Assume that we are at round $(t+1), 0 \leq t < \tau$, of the collaboration and the current model of $\mathcal{P}_k$ is $\mathcal{M}_{\theta_k^t}$. $\mathcal{P}_k$ performs one forward pass on its own training set $\mathcal{D}_k$ to obtain the predictions $\{\boldsymbol{p}_{j,(k,k)}\}_{j=1}^{N_k}$. The cross-entropy loss of party $\mathcal{P}_k$ can be obtained as:

$$\mathcal{L}_{CE_k} = \sum_{j=1}^{N_k} L_{CE}\left(\boldsymbol{p}_{j,(k,k)}, y_{j,k}\right). \tag{3}$$

To enable knowledge transfer between participants, we make use of the knowledge distillation (KD) approach [13], where a student model learns to mimic the predictions of a teacher model. In CaPriDe learning, there is no designated teacher model, and mutual KD between multiple peer models is used for knowledge transfer. Let $\mathcal{L}_{DL_{(k,l)}}$ denote the pairwise distillation loss between predictions of $\mathcal{P}_k$ and $\mathcal{P}_l$, which is defined as:

$$\mathcal{L}_{DL_{(k,l)}} = \sum_{j=1}^{N_k} D_{DL}\left(\boldsymbol{p}_{j,(k,k)}, \boldsymbol{p}_{j,(k,l)}\right), \tag{4}$$

where $D_{DL}(\boldsymbol{p_1}, \boldsymbol{p_2})$ denotes the mimic distance between two predictions $\boldsymbol{p_1}$ and $\boldsymbol{p_2}$. The total distillation loss for $\mathcal{P}_k$ can be computed as:

$$\mathcal{L}_{DL_k} = \frac{1}{K-1}\sum_{l=1, l\neq k}^{K} \mathcal{L}_{DL_{(k,l)}}. \tag{5}$$

Therefore, the overall loss of $\mathcal{P}_k$ is given by:

$$\mathcal{L}_k = \mathcal{L}_{CE_k} + \lambda_k \mathcal{L}_{DL_k}, \tag{6}$$

where λ_k is the weighting factor between the cross-entropy and distillation losses for $\mathcal{P}_k$. The model of participant $\mathcal{P}_k$ is then updated through a backward pass as follows:

$$\theta_k^{t+1} = \theta_k^t - \alpha_k \nabla_{\theta_k^t} \mathcal{L}_k, \tag{7}$$

where α_k is the learning rate of $\mathcal{P}_k$.

3.3. Encryption-friendly Distillation Loss

The key missing component in the above CaPriDe learning framework is the computation D_{DL} in Eq. 4. For participant $\mathcal{P}_k$, its logit vectors $\boldsymbol{z}_{\cdot,(k,k)}$ and predictions $\boldsymbol{p}_{\cdot,(k,k)}$ are available in the plaintext domain. However, since participant $\mathcal{P}_l$ receives the input samples of $\mathcal{P}_k$ in encrypted form, it can perform inference in the encrypted domain and obtain only $\mathcal{E}\left(\boldsymbol{z}_{\cdot,(k,l)}\right)$, where $\mathcal{E}\left(\boldsymbol{z}_{\cdot,(k,l)}\right) = \mathcal{M}_{\theta_l}\left(\mathcal{E}(\boldsymbol{x}_{\cdot,k})\right)$. It is not straightforward to obtain $\mathcal{E}\left(\boldsymbol{p}_{\cdot,(k,l)}\right)$ from $\mathcal{E}\left(\boldsymbol{z}_{\cdot,(k,l)}\right)$ because it involves computing a non-linear softmax function in the encrypted domain, which is a challenging task in the FHE domain and consumes a large multiplicative depth. Even if this softmax operation is somehow implemented, it is not possible to send $\mathcal{E}\left(\boldsymbol{p}_{\cdot,(k,l)}\right)$ back to $\mathcal{P}_k$, who can decrypt it and obtain $\boldsymbol{p}_{\cdot,(k,l)}$. Since $\mathcal{P}_k$ also has knowledge of $\boldsymbol{x}_{\cdot,k}$, it becomes feasible for $\mathcal{P}_k$ to carry out a model extraction attack based on $(\boldsymbol{x}_{\cdot,k}, \boldsymbol{p}_{\cdot,(k,l)})$ pairs to glean information about model $\mathcal{M}_{\theta_l}$. This violates the privacy requirements of participant $\mathcal{P}_l$. Hence, there is a need for a loss function that can be computed in the encrypted domain.

A straightforward solution is to compute the L2 distance between logit vectors $\boldsymbol{z}_{\cdot,(k,k)}$ and $\boldsymbol{z}_{\cdot,(k,l)}$. In this case, participant $\mathcal{P}_k$ can publish $\mathcal{E}\left(\boldsymbol{z}_{\cdot,(k,k)}\right)$ and the pairwise distillation loss $\mathcal{L}_{DL_{(k,l)}}$ can be computed by participant $\mathcal{P}_l$ as:

$$\mathcal{E}\left(D_{DL}\left(\boldsymbol{p}_{j,(k,k)}, \boldsymbol{p}_{j,(k,l)}\right)\right) \equiv \mathcal{E}\left(D_{L2}\left(\boldsymbol{z}_{j,(k,k)}, \boldsymbol{z}_{j,(k,l)}\right)\right)$$
$$= \left(\mathcal{E}\left(\boldsymbol{z}_{j,(k,k)}\right) - \mathcal{E}\left(\boldsymbol{z}_{j,(k,l)}\right)\right) \cdot \left(\mathcal{E}\left(\boldsymbol{z}_{j,(k,k)}\right) - \mathcal{E}\left(\boldsymbol{z}_{j,(k,l)}\right)\right)$$

However, our experiments indicate the above simple solution based on L2 distance does not lead to high utility U. On the other hand, we observed that using the KL divergence between predictions of $\mathcal{P}_k$ and $\mathcal{P}_l$ almost always leads to better utility. But, $D_{KL}\left(\boldsymbol{p}_{\cdot,(k,k)} || \boldsymbol{p}_{\cdot,(k,l)}\right)$ cannot be directly computed in the encrypted domain because participant $\mathcal{P}_l$ only has access to the encrypted logits $\mathcal{E}\left(\boldsymbol{z}_{\cdot,(k,l)}\right)$.

Fortunately, this problem can be solved by approximating the KL divergence. Combining Eqs. 1 and 2, we get:

$$D_{KL}(\boldsymbol{p}_2 || \boldsymbol{p}_1) = \sum_{m=1}^{M} p_2^m \log p_2^m - \sum_{m=1}^{M} p_2^m (z_1^m / T) + M \log C. \tag{8}$$

Here, the first term can be denoted as $(\boldsymbol{p}_2 \cdot \log \boldsymbol{p}_2)$ and measures the entropy (uncertainty) in $\boldsymbol{p}_2$. The second term measures the divergence (unnormalized cross-entropy) between $\boldsymbol{p}_2$ and $\boldsymbol{p}_1$ and can be expressed as $(\boldsymbol{p}_2 \cdot \boldsymbol{z}_1 / T)$. The final term $M \log C$ is just a normalization constant that does not affect the relative ordering of the class predictions. We observed that this normalization term does not appear to play a major role in model training. Hence, we ignore this value

Algorithm 1 CaPRiDe learning algorithm

```
Input: Number of participants $K$, weight parameter $\lambda_k$, learning rate $\alpha_k$ for each participant
Assumption: Each participant has local training set $\mathcal{D}_k = \{\boldsymbol{x}_{j,k}, y_{j,k}\}_{j=[1,N_k]}$ of size $N_k$, a fully homomorphic encryption function $\mathcal{E}$ with public key $\kappa_k$, a decryption function $d$ with secret key $\tilde{\kappa}_k$
  for each participant $k = 1, 2, \cdots, K$ do
      Train initial model $\mathcal{M}_{\theta_k^0}$ based on local data $\mathcal{D}_k$
      Publish encrypted data $\{\mathcal{E}(\boldsymbol{x}_{j,k})\}_{j=[1,N_k]}$
  end for
  for each round $t = 1, 2, \cdots, \tau$ do
      for each participant $k = 1, 2, \cdots, K$ do
          $\{\boldsymbol{p}_{j,(k,k)}\}_{j=[1,N_k]} \leftarrow \{\sigma(\mathcal{M}_{\theta_k^{t-1}}(\boldsymbol{x}_{j,k}))\}_{j=[1,N_k]}$
          $\mathcal{L}_{CE_k} \leftarrow$ Compute cross-entropy loss of participant $k$ using Equation 3
          $\hat{\mathcal{L}}_{KL_k} \leftarrow$ Compute local uncertainty of participant $k$ using Equation 11
          Publish encrypted inferences $\{\mathcal{E}(\boldsymbol{p}_{j,(k,k)})\}_{j=[1,N_k]}$
          for each participant $l = 1, 2, \cdots, K, l \neq k$ do
              $\{\mathcal{E}(\boldsymbol{z}_{j,(k,l)})\}_{j=[1,N_k]} \leftarrow \{\mathcal{M}_{\theta_l^{t-1}}(\mathcal{E}(\boldsymbol{x}_{j,k}))\}_{j=[1,N_k]}$ (Perform inference on encrypted data)
              $\mathcal{E}(\tilde{\mathcal{L}}_{KL_{(k,l)}}) \leftarrow$ Compute pairwise divergence in encrypted domain using Equation 12
              Send $\mathcal{E}(\tilde{\mathcal{L}}_{KL_{(k,l)}})$ to participant $k$
              $\mathcal{L}_{DL_{(k,l)}} \leftarrow$ Compute pairwise distillation loss between participants $k$ and $l$ using Equation 10
          end for
          $\mathcal{L}_{DL_k} \leftarrow$ Compute distillation loss of participant $k$ using Equation 5
          $\mathcal{L}_k \leftarrow$ Compute total loss of participant $k$ using Equation 6
          $\theta_k^t \leftarrow$ Update local model of participant $k$ using Equation 7
      end for
  end for
```

and define the ***approximate KL divergence*** is defined as:

$$\hat{D}_{KL}(\boldsymbol{p}_2||\boldsymbol{p}_1) = (\boldsymbol{p}_2 \cdot \log \boldsymbol{p}_2) - (\boldsymbol{p}_2 \cdot \boldsymbol{z}_1/T). \tag{9}$$

Going back to Eq. 4, the pairwise distillation loss based on the above approximate KL divergence can be easily computed in the encrypted domain as:

$$\mathcal{E}\left(D_{DL}\left(\boldsymbol{p}_{j,(k,k)}, \boldsymbol{p}_{j,(k,l)}\right)\right) \equiv \mathcal{E}\left(\hat{D}_{KL}\left(\boldsymbol{p}_{j,(k,k)}||\boldsymbol{p}_{j,(k,l)}\right)\right)$$
$$= \mathcal{E}\left(\boldsymbol{p}_{j,(k,k)} \cdot \log \boldsymbol{p}_{j,(k,k)}\right) - \mathcal{E}\left(\boldsymbol{p}_{j,(k,k)}\right) \cdot \mathcal{E}\left(\boldsymbol{z}_{j,(k,l)}\right)/T$$

The above distillation loss based on the approximate KL divergence is ***encryption-friendly*** because it involves only linear operations and scaling by a constant plaintext value T. In practice, our goal is to compute $\mathcal{L}_{DL_{(k,l)}}$ at participant $\mathcal{P}_k$. Therefore, Eq. 4 can be further reformulated as:

$$\mathcal{L}_{DL_{(k,l)}} = \hat{\mathcal{L}}_{KL_k} - d\left(\mathcal{E}\left(\tilde{\mathcal{L}}_{KL_{(k,l)}}\right)\right), \tag{10}$$

where

$$\hat{\mathcal{L}}_{KL_k} = \sum_{j=1}^{N_k}(\boldsymbol{p}_{j,(k,k)} \cdot \log \boldsymbol{p}_{j,(k,k)}), \tag{11}$$

$$\mathcal{E}\left(\tilde{\mathcal{L}}_{KL_{(k,l)}}\right) = \sum_{j=1}^{N_k}\mathcal{E}\left(\boldsymbol{p}_{j,(k,k)}\right) \cdot \mathcal{E}\left(\boldsymbol{z}_{j,(k,l)}\right)/T. \tag{12}$$

Since $\mathcal{P}_k$ has access to $\boldsymbol{p}_{j,(k,k)}$, it is straightforward to compute $\hat{\mathcal{L}}_{KL_k}$ in plaintext space. If participant $\mathcal{P}_k$ can publish $\left\{\mathcal{E}(\boldsymbol{p}_{j,(k,k)})\right\}_{j=1}^{N_k}$ in each round, $\mathcal{P}_l$ can compute $\mathcal{E}\left(\tilde{\mathcal{L}}_{KL_{(k,l)}}\right)$ in the encrypted domain because it already has access to $\left\{\mathcal{E}\left(\boldsymbol{z}_{j,(k,l)}\right)\right\}_{j=1}^{N_k}$. The encrypted value of $\tilde{\mathcal{L}}_{KL_{(k,l)}}$ is sent back to participant $\mathcal{P}_k$, who decrypts it to compute the overall loss via Eqs. 10, 5 and 6.

Privacy Analysis: In the CaPriDe learning framework, the only quantity that can potentially leak private information is the pairwise distillation loss between participants $\tilde{\mathcal{L}}_{KL_{(k,l)}}$, which has already been aggregated over all the N_k samples. A curious participant $\mathcal{P}_k$ attempting to reconstruct the logit values $z_{j,(k,l)}^m$ of participant $\mathcal{P}_l$ needs to dis-aggregate this sum and compute per-sample loss, which is infeasible in the absence of other constraints. Hence, the proposed CaPriDe learning framework is privacy-preserving. It must be emphasized that even gradient inversion attacks [9, 14] are significantly harder when the batch size is large (above 16) due to the same reason. The proposed method also has other advantages that enhance privacy: (i) A curious participant cannot query its collaborator adaptively. The encrypted samples are published once at the beginning and do not change. (ii) The responding participant can choose a random subset of samples in each round to compute the aggregate loss. This can improve efficiency as well as provide a privacy benefit by eliminating correlation across multiple rounds. (iii) We can incorporate differential privacy (DP) by adding noise to the aggregate pairwise distillation loss. Since the framework allows exchange of only encrypted input samples and encrypted predictions, which never get decrypted during the execution of the protocol, the CaPriDe learning approach also preserves data confidentiality.

4. Experiments

4.1. Datasets

CIFAR-10 [17] is a dataset consisting of 60000 32×32 RGB images from 10 classes, with 6000 images per class. It has 50000 training and 10000 test samples.

CIFAR-100 dataset [17] is similar to the CIFAR-10 dataset, but it has 100 classes containing 600 samples each. Training and evaluation sets are split in the same way as in CIFAR-10. CIFAR-100 dataset is used with data augmentation to account for the problem complexity. The data augmentations employed include random rotation (up to 15 degrees), random crop, and horizontal flip.

HAM10000 is a multi-class imbalanced dataset comprising of 10,015 dermoscopic images from diverse populations with 7 categories of pigmented lesions. We randomly perform an $85\%/15\%$ split for training/ testing.

4.2. Experimental Setup

We compare the CaPriDe learning framework against the following baselines: vanilla FL based on FedAvg (FedAvg) [20], FL with local differential privacy (FedAvg + DP) [24], and deep mutual learning (DML) [27]. We use ResNet-18 architecture for all our experiments. Stochastic Gradient Descent (SGD) algorithm with momentum is used as the optimizer. For all methods, 25 epochs of local training happens before 75 rounds of collaboration. Momentum is set to 0.9, the initial learning rate is set to $\alpha_k = 0.1$ and the learning rate decays in rounds 50 and 75 by a factor $\gamma = 0.1$. The batch size is set to 128 for CIFAR-10 and CIFAR-100 and 32 for HAM10000. The weight factor λ_k between cross-entropy and distillation losses is set to 50 for CIFAR-10 and CIFAR-100 and 20 for HAM10000. All algorithms are implemented using PyTorch and experiments are carried out on a single NVIDIA RTX A6000 GPU. Additionally, in CaPriDe and DML, the logits are scaled by a temperature parameter $T = 5.0$. Finally, for FedAvg + DP, the Gaussian noise mechanism is used with $\sigma_g = 0.01$. To implement an FHE-compatible version of ResNet-18, we use the Tile Tensors framework [1,2], which in turn relies on low-level FHE libraries such as HElib and SEAL. For encrypted inference, ReLU is replaced with a polynomial approximation of Gaussian Error Linear Unit

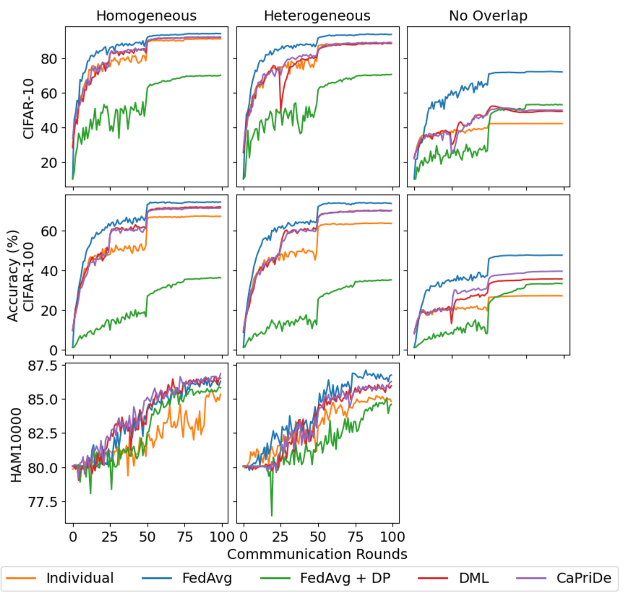


Figure 2. Accuracy trend of two ($K = 2$) collaboratively trained models on different datasets (rows) and for different partition strategies (columns).

($g(x) = 0.0711 + 0.5x + 0.2576x^2 - 0.0128x^4$).

We consider three types of strategies to partition the training dataset among the participants: (i) **Homogeneous**: each participant has an equal number of samples per class, (ii) **Heterogeneous**: each participant has an unequal number of samples determined randomly (both total number and number of samples per class), (iii) **Non-overlapping class distribution**: (denoted as "no class overlap") each participant has samples from a non-overlapping subset of classes (e.g., for $K = 2$, participants get even and odd classes). While homogeneous partitioning ensures that data distributions of participants are iid, heterogeneous split simulates typical non-iid scenarios, and the no class overlap case corresponds to extreme statistical heterogeneity.

4.3. Collaborative Learning Results

To begin with, we establish the utility (collaborative gain) of the proposed CaPriDe learning framework for the case of $K = 2$ participants on all three datasets and all three partition strategies [2]. The results of these experiments are shown in Figure 2. In each case, the collaboration gain of CaPriDe learning is positive and the actual collaboration gain values can be found in Table 1. We make the following observations based on these results.

- CaPriDe learning works not only in the iid setting but also equally well in the two non iid settings. This

[2]Since HAM10000 already has a massive class imbalance and an odd number of class, we omit the no class overlap setting for this dataset.

Dataset	Setting	K	FedAvg	FedAvg+DP	CaPriDe
CIFAR-10	Homogeneous	2	3.46	-20.54	1.58
CIFAR-10	Homogeneous	5	9.22	-11.8	3.55
CIFAR-10	Homogeneous	10	15.44	2.58	5.52
CIFAR-10	Heterogeneous	2	6.12	-17.04	0.93
CIFAR-10	Heterogeneous	5	13.78	-7.84	5.06
CIFAR-10	Heterogeneous	10	21.81	5.69	6.20
CIFAR-10	No class overlap	2	29.85	11.08	8.16
CIFAR-100	Homogeneous	2	7.08	-30.97	4.19
CIFAR-100	Homogeneous	5	22.68	-13.81	9.10
CIFAR-100	Homogeneous	10	26.10	-0.11	11.69
CIFAR-100	Heterogeneous	2	9.59	-28.89	6.28
CIFAR-100	Heterogeneous	5	22.50	-13.26	9.23
CIFAR-100	Heterogeneous	10	34.38	3.86	10.68
CIFAR-100	No class overlap	2	19.40	5.14	7.75
HAM10000	Homogeneous	2	0.97	0.48	1.81
HAM10000	Heterogeneous	2	1.93	-1.21	1.67

Table 1. Collaborative gain (%) on all datasets and all partition strategies for different number of participants K using different algorithms (columns 4-6). DML results are not reported here because there is negligible difference between CaPriDe and DML.

demonstrates that knowledge distillation is reasonably effective even under extreme statistical heterogeneity.

- There is a negligible difference between the performance of DML (which uses true KL divergence) and CaPriDe learning (which uses approximate KL divergence) in all the settings. This clearly justifies the dropping of the normalization (third) term in Eq. 8 to obtain the encryption-friendly distillation loss in Eq. 9. While DML offers no confidentiality or privacy, CaPriDe learning offers both without any utility cost.

- In almost all the settings, there is a huge difference in the collaboration gain between FedAvg and CaPriDe learning. This is unsurprising because FedAvg involves sharing and aggregation of full model parameters leading to high utility at a significant cost to privacy. On the other hand, FedAvg + DP provides strong privacy, but this comes with a significant degradation in utility. Hence, it can be argued that CaPriDe learning achieves a good trade-off between privacy and utility compared to standard FL algorithms.

Next, we evaluate the utility of the proposed method with the different number of participants $K = 2, 5, 10$ on CIFAR-10 and CIFAR-100 datasets based on the homogeneous partition (iid) setting. From Figure 3 and Table 1 we observe that the collaboration gain increases with the number of participants. This is to be expected because the training samples per participant become less when there are more participants, consequently resulting in lower individual accuracy.

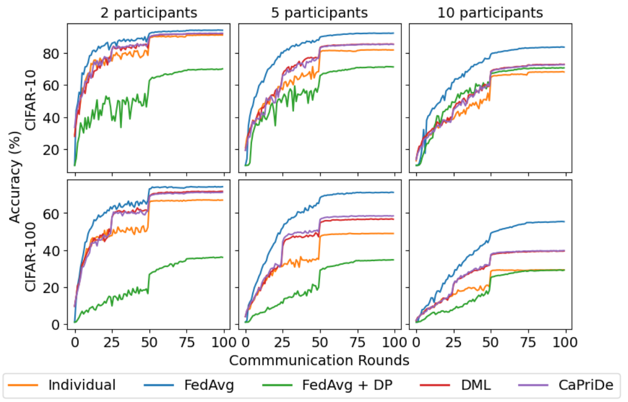
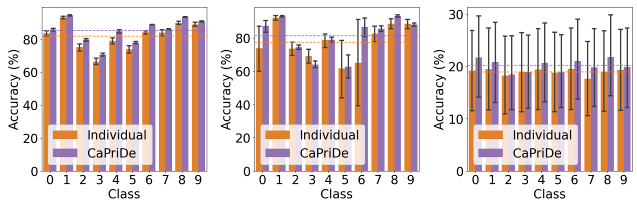
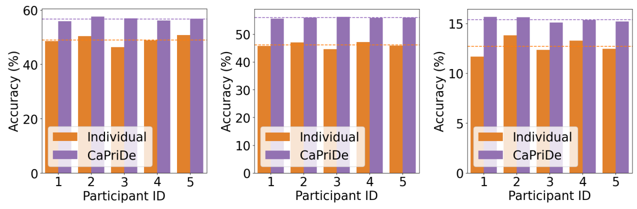


Figure 3. Accuracy with different number of participants K based on the homogeneous partition (iid) setting.



(a) Homogeneous (b) Heterogeneous (c) No overlap

Figure 4. Per-class accuracy of CaPriDe learning evaluated on CIFAR-10 averaged across 5 participants. Dashed lines represent mean accuracy and error bars indicate standard deviation.



(a) Homogeneous (b) Heterogeneous (c) No overlap

Figure 5. Per-participant accuracy of CaPriDe learning evaluated on CIFAR-100 dataset.

Furthermore, we evaluate the class-wise and participant-wise accuracy of our CaPriDe learning algorithm with $K = 5$ participants. Figure 4 shows class-wise accuracy averaged across 5 participants for each setting. From this figure, it is clear that almost all the classes benefit from collaboration, and the per-class accuracy of the individual models after collaboration has a lower standard deviation compared to the case without collaboration, especially in homogeneous and heterogeneous settings. Thus, in these two settings, the individual models become similar to each other after collaboration proving the successful transfer of knowledge between participants. Note that Figure 4 displays the average top-1 accuracy of the participants. The collaboration gains in the no class overlap case are significantly higher when top-5 accuracy is considered. Finally, Figure 5 illustrates participant-wise accuracy over all classes for each setting. It is obvious that each participant has a positive collaboration gain, which is the eventual goal that we set out to achieve. Moreover, all the 5 participant models have a similar accuracy at the end of the collaboration, once again demonstrating good knowledge transfer. Other results such as the sensitivity of the proposed method to weight factor λ_k, the sensitivity of FedAvg + DP to noise variance, and the comparison between approximate KL divergence and L2 loss are included in the supplementary material.

4.4. FHE Implementation Results

For FHE implementation using the Tile Tensors framework, Table 2 summarizes the information about all three datasets at the same security level of 128 bits. Two critical metrics to be kept in mind are the time required for encrypted inference and the memory consumed by a ciphertext. While encrypted inference time determines the computational complexity and latency involved in each round, the ciphertext size determines the communication burden. Note that encrypted inference can be performed on a per-sample basis or a batch of samples [4]. The single sample inference mode is preferred for machine learning as-a-service applications, where low latency is a critical requirement. However, the batched inference is better suited for the CaPriDe learning framework, where only the overall time required to complete inference on all the samples matter.

	CIFAR-10	CIFAR-100	HAM10000
Security Level	128	128	128
Number of slots	16384	16384	16384
Time taken to encrypt one sample	90 ms	103 ms	619 ms
Ciphertext size of one sample	29.101 KB	29.152 KB	1.359 MB
Time taken to encrypt a batch of 32 samples	1.31 s	1.26 s	15.57 s
Encrypted inference of a batch of 32 samples	110.21 s	112.09 s	896.12 s

Table 2. FHE configuration parameters and memory and computational requirements for the chosen datasets based on ResNet-18 architecture.

The encrypted inference time for a batch of 32 samples is approximately 110s for CIFAR-10 and CIFAR-100 and 900s for HAM10000. Thus, if the local dataset of each participant has 1024 training samples, it will take around 1 hour (8 hours) for encrypted inference per participant and then computation of the pairwise distillation loss in each communication round for CIFAR-10 and CIFAR-100 (HAM10000). While this is a significant computational load, it is less of an issue in the cross-silo collaborative learning setting, especially in sectors such as finance and healthcare, where data confidentiality and privacy as well as accuracy gain are more critical. Moreover, the ciphertext size for one sample is approximately, 30KB for CIFAR-10 and CIFAR-100 and 1.36MB for HAM10000. For the same local dataset size of 1024, the encrypted dataset will consume 30MB for CIFAR-10 and CIFAR-100 and 1.36GB for HAM10000. Fortunately, the encrypted dataset is published and needs to be downloaded only once by each participant at the beginning of the protocol. During every communication round, only the encrypted predictions and pairwise losses are exchanged. Depending on the local dataset size, the number of classes, and the ciphertext packing strategy used, these intermediate values can be packed into at most a few tens of ciphertexts (note that this does not depend on the image size). Hence, the per round communication burden of CaPriDe learning framework is relatively small.

Limitations and Potential Solutions: Note that all the above complexity computations are based on the assumption of $K = 2$ participants. As K increases, these requirements will increase linearly for each participant bringing the scalability of the system into question. Hence, the vanilla CaPriDe learning framework is more appropriate for a small network of participants. However, there are ways to improve scalability borrowing on ideas from cross-device FL. For example, instead of computing the pairwise distillation loss with respect to each participant, a random subset of participants can be sampled in each round for distillation. Similarly, the computational complexity for each participant can be significantly reduced by performing encrypted inference only on a randomly selected subset of encrypted samples. Furthermore, these actions can be determined at the local level without any central orchestration. The need to have FHE-friendly models for encrypted inference is a key limitation, which precludes the use of very deep models within the proposed framework. One potential solution is to employ two models at each participant: one deep model with large capacity and another smaller FHE-friendly model for encrypted inference on other participants' data and perform an additional mutual distillation between these two models after each round.

5. Summary

Collaborative learning without compromising on data confidentiality and privacy is a valuable tool in applications with stringent data privacy requirements. We have proposed a new framework called CaPriDe learning based on inference in the FHE domain to achieve this goal. Knowledge distillation is used to exchange useful information between models, and a new distillation loss based on approximate KL divergence has been proposed to enable secure loss computation in the encrypted domain. While the proposed approach has shown promising results, its efficiency can be further improved. Other vulnerabilities such as poisoning attacks will also be addressed in the future.

References

[1] Ehud Aharoni, Allon Adir, Moran Baruch, Nir Drucker, Gilad Ezov, Ariel Farkash, Lev Greenberg, Rami Masalha, Guy Moshkowish, Dov Murik, Hayim Shaul, and Omri Soceanu. HeLayers: A Tile Tensors Framework for Large Neural Networks on Encrypted Data. *arXiv:2011.01805v2*, 2021. 6

[2] Ehud Aharoni, Nir Drucker, Gilad Ezov, Hayim Shaul, and Omri Soceanu. Complex encoded tile tensors: Accelerating encrypted analytics. In *IEEE S&P 20 (5)*, pages 35–43, 2022. 6

[3] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy preserving machine learning. *ACM Conference on Computer and Communications Security (ACM CCS)*, 2017. 2

[4] Alon Brutzkus, Ran Gilad-Bachrach, and Oren Elisha. Low latency privacy preserving inference. In *International Conference on Machine Learning*, pages 812–821, 2019. 8

[5] Hongyan Chang, Virat Shejwalkar, Reza Shokri, and Amir Houmansadr. Cronus: Robust and heterogeneous collaborative learning with black-box knowledge transfer. In *arXiv preprint arXiv:1912.11279*, 2019. 2

[6] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic Encryption for Arithmetic of Approximate Numbers. 2016. 3

[7] Christopher A. Choquette-Choo, Natalie Dullerud, Adam Dziedzic, Yunxiang Zhang, Somesh Jha, Nicolas Papernot, and Xiao. Wang. CaPC Learning: Confidential and Private Collaborative Learning. *Proceedings of ICLR*, 2021. 2

[8] Deshpande, Nikhil and Shiffman, Gary M. Federated learning: The new thing in ai/ml for detecting financial crimes and managing risk. `https://morningconsult.com/opinions/federated-learning-the-new-thing-in-ai-ml-for-detecting-financial-crimes-and-managing-risk/`, 2021. 1

[9] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael. Moeller. Inverting gradients–How easy is it to break privacy in federated learning? *Advances in Neural Information Processing Systems (NeurIPS)*, 33, 2020. 1, 5

[10] Craig. Gentry. *Fully Homomorphic Encryption.* PhD thesis, 2009. 3

[11] Qiushan Guo, Xinjiang Wang, Yichao Wu, Zhipeng Yu, Ding Liang, Xiaolin Hu, and Ping. Luo. Online Knowledge Distillation via Collaborative Learning. *Proceedings of CVPR*, 2020. 2

[12] Jenny Hamer, Mehryar Mohri, and Ananda Theertha Suresh. FedBoost: A Communication-Efficient Algorithm for Federated Learning. *Proceedings of the 37th International Conference on Machine Learning, PMLR*, 119:3973–3983, 2020. 2

[13] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the Knowledge in a Neural Network. *NeurIPS*, 2014. 2, 4

[14] Jiwnoo Jeon, Kangwook Lee, Sewoong Oh, and Jungseul. Ok. Gradient Inversion with Generative Image Prior. *Advances in Neural Information Processing Systems (NeurIPS)*, 34, 2021. 1, 5

[15] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, and Mehdi. Bennis. Advances and Open Problems in Federated Learning. Now Publishers, 2021. 1

[16] Georgios A. Kaissis, Marcus R. Makowski, Daniel Rückert, and Rickmer F. Braren. Secure, privacy-preserving and federated machine learning in medical imaging. *Nature Machine Intelligence*, 2:305–311, 2020. 1

[17] Alex Krizhevsky. Gradient-based learning applied to document recognition. Technical report, University of Toronto, 2009. 6

[18] Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 2351–2363, 2020. 2

[19] Chuan Ma, Jun Li, Ming Ding, Howard H. Yang, Feng Shu, Tony Q. S. Quek, and Tony Q. S. Poor. On safeguarding privacy and security in the framework of federated learning. *IEEE Network*, pages 242–248, 2020. 2

[20] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y. Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2016. 2, 6

[21] Fan Mo, Hamed Haddadi, Kleomenis Katevas, Eduard Marin, Diego Perino, and Nicolas Kourtellis. PPFL: Privacy-Preserving Federated Learning with Trusted Execution Environments. In *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys)*, page 94–108, 2021. 1

[22] Willem G van Panhuis, Proma Paul, Claudia Emerson, John Grefenstette, Richard Wilder, Abraham J Herbst, and Donald S Heymann, David Burke. A systematic review of barriers to data sharing in public health. BMC Public Health, 2014. 1

[23] Micah J. Sheller, Brandon Edwards, G. Anthony Reina, Jason Martin, Sarthak Pati, Aikaterini Kotrotsou, Mikhail Milchenko, Weilin Xu, Daniel Marcus, Rivka R. Colen, and Spyridon. Bakas. Federated learning in medicine: facilitating multi-institutional collaborations without sharing patient data. *Nature Scientific Reports*, 10, 2020. 1

[24] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H. Yang, and Farhad. Farokhi. Federated Learning With Differential Privacy: Algorithms and Performance Analysis. *IEEE Transactions on Information Forensics and Security*, 15:3454–3469, 2020. 1, 6

[25] Jaehong Yoon, Wonyong Jeong, Giwoong Lee, Eunho Yang, and Sung Ju. Hwang. Federated Continual Learning with Weighted Inter-client Transfer. *Proceedings of the 37th International Conference on Machine Learning, PMLR*, 139:12073–12086, 2021. 2

[26] Chengliang Zhang, Suyi Li, Junzhe Xia, Wei Wang, Feng Yan, and Yang Liu. BatchCrypt: efficient homomorphic encryption for cross-silo federated learning. *Proceedings of the USENIX Conference*, page 493–506, 2020. 1

[27] Ying Zhang, Tao Xiang, Timothy M. Hospedales, and Huchuan Lu. Deep Mutual Learning. *CVPR*, 2017. 2, 6