MistNet: Towards Private Neural Network Training with Local Differential Privacy

Anonymous Author(s)
Affiliation
Address
email

Abstract

Deep neural networks (DNNs) generally learn parameters from massive amounts of high-quality training data to provide superb prediction performance. Classical machine learning approaches centralize decentralized data dispersed across devices in a common site for effective training but raise serious concerns of data privacy. In this paper, we design, implement, and evaluate MistNet, a privacy-preserving model training system that enables the cloud and the edge devices to collaboratively perform neural network training without revealing users' data as well as model parameters. MistNet partitions a DNN model into two parts, a lightweight feature extractor at the edge side to generate meaningful features from the raw training data, and a classifier including the most model layers at the cloud to be iteratively trained for specific tasks. Different from prior work, the feature extractor is transferred from pre-trained models for similar application domains and kept fixed during training, which eliminates the need to synchronize feature extractors across devices. Furthermore, MistNet enhances privacy via applying local differential privacy (LDP) to the intermediate features and assess the privacy leakage with two kinds of common attacks - membership inference and feature inversion attacks. We conduct an experimental study on multiple models and datasets, demonstrating that by choosing an appropriate partition layer and privacy budget, MistNet achieves acceptable model utility while greatly reducing privacy leakage from the released intermediate features.

1 Introduction

Deep neural networks (DNN) have been successfully applied to a wide range of areas including vision, speech, and natural language [1, 2, 3]. The superior prediction ability of DNN models relies on large amounts of data. Meanwhile, with the proliferation of mobile and IoT technology, tremendous valuable data are generated by edge devices but live in silos. There is an increasing demand to learn from the dispersed data, so as to better support machine learning (ML) tasks at the edge. However, in the most conventional training paradigm, models are placed at a central site. It requires collecting training data from users, which raises concerns about data privacy and violates data protection laws. As a result, privacy issues turn to be a barrier for empowering edge intelligence, and collaborative training without sharing the input training data is highly desired.

Recently, two learning paradigms have emerged to address this issue: *federated learning* [4] and *split learning* [5, 6]. Federated learning pushes the whole model to the edge and *model gradients* are exchanged across devices to learn a shared model. As DNNs have become deeper and more complex, model training has incurred prohibitive costs in computational resources, which poses a substantial challenge to resource-constrained devices. Split learning evolves to train the first few layers of the neural network at the edge and transmit the *intermediate features* to cloud servers with abundant computational resources to facilitate training the rest of the training.

However, the existing split learning method [5, 6] trains the model in a sequential fashion across edge devices, which slows down the training process - an edge device has to receive updated weights from the last trained device before training on its local data. There is a need to enable parallel local training among distributed edge devices. Exposing the intermediate features instead of the training data is assumed to be safer, but is it sufficient to protect the privacy of the training data? Recent study [7] devises an inversion attack to recover the inputs from the intermediate features even in the black-box setting without the need to know the parameters of the model at the edge.

In this paper, we propose MistNet[1], a privacy-preserving collaborative training framework, in which we divide the neural network into two parts, the first few layers as a feature extractor at the edge and the rest layers as a classifier at the cloud. We design the feature extractor based on an insight from transfer learning [8]: the early layer features are general to many datasets and tasks. Take computer vision tasks for example, we initialize the feature extractor with weights pre-trained on large public datasets like ImageNet [9]. The feature extractor thus is ready to produce meaningful features without the need to be further trained with the rest model on cloud servers. It thus supports edge devices to perform training in parallel by eliminating the need to synchronize updated feature extractors. To ensure that intermediate features do not reveal sensitive information about any particular training record, we adopt the rigorous local differential privacy technique - Randomized Response (RR), which was introduced by Warner et al. [10] for collecting sensitive statistics from survey respondents and later widely deployed in real systems by Google [11] and Apple [12]. Subject to RR constraints, we discretize the features of the partition layer by constraining the values to either 1 or 0. Each feature value will be independently randomized before leaving the edge devices, thus no raw features are leaked. The probability to preserve the original value is determined by the privacy budget ϵ. A small value of privacy budget ϵ strictly guarantees privacy, but also detriments model utility. We have limited understanding of the range of ϵ values for reasonable privacy-accuracy trade-off in practice. We thus assess the privacy leakage with two common attacks against ML models, model inversion, and membership inference attack. We summarize our major contributions as follows:

- We propose MistNet, which uses a fixed-weight pre-trained feature extractor to generate meaningful features and further apply local differential privacy on features to enhance privacy.
- Besides from privacy budget ϵ, we use model inversion and membership inference attack to quantitatively assess the privacy leakage.
- We experimentally show that MistNet achieves good prediction accuracy while preserving privacy under different models, datasets, and parameter settings.

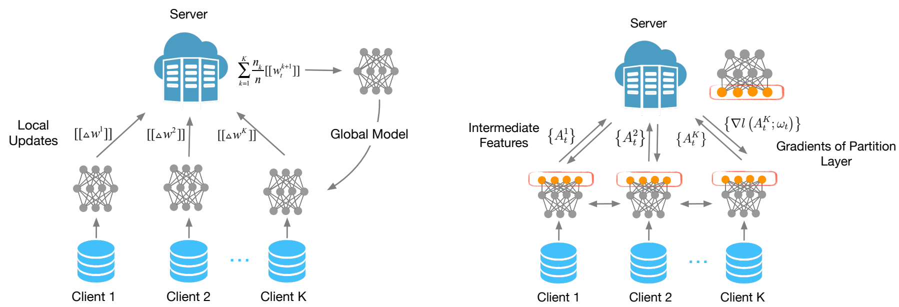
2 Preliminaries

2.1 Distributed Collaborative Learning

To support training on a substantial amount of training data from different sources, distributed collaborative learning emerges to enable multiple parties to contribute to learning a shared model. Depending on where the model is located, distributed collaborative learning systems can be categorized into the following paradigms.

Federated Learning. To build a more privacy-friendly collaborative training approach, Google proposes federated learning [4] to enable participating devices collaboratively to train a shared model, while keeping the training data on mobile devices. As illustrated in Figure 1a, each mobile device trains the model with its local data for several local epochs, and push the local updated parameters to the central server, where these updates are aggregated to compute a new model with model averaging. The updated global model is then sent back to edge devices in the next round. During the whole training procedure, only model parameters are shared, while the training data is kept locally at the devices.

[1] MistNet is a combination of the words **Mist** and Neural **Net**work. This name reflects the fact that preventing users' data from being revealed is like putting a mist around them.



(a) Federated Learning (b) Split Learning

Figure 1: Two distributed collaborative learning paradigms: federated learning and split learning

Split Learning. Instead of pushing the whole model to the edge, split learning [5, 6] is proposed as an alternative collaborative training approach, where a neural network model is partitioned between the cloud and edge. As shown in Figure 1b, with the training data, an edge device trains the network up to the partition layer and sends the intermediate features of the partition layer to the cloud. Upon receiving the features, the cloud takes over training the remaining layers to complete the forward pass. In the backward pass, with the gradients of the partition layer backpropagated from the cloud, the edge device updates local model parameters. For consistency of local models among devices, the edge device then synchronizes the updated model with the next device scheduled to participate in training. The sequential training manner results in severe under-utilization of resources on edge devices. Only one edge device is active in training with the cloud at any specific point in time. Split learning preserves privacy in a way that only the intermediate features are sent out to the cloud while the training data are still left to the edge devices. Unfortunately, transmitting features still has the risk to leak sensitive information of the input data. Recent work [7] shows the possibility to accurately recover the input image from the intermediate features even without access to the edge device.

2.2 Local Differential Privacy

Differential privacy is a statistical definition of privacy that is used to publish aggregate information about the entire population while constraining the privacy leakage of each individual. As a kind of differential privacy, local differential privacy (LDP) works without assuming a trusted data collector. The data owners directly add noise to their data before sharing them with the untrusted data collector, which provides a much stronger privacy guarantee [13, 14]. We provide a formal definition of local differential privacy below. As the privacy budget ϵ measures the extent of privacy leakage of the random mechanism π. A lower value of ϵ means more privacy.

Definition 2.1 (ϵ-LDP [13])**.** *An random mechanism $\pi : \mathcal{X} \to \mathcal{Y}$ satisfies ϵ-local differential privacy, where $\epsilon \geq 0$, if and only if for any inputs $x, x' \in \mathcal{X}$ and $y \in \mathcal{Y}$, we have $Pr[\pi(x) = y] \leq e^{\epsilon} Pr[\pi(x') = y]$.*

Randomized response (RR) [10] is a typical mechanism to implement LDP. It is initially proposed as a survey technique to collect answers to sensitive binary questions. The respondent uses a randomization method like a coin flip to randomize the answer "yes" or "no". She answers the question truthfully if the coin comes up heads, otherwise returns a false answer. Assume that we use a biased coin and the probability to provide a truthful answer (coin flip result is head) is p. The mechanism satisfies ϵ-differential privacy with $p = \frac{e^{\epsilon}}{1+e^{\epsilon}}$.

2.3 Attacks against ML Models

We concentrate on two types of attacks against ML models which are closely related to our work and suitable to be used to assess the privacy leakage.

Feature Inversion Attack As a specific case of model inversion attack [15], feature inversion attack [7] is recently devised for the edge-cloud collaborative learning system. The neural network f_{θ} is split into two parts: f_{θ_1} and f_{θ_2} between the edge and the cloud. The adversary aims to recover the input data sample x_0 from the intermediate layer features $f_{\theta_1}(x_0)$in both white-box and block-box settings. In the white-box setting, the adversary is assumed to compromise with participant and knows the structure and parameters of model $f_{\theta_1}(x_0)$. The adversary performs gradient descent technique on f_{θ_1} to find a generated sample x whose intermediate $f_{\theta_1}(x)$ is the most similar to $f_{\theta_1}(x_0)$ and at the same time following the same distribution as the input data. Recovering inputs is more challenging in the black-box setting, where the information of f_{θ_1} is totally unknown to the adversary. Suppose the attacker can feed arbitrary inputs to f_{θ_1} and receive the corresponding outputs, the attacker accordingly trains a network to approximate the inversion function of f_{θ_1} and then converts the intermediate output into the input sample with the trained network.

Membership Inference Attack Membership inference attack aims to find out whether a given sample is used to train a model or not, which is considered as a direct privacy breach. For example, knowing a patient record is used to train models for diagnosis (disease presence) reveals that the patient has the disease. Shokri et al. [16] introduces membership inference in the block-box ML setting where the model is not accessible by the attacker. The attacker firstly trains multiple "shadow models" to imitate the behavior of the target model and then trains the binary attack model with the labeled inputs and outputs of the shadow models. Taking a data sample's prediction output queried from the target model as input, the binary attack model infers whether the data sample is a member or non-member of the target model's training dataset.

3 Related Work

The idea of neural network partition in MistNet is inspired by a large body of previous work [17, 18, 19, 20, 21, 22]. These work mainly focus on optimizing performance (eg., latency and accuracy) and cost (eg., communication and computation overhead) without considering data privacy. We break related work on protecting data privacy into the following three categories, injecting noise on the intermediate features, censoring the intermediate features with private feature extractor, and leveraging secure computation techniques.

Noise Injection. Several efforts [23, 24] inject noise to reduce the mutual information between the input and the intermediate features. They assume that the attacker performs sensitive secondary inferences. It is unknown whether these noises could successfully defend other types of attacks. Differential privacy provides strict privacy guarantees in the worst-case scenario without knowing the types of privacy attacks. There exist different mechanisms to apply differential privacy mechanisms on the intermediate features, such as the Laplace mechanism [25], the Gaussian mechanism [26] adds Gaussian noise. MistNet extracts binarized features and perturb them with local differential privacy technique - randomized response.

Private Feature Extractor. To defend against attacks while not sacrificing much accuracy, proposals [27, 28] use adversarial training to find an appropriate feature extractor from two respects - the number of layers and the strategy to prune output channels of the partition layer. DPFE [24] and DeepObfuscator [29] train the feature extractor to hide information about sensitive attributes while keeping useful features for the target task.

Secure Computation. Secure computation techniques recently are used to deal with privacy-preserving machine learning. The first line is to use cryptographic protocols such as secure multi-party computation (MPC) [30, 31] and homomorphic encryption (HE) [32, 33, 34]. Two or more parties collaboratively train a neural network on encrypted data from clients without the need to decrypt them. However, the computational and communication cost is usually prohibitively high, which makes these cryptographic techniques too heavy to be deployed at resource-constrained edge devices. The second line leverages trusted execution environments (TEE) [35, 36, 37] such as Intel SGX [38] and ARM TrustZone [39] to protect the training data. Training data is used within an isolated secure environment which is invisible to unauthorized adversaries. Nevertheless, the potential drawbacks are the limited scalability of TEEs and the vulnerability to side-channel attacks.

4 Design

In this section, we present MistNet, a framework for privacy-preserving collaborative training between the cloud and the edge. We first provide an overview of the framework and then the role of every component of our architecture. Lastly, we show how the intermediate features are perturbed with local differential privacy to provide a strong privacy guarantee.

4.1 Workflow of MistNet

Figure 2 illustrates the collaborative training process between the edge and the cloud. To protect sensitive training data from being abused and support training complex models, we separate a model into two parts between the edge and cloud. The lightweight first few layers of the neural network are placed at the edge devices as the feature extractor. The rest layers with heavy computation are offloaded to the powerful cloud servers as the cloud classifier to make predictions. We detail the workflow of the local feature extractor and cloud classifier below.

Local Feature Extractor. As explained in §2, maintaining consistency of feature extractors among edge devices hinders parallel training, which lowers the efficiency of training among a large scale of edge devices. Taking the factor into account, MistNet uses a fixed feature extractor thus eliminate the need to perform synchronization among edge devices. To obtain meaningful features, the local feature extractor is transferred from pre-trained models that work on a similar application domain via transfer learning. Yosinski et al. [8] quantify the transferability of features from different layers in deep neural networks. Features from the early layers are more general than that from later layers, which show more flexibility to adapt to a wide range of related datasets and tasks. Meanwhile, to keep the feature extractor simple and lightweight, the partition point in MistNet is usually set at a very early layer in the model. The extracted features thus show high generalization capability and are ubiquitous to various tasks. Moreover, the feature extractor is less sensitive to the changes of input data, which provides the possibility to apply a fixed pre-trained feature extractor during training. With the fixed pre-trained feature extractor, edge devices transform the input training data into feature representations in parallel and send them to the cloud for the rest of the training. Thereafter, edge devices do not need to repeatedly send the feature representations for the same batch of training samples, nor receive backward-propagated feedbacks from the cloud.

As indicated in previous work [7], the value of intermediate features has the potential risk to reveal sensitive information about the input data. The volume of feature representations of the early layer in some models can be even larger than the raw input data, which incurs high communication costs. The intermediate representations should be transmitted in a secure and communication-efficient way. In MistNet, edge devices binarize each activation value with 1 bit and perturb the binarized feature representations conforming to local differential privacy before sending them to the cloud. We further explain the perturbing mechanism in §4.2.

Cloud Classifier. Upon receiving the perturbed feature representations from the edge devices, the cloud iteratively trains the rest layers of the network with stochastic gradient descent (SGD) algorithm to minimize the loss for a specific task. To reduce the communication cost, intermediate features for input samples without any data augmentation effect are transmitted to the cloud. We further apply random cropping on the intermediate features of each input sample as the data augmentation technique to alleviate overfitting. We have the intermediate features of each input sample reshuffled at each epoch during training. As shown in [8, 40], fine-tuning model on new related tasks is faster to converge to near optimum than training from scratch. We thus initialize the parameter weights for the cloud classifier with the transferred weights from the pre-trained model. During the backward pass, the cloud classifier does not need to propagate the loss back to the edge devices and only parameters of the cloud classifier are updated.

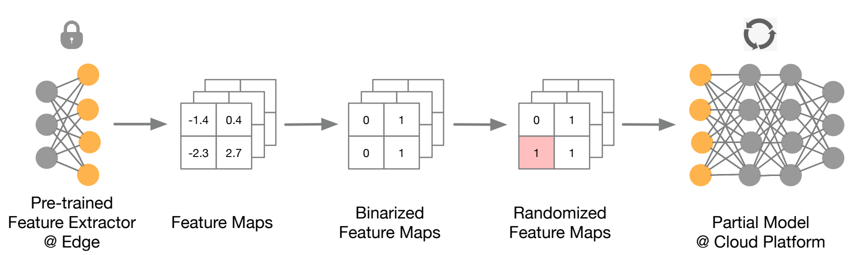


Figure 2: The overview of MistNet architecture. MistNet partitions the model between the edge and the cloud platform. The edge uses a pre-trained feature extractor to transform the local input data into a set of feature maps. Each edge device quantizes each activation into 1-bit and sends the randomized, differentially private version of the binarized feature values to the cloud platform. The partial model at the cloud platform is trained with the perturbed features collected from the edge devices.

4.2 Differentially Private Feature Representations

In this section, we detail how to protect the sensitive information about the training inputs from being revealed from the feature representations. Without assuming a trusted data collector, we apply local differential privacy to the intermediate feature representations from edge devices. The LDP mechanism we apply in MistNet is based on the randomized response method [10], which is widely deployed in practical systems [11, 12].

We denote the local feature extractor at the edge as $f : \mathbb{R}^d \to \mathbb{R}^r$, which transforms local data $x \in \mathcal{X}$ into feature representations A, with $r = \dim(A)$. With unary encoding [11], we encode each real-value feature A_i of the feature representations A into a bit B_i, whose value is either 1 or 0. The binarization function we use is as the following:

$$B_i = \begin{cases} 1 & \text{if } A_i > 0; \\ 0 & \text{if } A_i \leq 0. \end{cases} \tag{1}$$

We concatenate these bits as a binary string $B = (B_1, B_2, B_3, ..., B_r)$. Then we apply randomized response defined in Eq. (2) to perturb each bit B_i in B independently and submit the noisy version $\tilde{B}$ to the cloud. Each bit is preserved as its true value with probability p or responded with the other value with probability q. We have $p = 1 - q$ in this setting. The privacy budget ϵ is calculated with Theorem 4.1.

$$P(\tilde{B}_i = 1) = \begin{cases} p & \text{if } B_i = 1; \\ q & \text{if } B_i = 0. \end{cases} \tag{2}$$

Theorem 4.1. *Local feature extractor with randomized response defined in Eq. (2) satisfies ϵ-local differential privacy given that $p \geq q$, where $\epsilon = r \cdot \ln \frac{p}{q}$.*

See appendix A for the proof.

5 Evaluation

We evaluate the performance of MistNet on PyTorch and seek to answer the following questions: (1) How does MistNet perform with different privacy budgets for popular neural network models (§5.2)? (2) How does the partition layer selection affect the performance (§5.3)? (3) Whether MistNet is effective to defend model inversion and membership inference attack (§5.4)?

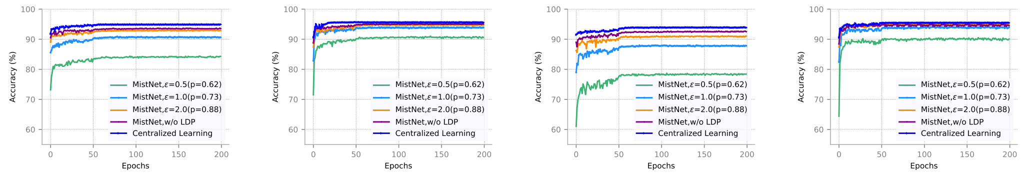
5.1 Experimental Setup

Datasets and Models. We evaluate MistNet for image classification on CIFAR-10 [41] and SVHN [42] dataset. CIFAR-10 has 10 classes and contains 60,000 32×32 color pixel images with 3 RGB channels (50,000 training images and 10,000 testing images). SVHN is a MNIST-like dataset of 32×32 images, including 73,257 training digits and 26,032 testing digits. To pre-train neural network models, we use ImageNet32×32 images extracted from CINIC dataset [43], which downsamples part of the original ImageNet images from 224×224 to 32×32 resolution with the Box algorithm from the Pillow Python library[2]. These Imagenet32×32 images have the same 10 classes as CIFAR-10 (the number of images for train/validation/test is 70,000/70,000/70,000 respectively) but do not include any image in neither CIFAR-10 dataset nor SVHN dataset. The models we choose are ResNet-18 [44] and VGG-16 [45].

Compared Schemes. We compare MistNet with the conventional training paradigm, centralized learning (CL), which collects data from users in a central site to train the model.

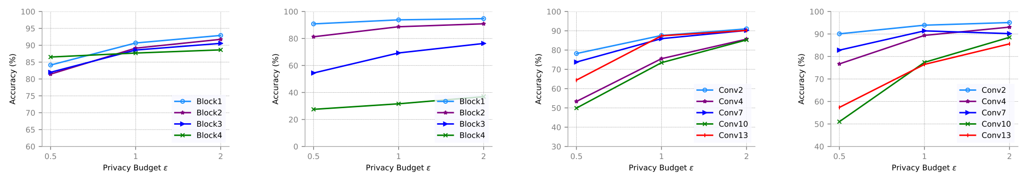
Evaluation Metrics. The performance of MistNet is evaluated from accuracy and privacy. Specifically, we detail how membership inference and model inversion attacks are used to assess privacy risks. For membership inference attack, we adopt two metrics, precision (the fraction of records inferred as members actually are members of the training dataset) and recall (the fraction of records which are correctly inferred as training samples over all training samples) as privacy metrics, which

[2] https://python-pillow.org



(a) ResNet-18, CIFAR-10 (b) ResNet-18, SVHN (c) VGG-16, CIFAR-10 (d) VGG-16, SVHN

Figure 3: Comparison among MistNet with varying privacy budgets, MistNet without local differential privacy and Centralized Learning for ResNet-18 and VGG-16 on CIFAR-10 and SVHN. ResNet-18 is partitioned at Block 1 and VGG-16 is partitioned at Conv 2.



(a) ResNet-18, CIFAR-10 (b) ResNet-18, SVHN (c) VGG-16, CIFAR-10 (d) VGG-16, SVHN

Figure 4: Comparison among MistNet with different partition layers for ResNet-18 and VGG-16 on CIFAR-10 and SVHN.

are consistent with previous work [16, 46, 47]. To quantify the quality of images recovered by model inversion attack, we use two commonly used image quality metrics, peak signal-to-noise ratio (PSNR) as well as the structural similarity index measure (SSIM). PSNR quantifies the pixel-level reconstruction quality of the image, which can be expressed as the ratio of the maximum possible value of a signal to the cumulative squared error between the reconstructed image and the original image. SSIM measures the reconstructed image quality by taking into account the structural information perceived by the human vision system including structure, luminance, and contrast.

5.2 Impact of Privacy Budget ϵ

In this section, we vary privacy budgets ϵ per feature to investigate its impact on model accuracy. As indicated in Sec. 2.2, the value of ϵ is proportional to the probability p of reporting the true value of each feature in the intermediate features. The smallest value of ϵ can be 0, which is equivalent to absolute privacy. The lowest value of ϵ we show in our experiments is 0.5, whose corresponding probability for each feature to report the true value is 62%. We evaluate MistNet with a range of privacy budgets $\epsilon \in \{0.5, 1, 2\}$ and use MistNet without LDP (i.e., do not randomize the intermediate features and is equivalent to $\epsilon = +\infty$) and centralized learning as two baselines. We partition ResNet-18 at Block 1 and VGG-16 at Conv 2 (the partition strategy is explained in Sec. 5.3), which both are early layers. In Figure 3, we observe that strong privacy is provided at the sacrifice of the utility. With a smaller value of ϵ (stricter privacy guarantee), the accuracy decreases and MistNet converges slower. Particularly, the performance of MistNet without LDP for both datasets achieves similar performance as centralized learning, which demonstrates the effectiveness of the pre-trained feature extractor.

5.3 Impact of Partition Layers

We explore the robustness of MistNet to different partition layers. More specifically, we partition the ResNet-18 model into 4 fused layer blocks (Block 1-4), with each containing 4 convolution and batch normalization layers. The VGG-16 model has 13 convolutional layers and is partitioned after convolutional layer 2, 4, 7 10, and 13. As shown in Figure 4, MistNet is robust to most partition layers and achieves acceptable utility. In most cases, partitioning at an earlier layer (i.e., Block 1 for ResNet-18 and Conv 2 for VGG-16) achieves better performance. The possible reason is twofold. First, the transferability of an earlier layer is better and partitioning at an earlier layer leaves more space to fine-tune the rest model to adapt to the perturbed features. Second, features from an earlier layer generally have more dimensions which contain redundant information. We also notice that

Table 1: Comparison of the quality of reconstructed images from features generated by different schemes with model inversion attack (ResNet-18 and CIFAR-10).

	SSIM					PSNR				
	$\epsilon = 0.5$	$\epsilon = 1$	$\epsilon = 2$	w/o LDP	CL	$\epsilon = 0.5$	$\epsilon = 1$	$\epsilon = 2$	w/o LDP	CL
Block 1	0.354	0.576	0.728	0.775	0.918	12.941	14.126	15.531	16.314	21.028
Block 2	0.211	0.306	0.453	0.515	0.690	12.466	12.918	13.621	14.099	15.724
Block 3	0.165	0.170	0.205	0.206	0.303	12.311	12.422	12.538	12.616	13.010
Block 4	0.155	0.154	0.149	0.169	0.164	12.274	12.296	12.299	12.314	12.366

Table 2: Membership inference attacks on different schemes (ResNet-18 and CIFAR-10).

	Precision					Recall				
	$\epsilon = 0.5$	$\epsilon = 1$	$\epsilon = 2$	w/o LDP	CL	$\epsilon = 0.5$	$\epsilon = 1$	$\epsilon = 2$	w/o LDP	CL
Block 1	0.5194	0.5488	0.5778	0.5973	0.5952	0.5267	0.6021	0.6962	0.7201	0.7551
Block 2	0.4988	0.5275	0.5673	0.5847	—	0.4860	0.5539	0.6595	0.7104	—
Block 3	0.5033	0.5155	0.5427	0.5788	—	0.4924	0.5371	0.5959	0.7101	—
Block 4	0.4997	0.5020	0.4991	0.5070	—	0.4975	0.5038	0.5030	0.5136	—

model partitioned at the last convolutional layer achieves reasonable performance for CIFAR-10 while performing poorly for SVHN. As described in Sec. 5.1, the chosen ImageNet images for pre-training models have the same classes with CIFAR-10. CIFAR-10 dataset is more similar than SVHN dataset. Features from later layers still transfer well for CIFAR-10.

5.4 Effect of Attack Mitigation

Feature Inversion Attack. We consider feature inversion attacks under the white-box setting where the attacker has access to the feature extractor at the edge, since white-box attacks are more challenging to defend than block-box attacks. In Table 1, we show the quality of images recovered from features generated by different schemes. Images are recovered from raw features for centralized learning. As indicated in [7], a recovered image with SSIM value below 0.3 is considered to be unrecognizable. We observe that, with ResNet-18 model partitioned at different layers, the value of SSIM for MistNet with various privacy budgets is consistently significantly lower than that for centralized learning, which demonstrates the effectiveness of applying LDP on the partition layer in protecting images from being recovered from the intermediate features. Partitioning ResNet-18 with $\epsilon = 0.5$ at Block 2 achieves a good trade-off between accuracy and privacy.

Membership Inference Attack. We perform membership inference attacks on the model trained with different schemes. As membership inference is a binary classification, the precision and recall value is always between 0.5 and 1. The value of 0.5 is equivalent to random guessing, which indicates that there is no privacy leakage. Table 2 shows that the precision and recall decreases with the value of ϵ and is reduced to around 0.5 with $\epsilon = 0.5$, while MistNet without LDP and centralized learning still remain a high precision and recall. An interesting observation is that ResNet-18 partitioned at Block 4 mitigates membership inference attacks even with a large ϵ. This is because the trainable cloud classifier only includes a linear layer, which is not sufficient to fit the training records.

6 Conclusion

In this paper, we presented MistNet as a privacy-preserving collaborative training system for resource-constrained edge devices. Our method uses a pre-trained feature extractor to eliminate the need to synchronize local weights across edge devices and enhances privacy by applying LDP to the intermediate features. We extensively evaluate MistNet with various settings on a variety of models, datasets, and attacks. The results show that MistNet partitioned at most layers with privacy budget $\epsilon = 0.5$ achieves acceptable utility while effectively reducing privacy leakage.

Broader Impact

Distributed machine learning is a widely used computing paradigm to learn from gigantic amounts of data generated by edge devices. Our work can be used to enhance data privacy in distributed machine learning systems, which follows the recent trend to comply with the EU General Data Protection Regulation (GDPR) law. It is possibly adopted by application developers and service providers as a tool to collect personal data from users. The possible negative aspect is it will be harder for the service provider to regulate the submitted training data from users, and detecting data with a significant detrimental impact on the prediction performance will be challenging. It is not likely to directly raise any ethical issues.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[2] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. IEEE, 2013.

[3] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

[4] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

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

[6] Praneeth Vepakomma, Otkrist Gupta, Tristan Swedish, and Ramesh Raskar. Split learning for health: Distributed deep learning without sharing raw patient data. *arXiv preprint arXiv:1812.00564*, 2018.

[7] Zecheng He, Tianwei Zhang, and Ruby B. Lee. Model inversion attacks against collaborative inference. In *Proceedings of the 35th Annual Computer Security Applications Conference*, ACSAC '19, page 148–162, New York, NY, USA, 2019. Association for Computing Machinery.

[8] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[10] Stanley L Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69, 1965.

[11] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, pages 1054–1067, 2014.

[12] Learning with privacy at scale. https://machinelearning.apple.com/2017/12/06/learning-with-privacy-at-scale.html. Accessed: December 2017.

[13] Shiva Prasad Kasiviswanathan, Homin K Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. What can we learn privately? *SIAM Journal on Computing*, 40(3):793–826, 2011.

[14] Björn Bebensee. Local differential privacy: a tutorial. *arXiv preprint arXiv:1907.11908*, 2019.

[15] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1322–1333, 2015.

[16] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18. IEEE, 2017.

[17] Yiping Kang, Johann Hauswald, Cao Gao, Austin Rovinski, Trevor Mudge, Jason Mars, and Lingjia Tang. Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. *ACM SIGARCH Computer Architecture News*, 45(1):615–629, 2017.

[18] Jiachen Mao, Xiang Chen, Kent W Nixon, Christopher Krieger, and Yiran Chen. Modnn: Local distributed mobile computing system for deep neural network. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, pages 1396–1401. IEEE, 2017.

[19] Hongshan Li, Chenghao Hu, Jingyan Jiang, Zhi Wang, Yonggang Wen, and Wenwu Zhu. Jalad: Joint accuracy-and latency-aware deep structure decoupling for edge-cloud execution. In *2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS)*, pages 671–678. IEEE, 2018.

[20] Jong Hwan Ko, Taesik Na, Mohammad Faisal Amir, and Saibal Mukhopadhyay. Edge-host partitioning of deep neural networks with feature space encoding for resource-constrained internet-of-things platforms. In *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–6. IEEE, 2018.

[21] Hyuk-Jin Jeong, Hyeon-Jae Lee, Chang Hyun Shin, and Soo-Mook Moon. Ionn: Incremental offloading of neural network computations from mobile devices to edge servers. In *Proceedings of the ACM Symposium on Cloud Computing*, pages 401–411, 2018.

[22] Xiaorui Wu, Hong Xu, Bo Li, and Yongqiang Xiong. Stanza: Layer separation for distributed training in deep learning. *arXiv preprint arXiv:1812.10624*, 2018.

[23] Fatemehsadat Mireshghallah, Mohammadkazem Taram, Prakash Ramrakhyani, Ali Jalali, Dean Tullsen, and Hadi Esmaeilzadeh. Shredder: Learning noise distributions to protect inference privacy. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 3–18, 2020.

[24] Seyed Ali Osia, Ali Taheri, Ali Shahin Shamsabadi, Kleomenis Katevas, Hamed Haddadi, and Hamid R Rabiee. Deep private-feature extraction. *IEEE Transactions on Knowledge and Data Engineering*, 32(1):54–66, 2018.

[25] Ji Wang, Jianguo Zhang, Weidong Bao, Xiaomin Zhu, Bokai Cao, and Philip S Yu. Not just privacy: Improving performance of private deep learning in mobile cloud. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2407–2416, 2018.

[26] Yunlong Mao, Shanhe Yi, Qun Li, Jinghao Feng, Fengyuan Xu, and Sheng Zhong. Learning from differentially private neural activations with edge computing. In *2018 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 90–102. IEEE, 2018.

[27] Meng Li, Liangzhen Lai, Naveen Suda, Vikas Chandra, and David Z Pan. Privynet: A flexible framework for privacy-preserving deep neural network training. *arXiv preprint arXiv:1709.06161*, 2017.

[28] Shuang Zhang, Liyao Xiang, Congcong Li, Yixuan Wang, Zeyu Liu, Quanshi Zhang, and Bo Li. Preventing information leakage with neural architecture search. *arXiv preprint arXiv:1912.08421*, 2019.

[29] Ang Li, Jiayi Guo, Huanrui Yang, and Yiran Chen. Deepobfuscator: Adversarial training framework for privacy-preserving image classification. *arXiv preprint arXiv:1909.04126*, 2019.

[30] Andrew Chi-Chih Yao. How to generate and exchange secrets. In *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*, pages 162–167. IEEE, 1986.

[31] Payman Mohassel and Yupeng Zhang. Secureml: A system for scalable privacy-preserving machine learning. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 19–38. IEEE, 2017.

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

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

[34] Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan. {GAZELLE}: A low latency framework for secure neural network inference. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pages 1651–1669, 2018.

[35] Zhongshu Gu, Hani Jamjoom, Dong Su, Heqing Huang, Jialong Zhang, Tengfei Ma, Dimitrios Pendarakis, and Ian Molloy. Reaching data confidentiality and model accountability on the caltrain. In *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 336–348. IEEE, 2019.

[36] Florian Tramer and Dan Boneh. Slalom: Fast, verifiable and private execution of neural networks in trusted hardware. *arXiv preprint arXiv:1806.03287*, 2018.

[37] Tyler Hunt, Congzheng Song, Reza Shokri, Vitaly Shmatikov, and Emmett Witchel. Chiron: Privacy-preserving machine learning as a service. *arXiv preprint arXiv:1803.05961*, 2018.

[38] Frank McKeen, Ilya Alexandrovich, Alex Berenzon, Carlos V Rozas, Hisham Shafi, Vedvyas Shanbhogue, and Uday R Savagaonkar. Innovative instructions and software model for isolated execution. *HASP@ ISCA*, 10(1), 2013.

[39] Tiago Alves. Trustzone: Integrated hardware and software security. *White paper*, 2004.

[40] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[41] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[42] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[43] Luke N Darlow, Elliot J Crowley, Antreas Antoniou, and Amos J Storkey. Cinic-10 is not imagenet or cifar-10. *arXiv preprint arXiv:1810.03505*, 2018.

[44] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[45] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[46] Bargav Jayaraman and David Evans. Evaluating differentially private machine learning in practice. In *28th {USENIX} Security Symposium ({USENIX} Security 19)*, pages 1895–1912, 2019.

[47] Daniel Bernau, Philip-William Grassal, Jonas Robl, and Florian Kerschbaum. Assessing differentially private deep learning with membership inference. *arXiv preprint arXiv:1912.11328*, 2019.

A Proof of Theorem 4.1

Proof. Given noisy binary vector $\tilde{B}$, for any $x, x' \in \mathcal{X}$ we have:

$$
\begin{aligned}
\frac{P[\tilde{B}|x]}{P[\tilde{B}|x']} &= \frac{P[\tilde{B}|B,x]P[B|x]}{P[\tilde{B}|B',x']P[B'|x']} = \frac{P[\tilde{B}|B]}{P[\tilde{B}|B']} \\
&= \frac{\prod_{i=1}^{r} P[\tilde{B}_i|B_i]}{\prod_{i=1}^{rl} P[\tilde{B}_i|B_i']} = \prod_{i=1}^{r} \frac{P[\tilde{B}_i|B_i]}{P[\tilde{B}_i|B_i]} \\
&\leq \prod_{i}^{r} \max\left\{ \frac{P[\tilde{B}_i = 1|B_i = 1]}{P[\tilde{B}_i = 1|B_i' = 0]}, \frac{P[\tilde{B}_i = 0|B_i = 0]}{P[\tilde{B}_i = 0|B_i' = 1]} \right\} \\
&= \prod_{i=1}^{r} \left\{ \frac{p}{q} \right\} \\
&= \left\{ \frac{p}{q} \right\}^{r}
\end{aligned}
$$

where the second equality follows from the fact that the mapping from input x to binary vector B is deterministic, while the fifth inequality is based on the assumption that $p \geq q$, under which we need not consider another two situations, where

$$
\frac{P[\tilde{B}_i = 1|B_i = 0]}{P[\tilde{B}_i = 1|B_i' = 1]} = \frac{q}{p} \quad \text{or} \quad \frac{P[\tilde{B}_i = 0|B_i = 1]}{P[\tilde{B}_i = 0|B_i' = 0]} = \frac{1-p}{1-q}
$$

. ∎