DENOISING DIFFERENTIAL PRIVACY IN SPLIT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

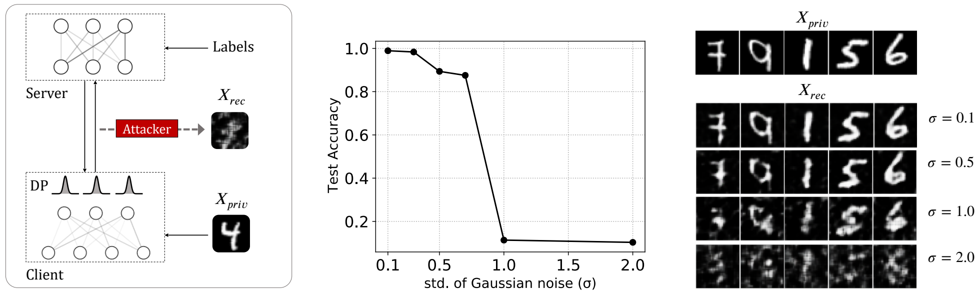
Differential Privacy (DP) is applied in split learning to address privacy concerns about data leakage. Previous work combines split neural network (SplitNN) training with DP by adding noise to the intermediate results during the forward pass. Unfortunately, DP noise injection significantly degrades the training accuracy of SplitNN. This paper focuses on improving the training accuracy of DP-protected SplitNNs without sacrificing the privacy guarantee. We propose two denoising techniques, namely scaling and random masking. Our theoretical investigation shows that both of our techniques achieve accurate estimation of the intermediate variables during the forward pass of SplitNN training. Our experiments with real networks demonstrate that our denoising approach allows SplitNN training that can tolerate high levels of DP noise while achieving almost the same accuracy as the non-private (i.e., non-DP protected) baseline. Interestingly, we show that after applying our techniques, the resultant network is more resilient against a state-of-the-art attack, compared to the plain DP-protected baseline.

1 INTRODUCTION

Privacy concerns in many application domains, such as finance, healthcare and on-line commerce, constraint the sharing of raw data that are necessary to train accurate deep neural networks (DNNs). Split learning (Gupta & Raskar, 2018; Vepakomma et al., 2018a) has recently emerged as a solution that allows different parties to learn a model collaboratively, without explicitly sharing raw input data. Typically, in two-party split learning the DNN model is divided between the *data owner*, a.k.a. the client, and the *label owner*, a.k.a. the server, as shown in Figure 1 (a). During training, in the forward pass the client forwards the intermediate results (IRs) (i.e., the parameters of the last layer in the client's part of the DNN) to the server. The server completes the forward pass and during the back-propagation it returns the gradients of the IRs to the client. Consequently, the client can train the joint model without revealing the private training data to the server.

Unfortunately, only sharing the IRs does not necessarily protect the private raw data. The shared IRs contain considerable latent information about the private data and can be used to stage powerful attacks, such as model inversion attack (He et al., 2019a; Zhang et al., 2020; Erdogan et al., 2021), label inference attack (Erdogan et al., 2021; Kariyappa & Qureshi, 2021; Li et al., 2021) and hijacking attack (Pasquini et al., 2021). Several works (Titcombe et al., 2021; Abuadbba et al., 2020; Mireshghallah et al., 2020; Wang et al., 2018) attempt to mitigate that risk through differential privacy (DP), which adds a certain amount of noise to the IRs before sharing with the other party. However, one of the fundamental issues of DP is the trade-off between its utility and privacy guarantee. While high levels of noise are favorable in terms of a strong privacy guarantee, the noise inevitably impacts the quality of the model (Abuadbba et al., 2020; Wang et al., 2020a; 2021); see Figures 1 (b) and (c) for an example. Thus, a fundamental question is: Can we improve the training accuracy of split learning under noise injection, without affecting the privacy guarantee of DP?

We answer the above question affirmatively by applying a post-processing denoising layer on top of noise-injected IRs in the split learning process. Our intuition is that *the injected noise introduces an error during the forward pass, which is dominated by variance when the noise level is high.* As long as we can reduce the variance by denoising techniques, the training quality should be improved. Such a post-processing layer will not impose any degradation on the privacy guarantee as long as it does not interact with the original private data (Dwork et al., 2014).



(a) (b) (c)

Figure 1: The trade-off between the security and utility in differentially private split learning. (a) Schematic representation of DP-SplitNN training, where DP is used to prevent private data leakage. (b) Best accuracy obtained in training a split CNN model on MNIST at different noise level (σ). (c) Training data reconstruction by hijacking attack when using DP with different noise level (σ).

Our main contributions are: (*i*) We propose two denoising techniques (i.e. scaling and masking) as post-processing layers on DP, to improve the training accuracy and stability of DP-SplitNN; see Section 3. (*ii*) Our theoretical investigation on a classification task in Section 3.2 shows that denoising can reduce the error caused by noise injection during the forward pass. (*iii*) We validate our claims through extensive numerical experiments on synthetic and real data (i.e., 4 DNN models on 4 different datasets) in Section 4. Moreover, (*iv*) we find that our masking technique, in addition to denoising, also enhances the resilience of split learning against the state-of-the-art hijacking attack (Pasquini et al., 2021); refer to Section 4.3.

2 Related Work

Differentially private federated learning (DP-FL). In horizontal FL (HFL), applying DP follows the same procedure as the centralized DP-SGD algorithm because the main privacy concern lies in the gradients/model updates during the communication. E.g., McMahan et al. (2017) has shown that directly integrating FedAvg with DP-SGD can protect user-level privacy for large language modeling tasks. Furthermore, many research (Hu et al., 2021; Liu et al., 2020; Agarwal et al., 2018; Kerkouche et al., 2021) propose combining DP with gradient compression techniques to further enhance the privacy guarantee as well as improve the communication efficiency. However, few attempts have been conducted to integrate DP with vertical FL (VFL) because VFL needs to transmit the intermediate results (IRs) instead of the gradients/model updates. Existing frameworks (Wang et al., 2020b; Chen et al., 2020) propose to add noise on participants' IRs to realize DP. Unfortunately, VFL (Chen et al., 2020) only demonstrates the impact of DP on the training accuracy when the noise scale is relatively low for some applications. The other framework, called HDP-VFL (Wang et al., 2020b), targeting linear model collaborative learning, proposes to directly perform sensitivity analysis on the IRs, which is not applicable for general DNN models.

Differentially private split learning (DP-SplitNN). Due to the vulnerability of SplitNN against model inversion attacks, Titcombe et al. (2021) proposed to apply DP on IRs during the inference time to prevent data reconstruction by the attacker. Shredder, proposed by Mireshghallah et al. (2020), adaptively generates a noise mask to minimize mutual information between input and intermediate data. However, these two methods only introduce noise injection during the inference time; thus, the privacy of training data is not preserved. Abuadbba et al. (2020) successfully applies noise to the IRs during the training to defend against model inversion attack on one-dimensional ECG data. It turns out that the noise has dramatically impacted the model's accuracy. Unlike previous works that only focus on the attack defense efficacy, we target to improve the training accuracy with a significant DP noise level.

Gaussian noise injections (GNIs) are a family of regularization methods for DNN training through adding random Gaussian noise on the activations or weights during the forward pass. It is similar to DP-SplitNN except for the following two differences: (*i*) There is no requirement of bounded sensitivity in injection objects. (*ii*) The noise scale is usually set small enough to avoid negative impacts on training accuracy. The explicit regularization effects of GNIs are well investigated in

(Camuto et al., 2020; Li & Liu, 2020; Lim et al., 2021), demonstrating better generalization for trained models over unseen data. In addition, GNIs can be used to improve the robustness of DNNs against adversarial attacks or data perturbations (Lim et al., 2021; He et al., 2019b). However, a recent study, Camuto et al. (2021) found that GNIs can also introduce some implicit bias on gradient updates, which inevitably degrades the overall training accuracy.

DP denoising. The denoising concept for DP has been well adopted in the field of statistical estimation (Hay et al., 2009; Nikolov et al., 2013; Bernstein et al., 2017), where they exploit some prior knowledge to design data release mechanism with better DP utility. Recently, Balle & Wang (2018) proposed an optimal denoising technique for Gaussian DP mechanism, where given $y \sim \mathcal{N}\left(f(x), \sigma^2 I\right)$ and their target is to find a postprocessing function g such that $g(y)$ is closer to $f(x)$ than y. This is different from denoising in DP-SplitNN as there are subsequent layers on top of the Gaussian mechanism in the training process, and we aim at denoising the output of each layer. Nasr & Shokri (2020) has also investigated using scaling as a denoising technique to improve the DP utility for DP-SGD. However, the authors scale up/down the noisy gradients based on the "usefulness" of gradients while we utilize scaling to minimize the estimation error of the noisy neural network outputs. Wang et al. (2020a) showed that adding Laplacian smoothing on Gaussian noise injected gradients can improve the utility of DP-SGD. To the best of our knowledge, we are the first to propose denoising techniques on the Gaussian noise injected intermediate results to improve the training accuracy of SplitNN.

3 Theoretical Guarantee

Notations. By $[n]$ we denote the set of n natural numbers $\{1, 2, \cdots, n\}$. By x_i, we denote the i^{th} component of vector x, while A_{ij} denote the $(i, j)^{\text{th}}$ component of a matrix, A. We use $\|x\|_2$ and $\|A\|_F$ to denote the ℓ_2 and the Frobenius norms of a vector x and a matrix A, respectively.

Problem setup. Let D be the training dataset with n elements, $\{(x_i, y_i^\star)\}_{i=1}^n$, drawn i.i.d. from some distribution, $P(\mathcal{X}, \mathcal{Y})$, where $x_i \in \mathbb{R}^d$ is the input feature vector, and $y_i^\star$ is the corresponding ground-truth label. We examine the performance of a machine learning algorithm, $\mathcal{A}$, with respect to a data distribution, $P(\mathcal{X}, \mathcal{Y})$, and denote $h_D := \mathcal{A}(D)$. We consider a split neural network classifying m classes. The network is divided among the client and the server, where the server network consists of fully connected (FC) layers and the output loss function. In general, there could be multiple FC layers, however, we only consider one FC layer as it is commonly adopted in practice and easy to analyze. Let $X \in \mathbb{R}^n$ be the input vector from the client-side split layer (i.e. IRs), and $M \in \mathbb{R}^{m\times n}$ be the trained weight matrix of the FC layer on the server side, which is independent of the input vector X during the loss calculation in the forward pass. At each iteration during training, the original split network processes a minibatch of training samples to calculate the loss and, during back-propagation, updates the weight. We follow this formalization in our theoretical analysis.

We consider the Gaussian mechanism of DP to protect the input vector X. For Laplace mechanism, please refer to Appendix E. Let the perturbed vector, $\tilde{X} \in \mathbb{R}^n$ follow the model: $\tilde{X} = X + \Delta$, where $\Delta_i \sim \mathcal{N}(0, \sigma)$, chosen from a zero mean Gaussian distribution and $\sigma \in \mathbb{R}^+$. Then $\tilde{X}$ is (ϵ, δ) differentially private w.r.t. X for $\sigma \geq \frac{\sqrt{2\ln\left(\frac{1.25}{\delta}\right)}\Delta_2(X)}{\epsilon}$, $\epsilon \in (0, 1)$ (Dwork et al., 2014), where $\Delta_2(X)$ denotes the ℓ_2-sensitivity of X. As we use `tanh` as the activation function in the client-side split layer, the sensitivity of X is naturally bounded. In general, denoising in DP-SplitNN can be formulated as follows: Let $f_i(X)$ be the output of the ith layer in the server model and f_i be a function that represents the first i layers, our goal is to find a post-processing function g, such that $f_i(g(\tilde{X}))$ is closer to $f_i(X)$ than $f_i(\tilde{X})$ for all possible i. In our setup, there are two possible f_i in the forward pass: (*i*) f_1– a linear layer, and (*ii*) f_2– a linear layer plus a nonlinear function (Softmax with negative log loss function). We consider one FC layer in our setup, and our primary focus is nonlinear classification tasks, although to understand the problem better denoising the linear layer is also important, which can be viewed as a regression task. For simplicity, we do not include iteration counter on M, X, or $\tilde{X}$. We use the following post-processing functions.

Random masking operator. Let R_p be a random matrix of 1 and 0 with identical and independently distributed entries, $(R_p)_{ij} \sim \text{Bernoulli}(p)$. Denote the support set, $\Omega_p \subset [m] \times [n]$ of R_p as $\Omega_p :=$

$\{(i,j)|(R_p)_{ij}=1\}$. Based on this, for a matrix, $A\in\mathbb{R}^{m\times n}$,

$$(\mathrm{R}_p[A])_{ij}=\begin{cases} A_{ij} & : i\in\Omega_p,\\ 0 & : \text{otherwise.}\end{cases}$$

From the definition, R_p is linear and is a projection operator, that is, $R_p^2=R_p$.

Scaling operator. For $\alpha>1$ and a matrix, $A\in\mathbb{R}^{m\times n}$, denote the element-wise scaling operator, $S_\alpha(\cdot):\mathbb{R}^{m\times n}\to\mathbb{R}^{m\times n}$, as $S_\alpha(A)=\frac{1}{\alpha}A$. While the random masking, R_p is a random operator, the scaling operator, S_α has no randomness in it.

3.1 A LINEAR LAYER

With the above setup, to explain our ideas more easily, we start with a neural network performing simple regression. Although this is not our primary focus, we believe this section will provide a better understanding. Because for the ℓ_2-regression task, no nonlinear activation function is required, so the problem is much simpler. That is, $y:=MX$ is the prediction of the output layer, and it does not involve any non-linearity. Theorem 1 describes results for a fully-connected DNN with an ℓ_2-regression task. Additionally, it explains how the scaling and masking parameters, α and p, respectively, are related to the noise scale σ, while denoising the output of a linear layer of a DNN for a given M and X. We calculate the expected test error, $\mathbb{E}\left[\|MX-Mh_D(\tilde{X})\|_2^2\right]$, where h_D is R_p and S_α, respectively, and compare against $\mathbb{E}\left[\|MX-M\tilde{X}\|_2^2\right]$, where $h_D=I_n$, an identity operator for original DP.

Theorem 1. *With the notations above, (*i*)* $\mathbb{E}\left[\|MX-MR_p(\tilde{X})\|_2^2\right]\le\mathbb{E}\left[\|MX-M\tilde{X}\|_2^2\right]$ *if and only if* $p\|M\odot\bar{X}\|_F^2+(1-p)\|MX\|_2^2\le\sigma^2\|M\|_F^2$*, where* $\bar{X}\in\mathbb{R}^{m\times n}$ *is a matrix obtained by stacking* $X^\top\in\mathbb{R}^{1\times n}$ *in each row, and* $\odot$ *denotes the elementwise product. (*ii*) Let* $\alpha>1$. $\mathbb{E}\left[\|MX-MS_\alpha(\tilde{X})\|_2^2\right]\le\mathbb{E}\left[\|MX-M\tilde{X}\|_2^2\right]$ *if and only if* $\frac{\|MX\|_2^2}{\|M\|_F^2}\le\left(\frac{\alpha+1}{\alpha-1}\right)\sigma^2$.

Remark 1. In Theorem 1, we considered the most commonly used mean square error (MSE), $\mathbb{E}\left[\|MX-MR_p(\tilde{X})\|_2^2\right]$ and $\mathbb{E}\left[\|MX-MS_\alpha(\tilde{X})\|_2^2\right]$, respectively, to compare against $\mathbb{E}\left[\|MX-M\tilde{X}\|_2^2\right]$. We use the MSE because it has nice mathematical properties; one can use other loss functions. This MSE is agnostic of the nature of the loss function used in DNN training.

Remark 2. Given $\mathcal{A}$, the expected MSE, $\mathbb{E}\left[\|MX-Mh_D(\tilde{X})\|_2^2\right]$, in Theorem 1, for different h_D was compared against $\mathbb{E}\left[\|MX-M\tilde{X}\|_2^2\right]$. Since, the expected MSE can be decomposed into bias and variance, by showing the relation between the expected MSEs, the bias-variance trade-off between different processes can be explained.

3.2 NONLINEAR LOSS FUNCTION FOR CLASSIFICATION TASK

In general, for classification problems such as image classification by CNN, movie review prediction by RNN, and many more, we require the softmax function for prediction and the negative log function as the loss function to train the DNN model; see their definitions in Appendix A.1. For a vector $z\in\mathbb{R}^m$, denote $s:\mathbb{R}^m\to(0,1)^m$ as the softmax function, and $\mathcal{L}_{LL}(y^\star,s(z))$ as the negative log loss function, where $y^\star$ is the true label.

In what follows, we show that for both masking and scaling operators, under certain conditions on the noise level, σ, it is possible to find parameters p and α, respectively, such that, by using any of these operations on DP-SplitNN, we incur a lower deviation in the loss value than using the DP alone when compared to the loss of the original SplitNN.

Masking operation. Quantifying the losses, $\mathcal{L}_{LL}(y^\star,s(MR_p(\tilde{X}))$ and $\mathcal{L}_{LL}(y^\star,s(M\tilde{X}))$ are critical tasks as they involve randomness from the masking operator and the Gaussian noise. Additionally, they are nonlinear functions, composed of softmax and logarithmic functions. Therefore, we require several intermediate results to prove the main result in Theorem 2. We provide the proofs in the Appendix, but stated them formally in the main body of the paper to present a sketch of proof of our main result, Theorem 2. The following Lemma [1], is the first intermediate result and instrumental in proving our main result as it approximates the expected logarithmic term in the log loss. In Lemma 1, we approximate $\mathbb{E}[\log(x)]$ by using Taylor's Theorem.

Lemma 1. *(Khuri, 2003, p. 117) Let x be a positive random variable. Then* $\mathbb{E}[\log(x)] = \log[\mathbb{E}(x)] - \frac{\mathrm{Var}(x)}{2(\mathbb{E}(x))^2} +$ *higher order terms, where* $\mathrm{Var}(x) = \mathbb{E}(x^2) - (\mathbb{E}(x))^2$.

Remark 3. The underline assumption is that x should have small higher order moments, $m_p = \mathbb{E}\left[|x - \mathbb{E}(x)|^p\right]$, for $p = 2, 3, \cdots$.

Note that, setting $x = \sum_{i=1}^m e^{(MR_p(\tilde{X}))_i}$ in Lemma 1 is the first step to quantify the expected loss value of DP-SplitNN with random masking, $\mathbb{E}\left[\mathcal{L}_{LL}(y^\star, s(MR_p(\tilde{X}))\right]$. Calculating $\mathbb{E}\left[\mathcal{L}_{LL}(y^\star, s(MR_p(\tilde{X}))\right]$ requires some auxiliary results on the $\mathbb{E}\left[\sum_{i=1}^m e^{(MR_p(\tilde{X}))_i}\right]$ and $\mathbb{E}\left[\left(\sum_{i=1}^m e^{(MR_p(\tilde{X}))_i}\right)^2\right]$. The following Lemma gives the details.

Lemma 2. *We have,* (i) $\mathbb{E}\left[\sum_{i=1}^m e^{(MR_p(\tilde{X}))_i}\right] = \sum_{i=1}^m \prod_{k=1}^n \left(pe^{m_{ik}x_k + \frac{m_{ik}^2\sigma^2}{2}} + (1-p)\right)$*; and* (ii) $\mathbb{E}\left[\left(\sum_{i=1}^m e^{(MR_p(\tilde{X}))_i}\right)^2\right] = \sum_{i,j}\left[\prod_{k=1}^n \left(pe^{(m_{ik}+m_{jk})x_k + \frac{(m_{ik}+m_{jk})^2\sigma^2}{2}} + (1-p)\right)\right]$.

Remark 4. Setting $p = 1$, in the loss function, we find the expected loss value, $\mathbb{E}\left[\mathcal{L}_{LL}(y^\star, s(M\tilde{X}))\right]$ due to DP-SplitNN (without random masking). Additionally, for $p = 1$, in Lemma 2, we recover $\mathbb{E}\left[\sum_{i=1}^m e^{(M\tilde{X})_i}\right] = \sum_{i=1}^m \prod_{k=1}^n (e^{m_{ik}x_k + \frac{m_{ik}^2\sigma^2}{2}})$.

Recall, we want to show that, by using random mask over a DP-SplitNN, we incur a lower deviation in the loss value than using the DP alone when compared to the loss of the original splitNN under certain condition. That is,

$$\mathbb{E}\left[|\mathcal{L}_{LL}(y^\star, s(MX)) - \mathcal{L}_{LL}(y^\star, s(MR_p(\tilde{X})))|\right] \leq \mathbb{E}\left[|\mathcal{L}_{LL}(y^\star, s(MX)) - \mathcal{L}_{LL}(y^\star, s(M\tilde{X}))|\right],$$

for all X. We formalize this in Theorem 2. Because the original splitNN, without DP, always produces the least loss, $\mathcal{L}_{LL}(y^\star, s(MX))$, the expressions in absolute values above are non-positive, and so we need only to verify that

$$\mathbb{E}\left[\mathcal{L}_{LL}(y^\star, s(MX)) - \mathcal{L}_{LL}(y^\star, s(M\tilde{X}))\right] \leq \mathbb{E}\left[\mathcal{L}_{LL}(y^\star, s(MX)) - \mathcal{L}_{LL}(y^\star, s(MR_p(\tilde{X})))\right].$$

for all X.

Theorem 2. *With the notations above, for classification problems, assume that $n \geq (MX)_i$ for $i = 1, 2, ..., m$. Then, if σ is large enough, there is some $\delta \in (0, 1)$ such that*

$$\mathbb{E}\left[|\mathcal{L}_{LL}(y^\star, s(MX)) - \mathcal{L}_{LL}(y^\star, s(MR_p(\tilde{X})))|\right] \leq \mathbb{E}\left[|\mathcal{L}_{LL}(y^\star, s(MX)) - \mathcal{L}_{LL}(y^\star, s(M\tilde{X}))|\right],$$

for $p \in (\delta, 1]$.

We remark that $n \geq (MX)_i$ is a technical assumption and can be easily satisfied in practice, basically requires the input dimension from the splitNN to be wide enough. We will pause here and provide a sketch of proof of Theorem 2. By the definitions of softmax and negative log loss, we have

$$\mathcal{L}_{LL}(y^\star, s(MR_p(\tilde{X}))) = -(MR_p(\tilde{X}))_{i^\star} + \log\left(\sum_{i=1}^m e^{(MR_p(\tilde{X}))_i}\right), \tag{1}$$

where $i^\star$ is the location of true label in $y^\star$. For fixed M and $\tilde{X}$, (1) is a function of p for $p \in (0, 1]$. Denote $\mathcal{F}(p) := \mathbb{E}\left[\mathcal{L}_{LL}(y^\star, s(MR_p(\tilde{X})))\right]$, and consequently, $\mathcal{F}(1) = \mathbb{E}\left[\mathcal{L}_{LL}(y^\star, s(M\tilde{X}))\right]$;

[1]See similar expression in Teh et al. (2006) with a restrictive assumption; assumption in Lemma 1 is more general.

see Remark 4. By using Lemma 1 on (1), we can approximate $\mathcal{F}(p)$ by

$$\mathcal{F}(p) \approx -p(MX)_i + \log\left(\mathbb{E}\left[\sum_{i=1}^{m} e^{(MR_p(\tilde{X}))_i}\right]\right) - \frac{\operatorname{Var}\left(\sum_{i=1}^{m} e^{(MR_p(\tilde{X}))_i}\right)}{2\left(\mathbb{E}\left[\left(\sum_{i=1}^{m} e^{(MR_p(\tilde{X}))_i}\right)\right]\right)^2}. \tag{2}$$

Our goal is to show that $\mathcal{F}(p) \leq \mathcal{F}(1)$ when $p \in (\delta, 1)$, for some $\delta \geq 0$. By using Lemma 2 in (2) and differentiating with respect to p, we can show, $\mathcal{F}'(1) \geq 0$. This would imply that $\mathcal{F}(p)$ is an increasing function of $p \in (\delta, 1]$, for some $\delta \in (0, 1)$. This in turn gives us $\mathbb{E}\left[\mathcal{L}_{LL}(y^\star, s(MR_p(\tilde{X})))\right] \leq \mathbb{E}\left[\mathcal{L}_{LL}(y^\star, s(M\tilde{X}))\right]$, and concludes the proof of Theorem 2; see details in Appendix A.3.1.

Scaling operation. Similarly, by using scaling over a DP-SplitNN, under certain condition on the noise, σ, we obtain a lower deviation in the loss than using the DP alone when compared to the loss of the original splitNN. We quote the result in Theorem 3; see Appendix A.3.2 for sketch of proof.

Theorem 3. *With the notations above, for classification problems, if* $\sigma^2 \geq \max_{i,i^\star} \frac{\sum_{k=1}^{n}(m_{i^\star k} - m_{ik})x_k}{\sum_{k=1}^{n} m_{ik}^2}$*, for* $i = 1, 2, ..., m$*, then there exists a* $\delta' \in (0, 1)$ *such that*

$$\mathbb{E}\left[|\mathcal{L}_{LL}(y^\star, s(MX)) - \mathcal{L}_{LL}(y^\star, s(MS_\alpha(\tilde{X})))|\right] \leq \mathbb{E}\left[|\mathcal{L}_{LL}(y^\star, s(MX)) - \mathcal{L}_{LL}(y^\star, s(M\tilde{X}))|\right],$$

for $\alpha \in (1, \frac{1}{\delta'}]$.

For concentration of the errors in Theorem 1 and Theorem 2; see Appendix A.3.3, and to understand how our post-processing techniques always preserves privacy bound; see Appendix B.

4 EXPERIMENTAL EVALUATION

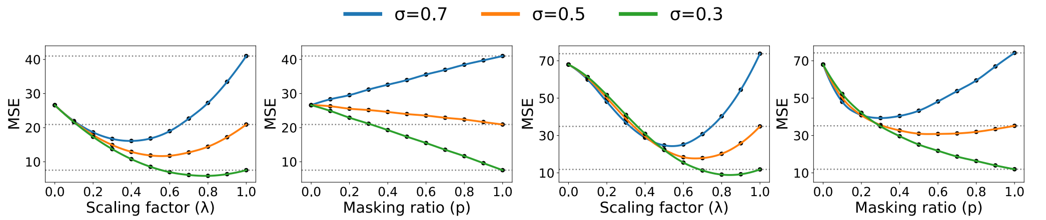
In Section 4.1, we validate our theoretical claims through numerical simulation on synthetic data, and in Section 4.2 we report results on DNN classification tasks.

4.1 SIMULATION

Setup. The following numerical simulations verify the results of Theorem 1 and 2. Since $X \in [-1, 1]$, output of `tanh` function, and M is usually randomly initialized around 0 in the actual training, in simulation, we sample the entries of X and M from a uniform distribution on $[-1, 1]$. The MSE corresponds to $\mathbb{E}\left[\|MX - MR_p(\tilde{X})\|_2^2\right]$, for linear case and $\mathbb{E}\left[|\mathcal{L}_{LL}(y^\star, s(MX)) - \mathcal{L}_{LL}(y^\star, s(MR_p(\tilde{X})))|\right]$, for nonlinear case, where R_p can be replaced by S_α for scaling. Moreover, for both operators, masking and scaling, when $p = 1$ and $\lambda = 1$, respectively, the MSE is $\mathbb{E}\left[\|MX - M\tilde{X}\|_2^2\right]$, for the linear case, and $\mathbb{E}\left[|\mathcal{L}_{LL}(y^\star, s(MX)) - \mathcal{L}_{LL}(y^\star, s(M\tilde{X}))|\right]$, for the nonlinear case and considered as baseline. In Figure 2, for each plot, we draw a line parallel to the X-axis from these baseline MSEs. The expectations are calculated by taking the average on k simulation results, where $k = 1000$.

Scaling simulation. In Figure 2 (a), each curve corresponds to a different noise scale, σ. By decreasing the scaling factor, λ for each σ, the MSE first decreases from the baseline to a minimum then increases, indicating an optimal λ for each σ. The NASC condition in Theorem 1 (*ii*) also infers that. For fixed M, X, this condition implies it is possible to find a smaller λ when σ is large. We make similar observations for the nonlinear case; see Figure 2 (c).

Masking simulation. Figure 2 (b) shows that by decreasing the masking ratio, p, the MSE does not necessarily become smaller unless σ is large enough. This verifies the claim of Theorem 1 (*i*). More importantly, there is an *almost linear* relationship between MSE and the masking ratio as $p \to 1$. This coincides with the expression of MSE with masking given in (7); see Appendix. We hypothesize that while both X, M are drawn from Uniform distribution, the coefficient of p^2 might become negligible. Hence, the coefficient of the linear term, p, which can be positive or negative



(a) $f_1(X) = MX$ (b) $f_1(X) = MX$ (c) $f_2(X) = \mathcal{L}_{LL}(y^\star, s(MX))$ (d) $f_2(X) = \mathcal{L}_{LL}(y^\star, s(MX))$

Figure 2: Simulation of how scaling factor ($\lambda = \frac{1}{\alpha}$) and masking ratio ($p$) influence the estimation error (MSE) under different noise levels (σ) for linear and nonlinear cases.

Table 1: SplitNN setup and training hyper-parameters

Model	**Dataset**	**Optimizer**	**Batch size**	**Epoch**	**lr**	**Split Config.** (client model || server model)	**Split Layer Size**
CNN	MNIST	SGD	64	4	0.1	2×Conv2d - MaxPool - Dense || Dense - Loss	256
ResNet20	CIFAR-10	SGD-M	128	160	0.1	Conv2d - 3×ResBlock - AvgPool || Dense - Loss	256
MLP	IMDB	Adam	64	2	0.01	Embedding - AvgPool - Dense || Dense - Loss	16
LSTM	Names	SGD	800	150	2	Embedding - RNN || Dense - Loss	128

depending on the noise scale σ, dominates MSE. Interestingly, the MSE decreases by reducing the noise level when $\sigma = 0.7$, implying a potential optimal denoising point given the lowest MSE. Results from Figure 2 (d), with the nonlinear loss, reflects the result in Theorem 2—σ^2 must be large for the *improvement to be possible*—when σ is too small (MSE curve for $\sigma = 0.3$), the masking does not work; the larger the σ, the more improvements one can expect by using masking. Moreover, when σ is large enough, there exists an $p \in (\delta, 1)$, for some $\delta \geq 0$ such that masking on top of DP incurs a lower MSE than the baseline. This indicates that an optimal denoising point is possible by using masking in DNN training for large noise.

Takeaway message. Figures 2 (a) and (c) indicate that regardless of the noise scale, σ, it is always possible to find a scaling factor such that using scaling over a DP-SplitNN incurs a lower MSE than the baseline. However, this is not always the case for the masking operator—σ must be significant for rendering the improvement from masking. Nevertheless, in practice, we see that masking performs better than scaling in some cases; see Section 4.2. Notably, these observations are agnostic of the nature of the FC layer of the DNN, linear or nonlinear. For the simulation of the MSE during backward pass, please refer to Figure 7 (e)(f) in Appendix F.

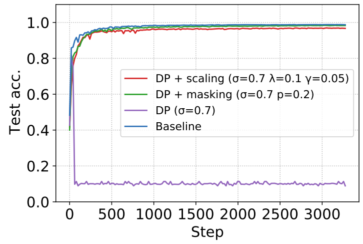
4.2 DNN Experiments

Datasets and models. We adapt the benchmarks from the popular `Pytorch` DP library `Opacus` [2] with split learning paradigm. It contains 2 vision tasks (image classification on MNIST (LeCun et al., 1998) and CIFAR-10 (Krizhevsky et al., 2009)), a recommendation task (movie review prediction on IMDB (Maas et al., 2011)) and a language modeling task (from `Pytorch` Tutorial [3]). The models also span across various architectures including convolutional neural network (CNN) , residual network (ResNet), recurrent neural network (RNN) and multi-layer perception (MLP). All training hyper-parameters are configured as default in order to maintain a fair comparison; see details in Table 1.

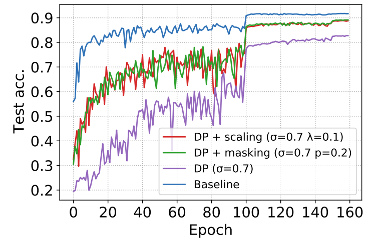
Setup and implementation. In our experiments, the target neural network is split at the last dense layer; see Table 1. The size of the split layer varies from 16 to 256. We use `tanh` activation function to bound the client's output in $[-1, 1]$ so that each input data's sensitivity is bounded. DP is implemented by injecting Gaussian noise on the `tanh` layer, with noise scale, σ, the standard deviation of Gaussian distribution. We implement both denoising techniques as a post-processing layer on top of the DP layer. The ratio $p \in (0, 1)$ describes the percentage of the elements we keep through masking. The scaling factor $\lambda = \frac{1}{\alpha} \in (0, 1)$ is used to scale down the tensor values. We run experiments on a local server equipped with one NVIDIA Tesla V100 GPU. The example code is provided in the supplemental material.

[2] https://github.com/pytorch/opacus

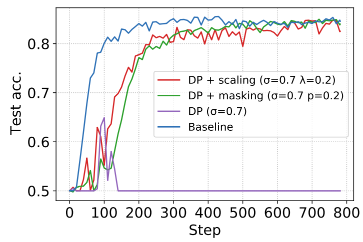
[3] https://pytorch.org/tutorials/intermediate/char_rnn_classification_tutorial.html



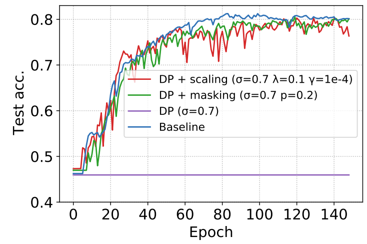
(a) CNN - MNIST



(b) ResNet20 - CIFAR10



(c) MLP - IMDB



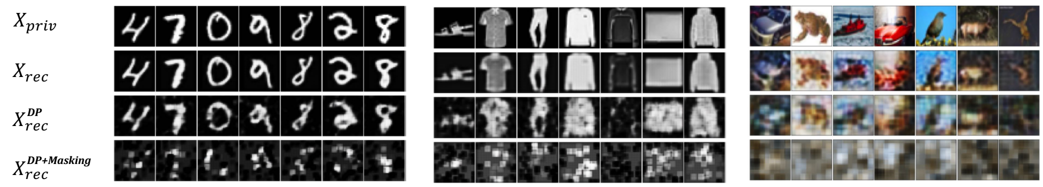
(d) LSTM - Names

Figure 3: Test accuracy of DP-SplitNN training with and without denoising (i.e. masking or scaling) in different training tasks. (σ: noise level, p: masking ratio, λ: scaling factor, γ: weight decay)

Denoising performance. We focus on demonstrating the effectiveness of both denoising techniques in various SplitNN training tasks. In Figure 3, we compare baseline SplitNN, SplitNN with DP, DP-SplitNN with the scaling or masking denoising technique. The DP noise level, σ is calibrated to a relatively high level such that the training accuracy of the plaintext DP-SplitNN suffers from the noise injection. Both scaling and masking are optimized by parameter tuning on λ, σ, γ; see Table 4 in Appendix F. In most cases, e.g. Figure 3 (a)(c)(d), once we inject a large noise ($\sigma = 0.7$), the overall training convergence is severely impacted so that the test accuracy is barely increased during the training. Such vulnerability is mainly due to two reasons: First, the dimension of the split layer is relatively small (less than 256), which is largely shrunk compared with other layers. Usually, high-dimensional data can better tolerate noise perturbation because it carries more information. Second, the learning rate is fixed as default in our experiment, but DP favors smaller learning rate in DNN training. After applying the scaling or masking layer on DP and fine-tuning some hyper-parameters, the training convergence vastly improves. In some cases, e.g. (a)(d), the improved accuracy due to masking is comparable with the baseline. Note that, in (a)(d), we introduce weight decay (γ) into the optimization of the scaling technique due to its stability issue, which we will further explain.

Denoising vs. Hyper-parameter tuning. To better understand the difference between denoising and traditional hyper-parameter tuning, we evaluate the MNIST image classification task by fine-tuning learning rate (lr), weight decay, masking ratio and scaling factor under high-level noise injection. We will present our main findings here. The detailed results are available in Appendix, Figure 6.

Noisy DNN training such as DP-SGD prefers smaller learning rate. We change lr from 0.1 to 0.001 and find that smaller lr indeed improves the training stability under a large noise injection ($\sigma = 0.7$). Weight decay, as a popular regularization method in DNN training, can be used to avoid over-fitting on noisy signals. We find that only heavy decay, e.g. $\gamma = 0.2, 0.4$, can help stabilize the noisy training till the end. However, both of them trade the convergence rate for stability and fail to reach the baseline accuracy after the training. Scaling can only improve the convergence at the beginning of the training and none of them manage to maintain the convergence till the end. This implies an inherent training stability issue with noise injection, which cannot be alleviated by pure denoising. Therefore, we combine scaling with weight decay and find that a small weight decay is sufficient to stabilize the training; see Figure 3a. On the contrary, the optimization of masking does not need weight decay. It can almost achieve the baseline convergence rate once the ratio p is optimized. Since there is a similarity between random masking and dropout, we hypothesize that the masking technique provides a similar regularization effect as weight decay. Both denoising techniques achieve significantly better training quality than pure hyper-parameter tuning.



(a) MNIST (b) Fashion-MNIST (c) CIFAR-10

Figure 4: Private training data recovery by FSHA in split learning on 3 public datasets: (a) MNIST, (b) Fashion-MNIST, and (c) CIFAR-10. In all cases, X_{priv}: the original private data, X_{rec}: obtained by FSHA without any protection of DP, X_{rec}^{DP}: obtained by FSHA under the protection of DP (σ = 0.7), $X_{rec}^{DP+Masking}$: obtained by FSHA under the protection of Random Masking enhanced DP ($\sigma = 0.7, p = 0.2$).

4.3 Attack defense

Setup. We demonstrate how the random masking technique can improve data privacy in defense against the recent feature-space hijacking attack (FSHA) (Pasquini et al., 2021) in split learning (see details in Appendix C). We evaluate the attack performance on three public datasets: MNIST, Fashion-MNIST (Xiao et al., 2017), and CIFAR-10. The client model configuration follows the setting in Table 1. The noise scale, σ and masking ratio, p are set to 0.7 and 0.2, respectively. We compare the attack performance by visualizing the recovered private data between the original FSHA (X_{rec}), FSHA under the protection of the vanilla DP (X_{rec}^{DP}), and FSHA under the protection of Random Masking enhanced DP ($X_{rec}^{DP+Masking}$). In all attack experiments, the training iteration is set to 10K to reach an adequate reconstruction result.

Results. Figure 4 shows that the original FSHA can reconstruct the private data with very high accuracy for MNIST and Fashion-MNIST but only keep the original images' looking for CIFAR-10. This is consistent with the attack performance in Pasquini et al. (2021)—attack on low-entropy images usually requires less effort and can produce a high-quality reconstruction. Next, we apply DP to the intermediate results and conduct data reconstruction on the perturbed data by FSHA. We can see that for MNIST, the digit on the reconstructed image is recognizable. For more complex images (Fashion-MNIST) and color images (CIFAR-10), although DP can hide most of the details in the images, we can still relate the constructed image with its original one by looking at the outline or the background color. The efficacy of DP is proportional to the noise scale; however, larger noise would cause a fatal error in the practical application. Lastly, when we combine DP with masking, the reconstructed images are fully damaged, and thus, the data privacy is greatly enhanced in this attack. See Figures 8-10 in Appendix F for results with different σ, p, λ.

5 Conclusion and Discussion

This paper proposes scaling and masking as denoising techniques for DP-SplitNN training without degrading the privacy guarantee. We show theoretically and empirically that denoising helps achieve more accurate intermediate outputs in DNN training under noise injection, which significantly improves the stability and accuracy of DP-SplitNN training. In addition, we show that the masking technique can provide additional security enhancement against powerful attacks, demonstrating the possibility of co-optimization of denoising and attack defense.

We noticed a few limitations of our work: First, there exist different split learning setups that we did not cover in this work, e.g., labels and data could reside on the same side. Second, we only focus on the client-side privacy; however, the server may also use DP to protect their labels during the backward propagation. Third, although we have empirically showed that our denoising techniques are also effective for the backward pass, the theoretical investigation remains challenging. We defer them in our future work.

REFERENCES

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

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

Sharif Abuadbba, Kyuyeon Kim, Minki Kim, Chandra Thapa, Seyit A. Camtepe, Yansong Gao, Hyoungshick Kim, and Surya Nepal. Can we use split learning on 1d CNN models for privacy preserving training? In *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*, pp. 305–318, 2020.

Naman Agarwal, Ananda Theertha Suresh, Felix Xinnan X Yu, Sanjiv Kumar, and Brendan McMahan. cpSGD: Communication-efficient and differentially-private distributed SGD. *Advances in Neural Information Processing Systems*, 31, 2018.

Borja Balle and Yu-Xiang Wang. Improving the gaussian mechanism for differential privacy: Analytical calibration and optimal denoising. In *International Conference on Machine Learning*, pp. 394–403. PMLR, 2018.

Garrett Bernstein, Ryan McKenna, Tao Sun, Daniel Sheldon, Michael Hay, and Gerome Miklau. Differentially private learning of undirected graphical models using collective graphical models. In *International Conference on Machine Learning*, pp. 478–487. PMLR, 2017.

Alexander Camuto, Matthew Willetts, Umut Simsekli, Stephen J. Roberts, and Chris C. Holmes. Explicit regularisation in Gaussian noise injections. *Advances in Neural Information Processing Systems*, 33:16603–16614, 2020.

Alexander Camuto, Xiaoyu Wang, Lingjiong Zhu, Chris Holmes, Mert Gurbuzbalaban, and Umut Simsekli. Asymmetric heavy tails and implicit bias in gaussian noise injections. In *International Conference on Machine Learning*, pp. 1249–1260. PMLR, 2021.

Tianyi Chen, Xiao Jin, Yuejiao Sun, and Wotao Yin. Vafl: A method of vertical asynchronous federated learning. *arXiv preprint arXiv:2007.06081*, 2020.

Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3-4):211–407, 2014.

Ege Erdogan, Alptekin Kupcu, and A Ercument Cicek. Unsplit: Data-oblivious model inversion, model stealing, and label inference attacks against split learning. *arXiv preprint arXiv:2108.09033*, 2021.

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

Michael Hay, Chao Li, Gerome Miklau, and David Jensen. Accurate estimation of the degree distribution of private networks. In *2009 Ninth IEEE International Conference on Data Mining*, pp. 169–178. IEEE, 2009.

Zecheng He, Tianwei Zhang, and Ruby B Lee. Model inversion attacks against collaborative inference. In *Proceedings of the 35th Annual Computer Security Applications Conference*, pp. 148–162, 2019a.

Zhezhi He, Adnan Siraj Rakin, and Deliang Fan. Parametric noise injection: Trainable randomness to improve deep neural network robustness against adversarial attack. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 588–597, 2019b.

Rui Hu, Yanmin Gong, and Yuanxiong Guo. Federated Learning with Sparsification-Amplified Privacy and Adaptive Optimization. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, 2021.

Sanjay Kariyappa and Moinuddin K Qureshi. Gradient inversion attack: Leaking private labels in two-party split learning. *arXiv preprint arXiv:2112.01299*, 2021.

Raouf Kerkouche, Gergely Ács, Claude Castelluccia, and Pierre Genevès. Compression boosts differentially private federated learning. In *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 304–318. IEEE, 2021.

André I Khuri. *Advanced calculus with applications in statistics*. John Wiley & Sons, 2003.

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

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Oscar Li, Jiankai Sun, Xin Yang, Weihao Gao, Hongyi Zhang, Junyuan Xie, Virginia Smith, and Chong Wang. Label leakage and protection in two-party split learning. *arXiv preprint arXiv:2102.08504*, 2021.

Yinan Li and Fang Liu. Adaptive Gaussian Noise Injection Regularization for Neural Networks. In *International Symposium on Neural Networks*, pp. 176–189. Springer, 2020.

Soon Hoe Lim, N. Benjamin Erichson, Liam Hodgkinson, and Michael W. Mahoney. Noisy recurrent neural networks. *Advances in Neural Information Processing Systems*, 34, 2021.

Ruixuan Liu, Yang Cao, Masatoshi Yoshikawa, and Hong Chen. Fedsel: Federated sgd under local differential privacy with top-k dimension selection. In *International Conference on Database Systems for Advanced Applications*, pp. 485–501. Springer, 2020.

Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pp. 142–150, 2011.

H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. *arXiv preprint arXiv:1710.06963*, 2017.

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*, pp. 3–18, 2020.

Milad Nasr and Reza Shokri. Improving deep learning with differential privacy using gradient encoding and denoising. *arXiv preprint arXiv:2007.11524*, 2020.

Aleksandar Nikolov, Kunal Talwar, and Li Zhang. The geometry of differential privacy: the sparse and approximate cases. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pp. 351–360, 2013.

Dario Pasquini, Giuseppe Ateniese, and Massimo Bernaschi. Unleashing the tiger: Inference attacks on split learning. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pp. 2113–2129, 2021.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Yee Teh, David Newman, and Max Welling. A collapsed variational bayesian inference algorithm for latent dirichlet allocation. *Advances in neural information processing systems*, 19, 2006.

Tom Titcombe, Adam J. Hall, Pavlos Papadopoulos, and Daniele Romanini. Practical defences against model inversion attacks for split neural networks. *arXiv preprint arXiv:2104.05743*, 2021.

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*, 2018a.

Praneeth Vepakomma, Tristan Swedish, Ramesh Raskar, Otkrist Gupta, and Abhimanyu Dubey. No peek: A survey of private distributed deep learning. *arXiv preprint arXiv:1812.03288*, 2018b.

Praneeth Vepakomma, Otkrist Gupta, Abhimanyu Dubey, and Ramesh Raskar. Reducing leakage in distributed deep learning for sensitive health data. *arXiv preprint arXiv:1812.00564*, 2019.

Bao Wang, Quanquan Gu, March Boedihardjo, Lingxiao Wang, Farzin Barekat, and Stanley J. Osher. DP-LSSGD: A stochastic optimization method to lift the utility in privacy-preserving ERM. In *Mathematical and Scientific Machine Learning*, pp. 328–351. PMLR, 2020a.

Chang Wang, Jian Liang, Mingkai Huang, Bing Bai, Kun Bai, and Hao Li. Hybrid differentially private federated learning on vertically partitioned data. *arXiv preprint arXiv:2009.02763*, 2020b.

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*, pp. 2407–2416, 2018.

Wenxiao Wang, Tianhao Wang, Lun Wang, Nanqing Luo, Pan Zhou, Dawn Song, and Ruoxi Jia. Dplis: Boosting utility of differentially private deep learning via randomized smoothing. *Proceedings on Privacy Enhancing Technologies*, 4:163–183, 2021.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Yuheng Zhang, Ruoxi Jia, Hengzhi Pei, Wenxiao Wang, Bo Li, and Dawn Song. The secret revealer: Generative model-inversion attacks against deep neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 253–261, 2020.