Secure Split Learning against Property Inference, Data Reconstruction, and Feature Space Hijacking Attacks

Yunlong Mao[1], Zexi Xin[1], Zhenyu Li[2], Jue Hong[3], Qingyou Yang[3], and Sheng Zhong[1]

[1] State Key Laboratory for Novel Software Technology, Nanjing University, China
[2] University of California San Diego, USA
[3] ByteDance Ltd.

Abstract. Split learning of deep neural networks (SplitNN) has provided a promising solution to learning jointly for the mutual interest of a guest and a host, which may come from different backgrounds, holding features partitioned vertically. However, SplitNN creates a new attack surface for the adversarial participant. By investigating the adversarial effects of highly threatening attacks, including property inference, data reconstruction, and feature hijacking attacks, we identify the underlying vulnerability of SplitNN. To protect SplitNN, we design a privacy-preserving tunnel for information exchange. The intuition is to perturb the propagation of knowledge in each direction with a controllable unified solution. To this end, we propose a new activation function named R^3eLU, transferring private smashed data and partial loss into randomized responses. We give the first attempt to secure split learning against three threatening attacks and present a fine-grained privacy budget allocation scheme. The analysis proves that our privacy-preserving SplitNN solution provides a tight privacy budget, while the experimental results show that our solution performs better than existing solutions in most cases and achieves a good tradeoff between defense and model usability.

Keywords: Privacy preservation · Inference attack · Reconstruction attack · Feature space hijacking attack · Split learning

1 Introduction

Private data, such as biological information and shopping history, is potentially valuable for commercial usage. Commercial agencies can learn users' preferences and make proper recommendations using their private data. Meanwhile, users could enjoy personalized services by sharing their privacy with service providers. However, both agencies and users are worried about their private data being abused. Besides, since commercial agencies nowadays are dedicated to providing high-quality services in vertical industries, like social networks or online shopping, their user profiles are business-relevant and highly homogeneous. But it commonly requires diversified features for building deep learning models. For

these reasons, using multifarious private data for building a satisfying model is essential but challenging.

Fortunately, the collaborative learning paradigm [26] has emerged as a promising solution. Collaborative learning enables participants from different interests to learn a shared model jointly. A well-known collaborative learning paradigm is federated learning [26], focusing on the coordination of distributed participants. Meanwhile, another paradigm split neural network (SplitNN for short) [11, 2] is designed explicitly for vertically partitioned features. Participants from different backgrounds could contribute with distinct feature representations. By combining different features, SplitNN is supposed to be more expressive in the real world. Notably, SplitNN has already been used for building industry-level frameworks, such as FATE [39] and Syft [31].

However, collaborative learning paradigms are faced with severe security issues. Roughly speaking, there are three kinds of threats, inference attack [29, 35], reconstruction attack [17, 34], and poisoning attack [37, 18]. An inference attack discloses attributes or membership information of specific data samples of the participants, while a reconstruction attack seeks to generate data samples similar to participants' private data. In [5], the reconstruction attack and the inference attack are studied simultaneously, enabling the host to recover the client's data and allowing an honest-but-curious host to infer the labels with desirable accuracy. Unlike these two kinds of threats, a poisoning attack aims to put harmful data into collaborative learning for malicious purposes rather than stealing private information. Moreover, a feature space hijacking attack considers a malicious participant, enlarging the attack effect of inference and reconstruction. Some excellent defensive solutions have been proposed for federated learning since it is a more general paradigm. According to the techniques used, these solutions can be roughly classified into differential privacy solutions [36, 22] and secure multiparty computation solutions [42, 30].

Unfortunately, security issues in SplitNN are barely discussed. The workflow of SplitNN has a unique asymmetric design. Therefore, most solutions for secure federated learning are not suitable for SplitNN. Secure multiparty computing solutions like homomorphic encryption can achieve ideal data confidentiality [41, 33], but the overhead introduced is still far away from practical uses. It has been proved in [12] that the defense performance of applying differential privacy in a general way to split learning is far away from the expectation when dealing with a feature space hijacking attack (FSHA) [32]. To defend against FSHA, two novel methods are designed in [6] for a split learning client to detect if it is being targeted by a hijacking attack or not. One approach is an active method relying on observations about the learning object, and the other one is a passive method at a higher computing cost. Thus, for the first attempt at privacy-preserving SplitNN for both the server and the client, we offer a unified solution for addressing different threats, including inference, reconstruction, and hijacking attacks. We focus on the abovementioned threats since they share a similar adversarial goal of privacy disclosure, while poisoning attacks need to be studied separately [7, 24].

We also note that there is an inherent contradiction [5] between privacy preservation and model usability, especially when private information of two sides should be considered in SplitNN. Through the investigation of the attack effect, we find that the attacker has sufficiently high success rates when disclosing the privacy of either party. To defeat the attacks [17, 35, 32] and give privacy guarantees on both sides in SplitNN, we make the following contributions:

- We investigate the privacy leakage issue in SplitNN by adapting inference and reconstruction attacks from federated learning. For the first attempt at securing SplitNN against multiple attacks, we propose a unified solution based on a newly designed activation function.
- We offer strong privacy guarantees for both sides of SplitNN. Moreover, a fine-grained privacy budget allocation scheme is given to achieve more efficient perturbations and improve privacy budget utilization.
- We implement and evaluate our solution using real-world datasets for different split learning tasks. The experimental result shows that our solution outperforms existing solutions when concurrently considering privacy preservation and model usability.

2 Problem Statement

2.1 Deep Learning

Given a training dataset $\boldsymbol{X}$ and DNN model parameters $\boldsymbol{\theta}$, a training task is to find approximately optimal $\boldsymbol{\theta}$ by minimizing a pre-defined loss function $\mathcal{L}$, regarding input $\boldsymbol{X}$. We assume that the optimizer used is a mini-batch stochastic gradient descent (SGD) algorithm, which updates $\boldsymbol{\theta}$ with a batch input of $\boldsymbol{X}$ iteratively. Assuming the batch size is M, then the total loss of $\boldsymbol{\theta}$ for a batch input $\boldsymbol{x} = \{x_i | x_i \in \boldsymbol{X}, i \in [1, M]\}$ should be $\sum_{x \in \boldsymbol{x}} \mathcal{L}(\boldsymbol{\theta}, x)$ in the t-th training iteration. The gradients of $\boldsymbol{\theta}$ for model updating should be estimated by $\frac{1}{M} \sum_{x \in \boldsymbol{x}} \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, x)$ approximately. Hence, parameters $\boldsymbol{\theta}$ can be updated as $\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t - \frac{1}{M} \sum_{x \in \boldsymbol{x}} \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, x)$. This mini-batch SGD-based optimizing procedure should be repeated until the model usability meets the requirement or the maximal count of iterations reaches.

2.2 Split Learning

SplitNN is an emerging collaborative learning paradigm partitioning the original neural network into different parts. There are commonly two types of participants in SplitNN, the host and the guest. For each training iteration, the forwarding input of the guest should be evaluated locally and passed to the host. Then the backpropagation should be initialized by the host and propagated to the guest. According to related studies [14], there exist several configurations of SplitNN. This paper will focus on the SplitNN design suitable for building models jointly with vertically partitioned features [2]. Assume that host and guest are two companies aiming to predict customer behaviors collaboratively. After an

embedding procedure, the host who holds label data merges embedded vectors using a predefined strategy, say averaging. Then the host will finish the rest of the forward propagation and initiate backward propagation. In Table 1, we give a benchmark of SplitNN using different merging strategies for building recommendation models with two public datasets, MovieLens [15] and BookCrossing [43]. For misaligned features where the server and the client may hold different shapes of features for the same data entry, we use zero padding to complement the missing features by zeros to retain the same shape of feature vectors. We give the top-10 hit ratio for the test in Table 1, which are the average of 30 runs in the same setting. During the experiment, we divide the original feature vector of 160 dimensions into two parts. One is 96 dimensional while the other is 64 dimensional. We notice that merging strategies have no significant influence on performance. Thus, averaging will be used as the default setting.

Table 1. Top-10 hit ratio (%) of SplitNN using different merging strategies. Batch size 32, learning rate 0.01. Model architectures are shown in the appendix.

		concat	element-wise					no split
			max	sum	avg.	mul.	min	
MovieLens	padding	56.62	56.26	56.35	56.89	55.23	**57.19**	57.21
	non-padding	55.38	54.75	54.95	**55.72**	54.52	55.08	
Book Crossing	padding	**61.70**	60.84	60.21	61.16	58.99	60.98	61.92
	non-padding	58.80	59.34	**59.44**	59.10	58.85	59.02	

2.3 Threat Model

Unlike updating local models separately in federated learning, participants of SplitNN are required to update local models cooperatively. Interactions between two parties pose new threats to each other [8, 21]. Hence, we will investigate privacy leakage threats from two perspectives. The host and the guest are honest but curious about the private data of each other. They are allowed to do any additional computations when they are following the split learning protocol. Assuming both parties are rational and privacy-aware, they will not exchange information except for the interactive interface. Please note that label leakage attacks and defense in SplitNN have attracted much attention recently. However, these topics are out of our discussion and need to be studied separately. Therefore, both parties may carry out certain attacks to infer or reconstruct each other's private data even simultaneously. The following part introduces the attacks which are considered potential threats against one party or both parties.

Property inference attack. Since the host has access to the output of the guest while the guest receives gradients containing private information of the host, an adversary can mount the property inference attack [9, 23] from both parties. Access to the output of the local model on the other side can be seen as a black-box query. In this setting, the adversary can infer properties of private data by observing query input and the corresponding output. By constructing elaborating shadow models, the adversary can steal substantial information from the target. In this way, the adversary acquires the capability of inferring some

properties (such as gender and age) of the data samples used for training. Denoted by F, T, $\mathcal{L}_F$ and l_i the inference model, target model, the loss function used for F and the label of each data, the adversarial goal is

$$\mathcal{A}_{PIA} = \arg\min_{F} \sum_{x_i \in \boldsymbol{X}} \mathcal{L}_F(F(T(x_i)), l_i), l_i \in \{0, 1\}. \tag{1}$$

Data reconstruction attack. A generative adversarial network (GAN) is an instance of generative models designed to estimate target data distribution [13]. Taking advantage of GANs, a data reconstruction attack is proposed in [17]. The adversary of a reconstruction attack aims to reconstruct the private training data of other participants in collaborative learning. The host or the guest or both can be adversarial. To mount the attack, the adversary augments the training data per iteration by inserting fake samples Z generated by a generator G. The global model will serve as a discriminator D. The adversary will affect global model updating by deceiving the target using fake training samples. For correcting the adversary, the target participant is supposed to put more private information into the learning. In this game-style training, the adversary may obtain substantial knowledge to reconstruct data samples as similar to target data as possible. Thus, the adversarial goal can be given as

$$\mathcal{A}_{DRA} = \min_{G} \max_{D} \frac{1}{|X|} \sum_{x \in X} logD(x) + \frac{1}{|X|} \sum_{z \in Z} log(1 - D(G(z))). \tag{2}$$

Feature space hijacking attack. Unlike attacks mentioned above, the feature space hijacking attack [32] considers a malicious adversary capable of manipulating the learning process. With the help of hijacking, the adversary can improve the performance of inference and reconstruction attacks. In this setting, the adversary can only be the host because label information is needed to mislead the victim. To mount the attack, the adversary uses a pilot model $\hat{f}$, an approximation of its inverse function $\hat{f}^{-1}$ with a shadow dataset and a discriminator D to distinguish the output from guest model f and the pilot model $\hat{f}$ during the training process. Then, the malicious host can send a suitable gradient to hijack the training of an honest guest by setting the goal as

$$\mathcal{A}_{FSHA} = log(1 - D(f(X_{PRIV}))).$$

3 Privacy-Preserving Split Learning

Our solution will be designed to preserve the privacy of the host and the guest concurrently. Ideally, the guest wants to collaborate with the host under the condition that the host should disclose no private data and vice versa. However, unlike conventional model publishing scenarios, the host and guest in SplitNN are required to exchange intermediate results continually. These continuous queries significantly increase the risk of privacy leakage for both sides. Moreover, the attack surface of SplitNN is inside the neural network, which is different from

situations studied in end-to-end models [25, 40]. Our work offers the first privacy-preserving SplitNN solution for defending against multiple attacks from two directions. The fundamental idea is to construct a bidirectional privacy-aware interface between the host and the guest. Noting that components of a neural network are loosely coupled, any output port may be a candidate for the interface. However, recent studies have proved that activation functions are more adaptive for perturbed operands [10]. Moreover, activation functions have various forms, which are flexible for configuration. As a result, we design a new variant of ReLU as an interface for SplitNN.

3.1 R^{3}eLU: Randomized-Response ReLU

Inspired by randomized response mechanisms [38], we design a new activation function R^{3}eLU (randomized-response ReLU) for SplitNN. Specifically, R^{3}eLU consists of a randomized-response procedure [38] and a Laplace mechanism [4]. This combination is not arbitrary but a complementary result. The original randomized response is good at statistical analysis of item sets. But the activation result of an input sample is commonly a continuous variable. The Laplace mechanism is a classic approach for differential privacy, handling continuous variables. But the perturbation is hard to be controlled, especially when the sensitivity degree of a query function is relatively large. It also means that it is highly risky to adopt the Laplace mechanism to an activation function directly. In SplitNN, we consider the model held by each party as a query function. Remember that both parties need to protect their privacy, so the sensitivity is bounded by the output of the cut layer in the process of forward propagation for the guest and by the gradient in the process of backward propagation for the host.

Recall that the original ReLU is $f(v) = \max(0, v)$, $v \in \mathcal{R}$. A randomized-response variant should yield a proper substitute for replacing real activations with a probability of p. We consider the activations of the cut layer as item sets and apply randomized-response on them. If we yield 0 as the substitute for $v > 0$, then we can inactivate a part of ReLU results, serving as artificial perturbations. But nothing has been changed for $v \leq 0$. Thus, it is not privacy-preserving since $f(v) = 0$ also reveals private information, indicating $v \leq 0$. To enforce a strict privacy policy, we integrate a Laplace mechanism into the ReLU variant by adding noise $z \xleftarrow{r} \mathit{Laplace}(0, \sigma)$. In this way, we can give the definition of R^{3}eLU as

$$\mathrm{R}^3\mathrm{eLU}(v) = \begin{cases} \max(0, v + z), & \text{with probability } p, \\ 0, & \text{with probability } (1 - p). \end{cases} \tag{3}$$

3.2 Forward Propagation with R^{3}eLU

In forward propagation, the guest needs to transfer local forwarding results to the host. At this point, an adversarial host can mount property inference or reconstruction attacks. To stem the leakage, we recommend replacing the original

activating function with R^{3}eLU while leaving the rest unchanged. Algorithm 1 gives R^{3}eLU-forward procedure by integrating essential operations. Denoted by $\boldsymbol{v}^g$ (the superscript may be omitted for concision) input of the original ReLU of the guest and N the cardinality of $\boldsymbol{v}$. The first operation is to select the top K largest elements of $\boldsymbol{v}$ and zero the rest. Then the top-K elements are clipped by a hyper-parameter C. The abovementioned pre-processing is defined as a procedure $ClipK$, taking as input $\boldsymbol{v}$, constants K and C, outputting $\hat{\boldsymbol{v}}$. We pre-process the inputs of R^{3}eLU to bound the sensitivity. The method $ClipK$ preserves the maximum K absolute values and clips each vector in the l_1 norm for a clipping threshold C. For randomized responding activation states, we calculate the probability

$$p_i = \frac{1}{2} + \frac{\hat{v}_i}{\|\hat{\boldsymbol{v}}\|_\infty} \cdot \left(\frac{e^{\frac{\epsilon_p}{K}}}{1 + e^{\frac{\epsilon_p}{K}}} - \frac{1}{2}\right), \tag{4}$$

where ϵ_p indicates the privacy budget of randomized responding. The state of $\hat{v}_i$ will be deactivated with probability $1 - p_i$ as per R^{3}eLU definition. Finally, a Laplace mechanism with privacy budget ϵ_l is integrated into the R^{3}eLU-forward completing the procedure. Now, the guest will transmit $\tilde{\boldsymbol{a}}^g = \text{R}^3\text{eLU-forward}(\boldsymbol{v}^g, C, K, N, \epsilon_p, \epsilon_l)$ instead of $\boldsymbol{a}^g = \text{ReLU}(\boldsymbol{v}^g)$ to the host.

Algorithm 1: R^{3}eLU-forward procedure.

Input: original input $\boldsymbol{v}^g$, cardinality N, constants C and K, privacy parameters ϵ_p and ϵ_l, probability $\boldsymbol{p}$.
Output: activation $\tilde{\boldsymbol{a}}^g$.

$\hat{\boldsymbol{v}}^g \leftarrow ClipK(\boldsymbol{v}^g, C, K, N)$ `// pre-process`
for $i \leftarrow 1$ **to** N **do**
  $r \xleftarrow{r} \mathcal{N}(0, 1)$
  if $r < p_i$ **then**
    $\tilde{a}_i^g \leftarrow \max(\hat{v}_i^g + \texttt{Lap}(0, \frac{2KC}{\epsilon_l}), 0)$ `// activate`
  else
    $\tilde{a}_i^g \leftarrow 0$ `// deactivate`
  end
end
return $\tilde{\boldsymbol{a}}^g \leftarrow \{\tilde{a}^1, \tilde{a}^2, \ldots, \tilde{a}^N\}$

3.3 Private Backward Propagation

Privacy leakage also exists from the host's perspective. When the host finishes the rest of forwarding propagation after aggregating activations $\tilde{\boldsymbol{a}}^g$ and $\boldsymbol{a}^h$, the loss produced for backward propagation contains data privacy of the host and guest. According to recent studies of backward propagation [27, 34], intermediate results of model updating can cause severe data privacy leakage. Since the loss must be propagated to the guest, it is crucial to prevent the host from being attacked by an adversarial guest. However, the partial loss propagated ranges widely. Integrating a DP mechanism directly into the original ReLU is

unrealistic. Due to the randomized-response design, we can construct a privacy-preserving tunnel for backward propagation atop the derivative of R^{3}eLU.

Recall that the derivative value of ReLU for any input is either one or zero. A randomized-response variant will perturb the binary output randomly. Besides, randomly flipping still discloses real partial losses when value ones are not flipped. Therefore, a Laplace mechanism is used in the backward procedure. Now, we give a randomized-response derivative of R^{3}eLU

$$\nabla \mathrm{R}^3\mathrm{eLU}(\boldsymbol{\delta}^g, \tilde{\boldsymbol{a}}^g, \boldsymbol{v}^g) = \begin{cases} \boldsymbol{\delta}^g + \boldsymbol{z}, & \text{with probability } p, \\ 0, & \text{with probability } (1-p), \end{cases} \tag{5}$$

where $\boldsymbol{\delta}^g$ is the partial loss for the guest model, $\boldsymbol{z}$ is artificial noise. Similar to R^{3}eLU-forward, R^{3}eLU-backward also needs some essential operations. Thus, the same $ClipK$ process for top-K selecting and scalar clipping can be adopted for R^{3}eLU-backward. However, different from R^{3}eLU, absolute values are used because the partial loss instructs the gradient descent direction. In this case, we have $\hat{\boldsymbol{\delta}}^g = ClipK(\boldsymbol{\delta}^g, C, K, N)$. Moreover, a $Sign$ process is used to obtain signs.

For randomized responding, the probability of retaining an actual loss is

$$p_i = \frac{1}{2} + \frac{|\hat{\delta}_i|}{\||\hat{\delta}_i|\|_\infty} \cdot \left(\frac{e^{\frac{\epsilon_p}{K}}}{1 + e^{\frac{\epsilon_p}{K}}} - \frac{1}{2}\right), \tag{6}$$

where ϵ_p is the privacy budget for the randomized response. We now give the backward procedure R^{3}eLU-backward in Algorithm 2. Please note that although some existing solutions choose to disturb the gradients of two parties, we choose to perturb the partial loss regarding the guest's backpropagation while keeping the partial loss of the host model unchanged. In this way, a slighter influence is caused for the host compared with disturbing all gradients directly.

Algorithm 2: R^{3}elu-backward procedure.

Input: partial loss $\boldsymbol{\delta}^g$, cardinality N, constants C and K, privacy parameters ϵ_p and ϵ_l, probability $\boldsymbol{p}$.
Output: partial loss $\tilde{\boldsymbol{\delta}}^g$.

$|\hat{\boldsymbol{\delta}}| \leftarrow ClipK(|\boldsymbol{\delta}|, C, K, N)$ `// pre-process`
for $j \leftarrow 1$ **to** N **do**
  $r \xleftarrow{r} \mathcal{N}(0,1)$
  if $r < p_i$ **then**
    $\hat{\delta}_i \leftarrow Sign(\delta_i) \cdot |\hat{\delta}_i|$
  else
    $\hat{\delta}_i \leftarrow 0$ `// randomized response`
  end
  $\tilde{\delta}_i^g \leftarrow \hat{\delta}_i^g + \mathtt{Lap}(0, \frac{2KC}{\epsilon_l})$
end
return $\tilde{\boldsymbol{\delta}}^g = \{\tilde{\delta}^1, \tilde{\delta}^2, ..., \tilde{\delta}^N\}$

3.4 Dynamic Privacy Budget Allocation

To further reduce privacy loss and improve the utilization of the privacy budget, we recommend allocating the privacy budget for parameters dynamically instead of allocating uniformly. Based on [28], the importance of a parameter during training can be quantified by the error introduced when it is removed from the model. In particular, the importance I_j of $\theta_j \in \boldsymbol{\theta}$ is the squared difference of prediction errors caused by removing θ_j, i.e.,

$$I_j = (\mathcal{L}(\boldsymbol{x}, \boldsymbol{\theta}) - \mathcal{L}(\boldsymbol{x}, \boldsymbol{\theta} \setminus \{\theta_j\}))^2. \tag{7}$$

For efficiency concern, an approximating method is given in [28], estimating the importance I_j by its first-order Taylor expansion as

$$\hat{I}_j = (\nabla_{\theta_j} \mathcal{L}(\boldsymbol{\theta}, x) \cdot \theta_j)^2. \tag{8}$$

Given the importance of each parameter in the cut layer, the importance of a feature can be derived further. Specifically, the importance of a feature U_j, $j \in [1, N_u]$, where N_u is the total number of neurons in the cut layer, can be calculated as joint importance of relevant parameters by summing them up. Thus, $U_j = \sum_{\theta_k \in \boldsymbol{\theta}_{U_j}} \hat{I}_k$, where $\boldsymbol{\theta}_{U_j}$ is the set of all parameters directly connected to the j-th neuron.

Please note that the original importance estimation is designed for a well-trained model and cannot be directly applied to intermediate models during training. To tackle the problem, we give a dynamic estimation method by deriving the original method into a cumulative form. The importance of a feature will be accumulated as the training epoch increases. Specifically, the importance of the j-th neuron in the q-th training epoch is

$$U_j^q = \frac{\sum_{\theta_k \in \tilde{\boldsymbol{\theta}}_j} \hat{I}_k + U_j^{q-1} \times (q \times \lfloor T/n_t \rfloor + (t \bmod n_t) - 1)}{q \times \lfloor T/n_t \rfloor + (t \bmod n_t)}, \tag{9}$$

where n_t indicates the iteration number within a training epoch, T is the maximum training iteration number, and t is the current training iteration. Assuming that $T \mod n_t = 0$, then $q \in [1, T/n_t]$.

Based on the importance estimated, now we can dynamically allocate the privacy budget for different features. The intuition is to give larger budgets to more important features. Before the q-th training epoch begins, we can estimate a feature importance vector $\boldsymbol{U} = \{U_1^q, U_2^q, \ldots, U_{N_u}^q\}$. Accordingly, the privacy budget allocated to each feature will be $\epsilon_j = \epsilon \times U_j^q$, if ϵ is one unit budget. Then the total privacy budget for all features is $\epsilon_F = \sum_{j=\in[1,N_u]} \epsilon_j$. Now, we can set the probability of randomized response for R^{3}eLU-forward and R^{3}eLU-backward using the dynamic budget allocation,

$$p_i = \frac{1}{2} + \frac{U_i^q}{\|\boldsymbol{U}\|_\infty} \cdot \left(\frac{e^{\frac{\epsilon_p}{K}}}{1 + e^{\frac{\epsilon_p}{K}}} - \frac{1}{2}\right). \tag{10}$$

On the other hand, we can also allocate different privacy budgets for different iterations for better budget utilization. Given the total budget ϵ_T for all iterations, we assign the privacy budget $\epsilon_i = \frac{\epsilon_T}{2^i}$ to the i-th iteration as suggested by [3]. Since $\sum_{i=1}^{\infty} \frac{\epsilon_T}{2^i} = \epsilon_T$, according to the sequential composition theory of differential privacy, we can ensure that the whole training process achieves ϵ_T-differential privacy.

We note that the additional computing cost will be caused by the dynamic privacy budget allocation, which is dominated by the computation of the importance of each neuron at the cut layer. As the gradient of the cut layer can be preserved during the process of each backward propagation, the cost generated by the product of the gradients and neurons for each round is $O(N_u)$ where N_u is the number of neurons of the cut layer.

4 Privacy Analysis

We give privacy analysis results regarding the host and the guest, respectively[4].

Corollary 1. *Given privacy budgets ϵ_p and ϵ_l for randomized response and Laplace mechanism respectively, the output of R^3eLU-forward procedure is $(\epsilon_p + \epsilon_l)$-DP.*

Corollary 2. *Split learning for the guest with R^3eLU-forward achieves (ϵ_g, δ_g)-DP , where $\epsilon_g = \gamma\epsilon\sqrt{2T\ln(\frac{1}{\delta_g})} + \gamma\epsilon T(e^{\gamma\epsilon} - 1)$.*

Since we construct R^{3}eLU-forward and R^{3}eLU-backward using the same method, these two procedures have the same analysis result if we constrain that they use the same input. In this way, we can conclude that the output of R^{3}eLU-backward procedure is $(\epsilon_p + \epsilon_l)$-DP, where ϵ_p and ϵ_l are budgets for randomized response and Laplace mechanism, respectively.

Corollary 3. *Split learning for the host with R^3eLU-backward achieves (ϵ_h, δ_h)-DP, where $\epsilon_h = \gamma\epsilon\sqrt{2T\ln(\frac{1}{\delta_h})} + \gamma\epsilon T(e^{\gamma\epsilon} - 1)$.*

We remark that the dynamic privacy budget allocation improves budget utilization without any additional privacy loss. In other words, an important feature gets a higher probability of retaining the activation state than an insignificant feature, which leads to a larger privacy budget. But the total privacy budget of all features remains unchanged.

Corollary 4. *When the guest runs R^3eLU-forward procedure with the dynamic privacy budget allocation, the output is still ϵ-DP, $\epsilon = \epsilon_p + \epsilon_l$.*

Since activation values are clipped with constant C, $\max_{i \in \boldsymbol{h}}\{p_i\}$ will not be affected. Thus, the activation state with the dynamic privacy budget allocation is still ϵ_p-DP. Given the activation state $\boldsymbol{s}$ and sensitivity Δ, then

$$\frac{\Pr[\tilde{\boldsymbol{a}}^g|\hat{\boldsymbol{v}}, \boldsymbol{s}]}{\Pr[\tilde{\boldsymbol{a}}^g|\hat{\boldsymbol{v}}', \boldsymbol{s}]} \leq e^{\sum_{i=1}^{N}(\frac{C\sum_{j=1}^{N} U_j}{\epsilon_l \Delta U_i})} \leq e^{\sum_{i=1}^{N}(\frac{\epsilon_l U_i}{\sum_{j=1}^{N} U_j})} = e^{\epsilon_l} \tag{11}$$

[4] More details can be found in another version, http://arxiv.org/abs/2304.09515.

For any arbitrary $\boldsymbol{v}$ and $\boldsymbol{v}'$, the difference of R^{3}eLU-forward outputs can be bounded by

$$\frac{\Pr[\tilde{\boldsymbol{a}}^g|\boldsymbol{v}]}{\Pr[\tilde{\boldsymbol{a}}^g|\boldsymbol{v}']} = \frac{\Pr[\tilde{\boldsymbol{a}}^g|\hat{\boldsymbol{v}}, \boldsymbol{s}] \cdot \Pr[\boldsymbol{s}|\boldsymbol{s}_{\boldsymbol{a}}]}{\Pr[\tilde{\boldsymbol{a}}^g|\hat{\boldsymbol{v}}', \boldsymbol{s}] \cdot \Pr[\boldsymbol{s}|\boldsymbol{s}_{\boldsymbol{a}'}]} \leq e^{\epsilon_l} \cdot e^{\epsilon_p} = e^{\epsilon_l + \epsilon_p} \tag{12}$$

5 Evaluation

We evaluate our privacy-preserving SplitNN solution from two aspects, model usability, and privacy loss. To be comprehensive, we will compare our solution with the baseline (without any protection) and the most relevant defensive solutions, i.e., a primitive Laplace mechanism [4] and DPSGD [1], the most well-known privacy-preserving deep learning solution, in the same setting. We will use the same fixed total privacy budget and the same split way (shown in the appendix) for all solutions. For the primitive Laplace mechanism, we simply add Laplacian noise to activations and partial losses to protect the privacy of the guest and host, respectively. For DPSGD, we add artificial noises to gradients of models on either side. For our solution, we set $\epsilon_p = \epsilon_l = \frac{\epsilon}{2}$ and K as half number of features. We set C as 10. We use the dynamic privacy budget allocation for features in our solution and set the initial importance as zero for all features.

All defensive solutions will be evaluated using three real-world datasets, MovieLens [15] and BookCrossing [43] for the recommendation, and MNIST [20] and CIFAR100 [19] for image classification. The *MovieLens* 1-M dataset contains 1 million ratings of 4,000 movies collected from 6,000 users and users' demographic information such as gender and age. The *BookCrossing* dataset includes 278,858 users' demographic information and 1,149,780 ratings of 271,379 books. The *MNIST* database has 70,000 handwriting image examples of digital numbers from 0 to 9. The *CIFAR100* database has 60,000 image examples for 100 classes. Each image example has one superclass as its rough label and one class as its accurate label. We will use batch size 32, a learning rate of 0.01, and an Adam optimizer as default. Since different datasets and defensive solutions may require various epochs for split learning, we will compare the metrics when the learning converges or the privacy budget is drained. All experimental results are averaged across multiple runs.

Before the evaluation, we verify the feasibility of our dynamic importance estimation method. During the verification, we observe the importance estimation of the neurons in the cut layer. The importance estimation of each neuron is calculated as Eq. (9). By accumulating all intermediate results of the importance estimated for neurons, we find that the final importance is almost the same as obtained by the original estimation method on a well-trained model's final state. The importance estimation results of neurons are shown in Figure 1, proving the correctness of our dynamic importance estimation method and the existence of unbalanced feature importance. As can be seen, the similarity between the accumulated (dynamic) estimation and the original (stable) estimation indicates that our accumulated approach to estimating the importance of a neuron works as well as the original approach.

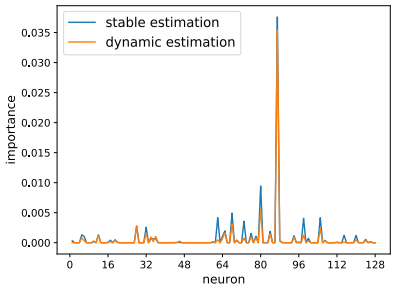

Fig. 1. Estimation results of neuron importance.

Table 2. Model usability results while preserving the privacy of the guest.

ϵ	MovieLens			BookCrossing			MNIST			CIFAR100		
	Laplace	DPSGD	**Ours**	Laplace	DPSGD	**Ours**	Laplace	DPSGD	**Ours**	Laplace	DPSGD	**Ours**
0.1	30.84%	32.29%	**34.03%**	57.02%	55.89%	**58.18%**	17.43%	30.21%	**32.41%**	20.25%	34.21%	**37.87%**
0.5	41.25%	43.69%	**43.87%**	57.67%	56.14%	**58.54%**	27.33%	58.43%	**60.38%**	39.74%	51.36%	**58.22%**
1.0	48.16%	49.09%	**50.56%**	58.02%	56.56%	**58.42%**	31.05%	75.58%	**76.60%**	46.19%	60.48%	**65.32%**
2.0	49.32%	50.38%	**50.49%**	58.74%	56.91%	**59.24%**	38.92%	92.90%	**93.53%**	56.30%	69.72%	**73.35%**
4.0	49.26%	**50.86%**	50.73%	59.01%	57.16%	**59.26%**	95.37%	**95.87%**	94.12%	57.04%	70.86%	**74.41%**

5.1 Model Usability

Since artificial perturbation may affect the learning procedure, we evaluate how SplitNN is affected by privacy-preserving solutions. Two asymmetric parties of SplitNN may have different influences on learning. Thus, we will evaluate model usability concerning privacy from the perspective of the guest or the host, respectively. We use an averaged test accuracy across all test samples for the evaluation of model usability. Precisely, the test accuracy of a recommendation model is calculated using a top-10 hit ratio, while the test accuracy of an image classifier is its prediction accuracy. In Table 2 and Table 3, we show the model usability results regarding various privacy budget values of the two parties. We note that model accuracy baselines of Movielens, BookCrossing, MNIST and CIFAR100 for SplitNN are 56.62%, 61.70%, 98.00% and 76.20%, respectively.

For the MovieLens model, our solution achieves the best model usability in most cases, especially with a smaller privacy budget. DPSGD has a better result when $\epsilon = 4$ for the guest. But a significant privacy leakage will be caused in this case. For the BookCrossing model, the model usability of our solution is relatively high in cases of protecting the guest and the host. Similarly, DPSGD achieves a better result when $\epsilon = 4$ by sacrificing the host's privacy. Results show some differences for the MNIST model. DPSGD has better results when protecting the host's privacy. The reason is that split learning for an image classification model segments image samples roughly, making our dynamic budget allocation approach malfunction. Meanwhile, DPSGD is not designed to protect partial loss in SplitNN, leading to an optimistic estimation of the threat against the host. On the contrary, our solution has a competitive performance in image classification. For the CIFAR100 model, our method outperforms other protection mechanisms. As the host has the major part of the model, the accuracy drops 5.5% for host

Table 3. Model usability results while preserving the privacy of the host.

ϵ	MovieLens			BookCrossing			MNIST			CIFAR100		
	Laplace	DPSGD	**Ours**	Laplace	DPSGD	**Ours**	Laplace	DPSGD	**Ours**	Laplace	DPSGD	**Ours**
0.1	31.47%	30.68%	**33.98%**	57.37%	57.46%	**58.26%**	27.64%	**33.45%**	32.36%	17.04%	34.22%	**38.96%**
0.5	41.75%	42.31%	**42.67%**	58.62%	58.24%	**58.59%**	55.38%	65.28%	**67.83%**	25.82%	43.96%	**53.72%**
1.0	47.43%	48.29%	**50.39%**	59.49%	58.44%	**59.77%**	71.95%	**89.74%**	88.14%	37.69%	55.28%	**61.48%**
2.0	49.86%	50.43%	**51.47%**	59.34%	59.97%	**60.27%**	89.15%	**92.66%**	92.52%	51.87%	65.67%	**69.60%**
4.0	49.57%	50.09%	**51.62%**	59.55%	**60.75%**	60.66%	94.61%	**95.37%**	95.01%	51.87%	66.89%	**70.70%**

protection while only 1.79% for guest protection. These results show that our method with dynamic privacy budget allocation can allocate appropriate privacy budget on different neurons and achieve high accuracy even on complex datasets and models while the primitive Laplace mechanism suffers a 24-percentage points drop in accuracy for CIFAR100 evaluation due to its indiscrimination on all neurons, as shown in Table 3.

5.2 Privacy Preservation

We evaluate the performance of privacy preservation by comparing attack results against SplitNN with and without the defense. We will mount property inference and data reconstruction attacks against the guest and the host, respectively. The prediction accuracy of the adversary's inference model will be used to measure the performance of the property inference attack. As for the data reconstruction attack, the adversary tries to generate data samples as similar as possible to the target's private data. In this case, a mean squared error (MSE) between a generated sample and a target data sample is commonly used for the adversarial effect measurement.

Table 4. Results of defending the guest against property inference attack.

ϵ	MovieLens			BookCrossing			MNIST			CIFAR100		
	Laplace	DPSGD	**Ours**	Laplace	DPSGD	**Ours**	Laplace	DPSGD	**Ours**	Laplace	DPSGD	**Ours**
0.1	66.99%	77.71%	**60.99%**	**54.76%**	73.29%	55.78%	**43.27%**	53.95%	44.33%	**50.76%**	79.14%	53.97%
0.5	66.16%	74.23%	**64.16%**	**54.97%**	74.52%	56.33%	46.92%	54.23%	**45.59%**	**51.47%**	79.26%	55.13%
1.0	**67.19%**	78.65%	68.18%	**55.03%**	74.96%	58.65%	47.58%	54.26%	**47.51%**	**50.40%**	79.37%	55.72%
2.0	68.65%	73.06%	**68.56%**	**54.85%**	74.26%	58.14%	**48.06%**	54.65%	52.87%	60.76%	79.37%	**58.81%**
4.0	**69.14%**	76.18%	71.91%	**54.92%**	74.33%	60.76%	**48.47%**	54.57%	55.73%	60.81%	79.35%	**58.03%**

Table 5. Results of defending the host against property inference attack.

ϵ	MovieLens			BookCrossing			MNIST			CIFAR100		
	Laplace	DPSGD	**Ours**	Laplace	DPSGD	**Ours**	Laplace	DPSGD	**Ours**	Laplace	DPSGD	**Ours**
0.1	53.46%	78.59%	**51.86%**	**54.55%**	74.35%	59.42%	60.34%	80.29%	**48.74%**	50.42%	51.46%	**41.89%**
0.5	53.46%	75.64%	**51.89%**	**54.62%**	74.36%	59.42%	59.82%	81.92%	**49.71%**	50.38%	52.23%	**44.26%**
1.0	53.46%	73.54%	**52.75%**	**54.95%**	74.39%	59.52%	59.74%	82.80%	**50.48%**	49.95%	51.95%	**44.78%**
2.0	**53.47%**	75.05%	59.77%	**54.40%**	74.39%	58.13%	60.38%	88.88%	**50.57%**	50.77%	51.67%	**50.16%**
4.0	**53.48%**	79.28%	56.52%	**54.95%**	74.39%	62.04%	60.62%	89.73%	**50.47%**	51.52%	51.76%	**51.27%**

Defense against property inference attack A property inference attack is to infer an existing property (or attribute) of data samples. For example, an adversarial host in our experiments infers the age attribute of the guest's data for a recommendation model. However, the host has no idea of the age distribution since training data is vertically partitioned. We carry out the same property

infence attack as [23]. We give evaluation results of the defensive effect of the guest and the host in Table 4 and Table 5, respectively. We use the prediction accuracy of the adversary's inference model as a criterion for evaluation. The higher the prediction accuracy, the more probable success the property inference attack may achieve. In other words, the worse the defensive effect is. As for an image classification model, an unknown patch of image samples will be inferred. The attack accuracy against baselines of MovieLens, BookCrossing, MNIST and CIFAR100 models can achieve above 80%, 79%, 94% and 87% by an adversarial host, 80%, 78%, 57% and 53% by an adversarial guest, respectively. However, our solution can effectively mitigate the adversarial effect during training and decrease the attack accuracy significantly. It should be noted that the primitive Laplace mechanism frustrates the inference attack because the artificial noise added by the Laplace mechanism is indiscriminate, leading to conspicuous damage to the model's usability. Even so, our solution has significant advantages on MovieLens and MNIST datasets. In contrast, the primitive Laplace mechanism cannot protect image classification models, while DPSGD cannot defeat the attack. On the BookCrossing dataset, the primitive Laplace mechanism seems to have a better performance. We infer that the simplicity of the BookCrossing dataset and its corresponding model may lead to this situation. As the noise generated by the primitive Laplace mechanism is haphazard, the model may fail to learn this certain property that the property inference attack aims for. As a result, the property inference attack performs worse on the primitive Laplace mechanism while it maintains a good level of accuracy due to its simplicity. For the CIFAR100 model, While protecting the guest, it seems that at a low privacy budget, the primitive Laplace is better. However, we remind that the model usability is extremely low when applying the primitive Laplace mechanism. Under other situations, our method can effectively decrease the effect of a property inference attack, especially at a low privacy budget.

Table 6. Results of defending the guest against data reconstruction attack.

ϵ	MovieLens			BookCrossing			MNIST			CIFAR100		
	Laplace	DPSGD	**Ours**	Laplace	DPSGD	**Ours**	Laplace	DPSGD	**Ours**	Laplace	DPSGD	**Ours**
0.1	0.2459	0.2455	**0.3223**	0.3216	0.2907	**0.3329**	1.8849	1.8885	**2.0181**	**12.5145**	2.8622	3.6983
0.5	0.2453	0.2451	**0.3222**	0.3202	0.2902	**0.3329**	1.8024	1.8137	**1.9875**	**12.5262**	2.8537	3.6891
1.0	0.2453	0.2451	**0.3222**	0.3202	0.2902	**0.3221**	1.7857	1.7509	**1.9533**	3.6419	2.8351	**3.6624**
2.0	0.2452	0.2451	**0.3222**	0.3202	0.2902	**0.3221**	1.7336	1.7469	**1.9391**	2.9453	2.7998	**3.6383**
4.0	0.2452	0.2451	**0.3222**	0.3202	0.2902	**0.3221**	1.7014	1.7440	**1.9206**	2.9502	2.7743	**3.6365**

Table 7. Results of defending the host against data reconstruction attack.

ϵ	MovieLens			BookCrossing			MNIST			CIFAR100		
	Laplace	DPSGD	**Ours**	Laplace	DPSGD	**Ours**	Laplace	DPSGD	**Ours**	Laplace	DPSGD	**Ours**
0.1	0.4032	0.2417	**0.5486**	0.4237	0.2758	**0.5066**	1.2887	1.0875	**1.8257**	**13.2849**	6.0999	6.5283
0.5	0.4024	0.2419	**0.5357**	0.4222	0.2756	**0.5149**	1.2778	1.0685	**1.7758**	**12.8057**	5.9302	6.3719
1.0	0.4008	0.2422	**0.5285**	0.4217	0.2743	**0.5235**	1.2602	1.0422	**1.7528**	**12.7936**	5.9283	6.3531
2.0	0.3982	0.2421	**0.5083**	0.4214	0.2697	**0.5150**	1.2613	1.0333	**1.7334**	6.0397	5.9256	**6.3453**
4.0	0.3960	0.2422	**0.4819**	0.4194	0.2683	**0.5046**	1.2549	0.9996	**1.7262**	6.0143	5.9247	**6.3396**

Defense against data reconstruction attack We take advantage of GANs and use the same reconstruction attack as [17]. We show the defense results

against an adversarial host and an adversarial guest in Table 6 and Table 7, respectively. We note that the MSE is measured after the attack model has been trained sufficiently in all cases. The MSEs measured for the attack against baselines of MovieLens, BookCrossing, MNIST and CIFAR100 models are 0.2412, 0.2629, 0.9612 and 2.6335 by an adversarial host, 0.2369, 0.2402, 1.6998 and 5.7534 by an adversarial guest, respectively. Please note that these attack results against the baselines are frustrating because the reconstruction attack is hard to succeed in the semi-honest setting. Meanwhile, data samples in the two recommendation datasets are similar and embedded with the same feature vectors. This leads to similar reconstruction results and similar MSEs because the reconstruction of structured data in MovieLens and BookCrossing largely depends on the embedding module. But we can still conclude from the results that our solution has a dominant performance in the defense against reconstruction attacks on either side. For the CIFAR100 model, when at a low privacy budget, the primitive Laplace mechanism sacrifices the prediction accuracy to reach a high difference. However, our method can maintain a satisfying prediction accuracy while protecting against the reconstruction attack.

Defense against feature space hijacking attack (FSHA) Please note that property inference and data reconstruction attacks implemented in FSHA [32] hijack the learning objective, offering the adversary an advantage over the previous attacks we have evaluated. In this setting, the malicious attacker trains a generator using SplitNN as a discriminator during the learning process. And a gradient-scaling trick is used to train the generator in FSHA. The sample generating process is essential to FSHA, meaning inference attacks depend on the reconstruction in FSHA. Thus, we will focus on the evaluation of defense against reconstruction attacks. If the generating part fails, the inference attack will be impossible. Since FSHA is comprehensively evaluated using the MNIST dataset, we give defense results of the MNIST model here. In Figure 2 and Figure 3, we give the reconstruction results of FHSA mounted by an adversarial host and an adversarial guest against the target samples used in [32], respectively. The second row of the two figures shows the results of FSHA against baselines. The following rows show that our solution can effectively preserve private data for both the guest and the host, even if the privacy budget is relaxed to 4. The results of our solution against FSHA using other datasets and the results of different budget values are in the appendix.

6 Conclusion

We investigate privacy leakage issues in SplitNN for two parties. By mounting property inference, data reconstruction, and hijacking attacks, we confirm that both the host and the guest in SplitNN are under severe threats of privacy leakage. To mitigate the leakage, we design a new activation function R^3eLU and its derivative in a randomized-response manner. By integrating R^3eLU into SplitNN as an interacting tunnel, we implement R^3eLU-forward and R^3eLU-backward

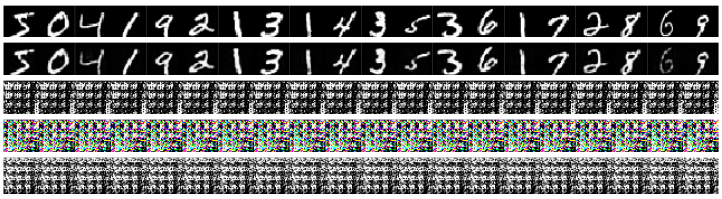

Fig. 2. Reconstruction results of FSHA against the guest's data in the first row. The following rows are attack results against the original SplitNN and our solution ($\epsilon = 0.1, 1.0, 4.0$), respectively.

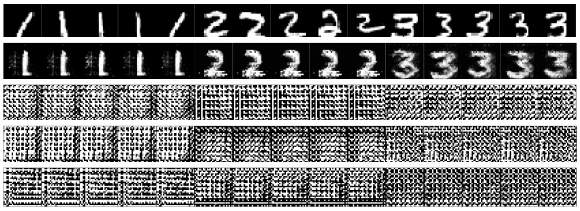

Fig. 3. Reconstruction results of FSHA against the host's data in the first row. The following rows are attack results against the original SplitNN and our solution ($\epsilon = 0.1, 1.0, 4.0$), respectively.

procedures. Through the privacy analysis, we confirm that SplitNN using R^{3}eLU-forward and R^{3}eLU-backward provides differential privacy for both two parties. Moreover, we propose a fine-grained privacy budget allocation scheme for assigning privacy budgets dynamically according to the parameters' importance. We finally conclude that our SplitNN solution outperforms the existing privacy-preserving solutions in model usability and privacy preservation through a comprehensive evaluation of different learning tasks. We also note that our solution concentrates on the leakage of private property and data samples. Other privacy issues, like label leakage in SplitNN, should be discussed separately. It is now unclear whether randomized-response solutions can deal with label leakage, which will be our work in the next.

Acknowledgement

The authors would like to thank our shepherd Prof. Stjepan Picek and the anonymous reviewers for the time and effort they have kindly put into this paper. Our work has been improved by following the suggestions they have made. This work was supported in part by the Leading-edge Technology Program of Jiangsu-NSF under Grant BK20222001 and the National Natural Science Foundation of China under Grants NSFC-62272222, NSFC-61902176, NSFC-62272215.

References

1. Abadi, M., Chu, A., Goodfellow, I., McMahan, H.B., Mironov, I., Talwar, K., Zhang, L.: Deep learning with differential privacy. In: ACM SIGSAC CCS (2016)
2. Ceballos, I., Sharma, V., Mugica, E., Singh, A., Roman, A., Vepakomma, P., Raskar, R.: Splitnn-driven vertical partitioning. preprint arXiv:2008.04137 (2020)
3. Du, J., Li, S., Chen, X., Chen, S., Hong, M.: Dynamic differential-privacy preserving sgd. preprint arXiv:2111.00173 (2021)
4. Dwork, C., Roth, A., et al.: The algorithmic foundations of differential privacy. Foundations and Trends® in Theoretical Computer Science (2014)
5. Erdogan, E., Kupcu, A., Cicek, A.E.: Unsplit: Data-oblivious model inversion, model stealing, and label inference attacks against split learning. In: Proceedings of the 21st Workshop on Privacy in the Electronic Society, WPES2022 (2022)
6. Erdogan, E., Teksen, U., Celiktenyildiz, M.S., Kupcu, A., Cicek, A.E.: Defense mechanisms against training-hijacking attacks in split learning. arXiv preprint arXiv:2302.0861 (2023)
7. Fang, M., Gong, N.Z., Liu, J.: Influence function based data poisoning attacks to top-n recommender systems. In: WWW 2020 (2020)
8. Fu, C., Zhang, X., Ji, S., Chen, J., Wu, J., Guo, S., Zhou, J., Liu, A.X., Wang, T.: Label inference attacks against vertical federated learning. In: USENIX Security 22 (2022)
9. Ganju, K., Wang, Q., Yang, W., Gunter, C.A., Borisov, N.: Property inference attacks on fully connected neural networks using permutation invariant representations. In: ACM SIGSAC CCS (2018)
10. Gao, H., Cai, L., Ji, S.: Adaptive convolutional relus. In: AAAI Conference on Artificial Intelligence (2020)
11. Gao, Y., Kim, M., Abuadbba, S., Kim, Y., Thapa, C., Kim, K., Camtep, S.A., Kim, H., Nepal, S.: End-to-end evaluation of federated learning and split learning for internet of things. In: SRDS (2020)
12. Gawron, G., Stubbings, P.: Feature space hijacking attacks against differentially private split learning. arXiv preprint arXiv:2201.04018 (2022)
13. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: NIPS (2014)
14. Gupta, O., Raskar, R.: Distributed learning of deep neural network over multiple agents. Journal of Network and Computer Applications (2018)
15. Harper, A.F.M., Konstan, J.A.: The movielens datasets: History and context. ACM Transactions on Interactive Intelligent Systems (2015)
16. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. CoRR (2015)
17. Hitaj, B., Ateniese, G., Perez-Cruz, F.: Deep models under the gan: information leakage from collaborative deep learning. In: ACM SIGSAC CCS (2017)
18. Huang, H., Mu, J., Gong, N.Z., Li, Q., Liu, B., Xu, M.: Data poisoning attacks to deep learning based recommender systems. In: NDSS (2021)
19. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
20. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE (1998)
21. Li, J., Rakin, A.S., Chen, X., He, Z., Fan, D., Chakrabarti, C.: Ressfl: A resistance transfer framework for defending model inversion attack in split federated learning. In: CVPR (2022)

22. Liu, R., Cao, Y., Chen, H., Guo, R., Yoshikawa, M.: Flame: Differentially private federated learning in the shuffle model. In: Proceedings of the AAAI Conference on Artificial Intelligence (2021)
23. Luo, X., Wu, Y., Xiao, X., Ooi, B.C.: Feature inference attack on model predictions in vertical federated learning. In: ICDE (2021)
24. Mao, Y., Yuan, X., Zhao, X., Zhong, S.: Romoa: Robust model aggregation for the resistance of federated learning to model poisoning attacks. In: ESORICS (2021)
25. Mao, Y., Zhu, B., Hong, W., Zhu, Z., Zhang, Y., Zhong, S.: Private deep neural network models publishing for machine learning as a service. In: IWQoS (2020)
26. McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Artificial intelligence and statistics (2017)
27. Melis, L., Song, C., De Cristofaro, E., Shmatikov, V.: Exploiting unintended feature leakage in collaborative learning. In: IEEE S&P (2019)
28. Molchanov, P., Mallya, A., Tyree, S., Frosio, I., Kautz, J.: Importance estimation for neural network pruning. In: CVPR (2019)
29. Nasr, M., Shokri, R., Houmansadr, A.: Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In: IEEE S&P (2019)
30. Nguyen, T.D., Rieger, P., Yalame, H., Möllering, H., Fereidooni, H., Marchal, S., Miettinen, M., Mirhoseini, A., Sadeghi, A.R., Schneider, T., et al.: Flguard: Secure and private federated learning. arXiv preprint arXiv:2101.02281 (2021)
31. OpenMined: Syft (2021), https://github.com/OpenMined/PySyft
32. Pasquini, D., Ateniese, G., Bernaschi, M.: Unleashing the tiger: Inference attacks on split learning. ACM SIGSAC CCS (2021)
33. Pereteanu, G.L., Alansary, A., Passerat-Palmbach, J.: Split he: Fast secure inference combining split learning and homomorphic encryption. arXiv preprint arXiv:2202.13351 (2022)
34. Salem, A., Bhattacharya, A., Backes, M., Fritz, M., Zhang, Y.: Updates-leak: Data set inference and reconstruction attacks in online learning. In: USENIX Security Symposium (2020)
35. Salem, A., Zhang, Y., Humbert, M., Fritz, M., Backes, M.: Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models. In: NDSS (2019)
36. Sun, L., Qian, J., Chen, X.: LDP-FL: practical private aggregation in federated learning with local differential privacy. In: IJCAI (2021)
37. Tolpegin, V., Truex, S., Gursoy, M.E., Liu, L.: Data poisoning attacks against federated learning systems. In: ESORICS (2020)
38. Warner, S.L.: Randomized response: A survey technique for eliminating evasive answer bias. Journal of the American Statistical Association (1965)
39. Webank: Fate (2021), https://github.com/FederatedAI/FATE
40. Yu, L., Liu, L., Pu, C., Gursoy, M.E., Truex, S.: Differentially private model publishing for deep learning. In: IEEE S&P (2019)
41. Zhang, C., Li, S., Xia, J., Wang, W., Yan, F., Liu, Y.: {BatchCrypt}: Efficient homomorphic encryption for {Cross-Silo} federated learning. In: 2020 USENIX annual technical conference (USENIX ATC 20) (2020)
42. Zheng, Y., Lai, S., Liu, Y., Yuan, X., Yi, X., Wang, C.: Aggregation service for federated learning: An efficient, secure, and more resilient realization. IEEE Transactions on Dependable and Secure Computing (2022)
43. Ziegler, C.N., McNee, S.M., Konstan, J.A., Lausen, G.: Improving recommendation lists through topic diversification. In: WWW (2005)

Appendix

.1 Model Architecture

The neural networks we used for MovieLens, BookCrossing, MNIST and CIFAR100 datasets after a split are shown in Table 8. These networks are widely used in related studies. We apply ResNet18 [16] for CIFAR100. We split them according to the interpretation of SplitNN in previous studies [2, 32].

Table 8. Model architectures used for evaluation.

MovieLens Model			BookCrossing Model		
Guest Layers	Dim.	Param.	Guest Layers	Dim.	Param. #
Linear(160,128)+ReLU	128	20608	Linear(160,128)+ReLU	128	20608
Host Layers	Dim.	Param.	Host Layers	Dim.	Param.
Linear(160,128)+ReLU	128	20608	Linear(160,128)+ReLU	128	20608
Merge Guest Output			Merge Guest Output		
Linear(128,128)+ReLU	128	16512	Linear(128,256)+ReLU	256	33024
Linear(128,64)+ReLU	64	8256	Linear(256,128)+ReLU	128	32894
Linear(64,3952)+Softmax	3952	256880	Linear(128,17384)+Softmax	17384	2242536

MNIST Model		
Guest Layers	Dim.	Param.
Linear(28*14,128)+BatchNormalization+ReLU	128	50304
Linear(128,64)	64	8256
Host Layers	Dim.	Param.
Linear(28*14,128)+BatchNormalization+ReLU	128	50304
Linear(128,64)+BatchNormalization+ReLU	64	8256
Merge Guest Output		
Linear(64,64)+BatchNormalization+ReLU	64	4160
Linear(64,10)+Softmax	10	650

Cifar100 Model		
Guest Layers	Dim.	Param.
conv1_x = Conv2D(3, 64, kernel=3, padding=1)	[1, 64, 32, 32]	1728
+ BatchNormalization+ReLU	[1, 64, 32, 32]	128
conv2_x = BasicBlock(64, 64, stride=1)	[1, 64, 32, 32]	73984
+ BasicBlock(64, 64, stride=1)	[1, 64, 32, 32]	73984
Host Layers	Dim.	Param.
conv1_x = Conv2D(3, 64, kernel=3, padding=1)	[1, 64, 32, 32]	1728
+ BatchNormalization+ReLU	[1, 64, 32, 32]	128
conv2_x = BasicBlock(64, 64, stride=1)	[1, 64, 32, 32]	73984
+ BasicBlock(64, 64, stride=1)	[1, 64, 32, 32]	73984
Merge Guest Output		
conv3_x = BasicBlock(64, 128, stride=2)	[1, 128, 16, 16]	230,144
+ BasicBlock(128, 128, stride=2)	[1, 128, 16, 16]	295,424
conv4_x = BasicBlock(128, 256, stride=2)	[1, 256, 8, 8]	919,040
+ BasicBlock(256, 256, stride=2)	[1, 256, 8, 8]	1,180,672
conv5_x = BasicBlock(256, 512, stride=2)	[1, 512, 4, 4]	3,673,088
+ BasicBlock(512, 512, stride=2)	[1, 512, 4, 4]	4,720,640
AdaptiveAvgPool2d(1, 1)+Linear(512, 1000)	[1, 100]	51300

.2 Supplement Results of Evaluation

To further investigate how our solution affects the learning process of SplitNN, we report learning results of a MovieLens recommendation model protecting the privacy of the guest and the host in Figure 4 and 5, respectively. In each plot, we show trends of training accuracy and testing accuracy as the training epoch increases. When $\epsilon = 0.1$ for the guest or the host, model usability will

be influenced seriously. Things get better when the privacy budget increases to 1 for either the guest or the host. We can conclude from the figures that our solution achieves satisfying model usability even with a small privacy budget for either side of SplitNN.

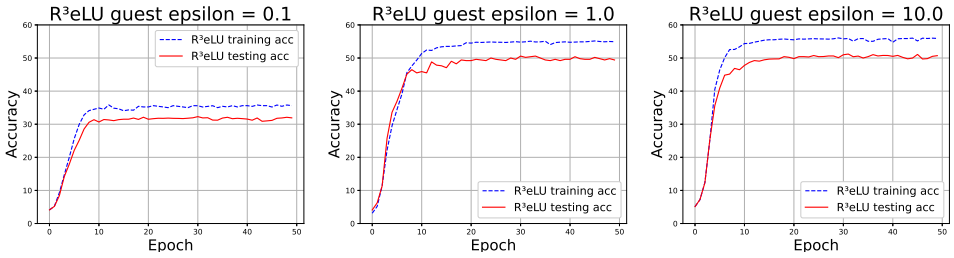

Fig. 4. SplitNN learning curve with the guest's privacy protected by our solution.

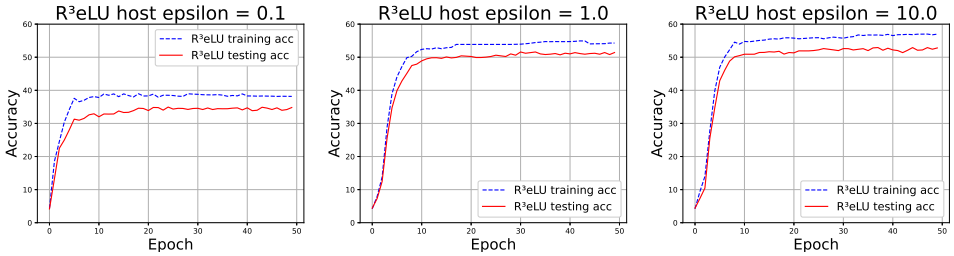

Fig. 5. SplitNN learning curve with the host's privacy protected by our solution.

In Table 9, we give a benchmark of SplitNN using different cut layers with two public datasets, MovieLens[15] and BookCrossing [43]. We also give the top-10 hit ratio for the test in Table 9. We use min as our merging strategy. We combine one linear layer with one ReLU as one cut layer. We notice that there is little difference between different cut layers. However, considering the computational cost at the guest part, it is a good tradeoff between computational cost and model usability to select the first layer as the cut layer.

Table 9. Top-10 hit ratio (%) of SplitNN using different cutlayers.

		layer1	layer2	layer3	layer4	no split
MovieLens	padding	**57.19**	56.97	56.72	57.07	57.21
	non-padding	55.08	55.76	**56.94**	56.75	
Book Crossing	padding	60.98	60.98	61.24	**61.12**	61.92
	non-padding	59.02	58.93	**60.16**	59.83	