ECAI 2023
K. Gal et al. (Eds.)

doi:10.3233/FAIA230330

On the Robustness of Split Learning Against Adversarial Attacks

Mingyuan Fan[a], **Cen Chen**[a;*], **Chengyu Wang**[b], **Wenmeng Zhou**[b] **and Jun Huang**[b]

[a]East China Normal University, China
[b]Alibaba Group, China

Abstract. Split learning enables collaborative deep learning model training while preserving data privacy and model security by avoiding direct sharing of raw data and model details (i.e., server and clients only hold partial sub-networks and exchange intermediate computations). However, existing research has mainly focused on examining its reliability for privacy protection, with little investigation into model security. Specifically, by exploring full models, attackers can launch adversarial attacks, and split learning can mitigate this severe threat by only disclosing part of models to untrusted servers. This paper aims to evaluate the robustness of split learning against adversarial attacks, particularly in the *most challenging setting where untrusted servers only have access to the intermediate layers of the model*. Existing adversarial attacks mostly focus on the centralized setting instead of the collaborative setting, thus, to better evaluate the robustness of split learning, we develop a tailored attack called *SLADV*, which comprises two stages: 1) shadow model training that addresses the issue of lacking part of the model and 2) local adversarial attack that produces adversarial examples to evaluate. The first stage only requires *a few unlabeled non-IID data*, and, in the second stage, *SLADV* perturbs the intermediate output of natural samples to craft the adversarial ones. The overall cost of the proposed attack process is relatively low, yet the empirical attack effectiveness is significantly high, demonstrating the surprising vulnerability of split learning to adversarial attacks.

1 Introduction

Split Learning (SL) emerges as a promising distributed learning paradigm for addressing the privacy issue in conventional centralization training [5]. This learning paradigm enables collaborative model training by splitting the whole network into sub-networks that are computed by different participants (i.e., clients and a server). It alleviates the privacy issue by keeping the sensitive raw data locally at the client side and only exchanging intermediate computations with the server. In the training process, the server bears most of the involved computational costs, making split learning lightweight and scalable for clients with limited computing resources.

Figure 1(a) presents the *vanilla split learning* [19], where a full model is split into two sub-networks: input layers and server layers. Clients train the input layers up to a specific cut layer, and the resulting outputs are sent to the server. The server then completes the rest of the forward process with the server layers without accessing

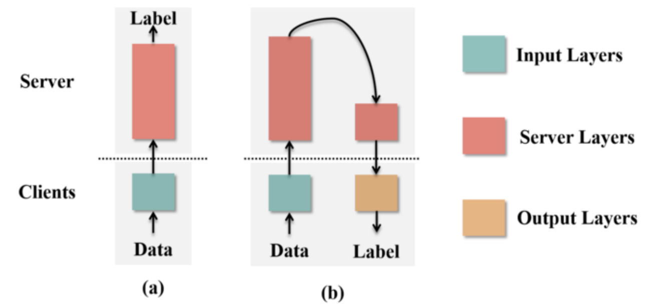

Figure 1: Sketch map of split learning: (a) vanilla split learning and (b) split learning with the U-shaped configuration.

clients' private raw data. The back-propagation process is also performed in a similar fashion. However, vanilla split learning requires clients to expose labels to the server, which may compromise label privacy. To mitigate this concern, an improved version of vanilla SL is presented in Figure 1(b), i.e., *split learning with the U-shaped configuration* [19], which further divides and transfers a few layers at the end of the full model (i.e., output layers) back to clients, eliminating the need for label sharing.

One of the most heated research topics of split learning is its trustworthiness and a sizable body of work focuses on examining whether split learning can truly protect privacy [14, 3]. However, there is a lack of research on discussing model security in split learning, which also is an important aspect of trustworthiness. In detail, attackers can manipulate models by employing *adversarial attacks* [15], which generates adversarial examples by imposing human-imperceptible adversarial noises into natural samples. Fortunately, launching adversarial attacks typically requires full access to target models, making them less dangerous in the conventional centralized learning paradigm. In centralized learning, the trainer centralizes data across different places into one data center. Untrusted individuals rarely are involved in the training process, thereby the models are not likely to be disclosed. In contrast, for split learning, the server may be untrusted while part of the model is delivered to the server [14, 3]. As a result, a key question naturally arises: can split learning maintain model security? To address this, to our best knowledge, this paper conducts the first systematic empirical study on the vulnerability of split learning against adversarial attacks.

In this paper, we focus on examining the vulnerability of split learning with the U-shaped configuration for three reasons. First, this type of split learning is more favorable since it allows protection for

[*] Corresponding Author. Email: cenchen@dase.ecnu.edu.cn.

both data and labels. Second, launching attacks in this setting is more challenging because the available resources for the server are greatly limited compared to vanilla split learning. Third, the attacks developed for this type of split learning are more generic and can be seamlessly applied to vanilla split learning. Thus, for simplicity, in the remainder of this paper, we use split learning to denote split learning with the U-shaped configuration (Figure 1(b)).

Existing adversarial attacks are mainly developed in centralized settings, which cannot fully exploit the characteristics of split learning [8]. To address this issue, we first present a practical attack scenario that aligns well with split learning and then develop a novel yet effective adversarial attack called *SLADV*. Adversarial attacks typically adopt input gradients as maliciously-designed noises [23]. Common adversarial attacks [8] assume either full access to the target model (i.e., white-box adversarial attacks) or pre-training a similar proxy model as a surrogate of the target model (i.e., transfer-based adversarial attacks). Our proposed *SLADV* follows the latter type of attack, as it is more practical than the former type.

Our proposed *SLADV* consists of two stages: shadow model training and local adversarial attack. In shadow model training, by leveraging the characteristics of split learning, *SLADV* only needs to train shadow input layers instead of a complete model. The training of shadow input layers requires only *a few hundred to a few thousand samples*, which do not necessarily have to be identical to the distribution of the target model's training data. In contrast, the training of proxy models in common transfer-based attacks needs to collect a large amount of data that is identically distributed with the training set of the target model. Additionally, the training in *SLADV* does not necessitate labeled data, eliminating the need for labeling data in common transfer-based attacks, which can be labor-intensive. In local adversarial attack stage, the concatenation of shadow input layers and server layers serves as the proxy model to produce adversarial examples. Extensive experiments show that *SLADV* achieves competitive attack performance at a significantly lower cost compared to common transfer-based attacks. In all, we believe this study sheds light on the vulnerability of split learning to adversarial attacks, alerts clients to potential model security issues, and may inspire further defenses in this area. Our contributions are four-fold:

- To our best knowledge, we are the first to identify the potential threat in split learning, i.e., the vulnerability of split learning against adversarial attacks, and, have conducted a seminal investigation into this threat.
- By delving into the bottom of split learning, we have designed a highly practical attack scenario with limited attack resources. Within this scenario, we developed a novel yet effective attack method named *SLADV* that fully exploits the characteristics of split learning.
- We have performed an in-depth theoretical analysis of the proposed *SLADV* attack and demonstrated its theoretical effectiveness under mild conditions.
- Extensive experiments have been conducted to examine the robustness of split learning against adversarial attacks, revealing the severe vulnerability of split learning and the effectiveness of our proposed *SLADV*.

2 Related Work

2.1 Model Security and Adversarial Attacks

Although DNNs achieve impressive performance over many complicated tasks, the internal operating mechanism of DNNs is still opaque to humans and this feature can be maliciously explored to launch adversarial attacks against DNNs [11]. Adversarial attacks impose little noises along model vulnerable directions into natural samples to craft adversarial examples. A lot of studies empirically show the remarkable vulnerability of DNNs against adversarial examples [8, 2]. Generally, if DNNs are allowed to be fully accessible, known as the white-box scenario, the input gradient directions are considered as the model vulnerable direction [6, 10], i.e., taking gradients of loss function w.r.t. natural samples as adversarial noises. In the conventional training paradigm, it is not likely for trained models to be disclosed to untrusted ones. Therefore, launching attacks in the black-box scenario is more appealing for real-world applications. Black-box attacks largely depend on the transferability of adversarial examples, i.e., adversarial examples locally crafted on a proxy model sometimes can also fool another unknown model. Furthermore, transfer-based black-box attacks particularly follow the following two stages [8, 12]: 1) collecting a sufficient amount of data together with their labels and train a proxy model from scratch, 2) locally crafting adversarial examples. For the first stage, obtaining a similar training dataset of the target model is difficult in the real world. Although [20] proposed data-free proxy model training methods, the methods are limited due to the prohibitive cost of extensively querying the target model. Our work proposes an effective attack method with limited attack resources.

2.2 Data Privacy and Split Learning

Data sharing is one of the primary challenges in collaboratively training DNNs [18]. Split learning allows the training of an effective DNN without sharing any raw data [5, 22]. In split learning, a full model is split into multiple parts, each of which is trained by a different participant. Split learning caters to practical needs for many scenarios such as IoT settings [5] and thus is ubiquitous. Split learning, however, typically involves an untrusted server [14, 3]. As shown in Figure 1(b), the full model $F_\theta(\cdot)$ is split into three parts, i.e., input layers $F_{\theta_1}(\cdot)$, server layers $F_{\theta_2}(\cdot)$, and output layers $F_{\theta_3}(\cdot)$. Wherein, input layers and output layers remain in clients, and server layers are located in the server. In each iteration, clients send the output of input layers into server layers. Next, the output of server layers is passed to output layers. Clients locally compute loss and generate the gradients from output layers that propagate back to server layers and input layers. Several recent studies [14, 3] show that the server is capable of stealing both data and labels. Yet, to our knowledge, no related work focuses on the model security in split learning, which is our research goal to fill the gap.

3 Threat Model

Before developing our attack, we present the threat model used in this paper, including attacker's goal, attacker's knowledge, and attacker's capability, which defines the available resource and constraints for attacks to be performed.

3.1 Attack Model

In split learning, multiple clients collaboratively train a model with the assistance of a server. Following previous works [14], we assume *the server to be a malicious attacker who may intentionally exploit the training process to pursue self-interest.* To be specific, in split learning, after training process finishes, the server delivers server layers to clients and clients concatenate input layers, server layers, and output layers into a complete model [1]. The complete model is then deployed into real-world applications such as medical diagnosis systems [1]. If the attacker can launch adversarial attacks to manipulate medical diagnosis systems, the attacker can fraudulently obtain government subsidies, insurance compensation, etc.

As white-box attacks necessitate full access to the target model and the server lacks access to input layers and output layers, the server can only resort to transfer-based adversarial attacks against the target model. Our aim is to investigate the feasibility of a successful transfer-based adversarial attack in such scenario even for a highly constrained attacker.

3.2 Attacker's Goal

The overall goal of the attacker is to launch transfer-based adversarial attacks against the target model. Specifically, the attacker aims to deceive the target model into making false or less confident predictions for given samples, in order to achieve personal gains.

3.3 Attacker's Knowledge

For the sake of the practicality of our attack, the attacker is considered to possess fairly restricted knowledge. First, the attacker only possesses knowledge of the output generated by the input layers, as its role in split learning is solely that of a transmitter. This entails that the server cannot obtain any information pertaining to the input or output layers, including their architecture and parameters. Second, in our SL setting, the attacker possesses only a vague understanding of the current task, rather than either complete or no knowledge. For instance, if the current task involves distinguishing between images of cats and dogs, the server may be able to deduce that it is an image-based task by analyzing the architecture of the server layers. However, the server is unlikely to be able to determine specific details about the task, such as the precise label space (e.g., cats and dogs), as the raw labeled data is not accessible to the attacker.

Note that, in common transfer-based adversarial attacks, attackers are typically assumed to possess prior knowledge of the label space of the target model in advance, enabling them to collect corresponding data associated with the label space to train a proxy model with high similarity. Conversely, our attacker only possesses knowledge of the type of current task, i.e., an image-based task, but lacks access to the label space. This constraint makes it unclear which images should be collected, significantly increasing the difficulty of launching attacks. Nevertheless, the restricted attacker's knowledge serves a highly practical purpose in our split learning context.

3.4 Attacker's Capability

The attacker's capabilities in this stage adhere to the principle of avoiding behaviors that violate split learning's training rules, as such behaviors would be easily detected by clients. As a result, we permit the following attacker's capabilities: 1) arbitrary modifications of the input and output of server layers, which is undetectable by clients, 2) data collection for training a proxy model with potentially dissimilar data distribution to the target model's training data[1], and 3) utilization of split learning's characteristics, such as input layer outputs, to train a proxy model during the training process without alerting clients. Notably, in existing transfer-based attacks, the proxy model's training is often delayed until the target model's training is complete. In contrast, in our case, the proxy model is trained together with the target model to exploit the characteristics of split learning.

[1] This aligns with real-world scenarios, where the attacker is assumed to have fairly limited knowledge.

Algorithm 1 Shadow Model Training

```
Input: $F_{\theta_1}$: input layers; $F_{\theta_2}$: server layers; $F_{\theta_3}$: output layers; $F_{\theta_1'}$: shadow input layers; $D_1$: dataset owned by clients; $D_2$: dataset owned by the attacker (server); $T$: the total number of iterations; $\alpha$: the hyperparameters to adjust the similarity magnitude; $L(\cdot,\cdot)$: a loss function.
Output: $F_{\theta_1}, F_{\theta_2}, F_{\theta_3}, F_{\theta_1'}$: the trained models.
1: for each iteration $i = 0$ to $T$ do
2:     Clients sample $x, y$ from $D_1$, compute $o_1 = F_{\theta_1}(x)$, and send $o_1$ to the server.
3:     The server computes $o_2 = F_{\theta_2}(o_1)$ and delivers $o_2$ to clients.
4:     The server sample $x'$ from $D_2$ and computes similarity loss $L_{sim} = ||F_{\theta_1'}(x') - o_1||_2$.
5:     Clients compute loss $L(F_{\theta_3}(o_2), y)$ and implement backpropagation algorithm to send gradient $\frac{\partial L(F_{\theta_3}(o_2),y)}{\partial o_2}$ to the server.
6:     The server computes $g_1 = \frac{\partial L(F_{\theta_3}(o_2),y)}{\partial o_2}\frac{\partial o_2}{\partial o_1}$ and $g_2 = \frac{\partial L_{sim}}{\partial o_1}$.
7:     The server fuses gradients $g = g_1 + \alpha g_2$ and returns $g$ to the clients.
8:     Clients and the server update model parameters based on gradients.
9: end for
```

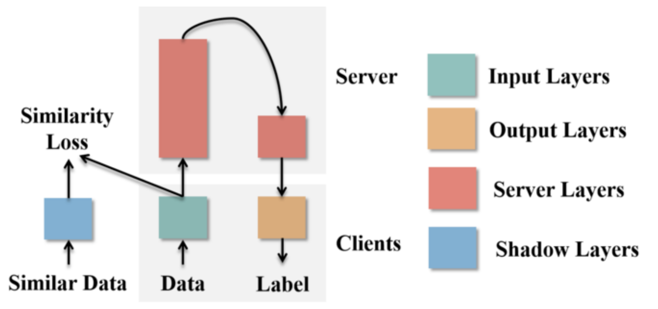

Figure 2: The sketch map of the shadow model training stage. We use shadow layers to substitute input layers. The resultant combination of shadow input layers and server layers serves as the proxy model. We leverage similarity loss to promote the similarity between input and shadow input layers for better attack effectiveness.

4 Approach

We elaborate on shadow model training and local adversarial attack of *SLADV* in Sections 4.1 and 4.2, respectively. Algorithms 1 and 2 outline the two stages. Figure 2 illustrates the overall idea of shadow model training stage. Section 5 gives theoretical analysis on *SLADV*. Finally, Section 4.3 compares the difference between *SLADV* and common-used transfer-based attacks.

4.1 Shadow Model Training

SLADV trains a proxy model to launch attacks. In split learning, the server holds server layers, which can be maliciously exploited by attackers to reduce the cost of launching attacks. In response, a straightforward idea is to train shadow input layers and shadow output layers (compatible with server layers) and then concatenate them with server layers to form a proxy model. However, attackers are unable to query ground-truth labels for the collected data, making it impossible to train the output layers without supervised signals. Furthermore, even if the labels of the collected data are available, the label space of the collected data may differ from the label space of the target model. This gap implies a low similarity between the proxy model and the target model, resulting in low attack success rates. To overcome this issue, we attempt to discard the shadow output layers. As a result, crafting adversarial examples can only rely on the shadow input layers and server layers. However, with the absence of an output layer, *SLADV* cannot generate adversarial examples by directly reducing the predicted probability of the corresponding label obtained from the proxy model. We defer the solution to this problem to Section 4.2.

In split learning, clients send outputs of input layers to the server. Intuitively, a simple idea is to adopt the outputs of input layers as supervised signals for shadow input layers. In each iteration, an extra difference loss item, L_2-norm, is introduced, which measures the difference between the outputs of input layers and shadow input layers associated with their respective data. In this way, shadow input layers can be trained. It is stressed that this way does not alert clients. Because the extra difference loss item is computed by the server, and, in each backpropagation process, the server just fuses the gradients of the original loss and gradients of the extra difference loss. The original loss denotes the loss used to train models over the task, e.g., cross-entropy loss. The fused gradients are returned to clients. Clients cannot detect gradient tampering by only observing the returned gradients. Formally, by denoting clients' and the server's data as D_1 and D_2, a forward process can be formally expressed as follows.

Clients transmit locally-computed output o_1 of input layers F_{θ_1} to the server: $o_1 = F_{\theta_1}(x), x \sim D_1$. With o_1, the server returns the output o_2 of server layers F_{θ_2} associated with o_1 and compute the extra difference loss L_{sim} to train shadow input layers:

$$o_2 = F_{\theta_2}(o_1),$$
$$L_{sim} = ||F_{\theta_1'}(x') - o_1||_2,\ x' \sim D_2.$$

Afterwards, clients complete the rest of forward process, i.e., computing the final outputs of the model and the original loss:

$$loss = L(F_{\theta_3}(o_2), y).$$

The back-propagation process is sequentially implemented in the following steps:

- Clients update θ_3 by $\frac{\partial loss}{\theta_3}$ and send $\frac{\partial loss}{o_3}$ back to the server;
- The server updates θ_2 and θ_1' by $\frac{\partial loss}{\partial o_2}\frac{\partial o2}{\theta_2}$ and $\frac{L_{sim}}{\theta_1'}$ respectively, and sends $\frac{\partial loss}{\partial o_2}\frac{\partial o2}{\partial o_1} + \alpha\frac{\partial L_{sim}}{o_1}$ back to the clients, where α is similarity constraints;
- Clients update θ_1 by $(\frac{\partial loss}{\partial o_2}\frac{\partial o2}{\partial o_1} + \alpha\frac{\partial L_{sim}}{o_1})\frac{\partial o_1}{\theta_1}$.

Besides exploiting the difference between the outputs of input layers and shadow input layers as supervised signals, an alternative is to utilize the difference of outputs of server layers with respect to clients' data and the attacker's data. However, the latter usually performs worse than the former. In particular, it is well-known that the extracted features of the model transmit from general to specific along with the model depth [7, 16]. In light of the observation, due to the non-negligible data distribution divergence between clients and the server, forcefully aligning outputs of server layers probably causes collapse in model performance. In contrast, it makes more sense to align the output of input layers since input layers typically extract features shared across different kinds of data. We also validate this point in experiments (Section 6.5).

Algorithm 2 Local Adversarial Attack

```
Input: $F_{\theta_1'}$: shadow input layers; $F_{\theta_2}$: server layers. $x$: samples; $K$:
    the total number of iterations; $\epsilon$: perturbation budget.
Output: $x + \delta$: the crafted adversarial examples.
1: Initialize the adversarial noises $\delta$ for $x$.
2: for each iteration $i = 1$ to $K$ do
3:     Compute $o_2 = F_{\theta_2}(F_{\theta_1'}(x))$ and $o_2' = F_{\theta_2}(F_{\theta_1'}(x + \delta))$.
4:     Compute loss $L_{attack}(o_2, o_2') = \frac{o_2 \cdot o_2'}{||o_2||_2||o_2'||_2}$.
5:     Optimize $\delta$ based on $L(o_2, o_2')$.
6:     Clip $\delta$ based on perturbation budget $\epsilon$.
7: end for
```

4.2 Local Adversarial Attack

After shadow model training, the trained shadow input layers serve as a substitute for input layers. We combine shadow input layers and server layers as the proxy model. The problem here is that the proxy model lacks output layers, indicating that common techniques for crafting adversarial examples in transfer-based attacks cannot be directly implanted into our case as these methods require prediction probability. Therefore, instead of decreasing the prediction probability, *SLADV* crafts adversarial examples by perturbing the intermediate outputs. Intuitively, since output layers make predictions based on the outputs of the server layers, it is possible to trick the target model by changing the outputs of the server layers. Furthermore, if shadow input layers and input layers are similar enough, such noises that induce considerable shifts in outputs of the combination of the shadow input layers and server layers also are likely to produce consistent influences on outputs of the combination of the input layers and server layers. Section 5 formally demonstrates the effectiveness of the above idea under some mild regularization conditions.

In practice, *SLADV* implements this idea by optimizing noises that reduce the cosine similarity between the intermediate outputs of perturbed and unperturbed samples. Mathematically, by initializing δ as zero vectors, *SLADV* iteratively optimizes δ for K times based on following steps:

- Compute the output of server layers with respect to original and corresponding adversarial samples: $o_2 = F_{\theta_2}(F_{\theta_1'}(x)), o_2' = F_{\theta_2}(F_{\theta_1'}(x + \delta))$.
- Compute loss function that measures the similarity between o_2 and o_2': $L_{attack}(o_2, o_2') = \frac{o_2 \cdot o_2'}{||o_2||_2||o_2'||_2}$.
- Update δ by $\delta = Clip_\epsilon\{\delta - \beta\frac{\partial L_{attack}(o_2,o_2')}{\partial \delta}\}$, where β is update step size, ϵ is given perturbation budget, and $Clip_\epsilon\{\cdot\}$ draws input into ϵ-ball (usually ∞-norm) if input beyond perturbation budget.

4.3 Detailed Comparison between SLADV and Commonly-used Transfer-based Attacks

Unlike commonly-used transfer-based attacks that focus on centralized training settings [8, 2], *SLADV* instead pays more attention to decentralized settings, i.e., split learning, which is more suitable in resource-restricted settings [5].

In the proxy model training stage, aside from more loose needs for datasets, *SLADV* enjoys two extra advantages over commonly-used transfer-based attacks. i) Attackers are assumed to own less information about the task performed by the target model, which is more practical as shown in Section 3.3. ii) *SLADV* is an immediate attack since the proxy and target models are trained together, suggesting that *SLADV* can launch attacks whenever the target model is deployed.

In the way of crafting adversarial examples, *SLADV* promotes the output of server layers w.r.t. crafted adversarial examples to have a large shift, rather than decreasing logits. Moreover, experimental results (Section 6) show that such method still can achieve competitive attack performance.

5 Theoretical Analysis

Section 5.1 suggests that local adversarial attack is effective when the input layers and shadow layers are similar. Additionally, Section 5.2 demonstrates that adding an extra loss term (i.e., L_2-norm) can increase the similarity between the input layers and shadow layers.

5.1 The Effectiveness of Shadow Model Training

In the following analysis, we set $||\delta|| \leq \epsilon$ where ϵ is a small positive constant to make Taylor expansion established. By adding δ into x, for the combination of shadow input layers and server layers, the output can be approximately estimated as follows:

$$\begin{aligned}F_{\theta_1'}(x+\delta) &= F_{\theta_1'}(x) + \nabla_x F_{\theta_1'}(x)^T \cdot \delta = o_1' + \nabla_x F_{\theta_1'}(x)^T \cdot \delta,\\ F_{\theta_2}(F_{\theta_1'}(x+\delta)) &= F_{\theta_2}(o_1') + \nabla_{o_1'} F_{\theta_2}(o_1')^T \cdot F_{\theta_1'}(x)^T \cdot \delta,\end{aligned} \tag{1}$$

where o_1' denotes the output of the shadow input layers. Similarly, for the combination of input layers and server layers, there is:

$$F_{\theta_2}(F_{\theta_1}(x+\delta)) = F_{\theta_2}(o_1) + \nabla_{o_1} F_{\theta_2}(o_1)^T \cdot F_{\theta_1}(x)^T \cdot \delta. \tag{2}$$

SLADV minimizes the cosine distance to craft δ as follows:

$$\begin{aligned}\delta &= \arg\min_{\delta} F_{\theta_2}(F_{\theta_1'}(x)) \cdot F_{\theta_2}(F_{\theta_1'}(x+\delta))\\ &= \arg\min_{\delta} F_{\theta_2}(o_1') \cdot (F_{\theta_2}(o_1') + \nabla_{o_1'} F_{\theta_2}(o_1')^T \cdot F_{\theta_1'}(x)^T \cdot \delta),\\ s.t., &\quad ||\delta|| \leq \epsilon, ||F_{\theta_2}(F_{\theta_1'}(x))|| = ||F_{\theta_2}(F_{\theta_1'}(x+\delta))|| = 1.\end{aligned} \tag{3}$$

The best solution of δ for the above optimization task is:

$$\begin{aligned}\delta &= -C \cdot F_{\theta_1'}(x) \cdot \nabla_{o_1'} F_{\theta_2}(o_1'),\\ C &= \frac{\epsilon}{||F_{\theta_1'}(x) \cdot \nabla_{o_1'} F_{\theta_2}(o_1')||}.\end{aligned} \tag{4}$$

Then, by substituting the best solution of δ into Equation 2, it is observed that the influence of δ for input layers and server layers:

$$\begin{aligned}F_{\theta_2}(F_{\theta_1}(x+\delta)) &= F_{\theta_2}(o_1) + \nabla_{o_1} F_{\theta_2}(o_1)^T \cdot F_{\theta_1}(x)^T \cdot \delta\\ &= F_{\theta_2}(o_1) - C \cdot \nabla_{o_1} F_{\theta_2}(o_1)^T \cdot F_{\theta_1}(x)^T \cdot F_{\theta_1'}(x) \cdot \nabla_{o_1'} F_{\theta_2}(o_1').\end{aligned} \tag{5}$$

The finding here is that if the inner dot between $F_{\theta_1}(x) \cdot \nabla_{o_1} F_{\theta_2}(o_1)$ and $F_{\theta_1'}(x) \cdot \nabla_{o_1'} F_{\theta_2}(o_1')$ is positive, δ can produce similar effects. Specifically, when directions of the two items are totally same, δ induces identical effects. However, there is no evidence to show the positive of the inner dot. To solve this problem, L_{sim} is added to increase the similarity between shadow input layers and input layers and we here demonstrate that L_{sim} can promote direction consistency between the two gradient items.

We first demonstrate $\nabla_x F_{\theta_1}(x) = \nabla_x F_{\theta_1'}(x), \forall x$ if $F_{\theta_1}(x) = F_{\theta_1'}(x)$. According to Lagrange mean value theorem, there is:

$$\begin{aligned}(F_{\theta_1}(x_1) - F_{\theta_1}(x_2))(x-y)^{-1} &= \nabla_{\xi_1} F_{\theta_1}(\xi_1), x1 \leq \xi_1 \leq x2,\\ (F_{\theta_1'}(x_1) - F_{\theta_1'}(x_2))(x-y)^{-1} &= \nabla_{\xi_2} F_{\theta_1'}(\xi_2), x1 \leq \xi_2 \leq x2.\end{aligned} \tag{6}$$

Due to $F_{\theta_1}(x) = F_{\theta_1'}(x), \forall x$, we have:

$$\begin{aligned}&(\nabla_{x_1} F_{\theta_1}(x_1) - \nabla_{x_2} F_{\theta_1}(x_2))(x-y)^{-1}\\ &= (\nabla_{x_1} F_{\theta_1'}(x_1) - \nabla_{x_2} F_{\theta_1'}(x_2))(x-y)^{-1},\\ &\nabla_{\xi_1} F_{\theta_1}(\xi_1) = \nabla_{\xi_2} F_{\theta_1'}(\xi_2).\end{aligned} \tag{7}$$

Let $x_2 \to x_1$, there is:

$$x_1 = \xi_1 = \xi_2, \nabla_{x_1} F(\theta_1)(x_1) = \nabla_{x_1} F(\hat{\theta}_1)(x_1). \tag{8}$$

Furthermore, $o_1 = o_1'$ because of $F_{\theta_1}(x) = F_{\theta_1'}(x)$. Therefore, if shadow input layers and input layers output identically for all inputs, the directions of the two items aforementioned are same, which inspire us to add L_{sim}.

Now, we relieve the condition $F_{\theta_1}(x) = F_{\theta_1'}(x), \forall x$. Let the maximum distance between $F_{\theta_1}(\cdot)$ and $F_{\theta_1'}(\cdot)$ equals to d over its input space, i.e. $d = \max_x ||F_{\theta_1}(x) - F_{\theta_1'}(x)||, \forall x$. Then, there is:

$$\begin{aligned}&||\nabla F_{\theta_1}(x) - \nabla F_{\theta_1'}(x)||\\ &\approx ||(F_{\theta_1}(x+\delta) - F_{\theta_1}(x))\delta^{-1} - (F_{\theta_1'}(x+\delta) - F_{\theta_1'}(x))\delta^{-1}||\\ &= ||(F_{\theta_1}(x+\delta) - F_{\theta_1}(x) - F_{\theta_1'}(x+\delta) + F_{\theta_1'}(x))\delta^{-1}||\\ &\leq \frac{1}{||\delta||}(||F_{\theta_1}(x+\delta) - F_{\theta_1'}(x+\delta)|| + ||F_{\theta_1}(x) - F_{\theta_1'}(x)||)\\ &= \frac{2d}{||\delta||}.\end{aligned} \tag{9}$$

As such, decreasing the distance of outputs of shadow input layers and input layers for same inputs still can enhance their gradient similarity.

5.2 The Effectiveness of Local Adversarial Attack

The overall goal of this part is to study whether or not the samples perturbed by adding noises δ can fool the target model. The target model is a combination of input layers, server layers, and output layers, and we discuss the impact of noises crafted by shadow input layers and server layers for the outputs of input layers and server layers before. Therefore, the overall goal can be converted into the impact of the intermediate output changes $-C \cdot \nabla_{o_1} F_{\theta_2}(o_1)^T \cdot F_{\theta_1}(x)^T \cdot F_{\theta_1'}(x) \cdot \nabla_{o_1'} F_{\theta_2}(o_1')$ to output layers. For the simplicity of symbols, $-C \cdot \nabla_{o_1} F_{\theta_2}(o_1)^T \cdot F_{\theta_1}(x)^T \cdot F_{\theta_1'}(x) \cdot \nabla_{o_1'} F_{\theta_2}(o_1')$ is denoted by Δ. We analyze the influence of Δ to loss function $L(\cdot)$ (the larger the loss function, the worse the target model performance):

$$L(F_{\theta_3}(o_2+\Delta), y) = L(F_{\theta_3}(o_2), y) + \nabla_{o_2} L(F_{\theta_3}(o_2), y)^T \Delta. \tag{10}$$

To maximize the above loss function, the directions of Δ should be aligned with $\nabla_{o_2} L(F_{\theta_3}(o_2), y)$ as possible.

As shown in [4, 17], the final part of DNNs usually exhibits high linearity, especially for the last fully-connected layer. Based on this, we can make the following approximation:

$$L(F_{\theta_3}(0), y) = L(F_{\theta_3}(o_2), y) + \nabla_{o_2} L(F_{\theta_3}(o_2), y)(0 - o_2). \tag{11}$$

Table 1: Accuracy drop (%) of the target model trained over CIFAR-10 with 40,000 instances. The proxy model is trained over four different data distributions and varying numbers of data instances.

# Instances	SVHN	CIFAR-10	CIFAR-100	TinyImageNet
128	14.07	16.31	20.31	15.47
256	27.86	29.43	30.31	25.16
1024	29.19	31.42	30.99	27.38
2048	29.60	31.65	33.21	28.31
4096	27.84	29.86	35.22	34.77
8192	28.28	30.20	34.29	33.15

Table 2: Attack performance of three attacks against the target model over CIFAR-10 measured by accuracy drop (%).

# Instances	VR	SSA	*SLADV*
1024	19.15	20.88	**31.42**
2048	22.48	24.60	**31.65**
4096	23.71	25.43	**29.86**

Table 3: Impact of parameter α. With a fixed dataset size of 4096, we use different α to train proxy models in four different data distributions to implement *SLADV* against the target model.

α	SVHN	CIFAR-10	CIFAR-100	TinyImageNet
0	10.87	12.58	15.15	14.79
0.01	15.02	16.77	20.53	20.46
0.1	21.58	21.33	26.45	25.04
1	27.84	29.86	35.22	34.77
10	28.28	30.52	38.01	35.88
100	29.51	31.08	41.58	37.11

Notice that, output layers for full-zero inputs also output zeros and then the loss $L(F_{\theta_3}(0), y)$ is huge. Besides, the target model is generally trained well, implying $L(F_{\theta_3}(o_2), y)$ being small. Thus, there is $\nabla_{o_2} L(F_{\theta_3}(o_2), y)^T o_2 < 0$.

Furthermore, the best solution for δ is to minimize the cosine direction, thereby the inner product between Δ and $\hat{o}_2 = F_{\theta_2}(F_{\theta_1}(x))$ being negative. Moreover, o_2 and $\hat{o}_2$ probably are close with the aid of L_{sim} shown in Part 1 and we assume $o_2^T \cdot \hat{o}_2 > 0$. Therefore, there is $o_2^T \Delta < 0$ and $\nabla_{o_2} L(F_{\theta_3}(o_2), y)^T \Delta > 0$. In short, if shadow input layers and input layers are similar, the noises crafted by *SLADV* can decrease the performance of the target model.

6 Experimental Evaluation

6.1 Setup

According to existing works [8], three factors (i.e., the difference in architecture, training data distribution, and the numbers of training instances between the proxy and target models) are empirically demonstrated as being significant to attack effectiveness, while the leaving ones are usually trivial. We primarily explore the robustness of split learning by varying these important factors (see Section 6.2 and Section 6.5) while keeping the trivial ones fixed at commonly-used values throughout evaluations [14].

Collaborative settings. A typical setting is considered here, where 10 clients collaboratively train a shared model under the coordination of the server. Four state-of-the-art model architectures are selected for evaluation, i.e., MobileNet, DenseNet, ResNet18, and EfficientNet. The models are sequentially split into three parts to be allocated to clients and the server. Moreover, the ResNet18 architecture serves as the default architecture of the shared model and we discuss the impact of architecture in Section 6.5. ResNet18 composes of 17 convolutional layers tailed with a fully-connected layer and input layers, server layers, and output layers hold 2, 15, and 1 layers. Moreover, similar to [14], we suppose that clients solve a benchmark image classification task of CIFAR-10 to evaluate and the attacker's data is randomly extracted from SVHN, CIFAR-10, CIFAR-100, and TinyImageNet datasets. 80% of the training data in CIFAR-10 are evenly distributed to clients for training with a momentum optimizer with learning rate of 0.01 for 3000 iterations. In this stage, the server trains shadow input layers with similarity constraint α of 1 and an SGD optimizer with learning rate of 0.01. Since the server (Section 3.3) is assumed to have some knowledge about the AI domain and can infer the rough type of the current task, the default architecture of the shadow input model is set to two plain convolutional layers with skip connections.

Table 4: The impact of similarity constraints α of *SLADV* in terms of the accuracy of the target model.

α	SVHN	CIFAR-10	CIFAR-100	TinyImageNet
0	81.47	81.25	80.36	80.47
0.01	81.81	80.58	82.03	80.47
0.1	78.91	80.25	79.35	80.25
1	78.45	78.12	78.23	78.01
10	76.00	76.34	76.23	76.79
100	75.33	75.67	77.34	75.89

Attack settings. In the attack stage, the server exploits the concatenation of the trained shadow input model and server layers as the proxy model. We implement Algorithm 2 with perturbation budget of 0.3, step size of 0.3, and K of 1 (small iterations can make cheaper attacks) to craft adversarial examples against the target model.

Metric. We assess the effectiveness of *SLADV* using accuracy drop, which is defined as the difference between the accuracy of the model in the presence of natural samples and adversarial samples, over the test set of CIFAR-10. Higher accuracy drop indicates better attack performance.

6.2 The Impact of Data

The training data of the proxy model is of crucial importance for attack effectiveness. Generally, attack effectiveness grows with an increase in the number of available data and a decrease in the divergence between the data distributions of the target and proxy models. This is because attack effectiveness heavily depends on the similarity of learned features between proxy and target models [7]. A small volume of data makes it difficult to identify discriminative features while significant divergence in data distribution implies the underlying discriminative features behind these data are distinct. Table 1 reports the results by altering the data number and data distribution of the attacker. For SVHN, CIFAR-100, and TinyImageNet, we randomly extract data from the training dataset as the attacker's data. For CIFAR-10, we randomly sample the remaining training dataset (different from the data used for training the global model).

Data volume. The attack effectiveness considerably raises at the start of increasing data volume until it peaks at around 2000 to 4000, after which attack effectiveness tends to fluctuate moderately. Surprisingly, we find that even with a tiny amount of data, *SLADV* can still achieve a non-trivial performance. Taking CIFAR-100 as an example, *SLADV* produces a 20% accuracy drop in the target model by using 128 Non-IID data instances, highlighting the extreme vulnera-

Table 5: The attack effectiveness of *SLADV* with different sizes of the shadow input layers and input layers.

Layer Depth	SVHN			CIFAR-10			CIFAR-100			TinyImageNet		
	Single	Multi	Diff	Single	Multi	Diff	Single	Multi	Diff	Single	Multi	Diff
1	27.84	27.84	0.00	29.86	29.86	0.00	35.22	35.22	0.00	34.77	34.77	0.00
2	23.68	23.94	0.26	25.74	26.19	0.45	34.98	35.10	0.12	32.77	32.99	0.21
3	18.55	20.48	1.93	21.79	22.90	1.11	32.11	33.76	1.66	28.56	30.20	1.64
4	13.59	17.68	4.09	15.89	18.56	2.67	27.47	30.30	2.83	24.35	27.62	3.28

Table 6: The impact of the target model architectures. The proxy model is trained over four datasets with 4096 instances).

	ResNet	EfficientNet	DenseNet	MobileNet
SVHN	27.84	18.80	30.63	32.97
CIFAR-10	29.86	28.28	39.56	38.45
CIFAR-100	35.22	30.75	44.37	45.83
TinyImageNet	34.77	31.64	39.23	38.90
Avg.	31.92	27.37	38.45	39.04

bility of split learning against adversarial attacks.

Data distribution. Compared to data volume, the impact of data distribution indicates more striking observations. To begin with, intuitively, the best attack effectiveness should be achieved when the attacker's data is similar to the training data of the target model, i.e., CIFAR-10. However, Table 1 shows the attack effectiveness of employing different datasets as training data for the shadow input layers as follows: CIFAR100 $>$ CIFAR10 $>$ TinyImageNet $>>$ SVHN. This phenomenon can be explained by the fact that early layers primarily capture low-level features [7], which are shared among these datasets despite the divergence in their high-level features. Notably, the attack effectiveness using SVHN is slightly lower than that of the other datasets. We speculate this is because SVHN is a digital image dataset, and consequently, its low-level feature distribution differs from animal or plant images (CIFAR-10, CIFAR-100, and TinyImageNet). In summary, the quality requirement of the training data distribution for shadow input layers to launch adversarial attacks is lower than we thought.

6.3 Comparison to State-of-the-art Attacks

We compare the attack performance of *SLADV* with two state-of-the-art transfer-based attacks, namely VR [21] and SSA [13]. For VR and SSA, we train a proxy model of the same architecture of the target model in CIFAR-10 with various sizes of training sets (the same training dataset of *SLADV*) and then craft adversarial examples using the hyperparameters suggested in their original paper. Table 2 reports the attack effectiveness of the three attacks. Wherein, the attack performance of *SLADV* surpasses both baselines by a large margin, regardless of the dataset size. In addition, the proxy model training of VR and SSA uses labels. Therefore, *SLADV* is a more powerful and cost-effective attack than baselines.

6.4 The Impact of Similarity Constraints

Section 5 presents theoretical evidence that adding similarity constraints α for input and shadow input layers can improve attack effectiveness, which is further verified in this section. Table 3 reports the attack effectiveness of *SLADV* over varying magnitudes of similarity constraints. As shown in Table 3, increasing the magnitude of similarity constraints substantially strengthens the attack effectiveness, demonstrating consistency with our theoretical analysis.

However, we emphasize that increasing the magnitude of similarity constraints also has a negative side. Specifically, Table 4 shows the accuracy of the shared model over the test set at different values of similarity constraints. As can be seen, larger similarity constraints lead to greater loss in model performance.

6.5 The Impact of Model

The impact of input layer depth. By holding shadow input layers fixed while incrementally increasing input layer depth, we study the impact of the depth of the input layers on the attack effectiveness of *SLADV*. Table 5 reports the attack effectiveness for the shared model with input layers of different depths, where we use Single to denote the fixed shadow input layers. Overall, an increase in the depth of input layers leads to a decrease in the attack effectiveness of *SLADV* and this can be attributed to the model similarity. On the one hand, increasing the depth of input layers makes a bigger difference between input and shadow input layers. On the other hand, input layers with big depth tend to capture high-level features rather than low-level features [9]. To validate this point, we increase shadow input layers denoted in Multi in Table 5. Using shadow input layers with a deeper depth can partly offset the effect of increasing the depth of input layers but cannot completely offset.

The impact of target model architecture. We change the architecture of the target model and then implement *SLADV* to further examine the effectiveness of *SLADV*. Table 6 reports the attack performance of *SLADV* against the target model of different architectures. Simply speaking, *SLADV* can effectively attack four model architectures. Moreover, the results in Table 6 indicate that DenseNet and MobileNet exhibit a higher vulnerability to adversarial attacks when compared to ResNet and EfficientNet. This observation aligns with findings reported in baselines (VR and SSA). A conjecture is that the simpler and more lightweight design of MobileNet renders it less robust against attacks (simple architectures may be not able to learn enough discriminative features). On the other hand, DenseNet's distinctive characteristic of having more skip connections allows errors from earlier layers to propagate more easily to the last layer, potentially increasing its vulnerability to attacks.

7 Conclusion

We designed an attack method called *SLADV* to sufficiently harness the characteristics of split learning.In detail, *SLADV* comprises two stages: shadow model training and local adversarial attack. Shadow model training enables attackers to train shadow input layers inexpensively, compensating for the lack of input layers. Local adversarial attack employs shadow input layers and server layers to generate adversarial examples against the target model (i.e., the shared model). Our study also includes a thorough theoretical analysis of *SLADV*'s attack effectiveness. Extensive empirical experiments demonstrated the superior attack performance of *SLADV* and reveal the striking vulnerability of split learning to adversarial attacks.

Acknowledgements

This work was supported by the National Natural Science Foundation of China under grant number 62202170 and Alibaba Group through the Alibaba Innovation Research Program.

References

[1] Amir Aminifar, 'Minimal adversarial perturbations in mobile health applications: The epileptic brain activity case study', *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1205–1209, (2020).

[2] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li, 'Boosting adversarial attacks with momentum', in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 9185–9193, (2018).

[3] Ege Erdogan, Alptekin Kupcu, and A. Ercument Cicek, 'Unsplit: Data-oblivious model inversion, model stealing, and label inference attacks against split learning', *IACR Cryptol. ePrint Arch.*, **2021**, 1074, (2021).

[4] Mingyuan Fan, Yang Liu, Cen Chen, Ximeng Liu, and Wenzhong Guo, 'Defense against backdoor attacks via identifying and purifying bad neurons', *ArXiv*, **abs/2208.06537**, (2022).

[5] Yansong Gao, Minki Kim, Sharif Abuadbba, Yeonjae Kim, Chandra Thapa, Kyuyeon Kim, Seyit Ahmet Çamtepe, Hyoungshick Kim, and Surya Nepal, 'End-to-end evaluation of federated learning and split learning for internet of things', *2020 International Symposium on Reliable Distributed Systems (SRDS)*, 91–100, (2020).

[6] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy, 'Explaining and harnessing adversarial examples', *arXiv preprint arXiv:1412.6572*, (2014).

[7] Andrew Ilyas, Shibani Santurkar, Logan Engstrom, Brandon Tran, and Aleksander Madry, 'Adversarial examples are not bugs, they are features', *Advances in neural information processing systems*, **32**, (2019).

[8] Wei Jiang, Zhiyuan He, Jinyu Zhan, Weijia Pan, and Deepak Adhikari, 'Research progress and challenges on application-driven adversarial examples: A survey', *ACM Transactions on Cyber-Physical Systems (TCPS)*, **5**, 1 – 25, (2021).

[9] Asifullah Khan, Anabia Sohail, Umme Zahoora, and Aqsa Saeed Qureshi, 'A survey of the recent architectures of deep convolutional neural networks', *Artificial Intelligence Review*, 1 – 62, (2020).

[10] Alexey Kurakin, Ian J Goodfellow, and Samy Bengio, 'Adversarial examples in the physical world', *arXiv preprint arXiv:1607.02533*, (2016).

[11] Ximeng Liu, Lehui Xie, Yaopeng Wang, Jian Zou, Jinbo Xiong, Zuobin Ying, and Athanasios V. Vasilakos, 'Privacy and security issues in deep learning: A survey', *IEEE Access*, **9**, 4566–4593, (2021).

[12] Yuyang Long, Qi li Zhang, Boheng Zeng, Lianli Gao, Xianglong Liu, Jian Zhang, and Jingkuan Song, 'Frequency domain model augmentation for adversarial attack', *ArXiv*, **abs/2207.05382**, (2022).

[13] Yuyang Long, Qi li Zhang, Boheng Zeng, Lianli Gao, Xianglong Liu, Jian Zhang, and Jingkuan Song, 'Frequency domain model augmentation for adversarial attack', in *European Conference on Computer Vision*, (2022).

[14] Dario Pasquini, Giuseppe Ateniese, and Massimo Bernaschi, 'Unleashing the tiger: Inference attacks on split learning', *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, (2021).

[15] Alexandru Constantin Serban, Erik Poll, and Joost Visser, 'Adversarial examples on object recognition: A comprehensive survey', *ArXiv*, **abs/2008.04094**, (2020).

[16] Muhammad Shafay, Raja Wasim Ahmad, Khaled Salah, Ibrar Yaqoob, Raja Jayaraman, and Mohammed A. Omar, 'Blockchain for deep learning: review and open challenges', *Cluster Computing*, 1 – 25, (2022).

[17] Jingwei Sun, Ang Li, Binghui Wang, Huanrui Yang, Hai Li, and Yiran Chen, 'Soteria: Provable defense against privacy leakage in federated learning from representation perspective', *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 9307–9315, (2021).

[18] Chandra Thapa, Pathum Chamikara Mahawaga Arachchige, and Seyit Ahmet Çamtepe, 'Advancements of federated learning towards privacy preservation: from federated learning to split learning', *ArXiv*, **abs/2011.14818**, (2020).

[19] Praneeth Vepakomma, Otkrist Gupta, Tristan Swedish, and Ramesh Raskar, 'Split learning for health: Distributed deep learning without sharing raw patient data', *ArXiv*, **abs/1812.00564**, (2018).

[20] Wenxuan Wang, Bangjie Yin, Taiping Yao, Li Zhang, Yanwei Fu, Shouhong Ding, Jilin Li, Feiyue Huang, and X. Xue, 'Delving into data: Effectively substitute training for black-box attack', *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 4759–4768, (2021).

[21] Xiaosen Wang and Kun He, 'Enhancing the transferability of adversarial attacks through variance tuning', *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 1924–1933, (2021).

[22] Danyang Xiao, Chengang Yang, and Weigang Wu, 'Mixing activations and labels in distributed training for split learning', *IEEE Transactions on Parallel and Distributed Systems*, **PP**, 1–1, (2021).

[23] Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li, 'Adversarial examples: Attacks and defenses for deep learning', *IEEE Transactions on Neural Networks and Learning Systems*, **30**, 2805–2824, (2019).