ExPLoit: Extracting Private Labels in Split Learning

***Abstract*—Split learning is a popular technique used to perform vertical federated learning, where the goal is to jointly train a model on the private input and label data held by two parties. To preserve privacy of the input and label data, this technique uses a split model trained end-to-end, by exchanging the intermediate representations (IR) of the inputs and gradients of the IR between the two parties. We propose *ExPLoit* – a label-leakage attack that allows an adversarial input-owner to extract the private labels of the label-owner during split-learning. ExPLoit frames the attack as a supervised learning problem by using a novel loss function that combines gradient-matching and several regularization terms developed using key properties of the dataset and models. Our evaluations on a binary conversion prediction task and several multi-class image classification tasks show that ExPLoit can uncover the private labels with near-perfect accuracy of up to 99.53%, demonstrating that split learning provides negligible privacy benefits to the label owner. Furthermore, we evaluate the use of gradient noise as a defense and show that the protection against our attack comes at the cost of a significant loss in model utility. Our findings underscore the need for better privacy-preserving training techniques for vertically split data.**

***Index Terms*—Federated Learning, Split Learning, Data Privacy, Label Leakage**

I. Introduction

The plethora of apps and services that we use for our everyday needs, such as online shopping, social media, communication, healthcare, finance, etc., have created distributed silos of user data. While aggregating this distributed data would improve the performance of machine learning models, doing so is not always feasible due to privacy constraints. For instance, in healthcare, laws like HIPAA require hospitals to keep medical records private. For finance and internet companies, user agreements and privacy laws might prevent them from sharing data. These challenges have led to the development of several techniques for *federated learning*, which allow models to be trained on distributed private data, without the data owner having to share their data explicitly. *Split learning* [11], [25] is one such technique that allows federated learning to be performed when the inputs and the corresponding labels are held by two different parties. Split learning uses a composition of two models $f : \mathcal{X} \rightarrow \mathcal{Z}$ and $g : \mathcal{Z} \rightarrow \mathcal{Y}$ that is split between the input and label owners as shown in Fig. 1. The composition network $g \circ f$ is trained end-to-end by requiring the input owner to transmit the embedding z_i (intermediate representation) to the label owner in the forward pass and the label owner returning the gradient $\nabla_z L_i$ to the input owner during the backward pass. This allows the parameters of the split model to be trained on the distributed data while keeping the sensitive data with their respective owners.

Unfortunately, split learning does not have formal privacy guarantees, and it is not clear if it allows the input and label owners to hide their private data from each other. We set out to answer this question by considering an adversarial input owner who wants to break label privacy by extracting the private labels in two-party split learning. To this end, we propose *ExPLoit*– a label-leakage attack that frames the problem of learning the private labels as a supervised learning task, by leveraging the gradient information ($\nabla_z L_i$) obtained during split learning. ExPLoit "replays" split learning by replacing the label owner's model g and labels $\{y_i\}$ with a randomly initialized surrogate model g' (with parameters $\theta_{g'}$) and surrogate labels $\{y'_i\}$ respectively. We aim to learn the private labels by training these surrogate parameters using the following key objectives:

1) *Gradient Matching Objective:* The gradient computed using the surrogate model and labels during *replay split learning* should match the gradients received from the label owner during the original split learning process.
2) *Label Prior Objective:* The distribution of surrogate labels must match the expected label prior distribution. For instance, if the classification problem in consideration has a uniform label prior, the surrogate labels must also have a uniform distribution.
3) *Label Entropy Objective:* Since we consider datasets with hard labels, each individual surrogate label y'_i must have low entropy.
4) *Accuracy Objective:* The predictions made by the surrogate model must be close to the surrogate labels, achieving high accuracy.

Combining the above objectives yields a loss function that can be used to train the surrogate parameters and labels. By minimizing this loss function over all the embedding, gradient pairs $\{z_i, \nabla_z L_i\}$ received during split learning, the surrogate labels $\{y'_i\}$ can be trained to match the private labels of the label owner $\{y_i\}$, allowing an adversarial input owner to carry out a label leakage attack with high accuracy. Our paper makes the following key contributions:

Contribution 1: We propose *ExPLoit* – a label-leakage attack for two-party split learning. Our proposal replaces the label owner's private labels and model with trainable surrogate parameters and frames the attack as a supervised learning problem. We combine gradient-matching and regularization terms obtained by leveraging several key properties of the model and dataset to develop a novel loss function. By training the surrogate parameters to minimize this loss function, our attack uncovers the private labels of the label owner corresponding to each input.

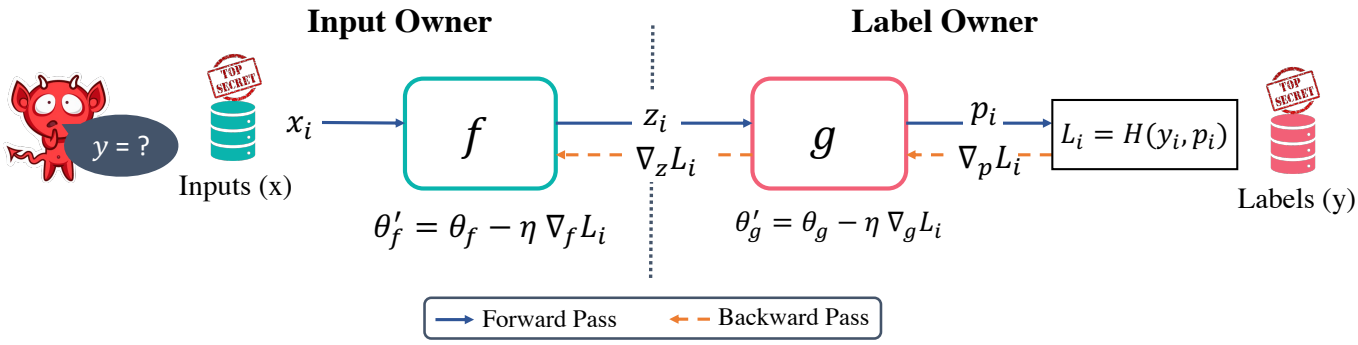

Fig. 1. Split learning can be used for vertical federated learning by training a composition model $g \circ f$, split between the input and model owner. In this work, we demonstrate that an adversarial input owner can learn the private labels using gradient information obtained during split learning, compromising the label owner's privacy.

Contribution 2: We carry out extensive evaluations on the Criteo conversion prediction [23] task, and several image classification datasets, including FashionMNIST, CIFAR-10, CIFAR-100 and Tiny-ImageNet to show that ExPLoit can leak private labels with near-perfect accuracy (up to 99.53%) for most datasets. Our attack is effective across multiple model architectures and also outperforms several recently proposed label-leakage attacks for split learning.

Contribution 3: We evaluate perturbing the gradient with noise as a defense against our attack. Our results show that gradient noise allows the label owner to trade off model accuracy (lower utility) for improved privacy against ExPLoit (better label privacy). However, our results show that a large amount of noise is necessary to offer protection against our attack. This results in a significant degradation of accuracy for several datasets like CIFAR-10, CIFAR-100 and Tiny-Imagenet.

Our findings in this work demonstrate that split learning does not protect the privacy of labels, emphasizing the need for better techniques for VFL.

II. Related Work

We discuss prior work on label leakage attacks and describe their limitations. For an overview of techniques besides split learning that can be used to learn on vertically partitioned data, we refer the reader to Section X.

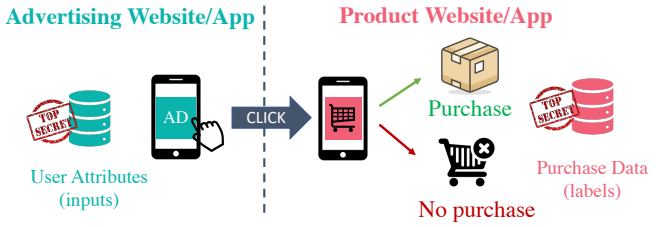

Fig. 2. Conversion prediction estimates the likelihood of a purchase when a user clicks on an ad. The training data is vertically partitioned, with the user attributes (inputs) held by the ad company and the purchase data (outputs) held by the product company (Figure adapted from [15]).

A. Norm-Based Attack for Conversion Prediction

Recently, Li et al. proposed *Norm-Based Attack* [16] – a label leakage attack on two-party split learning, specifically for the conversion prediction problem. We first provide background on the conversion prediction problem and then describe the Norm-Based attack.

Conversion Prediction: Given the attributes of a user and an ad, conversion prediction estimates the likelihood of a user purchasing the product. Conversion prediction is an essential component of ad-ranking algorithms, as ads with a high likelihood of conversion are more relevant to the user and need to be ranked higher. The data required to train the model are split between the advertising and product websites, as depicted in Fig. 2. The user attributes, which serve as the inputs, are stored with the advertising company, while the purchase data, which serve as the labels, are held with the product company. The companies are interested in training a model to predict the conversion likelihood while keeping their datasets private.

Norm-based Attack: Norm-based attack [16] leverages the observation that only a small fraction of ad clicks result in a purchase. Consequently, there is a high class imbalance in the training dataset of the conversion prediction task. This imbalance results in the magnitude of the gradients being higher when the infrequent class is encountered. Thus, by considering the norm of the loss gradient ($\|\nabla_z L_i\|_2$), an adversarial input owner can infer the private labels. Note that a key limitation of this attack is that it only works on binary classification problems with high class imbalances. In contrast, our proposed ExPLoit attack does not require a class imbalance and works for multi-class classification problems.

B. UnSplit: Gradient Matching Attack

Similar to our attack, a recent concurrent work *UnSplit* [8] also proposes to learn the private labels in split learning using a gradient-matching objective [31] by minimizing the mean squared error (MSE) between the surrogate and true gradients using the following objective: $\min_{\theta'_g, \{y'_i\}} MSE(\nabla_z L'_i, \nabla_z L_i)$. Here, θ'_g are the parameters of the surrogate model and $\{y'_i\}$ are the surrogate labels. Results from this work show that

UnSplit only works well when the label-owner's model g is one-layer deep. In contrast, ExPLoit provides high accuracy of up to 99.53%, even when the label owner uses multi-layer networks (up to 8 layers deep in our experiments). This is because ExPLoit uses additional regularization terms in the loss that help avoid poor local minima during training.

C. Model Completion Attack

Model Completion Attack [9] uses unlabeled embeddings $\mathcal{D} = \{z_i\}$ and a small number of labeled embeddings $\mathcal{D}^l = \{z_i^l, y_i^l\}$ to train a surrogate model $g' : \mathcal{Z} \rightarrow \mathcal{Y}$ using semi-supervised learning. Since g' functionally approximates the label-owner's model g, it can be used to predict the labels for the input embeddings $y_i' = g'(z_i)$, allowing the input owner to guess the private labels. This proposal suffers from two key drawbacks. First, it requires the adversary to have access to labeled examples, which may not always be available. For instance, in case of conversion prediction, labels for the input data cannot be gathered even using human annotators, as it is not readily apparent from the data. Second, the efficacy of this attack is limited by the accuracy of the model that can be trained by the adversary. Thus, the attack accuracy is highly dependent on the number of labeled examples available to the attacker and the difficulty of the prediction problem at hand. For example, this attack provides a label leakage accuracy of 75.06% for FashionMNIST, where a high-accuracy g' can be trained with just a few labeled examples. However, the accuracy is much lower (15.3%) for CIFAR-100, where the surrogate model is harder to train. In contrast, ExPLoit does not require any labeled examples and can achieve a high accuracy of 94.38% even for complex datasets like CIFAR-100 (after 10 training epochs of the split-model).

III. Preliminaries

In this section, we provide background on the two-party split learning framework and formally state the objectives of the label leakage attack and defense.

A. Two-Party Split Learning

Two-party split learning is used for VFL, where the data is vertically partitioned between the input and label owner. The input owner owns the inputs $\mathcal{D}_{inp} = \{x_i\}$ and the label owner owns the labels $\mathcal{D}_{label} = \{y_i\}$, corresponding to each input. The goal of split learning is to train a composition model $g \circ f$ that is distributed between the two parties. Training with supervised learning requires mapping the entries in the input set to the corresponding entries in the label set. If this mapping is not known, private set intersection algorithms [5] can be used to link the corresponding entries in the two datasets. A single training iteration involves a forward and a backward pass (as shown in Fig. 1), which proceeds as follows:

- *Forward pass:* The input owner samples a batch of inputs $\{x\}_{batch} \sim \mathcal{D}_{inp}$ and performs forward propagation through $f : \mathcal{X} \rightarrow \mathcal{Z}$ and produces the corresponding embeddings $\{z\}_{batch}$. These embeddings, along with the corresponding *inputIDs* are sent to the label owner. The label owner feeds the embeddings to $g : \mathcal{Z} \rightarrow \mathcal{Y}$ to produce the predictions $\{p\}_{batch}$, which along with the labels $\{y\}_{batch}$ are used to compute the model's loss $L = \mathbb{E}[H(y, p)]$.
- *Backward pass:* The label owner initiates backpropagation and returns the loss gradient $\{\nabla_z L\}_{batch}$ to the input owner. Both the label and input owner compute the gradient of the loss with respect to the model parameters and update model parameters using gradient descent as shown below:

$$\theta_g^{t+1} = \theta_g^t - \eta \nabla_{\theta_g} L; \quad \theta_f^{t+1} = \theta_f^t - \eta \nabla_{\theta_f} L. \tag{1}$$

Privacy Objectives: There are two key privacy objectives that split learning aims to achieve:

1) *Input privacy:* The label owner should not be able to infer the input owner's private inputs $\{x_i\}$.
2) *Label privacy:* The input owners should not be able to infer the label owner's private labels $\{y_i\}$.

B. Label Leakage Attack Objective

In this work, we propose a label leakage attack, where an adversarial input owner tries to learn the label owner's private labels $\mathcal{D}_{label} = \{y_i\}$. We consider an honest-but-curious adversary, where the input owner tries to infer the private labels while honestly following the split learning protocol. During the training process of split learning, for each input x_i, the adversarial input owner transmits the embeddings z_i and receives the gradient $\nabla_z L_i$ from the label owner. The adversary uses an algorithm $\mathcal{A}$ to estimate the private labels y_i' for each input using these $\{z_i, \nabla_z L_i\}$ pairs obtained during split learning as shown below:

$$\mathcal{A}(\mathcal{D}_{grad}) \rightarrow \mathcal{D}_{y'}, \quad \text{where } \mathcal{D}_{grad} = \{z_i, \nabla_z L_i\},\ \mathcal{D}_{y'} = \{y_i'\}. \tag{2}$$

The attack objective is to maximize the accuracy of estimated labels as follows:

$$\max \mathop{\mathbb{E}}_{y_i \sim \mathcal{D}_{label}} [Acc(y_i, y_i')]. \tag{3}$$

Since the private labels $\{y_i\}$ are unavailable to the input owner, evaluating Eqn. 3 is not possible. Instead, our proposed attack uses a surrogate objective that can be optimized to uncover the private labels with high accuracy.

Attack Constraints: We assume that the parameters and architecture of the label-owner's model g is unknown to the attacker. While hiding the label-owner's model architecture does not provide any theoretical security benefits, our assumption is intended to simulate a hard setting for the attacker. The number of output classes and the class prior distribution are assumed to be known[1].

[1]We refer the reader to Section VIII) for a justification of this assumption.

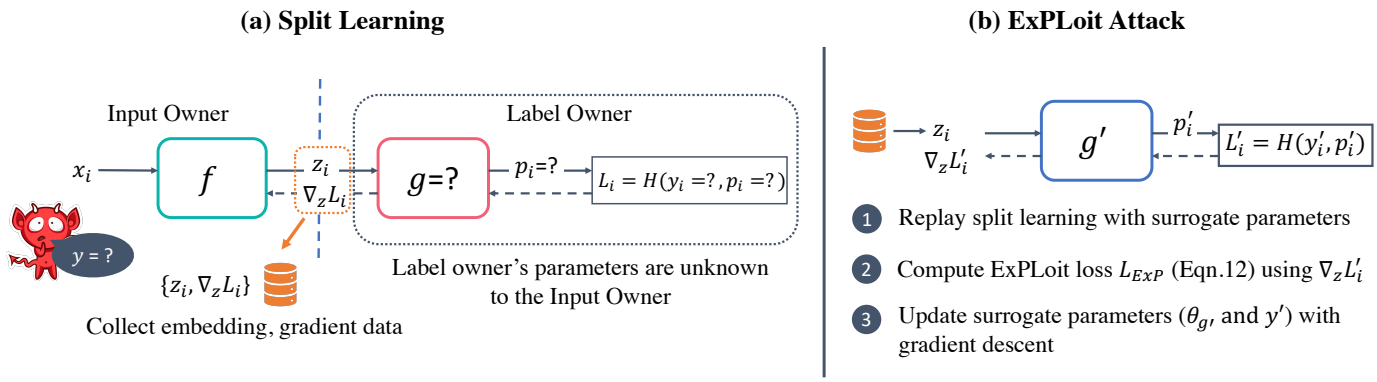

Fig. 3. (a) *Split Learning:* The adversarial input owner collects the embedding and gradient data $\{z_i, \nabla_z L_i\}$ when performing split learning with the label owner. (b) *ExPLoit Attack:* The embedding and gradient data is used to train surrogate model and label parameters (g' and $\{y_i'\}$) and uncover the private labels.

C. Label Leakage Defense Objective

Defending against label leakage attack requires balancing the following utility and privacy objectives:

Utility Objective: Train the composition model $g \circ f$ to have high classification accuracy on an unseen validation set associated with the classification task (Eqn. 4).

$$\max \mathop{\mathbb{E}}_{x_i, y_i \sim \mathcal{D}_{val}} [Acc(y_i, g(f(x_i))] \tag{4}$$

Privacy Objective: Minimize the accuracy of the estimated labels $\{y_i'\}$ that can be recovered from the gradient information used in split learning (Eqn. 5).

$$\min \mathop{\mathbb{E}}_{y_i \sim \mathcal{D}_{label}} [Acc(y_i, y_i')] \tag{5}$$

IV. Our Proposal: ExPLoit

We propose *ExPLoit*– a label leakage attack that can be used by a malicious input-owner to learn the private labels in split learning. Our key insight is that the label leakage attack can be framed as a supervised learning problem by replacing the unknown parameters of the label owner with learnable surrogate parameters. This allows the adversarial input owner to "replay" the split learning process with surrogate parameters. We train these surrogate parameters by using a novel loss function that uses gradient-matching and several regularization terms that leverage key properties of the model and training data. By optimizing the surrogate parameters to minimize this loss function, we can recover the label owner's private labels with high accuracy. The rest of this section describes our proposed attack in greater detail.

A. Surrogate Parameter Substitution

From the input owner's point of view, the split learning process has two sets of unknown parameters associated with the label-owner: the label owner's model g and the private labels $\{y_i\}$ (see Fig. 3a). Our goal is to uncover these unknown values by treating them as learnable parameters. To do so, we start by substituting these unknowns with randomly initialized surrogate parameters, as shown in Fig. 3b. We replace g with a surrogate model g' (with parameters $\theta_{g'}$), and $\{y_i\}$ with a set of surrogate labels $\{y_i'\}$. We want y_i' to be a point on an $n-1$ dimensional probability simplex for an n-class classification problem. To enforce this property, we set $y_i' = Softmax(\hat{y}_i)$, where $\hat{y}_i \in \mathcal{R}^n$. With the surrogate parameters in place, the goal of our attack is to learn the surrogate labels $\{y_i'\}$ (or equivalently to learn $\{\hat{y}_i\}$).

B. Replay Split Learning

To train the surrogate parameters, we first "replay" the split learning process using the surrogate parameters (Fig. 3b). First, in the forward pass, the embedding z_i (collected during split learning) is fed into g' to get the prediction p_i', which along with the surrogate labels y_i' can be used to compute the loss $L_i' = H(y_i', p_i')$. Next, we perform backpropagation through g' and compute the gradient of the loss with respect to the embedding $\nabla_z L_i'$. We use this gradient data as part of our loss function to learn the private labels, as described below.

C. ExPLoit Loss

To train the surrogate parameters $\theta_{g'}$ and $\hat{y}$, we formulate a loss function using four key objectives:

1. *Gradient Objective*: The loss gradient $\nabla_z L_i'$, obtained during *replay split learning*, must match the original gradients $\nabla_z L_i$, obtained during the original split learning process. This can be achieved by minimizing the l^2 distance between $\nabla_z L_i'$ and $\nabla_z L_i$ as shown below:

$$\min_{\theta_g', \{\hat{y}_i\}} \mathbb{E} \left\| \nabla_z L_i' - \nabla_z L_i \right\|_2 . \tag{6}$$

2. *Label Prior Objective*: The distribution of surrogate labels must match the label prior P_y of the dataset. The probability distribution of the surrogate labels can be computed by taking the expectation of the surrogate labels[2] $P_{y'} = \mathbb{E}(y_i')$. We perform the following optimization to match the distributions of the original and surrogate labels:

$$\min_{\theta_g', \{y_i'\}} \mathcal{D}_{KL}(P_y \| P_{y'}). \tag{7}$$

[2]Each surrogate label y_i' is a n-dimensional probability vector that represents the probability distribution over the n output classes.

3. *Label Entropy Objective*: Each individual surrogate label y'_i must have low entropy as the datasets we consider have zero entropy one-hot labels.

4. *Accuracy Objective*: The surrogate model must have high prediction accuracy with respect to the surrogate labels. In other words, the predictions of the surrogate model p'_i must be close to the surrogate labels y'_i.

To achieve the label entropy and accuracy objectives, we can minimize the normalized cross-entropy loss between p'_i and y'_i as follows:

$$\min_{\theta'_g, \{y'_i\}_i} \frac{\mathbb{E}[H(y'_i, p'_i)]}{H(P_y)}. \tag{8}$$

Note that the cross-entropy term $H(y'_i, p'_i)$ in Eqn. 8 can be expressed as a sum of the label entropy and KL divergence between the surrogate label and prediction: $H(y'_i, p'_i) = H(y'_i) + \mathcal{D}_{KL}(y'_i \| p'_i)$. Thus, by minimizing cross-entropy, we can minimize the entropy of surrogate labels (label entropy objective) and match the model's predictions with the surrogate labels (accuracy objective). We normalize cross-entropy with the entropy of the label prior $H(P_y)$ to ensure that the metric is insensitive to the number of label classes and the label priors [13].

We combine all the learning objectives described above to derive the final loss function as shown below:

$$L_{ExP} = \mathbb{E}\Big[\|\nabla_z L_i - \nabla_z L'_i\|_2 \Big] + \lambda_{ce} \cdot \mathbb{E}\Big[H(y'_i, p'_i)/H(P_y) \Big] + \lambda_p \cdot \mathcal{D}_{KL}(P_y \| P_{y'}). \tag{9}$$

Here, λ_{ce} and λ_p dictate the relative importance of the cross-entropy and label prior terms compared to the gradient loss term (first term in Eqn. 9). By optimizing the surrogate model and label parameters using this loss function, we can recover the private labels of the label owner with high accuracy. We consider the gradient loss term to be the primary optimization objective of our loss function. The cross-entropy and label prior terms act as regularizers and help us achieve a better label leakage accuracy (see Section VII for an ablation study).

D. Putting It All Together

The individual components described thus far can be combined to carry out our label leakage attack. Our attack starts with the input owner performing split learning process with the label owner, as shown in Fig. 3a. During this process, the input owner collects the embedding z_i and the corresponding loss gradient $\nabla_z L_i$ for each input. Using this data, the input owner can use the *ExPLoit* attack to leak the private labels. Our attack is described in Algorithm 1. In the outer loop, we pick values for λ_p, λ_{ce} and the learning rates $\eta_{g'}, \eta_{\hat{y}}$ using a Bayesian hyperparameter optimization algorithm. The surrogate parameters $\{\hat{y}_i\}_i$ and $\theta_{g'}$ are randomly initialized, and each inner loop of the attack proceeds as follows:

1) Replay split learning with surrogate parameters with the following steps:
   a. Sample a batch of embeddings, gradients and surrogate labels: $\{z, \nabla_z L, \hat{y}\}_{batch}$.
   b. Perform forward pass and compute loss $\{L'\}$ using predictions $\{p'\}$ and surrogate labels $\{y'\}$.
   c. Perform backpropagation to compute the loss gradients $\{\nabla_z L'\}$.
2) Compute the ExPLoit loss: L_{ExP} (Eqn. 9).
3) Update surrogate parameters $\theta_{g'}$ and $\{\hat{y}_i\}$ to minimize L_{ExP}.

We repeat the above steps until the values of the surrogate parameters converge.

Algorithm 1: ExPLoit Attack

```
Input: {z_i}, {∇_z L_i}, P_y, N_iter
Output: {y_i^*}
D_train = {z_i, ∇_z L_i, y'_i}
for i ← 0 to N_iter do
    λ_p, λ_ce, η_g', η_ŷ ← BayesOpt()
    Initialize {ŷ_i}, g'(·; θ_g')
    repeat
        for {z, ∇_z L, ŷ}_batch in D_train do
            {y'_i} = {Softmax(ŷ_i)}
            P_y' = E(y'_i)

            // 1. Replay Split Learning
            for {z, ∇_z L, ŷ}_i in {z, ∇_z L, ŷ}_batch do
                p'_i = g'(z_i; θ_g')
                L'_i = D_KL(y'_i || p'_i)
                Compute ∇_z L'_i
            end

            // 2. Compute ExPLoit loss (Eqn. 12)
            L_ExP = E[||∇_z L'_i − ∇_z L_i||_2] + λ_ce ·
              E[H(y'_i, p'_i)/H(P_y)] + λ_p · D_KL(P_y || P_y')

            // 3. Update surrogate model, label
              parameters
            θ_g' ← θ_g' − η_g' · ∇_θ_g' L_ExP
            ŷ ← ŷ − η_ŷ · ∇_ŷ L_ExP
        end
    until Convergence;
    NewBest =
      UpdateBayesOpt(E[||∇_z L'_i − ∇_z L_i||_2])
    if NewBest then
        {y_i^*} ← {Softmax(ŷ_i)}
    end
end
```

Hyperparameter Optimization: We learn a set of surrogate labels $\{y'_i\}$ in each outer loop for different selections of the hyperparameters. Unfortunately, evaluating the accuracy of the surrogate labels produced in each iteration is not possible since the input owner is unaware of any of the true labels. Consequently, we cannot use accuracy to guide the hyperparameter search. Instead, we evaluate the gradient loss term $\mathbb{E}[\|\nabla_z L'_i - \nabla_z L_i\|_2]$ after completing each outer iteration and use this as our objective function to be minimized

TABLE I
DATASETS AND THE CORRESPONDING SPLIT-MODELS USED IN OUR EXPERIMENTS.

Dataset	Config-1		Config-2		g'
	f	g	f	g	
FashionMNIST	$Conv \times 4$	$FC \times 2$	$Conv \times 3$	$Conv - FC \times 2$	$FC \times 3$
CIFAR-10	$Conv - Res \times 9$	$FC \times 2$	$Conv - Res \times 6$	$Res \times 3 - FC \times 2$	$FC \times 3$
CIFAR-100	$Conv - Res \times 9$	$FC \times 2$	$Conv - Res \times 6$	$Res \times 3 - FC \times 2$	$FC \times 3$
Tiny-ImageNet	$Conv - Res \times 9$	$FC \times 2$	$Conv - Res \times 6$	$Res \times 3 - FC \times 2$	$FC \times 3$
Criteo	$Emb - FC \times 2$	$FC \times 2$	$Emb - FC$	$FC \times 3$	$FC \times 4$

by tuning the hyperparameters. We report the accuracy of the surrogate labels obtained for the best set of hyperparameters that minimizes this objective.

V. EXPERIMENTS

We evaluate ExPLoit with multiple datasets and model architectures to show that it can leak private labels with high accuracy, across different settings. We describe our experimental setup followed by the results in this section.

A. Experimental Setup

The datasets and the corresponding split-models ($f \circ g$) used in our evaluations are shown in Table I.

Datasets: FashionMNIST, CIFAR-10, CIFAR-100, and Tiny-ImageNet are computer vision datasets used to perform multi-class image classification. The Criteo dataset consists of conversion logs for online ad-clicks, with each entry consisting of 3 continuous, and 17 categorical features, along with a binary label indicating if the ad-click resulted in a purchase (conversion). Note that the Criteo dataset has a large class imbalance (90% of the labels are 0's, and the rest are 1's).

Models: We use a 4-layer convolutional neural network for FashionMNIST and a 21-layer ResNet model for CIFAR-10, CIFAR-100 and Tiny-ImageNet. The model for the conversion prediction task (Criteo) consists of a learnable embedding layer (to handle categorical features) followed by four fully connected (FC) layers. All the models are split into two sub-models f (input-owner's model) and g (label-owner's model), which are jointly trained using split learning. The layer at which the model is split is referred to as the *cut-layer*. To test the sensitivity of our attack to the cut-layer, we perform experiments with two different configurations: Config-1 and Config-2, which splits the model at different points as shown in Table I. The label-owner's model g only consists of FC layers in Config-1. In contrast, the g model in Config-2 is larger and consists of both convolutional (Conv) and FC layers. The vision models are trained for 10 epochs and the CVR model for 5 epochs using the Adam optimizer with a learning rate of 0.001. We evaluate the efficacy of label leakage attacks at different points during the split-model training by carrying out the attack after each training epoch.

Attack parameters: We assume that the architecture of the label owner's model g is not known to the input owner. Thus, we use a 3-layer fully connected DNN (FC[128-64-10] for image classification and FC[32-32-10] for Criteo) as the surrogate model g'. We set $N_{iter} = 500$, and the learning rate range to $[10^{-5}, 10^{-4}]$ for $\eta_{g'}$ and $[10^{-2}, 10^{-1}]$ for $\eta_{\hat{y}}$. The range for λ_{ce} and λ_p is set to $[0.1, 3]$. We carry out the ExPLoit attack for each training epoch of split learning.

Evaluation Metric: By learning the surrogate labels associated with each input, ExPLoit groups the inputs that belong to the same class together. However, the true class labels corresponding to each input group remains to be determined. For an unbalanced dataset, the label prior information can be used to infer the label by considering the size of each input group. For a balanced dataset, the adversary can use inputs with known labels (one for each output class) to determine the label associated with each input group. We report the clustering accuracy [28] obtained with the best hyperparameters (corresponding to the lowest gradient loss) in our results below.

B. Results

We plot label leakage accuracy for ExPLoit and other label leakage attacks and compare it with the test accuracy/normalized cross entropy[3] of the split-model, across various datasets and model configurations in Fig. 4. ExPLoit achieves near-perfect label leakage accuracy for most datasets (99.53% for CIFAR-10) and significantly outperforms all prior works. We provide explanations for the attack sensitivity to various parameters like dataset, split-model training epoch, cut layer, and a detailed comparison with recent prior works below. We use the accuracy numbers from Config-1 (Fig. 4a), Epoch-10 to discuss the results, unless specified otherwise.

Sensitivity to Split-model Training Epoch: Our results show that the efficacy of ExPLoit improves when attacking the later epochs of the split-model training. For instance, the label leakage accuracy of ExPLoit for CIFAR-100 is just 30.65% for Epoch-1 and improves to 94.4% for Epoch-10. The reason for this trend is because our attack approximates the label owner's model g using a fixed surrogate model g'. However, in reality, g is not fixed, and changes as training progresses. The rate of this change is smaller for the latter epochs. Consequently, our ability to approximate g with a fixed surrogate model improves for the later epochs, which improves our attack's efficacy. While ExPLoit already achieves near perfect accuracy for most datasets with just 10 epochs of split-model training,

[3]We use normalized cross-entropy (NCE) instead of test accuracy to measure the model performance for the Criteo dataset as it has a high class imbalance. A lower value of NCE indicates better performance.

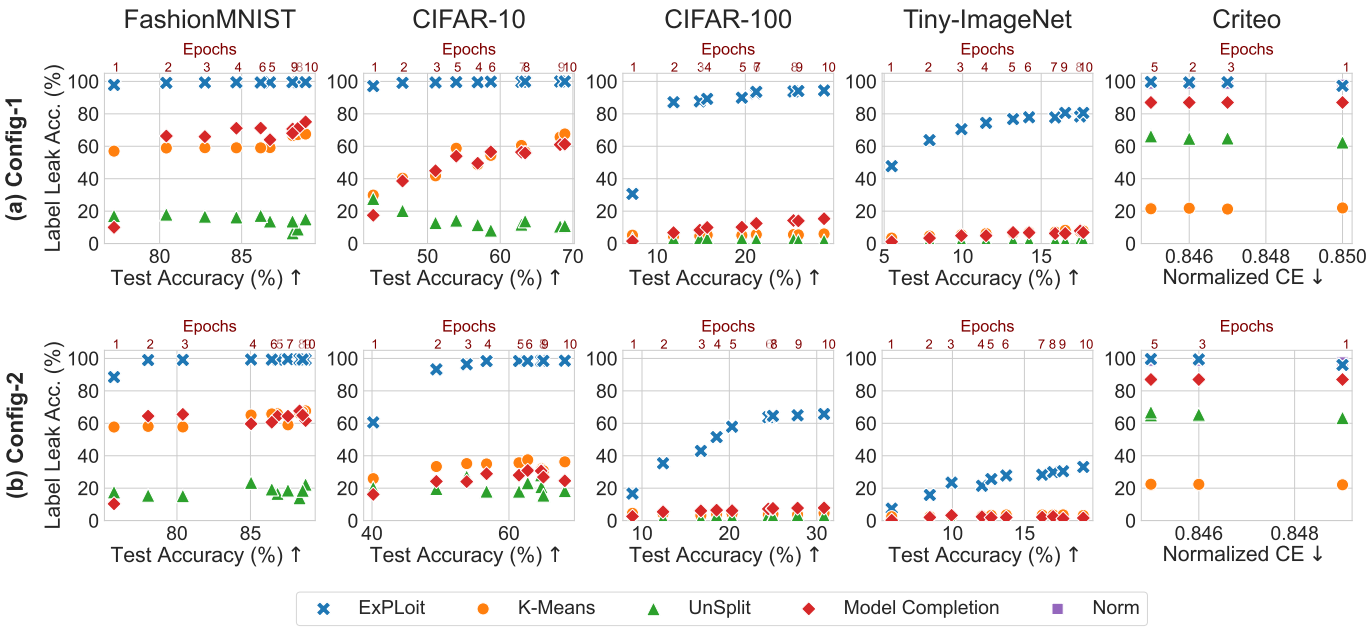

Fig. 4. Results comparing *ExPLoit* with the K-means baseline and prior works (UnSplit, Model Completion and Norm-based attacks) for two configurations of the split model: (a) Config-1 and (b) Config-2 (see Table I). ExPLoit significantly outperforms prior works and can leak private labels with a high accuracy (up to 99.53%) for most datasets.

we expect this accuracy to improve further as the split-model is trained for a greater number of epochs.

Sensitivity to Datasets: ExPLoit is more effective for datasets with lower input dimensionality and fewer classes. For instance, ExPLoit has a label leakage accuracy of 99.5% for FashionMNIST, whereas the accuracy drops to 94.38% for CIFAR-100 and 80.61% for Tiny-ImageNet. This is because our attack has a higher number of surrogate parameters for CIFAR-100 and Tiny-ImageNet compared to FashionMNIST (due to larger input size and number of output classes), which increases the difficulty of the learning problem in our attack.

Sensitivity to Cut layer and Model Architecture: We evaluate our attack on two split network configurations: Config-1 and Config-2, which represent two different choices of the cut layer. ExPLoit achieves a higher label-leakage accuracy for Config-1 compared to Config-2. For instance, in the case of CIFAR-100, our attack produces a label leakage accuracy of 94.38% for Config-1 and 65.74% for Config-2. The reason for this discrepancy is two-fold. First, g is larger for Config-2, compared to Config-1, which makes it harder to approximate with a surrogate model g' for Config-2. Second, our g' model is architecturally similar to the g model in Config-1 as both these models use fully connected layers, while the g models in Conv-2 uses FC and Conv layers. Thus, the efficacy of our attack reduces when g is larger and has a dissimilar model architecture compared to g'.

Comparisons with Baseline and Prior Work: We compare the performance of ExPLoit against an unsupervised learning baseline (K-Means Clustering) and three recent attacks: UnSplit, Model Completion and Norm based attack. The experimental setup for these prior works is described in Section IX.

K-Means Attack: Through the split-learning process, the label-owner is able to generate embeddings $z_i = f(x_i)$, for each input x_i. One way to estimate the private labels is by using unsupervised learning to group these embeddings. We perform K-Means clustering using the embeddings $\{z_i\}$ and report the resulting label leakage accuracy in Fig. 4. The efficacy of the attack improves with the quality of embeddings. Consequently, the attack performs well for simpler datasets like FashionMNIST, where it is easier to learn good embeddings with relatively few training epochs. The attack accuracy accuracy also improves for later epochs as the model f learns better embeddings as training progresses.

UnSplit Attack [8]: The UnSplit attack uses the gradient matching loss to learn the private labels. The authors of [8] showed that this technique is effective only when g is a single layer network and does not work for multi-layer networks. Consistent with their results, our experiments with the UnSplit attack also showed very low efficacy (10.99% attack accuracy for CIFAR-10, which is comparable to a random guess).

Model Completion Attack [9]: This attack proposes to train a surrogate model g' using semi-supervised learning to predict the private labels corresponding to the inputs. Similar to the K-Means attack, the efficacy of this attack depends on the quality of emebddings produced by f. Consequently, this attack works well for simpler datasets, providing an accuracy of 75.06% for FashionMNIST, while the accuracy degrades for more complex datasets like CIFAR-10 (61.3% accuracy), CIFAR-100 (15.3% accuracy) and Tiny-ImageNet (6.875% accuracy).

Norm-based attack [16] : The norm-based attack uses gradient norm to predict the labels in imbalanced binary classification problems. Evaluations on the Criteo dataset shows

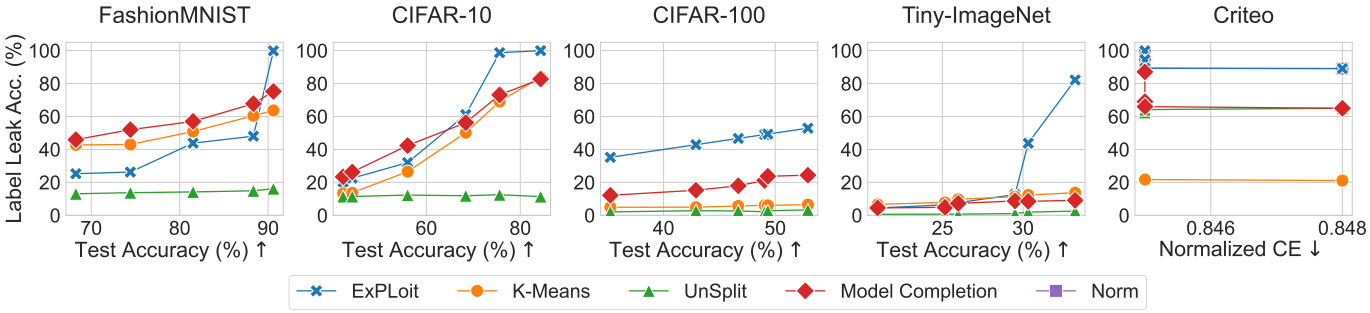

Fig. 5. Results showing the utility-privacy trade-off for various attacks under the gradient noise defense. A larger gradient noise reduces the efficacy of label leakage attacks at the cost of degradation in model accuracy.

that this attack can achieve high accuracy (comparable with ExPLoit). The leakage accuracy also improves for the later epochs of the split-model training as the difference in gradient norms becomes more pronounced.

ExPLoit significantly outperforms all prior works, providing a near-perfect label leakage accuracy for most datasets. E.g. ExPLoit achieves a label-leakage accuracy of 99.53% for CIFAR-10, which is 32.38% higher than the next best attack. This difference is even more pronounced when the attack is carried out at an earlier training epoch (67.2% higher accuracy compared to next best attack at Epoch-1 for CIFAR-10). This is because, unlike prior works, the efficacy of ExPLoit does not depend on the quality of embeddings produced by f. The efficacy of our attack demonstrates that split learning offers negligible privacy benefits for the label owner.

VI. Gradient Noise Defense

ExPLoit uses the gradient information obtained from the label owner during split learning to leak the private labels. One way to defend against ExPLoit is by perturbing the loss gradients $\nabla_z L_i$ with noise as shown in Eqn. 10.

$$\nabla_z \hat{L}_i = \nabla_z L_i + \eta, \quad where\ \eta \sim \mathcal{N}(0, \sigma) \tag{10}$$

The label owner can transmit these noisy gradients $\nabla_z \hat{L}_i$ to the input owner to perform split learning. Adding noise prevents the input owner from having reliable access to the true gradients. This reduces the efficacy of ExPLoit, providing better privacy to the label owner. On the other hand, noisy gradients are detrimental to training the split model and results in lower accuracy, thus impacting utility. To evaluate the utility-privacy trade-off offered by this defense, we perform split learning with different amounts of gradient noise by sweeping σ in Eqn. 10. For each value of σ, we train the split-model till convergence and report the test and label leakage accuracy with ExPLoit. Since the gradients obtained during split learning are noisy, optimizing the hyperparameters of our attack using the gradient loss objective is not optimal. Instead we tune the hyperparameters using $L_{ExP}(\lambda_{ce} = 1, \lambda_p = 1)$ as the optimization objective.

The utility-privacy trade-off with gradient noise defense for ExPLoit is shown in Fig. 5. Against the ExPLoit attack, this defense provides a better utility-privacy trade-off for lower dimensional datasets like Criteo and FashionMNIST. For instance, gradient noise degrades the label leakage accuracy for FashionMNIST by 51.8% with only a 2.26% reduction of test accuracy. In contrast, for CIFAR-10, a 79% reduction in label leakage accuracy incurs a 38% reduction in test accuracy. We hypothesize that this discrepancy could be attributable to the degradation in the quality of the input owner's model f. Lower-dimensional datasets could be more resilient to this degradation, whereas higher-dimensional datasets like CIFAR-10, CIFAR-100 and Tiny-ImageNet may be impacted more if f is not trained properly.

Additionally, we evaluate the gradient noise defense against other attacks. While the gradients are not directly used by the K-Means and model completion attacks, gradient noise reduces the quality of embeddings learnt by f, which degrades the efficacy of these attacks. We find that these attacks are more resilient to gradient noise for FashionMNIST, as the quality of the embeddings does not degrade significantly. However, ExpLoit performs better than these two attacks for datasets like CIFAR-100 and Tiny-ImageNet. The UnSplit attack continues to provide no benefit under gradient noise.

VII. Ablation Study

ExPLoit replaces the unknown labels and model parameters of the label owner with surrogate parameters and uses the gradient information obtained during split learning to train these surrogate parameters. In addition to matching the surrogate gradients obtained during "replay" split learning with the original gradients, our loss function consists of two regularization terms as shown in Eqn. 11.

$$\begin{aligned} L_{ExP} =& \mathbb{E}\Big[\, \|\nabla_z L_i - \nabla_z L'_i\|_2 \Big] + \\ & \lambda_{ce} \cdot \mathbb{E}\Big[H(y'_i, p'_i)/H(P_y)\Big] + \lambda_p \cdot \mathcal{D}_{KL}(P_y \| P_{y'}) \end{aligned} \tag{11}$$

The cross-entropy regularization (CER) term $\mathbb{E}\Big[H(y'_i, p'_i)/H(P_y)\Big]$ achieves the dual objective of minimizing the entropy of the individual surrogate labels and improving the accuracy of the surrogate model (g'). The label prior regularization (LPR) term $\mathcal{D}_{KL}(P_y \| P_{y'})$ tries to match the distribution of the surrogate labels with the label prior.

TABLE II
ABLATION STUDY SHOWING THE LABEL LEAKAGE ACCURACY OF EXPLOIT WHEN THE TWO REGULARIZATION TERMS: LABEL PRIOR REGULARIZATION (LPR) AND CROSS ENTROPY REGULARIZATION (CER), ARE NOT USED.

Dataset	Original (%)	No LPR (%)	No CER (%)	No LPR, CER (%)
FashionMNIST	99.84	69.78	59.31	27.83
CIFAR-10	99.96	99.28	99.35	99.84
CIFAR-100	94.38	60.78	17.04	20.38
Tiny-ImageNet	80.61	23.58	10.08	8.88
Criteo	99.68	99.65	99.87	97.75

We conduct an ablation study to understand the importance of the two regularization terms by carrying out ExPLoit without using LPR, without using CER, and without using both LPR and CER. The resuls of this study are shown in Table II. As expected, we find that there is a degradation in accuracy when regularization terms are not used. CER seems to be more important compared to LPR as the degradation is higher when CER is not used. For CIFAR-10 and Criteo, the regularization terms seem to matter less as the accuracy is high even when we disable both regularization terms.

VIII. LIMITATIONS

Our proposed ExPLoit Attack uses the gradient information obtained during split learning to leak the private labels. We assume that the input owner has knowledge of the number of classes and the distribution of the labels over these classes (prior information) to develop our loss function (Eqn. 9). If the attacker is completely unaware of the downstream classification task, this information could be hard to estimate, making our attack less effective (see Section VII). However, we argue that it is rare for the input owner (attacker) to be completely unaware of the downstream classification task. Even if the label prior is unknown, knowledge of the task in itself might be sufficient to make an educated guess about the label prior. For instance, in conversion prediction, the average conversion rate for online advertising is publicly available [4]. For disease prediction, the prevalence rate of a disease is often known and can be used as the label prior. Developing attacks that do not require label prior information is an interesting avenue of exploration for future work.

IX. EXPERIMENTAL SETUP FOR PRIOR WORKS

We describe the experimental setup and evaluation methodology for the prior works used in our experiments. The Unsplit and Model Completion attacks both require a surrogate model. To have a fair comparison, we use the same surrogate model as ExPLoit (see Table I) for both of these attacks.

UnSplit Attack: The UnSplit attack [8] aims to learn the surrogate labels $\{y'\}$ and model parameters $\theta_{g'}$ by minimizing the mean square error loss between the original and the surrogate gradients using the following objective: $\min_{\theta'_g, \{y'_i\}} MSE(\nabla_z L'_i, \nabla_z L_i)$. We use an Adam optimizer with a learning rate of 0.001 and train for 50 epochs.

Model Completion Attack: Model completion attack [9] trains the surrogate model using semi-supervised learning by using a small number of labeled embeddings $\mathcal{D}^l = \{z_i^l, y_i^l\}$ and unlabeled embeddings $\mathcal{D} = \{z_i\}$. We assume that the attacker has 4 labeled examples per class for the vision datasets and 50 examples per class for the Critio dataset. We use the parameters from the original paper [9] to perform semi-supervised learning. We set temperature T=0.8 for sharpening the predictions, $\lambda_u = 50$ as the weight for the loss on the unlabeled dataset and $\alpha = 0.5$ for MixUp. We train all the models for 100 epochs and report the accuracy by using the predictions of this trained model.

Norm-Based Attack: The norm-based attack uses the difference in the magnitude of gradient norms in imbalanced binary datasets to predict the private labels. This difference can be seen clearly from Fig. 6, which shows the distribution of gradient norms $\|\nabla_z L\|_2$ obtained during split learning for different epochs of the conversion prediction model trained on the Criteo dataset. The norm-based attack exploits this difference in the gradient norms and uses it to infer the private labels in split learning. This attack uses a threshold T to classify the examples into positive and negative classes as shown below.

$$y' = \begin{cases} 1 & \|\nabla L_z\|_2 > T \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

We sweep the value of T and pick a value that yields the best accuracy to report the Norm-based attack results in Section V-B. Note that in a real attack setting, the adversary does not have the ability to check the accuracy for different values of T. However, we pick an ideal value of T in our experiments to understand the best possible accuracy that can be obtained with the Norm-based attack.

X. OTHER RELATED WORK

We describe prior works in vertical federated learning that are not directly related to our proposed attack in this section.

A. Privacy-Preserving Training Techniques for Vertical Federated Learning

In addition to split learning, several methods have been proposed to train a model on vertically partitioned private data. These methods can broadly be classified into three categories: 1. Differential Privacy (DP) 2. Multi-Party Compute (MPC) and 3. Trusted Execution Environment (TEE). We discuss solutions in each category and describe their limitations.

Label Differential Privacy: Differential privacy [7] is a principled system for training on a private database that

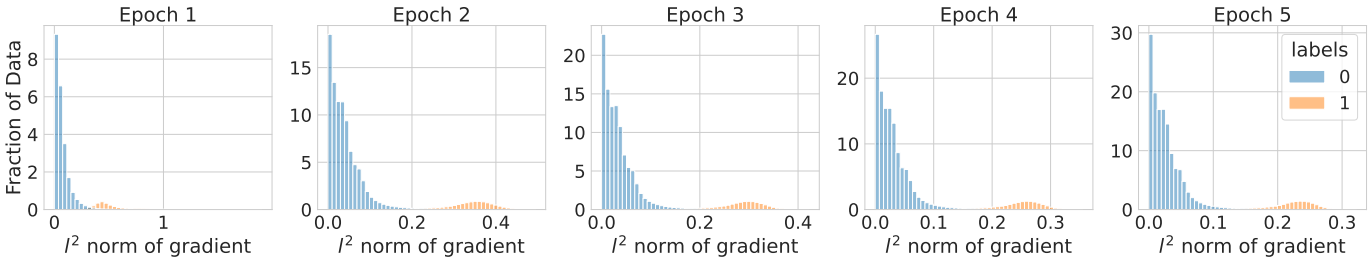

Fig. 6. Distribution of gradient norms $\|\nabla_z L\|_2$ for the conversion prediction task with Criteo dataset. Positive classes are infrequent and typically produce higher gradient norms.

restricts the influence of any single entry of the database on the outcome by adding noise to a query's response. A recent work [10] proposed *Label Differential Privacy (LDP)* to train a model on vertically partitioned data with sensitive labels. LDP relies on a randomized response algorithm to provide a noisy version of the labels to the input owner by defining a probability distribution over the class labels as follows:

$$\Pr[\tilde{y} = \hat{y}] = \begin{cases} \frac{e^{\varepsilon}}{e^{\varepsilon}+K-1} & \text{for } \hat{y} = y \\ \frac{1}{e^{\varepsilon}+K-1} & \text{otherwise} \end{cases} \tag{13}$$

The label owner uses the noisy labels sampled from this distribution and the input data to train a model. To prevent the model from overfitting on the incorrect labels, the authors propose using the Mixup technique [30] to train the model, which provides resilience to label noise. One drawback of this technique is that it allows the input owner to have complete ownership of the model. In contrast, split learning enables the input and label owners to jointly own the model, which might be desirable if the label owner wants to exercise control over the usage of the model.

Multi-Party Compute (MPC): Several works [18], [20], [27] have proposed using cryptographic techniques to enable private computations over distributed data held by multiple parties. These works use a combination of cryptographic primitives such as oblivious transfer [2], [14], [22], garbled circuits [29], secret sharing [6] and homomorphic encryption [21] to train the model. Unfortunately, these methods have significant computational overheads and require multiple rounds of communication between the parties involved. Consequently, even training a simple 2-layer network incurs a 30× overhead [18] compared to training without privacy, making it impractical for training larger networks.

Trusted Execution Environment (TEE): Trusted Execution Environments use hardware enclaves to enable remote computations with confidentiality and integrity. A centrally hosted TEE can be used to train a model on distributed data. The data owners can communicate data securely over an encrypted channel to the trusted enclave. Training is performed while ensuring data confidentiality, and the resulting model is transmitted securely to the data owners. Unfortunately, TEEs have slow memory due to the overheads associated with encryption and integrity checks [12], [19]. Moreover, commercially available TEEs such as Intel SGX [17] and Arm Trustzone [26] are CPU-based and offer less parallelism compared to GPUs. The combination of these two factors results in orders of magnitude [3] increase in training times of models. Additionally, this solution requires specialized hardware, which adds to the cost of implementation.

B. Input Privacy Attacks and Defenses in Split Learning

Recent works have proposed attacks to break input privacy in split learning. The goal of these attacks (a.k.a model inversion attack) is for an adversarial label owner to recover the private inputs $\{x_i\}$ of the input owner using the embedding information $\{z_i\}$ obtained during split learning. A recent work [1] showed that, for simple 1-d time-series signals, the embedding data obtained in split learning might not preserve privacy as it has a high distance correlation with the original input data. Model inversion attacks have also been demonstrated on split learning with more complex datasets in the image domain [24]. To carry out the attack, the adversary uses examples from the input data distribution $\{x'_i\}$ to query the input owner's model and generate embeddings $z'_i = f(x'_i)$. The input and embedding data can be used to train an inversion model f_{inv} that maps the embedding to the input: $\mathcal{Z} \rightarrow \mathcal{X}$. This inversion model can be used to reconstruct the input data using the embeddings during the attack. Note that such attacks require access to the examples from the input data distribution and black-box query access to the input owner's model. In contrast, our label leakage attack does not require black-box access to the label owner's model or access to the ground truth label data. [1] also proposes using additive noise to perturb the embedding to defend against such attacks. This is similar in spirit to our work, which uses gradient noise to deter label leakage attacks.

XI. Conclusion

Split learning has been proposed as a method to train a model on vertically split data while keeping the data private. We investigate the privacy properties of two-party split learning by proposing *ExPLoit* – a label-leakage attack that allows an adversarial input owner to learn the label owner's private labels during split learning. Our key insight is that the attack can be framed as a learning problem by substituting the unknown parameters of the label owner with learnable surrogate parameters. We use the gradient data collected during split learning and a novel loss function to train these surrogate

parameters. Our evaluations on several image-classification tasks and a converstion prediction task show that ExPLoit can leak private labels with near-perfect accuracy of up to 99.53%, proving that split learning provides a negligible amount of label privacy. ExPLoit also outperforms recent prior works, offering up to 67.2% improvement in label leakage accuracy. We also evaluate gradient noise as a defense to improve label privacy. While this provides a reasonable defense for simpler datasets, we find that the utility-privacy tradeoff of this technique is unfavorable for more complex datasets. Our findings in this work underscore the need for better techniques to perform vertical federated learning.

REFERENCES

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

[2] Gilad Asharov, Yehuda Lindell, Thomas Schneider, and Michael Zohner. More efficient oblivious transfer and extensions for faster secure computation. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 535–548, 2013.

[3] Aref Asvadishirehjini, Murat Kantarcioglu, and Bradley A Malin. Ginn: Fast gpu-tee based integrity for neural network training. 2020.

[4] Conor Bond. Conversion rate benchmarks: Find out how your conversion rate compares. https://www.wordstream.com/blog/ws/2019/08/19/conversion-rate-benchmarks, October 2021.

[5] Hao Chen, Kim Laine, and Peter Rindal. Fast private set intersection from homomorphic encryption. In *CCS '17 Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1243–1255. ACM New York, NY, USA ©2017, October 2017.

[6] Daniel Demmler, Thomas Schneider, and Michael Zohner. Aby-a framework for efficient mixed-protocol secure two-party computation. In *NDSS*, 2015.

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

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

[9] Chong Fu, Xuhong Zhang, Shouling Ji, Jinyin Chen, Jingzheng Wu, Shanqing Guo, Jun Zhou, Alex X Liu, and Ting Wang. Label inference attacks against vertical federated learning.

[10] Badih Ghazi, Noah Golowich, Ravi Kumar, Pasin Manurangsi, and Chiyuan Zhang. On deep learning with label differential privacy. *arXiv preprint arXiv:2102.06062*, 2021.

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

[12] Danny Harnik and Eliad Tsfadia. Impressions of intel sgx performance. *Retrieved April*, 19:2020, 2017.

[13] Xinran He, Junfeng Pan, Ou Jin, Tianbing Xu, Bo Liu, Tao Xu, Yanxin Shi, Antoine Atallah, Ralf Herbrich, Stuart Bowers, et al. Practical lessons from predicting clicks on ads at facebook. In *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*, pages 1–9, 2014.

[14] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In *Annual International Cryptology Conference*, pages 145–161. Springer, 2003.

[15] Oscar Li, Jiankai Sun, Xin Yang, Weihao Gao, Hongyi Zhang, Junyuan Xie, Virginia Smith, and Chong Wang. Label leakage and protection in two-party split learning.

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

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

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

[19] Meni Orenbach, Pavel Lifshits, Marina Minkin, and Mark Silberstein. Eleos: Exitless os services for sgx enclaves. In *Proceedings of the Twelfth European Conference on Computer Systems*, pages 238–253, 2017.

[20] Wei Ou, Jianhuan Zeng, Zijun Guo, Wanqin Yan, Dingwan Liu, and Stelios Fuentes. A homomorphic-encryption-based vertical federated learning scheme for rick management. *Computer Science and Information Systems*, (00):22–22, 2020.

[21] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *International conference on the theory and applications of cryptographic techniques*, pages 223–238. Springer, 1999.

[22] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In *Annual international cryptology conference*, pages 554–571. Springer, 2008.

[23] Marcelo Tallis and Pranjul Yadav. Reacting to variations in product demand: An application for conversion rate (cr) prediction in sponsored search. *arXiv preprint arXiv:1806.08211*, 2018.

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

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

[26] Johannes Winter. Trusted computing building blocks for embedded linux-based arm trustzone platforms. In *Proceedings of the 3rd ACM workshop on Scalable trusted computing*, pages 21–30, 2008.

[27] Kai Yang, Tao Fan, Tianjian Chen, Yuanming Shi, and Qiang Yang. A quasi-newton method based vertical federated learning framework for logistic regression. *arXiv preprint arXiv:1912.00513*, 2019.

[28] Yi Yang, Dong Xu, Feiping Nie, Shuicheng Yan, and Yueting Zhuang. Image clustering using local discriminant models and global integration. *IEEE Transactions on Image Processing*, 19(10):2761–2773, 2010.

[29] Andrew C Yao. Protocols for secure computations. In *23rd annual symposium on foundations of computer science (sfcs 1982)*, pages 160–164. IEEE, 1982.

[30] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[31] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. *Advances in Neural Information Processing Systems*, 32, 2019.