A Stealthy Inference Attack on Split Learning with a Split-Fuse Defensive Measure

Sean Dougherty*, Abhinav Kumar*, Jie Hou*, Reza Tourani*, Atena M. Tabakhi†
*Saint Louis University, {sean.dougherty, abhinav.kumar, jie.hou, reza.tourani}@slu.edu
†Washington University in St. Louis, amtabakhi@wustl.edu

2023 IEEE Conference on Communications and Network Security (CNS) | 979-8-3503-3945-1/23/$31.00 ©2023 IEEE | DOI: 10.1109/CNS59707.2023.10288661

***Abstract*—The privacy vulnerabilities and communication inefficiencies of federated learning have motivated the development of the split learning architecture. Google's recent Federated Reconstruction architecture combines federated and split learning architectures into a unified design, aiming to improve communication and computation scalability. While split learning aims to protect the privacy of clients' data, recent work has revealed its vulnerability to malicious adversaries through model poisoning.**

We aim to investigate the privacy aspect of split learning, as an independent architecture or the significant component of federated reconstruction architecture and expose its shortcomings. Different from the existing literature, we illustrate that an *honest-but-curious* adversary can infer the private properties of clients' data without model poisoning or manipulation. We demonstrate the practicality of the property inference attack against split learning using various datasets. To reduce information leakages and protect clients' privacy, we propose *Bundle-Net* architecture as a privacy-preserving distributed learning mechanism and assess its effectiveness in thwarting inference attacks.

***Index Terms*—Distributed learning, privacy, inference attack, split learning, model inversion.**

I. Introduction

Advanced machine learning techniques such as deep learning have recently overcome many challenges that were traditionally deemed to be impossible. Remarkable success stories include mastering the game of Go and generating realistic images [1] or text [2]. The complexity and computational demands of training deep neural networks motivated the deployment of Machine Learning-as-a-Service (MLaaS) platforms by major cloud providers such as Amazon and Google, in which clients share their data with a centralized server for training and prediction purposes. Such centralized MLaaS platforms, however, engender extensive communication overhead. Moreover, with recent privacy laws, such as *General Data Privacy Regulation* (GDPR) and the *California Privacy Rights Act* (CPRA), data owners are more reluctant to share their privacy-sensitive data. In light of such concerns, distributed learning architectures such as federated learning (FL) [3], [4] have emerged, promising to augment clients' privacy and improve communication efficiency. The FL architecture, however, is vulnerable to a range of privacy attacks, including membership inference [5], [6], property inference [7], [8], and data reconstruction attacks [7], [9], [10]. Moreover, sharing local models incurs high communication overhead, particularly in deeper networks with millions of parameters.

This research was partially supported by US NSF awards #2133407.

Recently, split learning (SL) [11], [12] has emerged as a novel distributed learning architecture, aiming to reduce the communication overhead of FL and address its privacy vulnerabilities [13]. The primary premise of SL is to divide a given neural network into two sequential sub-models. The clients will execute the first few layers (*i.e.,* a lightweight sub-model) on their data while the server will execute the follow-up layers (*i.e.,* a more complex sub-model) on the output of the first sub-model. Thus, SL preserves clients' privacy by allowing them to retain the ownership of their private data and reducing the communication overhead by only sharing the intermediate states of the clients' sub-models with the server. Such traits have attracted more attention to SL architecture and led to its adoption in industry and academia [14], [15]. The most notable adoption of SL is Google's recent Federated Reconstruction architecture [16], which is materialized by integrating federated and split learning architectures. This architecture was used in the deployment of a mobile keyboard application with hundreds of millions of clients.

Despite SL's success in various applications, its privacy and security considerations did not receive much attention. Recent work has highlighted the vulnerabilities of SL to inference and data reconstruction attacks. The authors in [17] have shown that an active adversary can hijack the training process by maliciously forging gradients for the client's sub-model to its benefit. More specifically, the malicious adversary orchestrates the proposed attack by poisoning the client's sub-model during the training phase, *i.e.,* hijacking the training using forged gradients. However, the proposed training-hijacking attack, similar to other adversarial model manipulation and poisoning attacks, can be detected using the existing techniques [18], [19]. Moreover, we argue that well-known MLaaS providers like Google do not compromise their clients' model training.

In this work, we aim to assess the privacy guarantees of the SL architecture by considering a more realistic *honest-but-curious* adversary that neither compromises the training process nor poisons the client's sub-model. We will demonstrate how such a non-malicious adversary can infer the private properties of the client's training/test data in the SL setting. In effect, we transform the problem of inferring the features of private data from the output of intermediate layers into a binary classification problem. We orchestrate our proposed attack using various datasets and demonstrate its practicality and accuracy. We will also propose one novel defensive measure to protect

979-8-3503-3945-1/23/$31.00 ©2023 IEEE

clients' privacy: the *Bundle-Net* architecture. Bundle-Net is a distributed MLaaS architecture suitable for distributed computing platforms. It effectively mitigates information leakage through the utilization of model and data-splitting techniques. In the Bundle-Net training process, clients partition their input data based on features and share the representations of the partial feature data with the server. The server then utilizes these transformed data representations for each feature subset to train the model. Finally, clients aggregate these representations to complete the training process, ensuring both high model accuracy and privacy preservation. We will assess the efficacy of the approach in reducing unintentional information leakage and attack accuracy.

In summary, the **novel contributions** of this paper include: *(i)* Demonstration of privacy vulnerabilities in split learning. In particular, we illustrate the property inference and model inversion attacks by an honest-but-curious adversary, who neither has the model's input data nor compromises the training process by poisoning the client's sub-model. *(ii)* Extensive evaluation and analysis of property inference and data reconstruction attacks using multiple datasets and neural network architectures. *(iii)* Design, development, and comprehensive evaluation of Bundle-Net, an architectural solution for protecting clients' privacy against inference attacks by minimizing the unintentional information leakage of split learning.

The paper is organized as follows. Section II reviews the existing literature. Section III outlines our models and assumptions. Section IV presents our attack design and evaluation. We elaborate on the Bundle-Net architecture in Section V and conclude our work in Section VI.

II. Related Work

The privacy promise of SL relies on the fact that *(i)* clients do not share their raw data with the server, and *(ii)* the server is unaware of the client's sub-model parameters. As a result, a malicious server cannot rebuild the client's sub-model via model inversion [9] and hence, can neither reconstruct nor reveal any private feature of the training/test samples [11]. However, a few recent works have shown successful privacy attacks against SL [17], [20]. In particular, the high distance correlation between the raw input data and the cut layer activation can lead to the reconstruction of 1-dimensional data [20]. The authors in [21] employed a model inversion attack to reconstruct the user's private data in an edge–cloud collaborative system. While data reconstruction was shown possible, the authors only used the simple MNIST dataset, which is not representative of the existing complex data.

In [17], the authors exposed the privacy vulnerability of the SL architecture through property inference and reconstruction attacks. In the proposed attack, the malicious server hijacks the training process to drive the client's sub-model to an insecure state. The attack starts during the training phase, in which the malicious server completes the forward pass of its sub-model on the client's smashed data, *i.e.,* output of the cut layer. In parallel, aiming to mimic the client's sub-model, the server trains a pilot model and its inverse function using a publicly available dataset of the same distribution. The server uses a discriminator model to derive a set of forged gradients and construct an adversarial loss function to maliciously modify the client's sub-model. During inference, the server uses the smashed data and the inverse of the pilot model to reconstruct the client's private data. Note that the active adversary in [17] can hijack the training process and manipulate the client's sub-model. Moreover, the authors evaluated the impact of the attack on the target model in the white-box setting, where the adversary has complete knowledge of the target model.

TABLE I: Notations Used.

Notation	Description
$\mathcal{M}(.)$	Target model
$\mathcal{F}_c(.)$	Client's sub-model of $\mathcal{M}(.)$
$\mathcal{F}_s(.)$	Server's sub-model of $\mathcal{M}(.)$
K	Number of shadow models
$\widehat{\mathcal{F}_c}^i(.)$	The i-th shadow model (total of K)
$\overline{\mathcal{F}_c}(.)$	The aggregated shadow model
$\mathcal{C}(.)$	Attack model (binary classifier)
D_C^{Target}	Client's private dataset for target model
$D_{\mathcal{A}}^{Target}$	Attacker's Shadow dataset for shadow model
$D_{\mathcal{A}}^{Attack}$	Attack dataset (generated by shadow models)
d_{priv} ($\in D_C^{Target}$)	A private training/test data sample
d_i ($\in D_{\mathcal{A}}^{Target}$)	A public training/test data sample
y	Private property of a data sample
$[a]$	Smashed data generated by shadow model

In contrast, in this work, we consider a passive and honest-but-curious adversary that neither *(i)* compromises the training process *(ii)* nor poisons the client's sub-model. More specifically, our adversary remains passive concerning the client's sub-model and does not tamper with the training process. In addition, we assume that the adversary has only partial information about the model architecture and its parameters. Finally, our proposed property inference attack targets inferring the properties of both the training and inference data.

III. Models and Assumptions

A. System Model

We consider a computing ecosystem, such as pervasive edge computing [22], which offers machine learning-as-a-service (MLaaS). Cloud providers, such as Amazon and Google, offer MLaaS platforms equipped with model training and prediction interfaces to their clients for compute-intensive ML applications, *e.g.,* smartphones offloading Augmented/Virtual Reality applications to mobile edge-cloud systems. We particularly consider the split learning MLaaS architecture, in which the original model ($\mathcal{M}(.)$) is horizontally (layer-wise) divided into $\mathcal{F}_c(.)$ and $\mathcal{F}_s(.)$ sub-models, where the predictions are made by $\mathcal{M}(d_i) = \mathcal{F}_s\big(\mathcal{F}_c(d)\big)$ and d is a training/test sample. Following the conventional split learning MLaaS, we assume the client owns the first few layers of the model, *i.e.,* $\mathcal{F}_c(.)$, while the server hosts the rest of the layers, *i.e.,* $\mathcal{F}_s(.)$. The proportions of the two sub-models, *i.e.,* the cut layer, can be negotiated between the client and the server based on various factors,

such as the client's expected privacy requirements or computing resources. Finally, we consider the architecture proposed in [16] for scenarios where multiple clients collaboratively train a shared model. Our notation is summarized in Table I.

B. Threat Model and Security Assumptions

In this work, we consider the adversary to be the back-end server, hosted either on the cloud or at the edge, responsible for running the $\mathcal{F}_s(.)$ sub-model for the client during model training and inference. The adversary constructs the attack during the training phase and orchestrates it during model inference. The adversary in this scenario is honest-but-curious, in that it attempts to infer information about the client's sub-model ($\mathcal{F}_c(.)$) or data while adhering to the standard model training and inference processes, without maliciously sabotaging the training process or poisoning the client's sub-model. Such an attacker represents scenarios where the MLaaS is offered by major cloud providers, who would not risk their reputation by intentionally poisoning their clients' models [23]. The rationale for choosing such an adversary, which contrasts the powerful attacker presented in [17], is to demonstrate that breaching clients' privacy in split learning neither requires a powerful attacker nor demands model poisoning. Moreover, existing model poisoning detection techniques [18], [19] can detect active adversaries that perform training-hijacking attacks, undermining the practicality of the attack in [17].

We consider a scenario where the client possesses a private training dataset, denoted as D_C^{Target}. We assume that the adversary has access to a public shadow dataset, denoted as $D_{\mathcal{A}}^{Target}$, which is assumed to follow the same underlying distribution as D_C^{Target} but has no overlapping data points ($D_C^{Target} \cap D_{\mathcal{A}}^{Target} = \emptyset$). Note that there are various approaches for building shadow datasets [24] and the adversary's access to the shadow dataset is a common assumption [6], [17], [24]. It is worth noting that our attack methodology remains effective even when the adversary possesses partial knowledge of the client's private dataset [14].

In this work, we assume that the adversary possesses partial knowledge of architecture, denoted as $\mathcal{F}_c(.)$, of the neural network under consideration. Specifically, the adversary is aware of the network's structure and design but does not have any knowledge of its specific weight values. This is a fair assumption as there are established methods available for the adversary to extract the architecture of neural networks [25]. In our framework, we consider that the client can request the parameters of $\mathcal{F}_s(.)$ from the edge server. This allows the client to verify the accuracy and effectiveness of the overall model, denoted as $\big(\mathcal{F}_s(\mathcal{F}_c(\cdot))\big)$. Moreover, this provision enables the client to detect and potentially mitigate any model poisoning attacks that may be present within the system [17].

IV. Attack Methodology

In this section, we discuss our attacks foundation and further evaluate their efficacy under various configurations.

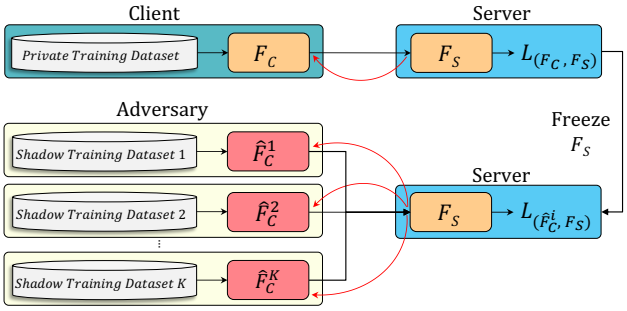

Fig. 1: To generate the attack dataset, the adversary freezes its sub-model ($\mathcal{F}_s$) to train a set of shadow models $(\{\widehat{\mathcal{F}_c}^1(.), \cdots, \widehat{\mathcal{F}_c}^K(.)\})$ for imitating the behavior of $\mathcal{F}_c$.

A. Attack Objective and Overview

In split learning, the server (hereafter adversary) actively participates in the training of the sub-model ($\mathcal{F}_s(\cdot)$) by utilizing the smashed data provided by the client. However, this participation raises concerns regarding sensitive information leakage from the client's private dataset [17]. The correlation between the model's input, denoted as d_{priv}, and the smashed data, denoted as $\mathcal{F}_c(d_{priv})$, amplifies the risk of such leakage. The server's involvement in the training process can inadvertently expose confidential information from the client's private dataset, posing a potential privacy breach. As such, we propose an inference attack, in which the adversary's primary objective is to infer the properties of private training and inference data – those data features that are seemingly unrelated to the model's primary goal [26], [27], *e.g.,* inferring the "gender" attribute from a multi-attributed face image by exploiting an inference attack on a deep learning model that was initially trained to predict the "smiling" attribute. The adversary's secondary objective is to reconstruct the client's data instance using only the intermediate representations obtained from the layers of the deep learning models. Different from recent works [17], [26], [27], our attack targets the live inference data that clients provide during the inference phase. We also show the impact of our attacks on the training data for completeness.

B. Attack Modeling

Our proposed attack follows a two-step process. In the *target model shadowing* step, the adversary creates a set of shadow models $\big(\{\widehat{\mathcal{F}_c}^1(.), \cdots, \widehat{\mathcal{F}_c}^K(.)\}\big)$ to mimic the behavior of the client's sub-model $\mathcal{F}_c(.)$; K is the number of shadow models and each shadow model is trained on a different shadow dataset. In the *attack training and orchestration* step, the adversary trains an attack model $\big(\mathcal{C}(.)\big)$ – a binary classifier that acts similar to discriminator models – to identify whether d_{priv} features a specific property or not. For attack orchestration, the adversary obtains the client's smashed data during the inference phase and uses $\mathcal{C}(.)$ to infer the desired properties or reconstruct the client's data. Here, we elaborate on the attacks.

1) Target Model Shadowing: Considering that $\mathcal{F}_c$ is the client's private sub-model and not publicly available, the adversary trains a set of shadow models to mimic its behavior

(Fig. 1). The adversary executes this step in parallel with the client's model training over multiple iterations. In particular, the adversary trains the following neural networks:

- $\mathcal{F}_s$, which is the sub-model requested by the client to complement $\mathcal{F}_c$ in training the final target model, such that $\mathcal{M}(.) = \mathcal{F}_s\big(\mathcal{F}_c(.)\big)$. The client generates the smashed data, $\mathcal{F}_c(D_C^{Target})$, as the training data of the $\mathcal{F}_s$ sub-model.
- Shadow models, $\widehat{\mathcal{F}_c}(.) = \{\widehat{\mathcal{F}_c}^1(.), \cdots, \widehat{\mathcal{F}_c}^K(.)\}$, which mimic the behavior of the client's private sub-model ($\mathcal{F}_c(.)$). The adversary trains these shadow models using the shadow dataset ($D_{\mathcal{A}}^{Target}$).

The target model shadowing process starts with the adversary performing one training iteration of $\mathcal{F}_s$ on the client's smashed data, including back-propagation. At this stage, the adversary freezes $\mathcal{F}_s$ and proceeds with the forward pass of $\widehat{\mathcal{F}_c}(.)$ using the shadow dataset $D_{\mathcal{A}}^{Target}$ up to the cut layer and the already trained $\mathcal{F}_s$. The rationale for freezing $\mathcal{F}_s$ after each training iteration is to influence the training of $\widehat{\mathcal{F}_c}(.)$ such that its weights and biases find a local minima of loss function that is similar to the local minima of $\mathcal{F}_c$. Thus, helping reduce the distance correlation between $\mathcal{F}_c$ and $\widehat{\mathcal{F}_c}(.)$ without adversarial manipulation of $\mathcal{F}_c$. The adversary then completes one training iteration of $\widehat{\mathcal{F}_c}(.)$ by back-propagating the gradients from $\mathcal{F}_s$ to $\widehat{\mathcal{F}_c}(.)$. The adversary iteratively trains the target's $\mathcal{F}_s$ and $\widehat{\mathcal{F}_c}(.)$ until the model converges. Finally, the adversary trains K shadow models and extracts the aggregated shadow model – $\overline{\mathcal{F}_c}(.)$ – by weighted averaging of parameters:

$$\overline{\mathcal{F}_c}(.) = \sum_{p=1}^{P} w_p \widehat{\mathcal{F}_c}^p(.), \;\; \sum_{p=1}^{P} w_p = 1.$$

While various aggregating strategies can be applied to combine shadow models, we averaged the models' parameters as done in federated averaging [28]. In our experiments, we assigned equal weight to all shadow models due to the homogeneity of the datasets.

2) Property Inference Attack Training and Orchestration: The adversary uses the aggregated shadow model $\overline{\mathcal{F}_c}(.)$ for generating attack datasets and training the attack model(s) – a collection of models, one per data property of interest. To generate $D_{\mathcal{A}}^{Attack}$, the adversary queries the resultant aggregated shadow models using the shadow test dataset. Per Section III-B, the shadow and client datasets are disjoint with no intersection, resulting in the worst-case scenario for the adversary. Formally, we define the attack dataset, $D_{\mathcal{A}}^{Attack}$:

$$D_{\mathcal{A}}^{Attack} = \Big\{ \big([a], y\big) : [a] = \overline{\mathcal{F}_c}(d), \forall d \in D_{\mathcal{A}}^{Target} \Big\},$$

in which $[a]$ is the smashed data generated by aggregated shadow models on the shadow dataset and y is the label of the corresponding property; $y = 1$ represents the presence of desired property in the input and $y = 0$ indicates its absence.

The adversary uses the attack dataset, $D_{\mathcal{A}}^{Attack}$, to train the attack model, $\mathcal{C}(.)$. We define $\mathcal{C}(.)$ as a binary classifier, aiming to discriminate between the data samples that feature the desired property and those that do not. For the attack orchestration during inference, the client sends $\mathcal{F}_c(d_{priv})$ to the server for prediction. At this stage, the server executes the requested service as $\mathcal{F}_s\big(\mathcal{F}_c(d_{priv})\big)$ and returns the result to the client. At the same time, the server executes the attack model, $\mathcal{C}\big(\mathcal{F}_c(d_{priv})\big)$, to infer d_{priv} properties without direct access to it. If the training data features various properties, the adversary needs to create a unique $D_{\mathcal{A}}^{Attack}$ per property for training the corresponding attack model. For instance, we generated multiple attack datasets and models for MNIST, one per property, *e.g.,* samples with horizontal and vertical lines.

3) Data Reconstruction Attack: We also performed the data reconstruction attack on the target model in the split learning setting, in which the adversary aims to reconstruct the raw input data by observing only the output of the client's sub-model $\mathcal{F}_c(.)$. Considering the adversary's access to the smashed data from $\mathcal{F}_c(.)$, the data reconstruction attack can be realized using the shadow model $\widehat{\mathcal{F}_c}(.)$. We adopt the implementation in work [21] to perform the model inversion attack in a splitting learning environment. First, for any input sample x_c from the client dataset $D_{\mathcal{C}}^{Target}$, the adversary receives the feature representation $\mathcal{F}_c(x_c)$ from the client's sub-model. Second, the adversary creates one noise input x_r with the initial random values and calls the pre-trained shadow models to generate the hidden features $\widehat{\mathcal{F}_c}(x_r)$. The adversary then performs the back-propagation over the parameters of $\widehat{\mathcal{F}_c}(.)$ to derive the gradients of input x_r and update the input data using gradient descent algorithms. The back-propagation process will be repeated until the input sample is optimized to achieve the minimum distance error between $\mathcal{F}_c(x_c)$ and $\widehat{\mathcal{F}_c}(x_r)$. The optimized input is then derived as the reconstructed data for the client's sample.

C. Attack Evaluation

1) Dataset: We use the following datasets:

- MNIST [29] dataset consists of 70,000 images of handwritten digits. Each digit image has been annotated with three primary attributes: *Loop*, *Horizontal* line, and *Vertical* line – digits $\{2, 4, 5, 7\}$ have "Horizontal" property, digits $\{1, 4, 7, 9\}$ have "Vertical" property, and digits $\{0, 6, 8, 9\}$ have "Loop" property. For our experiment, we trained a binary classifier as the target model to predict the presence of the "Loop" attribute in an image. Additionally, we trained two attack models to infer the presence of the "Vertical" and "Horizontal" lines, respectively.
- UTKFace [30] dataset comprises over 20,000 face images with annotations for age, gender, and ethnicity. These images exhibit a wide range of variations in pose, facial expression, illumination, occlusion, and resolution. In our attack evaluation, the binary target classifier focuses on predicting "Race" while the attack aims to infer the "Gender" property.
- CelebA [31] dataset contains 202,599 face images of celebrities with annotations of 40 binary labels indicating the presence of attributes in each image, such as hair color, gender, and age. Note that most of these attributes are highly imbalanced, making it challenging to train accurate classifiers. For instance, the attribute "Eyeglasses" is present in only 6.5%

of the images, while the attribute "Wearing Hat" is present in only 4.8% of the images. Following the data configuration in [32], we selected and combined the three most balanced attributes, namely "Heavy Makeup," "Mouth Slightly Open," and "Smiling," referred to as HMS. We then concatenated these attributes to create eight classification classes as labels for our target model training. Meanwhile, our attack model aims to infer the combined attributes, "Young" and "Male", and assigns them to four classification classes, denoted as YM.

We applied the same preprocessing pipeline to all datasets, resulting in the construction of three types of data: D_C^{Target}, $D_{\mathcal{A}}^{Target}$, and $D_{\mathcal{A}}^{Attack}$. Initially, we randomly divided each dataset into two equal-sized and non-overlapping subsets: D_C^{Target} and $D_{\mathcal{A}}^{Target}$. Furthermore, both D_C^{Target} and $D_{\mathcal{A}}^{Target}$ were split into training and test sets in a 5:5 ratio to evaluate the performance of the target classification and attack models. Lastly, we selected a validation set by sampling 10% of the images from each training set to identify the optimal deep learning model during training. To validate the robustness of the pipeline, we repeated the above data generation process 10 times, each time using different random seeds.

2) Models Specification and Experiment Setup: We evaluated our attack in a scenario where the client sends one data point to the server at a time. However, our attack remains applicable if the client decides to send a batch of data at once. To ensure consistency across datasets with varying input sizes, we resized input images to a uniform size of 64×64 before the training phase. In our experiments, we utilized a convolutional neural network (CNN) that consisted of three convolutional blocks, each containing a convolution layer with kernel sizes of 32, 64, and 128, followed by a ReLU layer, and a MaxPooling layer. Furthermore, we incorporated two fully connected dense layers (with a hidden layer comprising 512 neurons) to perform target classification. For the purpose of comparison in target classification and property inference attacks, we also employed the ResNet-18 architecture alongside the CNN. We define the attack model $\mathcal{C}(.)$ as a shallow feed-forward neural network (with one hidden layer comprising 128 neurons) that infers the presence of properties in the input data. We conduct experiments on the target model by splitting the network at four different layers, including three splits at the convolutional layers (*i.e.,* $\big(\mathcal{F}_c(\cdot)\big)$ owns 20%, 40%, or 60% of the layers) and one split at the fully connected layer. Additionally, we evaluate the classification accuracy of the target model and the effectiveness of the attack for these splits.

3) Results Analysis: We first present the training and test accuracy of the target and shadow models (Fig. 2). The shadow models demonstrate similar classification accuracy in both training and test scenarios when compared to the target models across all datasets, architectures, and cut layers. The averaged test classification accuracy for target/shadow models, considering different cut layers, is 98.97%/99.08% (MNIST), 82.65%/83.07% (UTKFace), and 63.22%/62.68% (CelebA) for the CNN model, and 99.27%/99.30% (MNIST), 80.27%/79.86% (UTKFace), and 69.58%/69.43% (CelebA) for

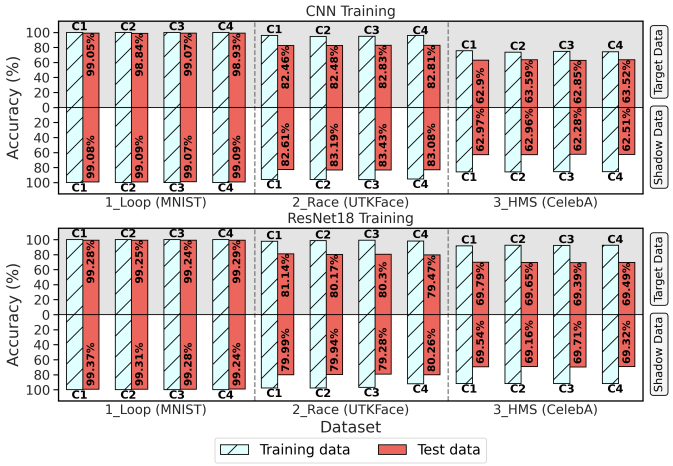

Fig. 2: The training and testing accuracy of the target and shadow models in split learning (SL), considering different datasets and splitting proportions. The classification performance of both the target and shadow models is assessed at four cut layers (C1, C2, C3, C4).

ResNet18, respectively. The results also demonstrate the complexity of the three datasets. Notably, the CelebA dataset exhibits lower test accuracy for both the target and shadow models compared to the training accuracy. Conversely, the MNIST dataset shows nearly identical training and test accuracy for both the target and shadow models.

Fig. 3 showcases the effectiveness of our proposed attack in accurately inferring private properties of the input data, both in training and test scenarios, across all datasets and cut layers for the CNN and ResNet architectures. First, shifting the cut layer towards the last layer results in a lower attack accuracy, irrespective of the data or architecture – particularly when $\mathcal{F}_c(.)$ includes a dense layer. The outcome is partly due to the large number of functionally equivalent neural networks that can be generated by flattening the neurons – by reordering the neurons without changing weights – which misleads the adversary [11]. Second, we observed that the attack achieves greater success rates on less complex datasets. For example, when using feature representation from the first layer of the CNN to infer the "vertical" attribute in MNIST data, the attack accuracy reaches 98.25%. In contrast, the accuracy for inferring "Gender" in UTKFace is 82.54%, and 69.61% for inferring "YM" in CelebA. ResNet18 achieves similar attack performance by leveraging feature representation derived from earlier layers in the deep learning architecture. The results closely follow the state-of-the-art property inference attack against deep learning [32]. Finally, one can observe that the accuracy of the attack on test data closely follows the attack accuracy of training data, which shows the robustness of the underlying property inference attacks.

4) Data Reconstruction Attack: We evaluated data reconstruction attacks on split learning using the structural similarity index measure (SSIM) and the mean squared error (MSE) metrics (Fig. 4). Due to space limitations, we will focus our presentation solely on the results for UTKFace and CelebA. One can observe that reconstructing the input data solely based

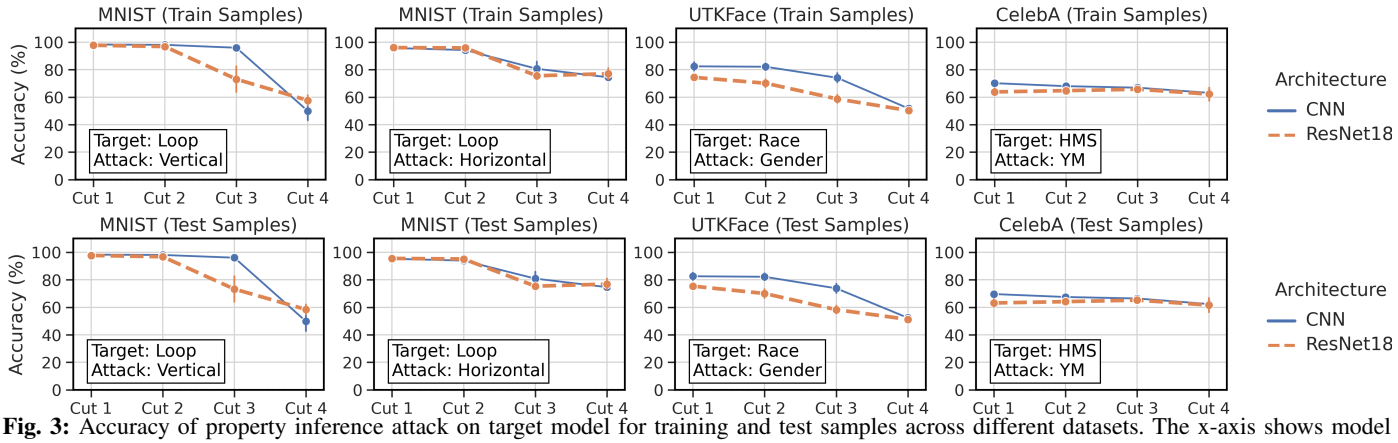

Fig. 3: Accuracy of property inference attack on target model for training and test samples across different datasets. The x-axis shows model splitting at different layers. The attack accuracy consistently drops on training and testing samples as the ratio of the model split increases.

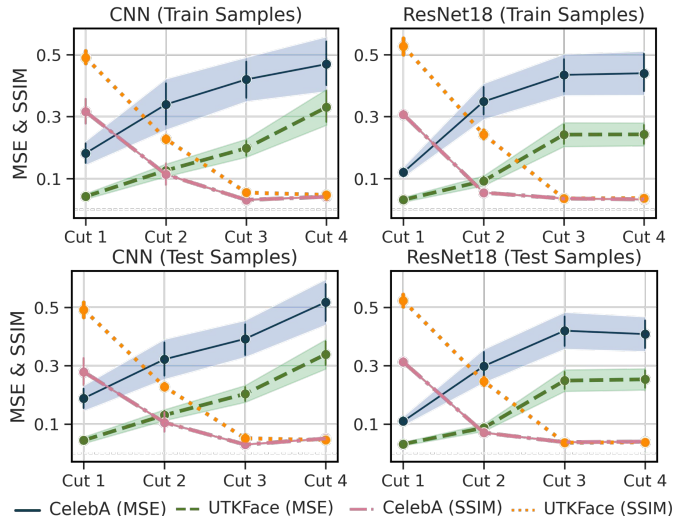

Fig. 4: MSE & SSIM of model inversion attack on Target model for training and test instances using shadow training. The attack efficacy decreases as the cut layer approaches the model's last layer.

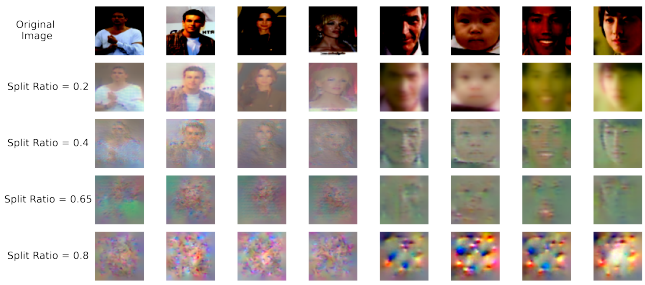

Fig. 5: Model inversion attack on CelebA and UTKface data using shadow training in split learning. Columns 1-2: Training images of CelebA data in target model training. Columns 3-4: Testing images of CelebA data in target model training. Columns 5-6: Training images of UTKface data in target model training. Columns 7-8: Testing images of UTKface data in target model training.

on the activations of the cut layer yields higher precision when using the output of the convolution layers compared to reconstructing from the fully connected layer. The earlier model splits result into lower errors (*i.e.,* MSE) and higher similarities (*i.e.,* SSIM). Furthermore, the results indicate that the complexity of the dataset influences the effectiveness of the attack. Specifically, data reconstruction of lower complexity datasets, such as UTKFace, leads to a more potent attack compared to the more complex CelebA dataset. This observation aligns with our findings from the property inference attack. Fig. 5 showcases the quality of the reconstructed data samples from both the UTKFace and CelebA datasets. The visual representation of the reconstructed data samples complements our evaluation using the MSE and SSIM metrics, confirming the negative impact of shifting the cut layer towards the models' end on attack accuracy.

V. Bundle-Net Architecture

Besides being vulnerable to property inference and data reconstruction attacks, the split learning architecture also exposes the intermediate layer outputs of the model to the server. Moreover, the privacy advantage of split learning becomes more evident when the client's sub-model ($\mathcal{F}_c(.)$) includes a greater number of layers, despite the potential increase in computational demands on the client side. To address these shortcomings, we introduce *Bundle-Net,* a distributed learning architecture designed to mitigate the unintentional information leakage associated with split MLaaS. We elaborate on the design of Bundle-Net and evaluate its effectiveness in countering inference attacks.

A. Bundle-Net Architectural Design

The fundamental concept behind the Bundle-Net architecture combines horizontal model splitting with vertical data feature partitioning. For horizontal model splitting, Bundle-Net first splits the model in a layer-wise fashion, *i.e.,* horizontally, into three sequential sub-models: $\mathcal{F}_c^1$, $\mathcal{F}_s$, and $\mathcal{F}_c^2$. The client retains $\mathcal{F}_c^1$ and $\mathcal{F}_c^2$ to perform data transformation and label prediction, while the server handles the intermediate sub-model, $\mathcal{F}_s$, responsible for feature extraction. For data partitioning, Bundle-Net divides the features of each data into T chunks. Processing these chunks involves vertically mapping the server's sub-model ($\mathcal{F}_s$) into T feature-wise sub-models $\mathcal{F}_s^i$ (for all i in T). This enables parallel feature extraction for model prediction (see Fig. 6). The Bundle-Net architecture

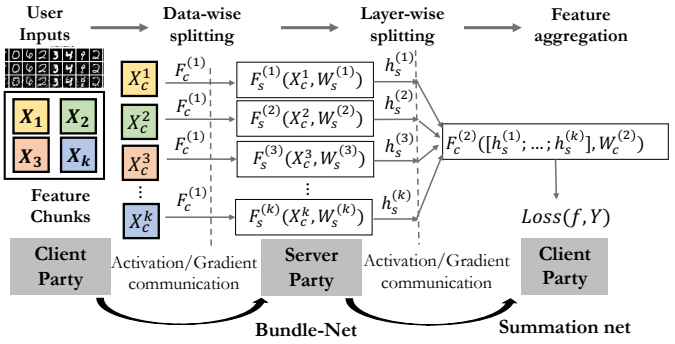

Fig. 6: Bundle-Net architecture for distributed MLaaS.

empowers the client to regulate the information they disclose to each server by assigning each data chunk to a specific server that possesses the corresponding sub-model $\mathcal{F}_s^i$. As a result, unintended information leakage is substantially reduced, while minimizing computational burden on the clients. Similar to various existing security frameworks, such as multi-party computation, we operate under the assumption that the majority of servers are trustworthy and do not engage in collusion.

1) Bundle-Net Training and Inference: The training phase initiates with the client and servers loading the corresponding sub-models of a predefined architecture with random parameters. In this phase, assuming an image-type input, the client partitions the input data into T chunks, where each chunk represents a subset of features. Subsequently, the client applies $\mathcal{F}_c^1$ to each chunk to generate the transformed data, which is then transmitted to the servers' models. On receiving its unique smashed data (chunk), each server performs the forward pass to the corresponding intermediate sub-model, *i.e.,* $\mathcal{F}_s^i$ for the i^{th} data chunk, for feature representation learning. Finally, the participating servers return the outputs of their respective sub-models back to the client. The client then aggregates the servers' output, executes $\mathcal{F}_c^2$ for label prediction, and calculates the loss (Fig. 6). Several aggregation functions, including average, sum, and concatenation, have been explored. In this study, we employ mean aggregation of the servers' outputs for label prediction. The client then initiates the back-propagation process on $\mathcal{F}_c^2$ by deriving and splitting the input's gradients for each intermediate sub-models, $\mathcal{F}_s^i$ $(\forall i \in T)$. Upon receiving the gradients, the servers continue the back-propagation using their respective sub-models and send their derived input gradients back to the client to be processed by $\mathcal{F}_c^1$. All the weights of $\mathcal{F}_c^1$, $\mathcal{F}_s^i$ $(\forall i \in T)$, and $\mathcal{F}_c^2$ are then updated based on the collected gradients using an SGD optimizer. It concludes the first training iteration, wherein each of the server's sub-models will be trained on a particular data chunk, resulting in similar architectures but different parameters. Furthermore, Bundle-Net allows the client to enhance classification accuracy by incorporating the representation of original features, which are not shared with the servers, into the aggregation function through an independent shortcut layer.

During the inference phase, the client executes $\mathcal{F}_c^1$ and divides its output into the respective data chunks. Each chunk, along with the identifier of the corresponding intermediate sub-model is then sent to a designated server. The selected servers run the requested sub-models and return their smashed data to the client. Finally, the client aggregates the received tensors into the smashed data for label prediction using $\mathcal{F}_c^2$.

2) Data Splitting Strategy: We consider three data-splitting strategies to partition the features of each data sample into T chunks, including serious overlapping (SO), non-overlapping (NO), and sparse non-overlapping (SPO). The serious overlapping strategy divides data such that a subset (or all) of its features (*i.e.,* pixels) belongs to two data chunks; hence, processed by two intermediate sub-models. The non-overlapping strategy divides the data into equal-size chunks, in which each data feature belongs to only one data chunk. Finally, the sparse non-overlapping strategy performs sub-sampling of data features in a non-overlapping manner, which leads to only a subset of data features being processed by servers. In our evaluation, we used the three strategies under two extreme configurations – generating four and sixteen chunks for each image data:

- *Four* and *sixteen* serious overlapping chunks (*i.e., SO-K*, where K represents the number of feature chunks) with sizes of 1/2 and 1/8 features, respectively. Each chunk is derived by splitting the input data horizontally and vertically into one-half and one-quarter.
- *Four* and *sixteen* non-overlapping chunks (*i.e., NO-K*) with sizes of 1/4 and 1/16 features, respectively. We split data into four and sixteen equal chunks, fully offset from one another.
- *Four* and *sixteen* sparse non-overlapping chunks (*i.e., SNO-K*) with sizes of 1/16 and 1/64 features, respectively. The input data is split into four and sixteen non-overlapping equal pieces with each chunk located centrally in each piece.

3) Bundle-Net Privacy and Efficiency Implications: The Bundle-Net architecture offloads the majority of compute-intensive layers of neural networks to the servers, suitable for resource-constrained devices. Moreover, Bundle-Net protects the client's input data and the model prediction by virtue of running the $\mathcal{F}_c^1$ and $\mathcal{F}_c^2$ on the client. Finally, by splitting the input data into multiple smaller chunks, Bundle-Net drastically reduces the accuracy of the property inference attack as each server only observes a smaller chunk of the input data. Our proposed data splitting is a particular case of data cropping where we intentionally or randomly drop the feature block of a data sample. The truncated data inputs will be less informative for the corresponding machine-learning tasks. Note that our data splitting strategy can be generalized to any number of structured chunks without limiting them to four and sixteen chunks. The data chunking strategy can also be extended to random sub-sampling by varying factors to get unstructured feature chunks sent to intermediate sub-models.

B. Bundle-Net Evaluation

1) Bundle-Net Attack Modeling: The proposed inference attack against the Bundle-Net architecture closely follows the methodology described in Section IV-B. In particular, we consider the adversary to be one of the servers, *e.g.,* server i who own an intermediate sub-model $\mathcal{F}_s^i$, participating in

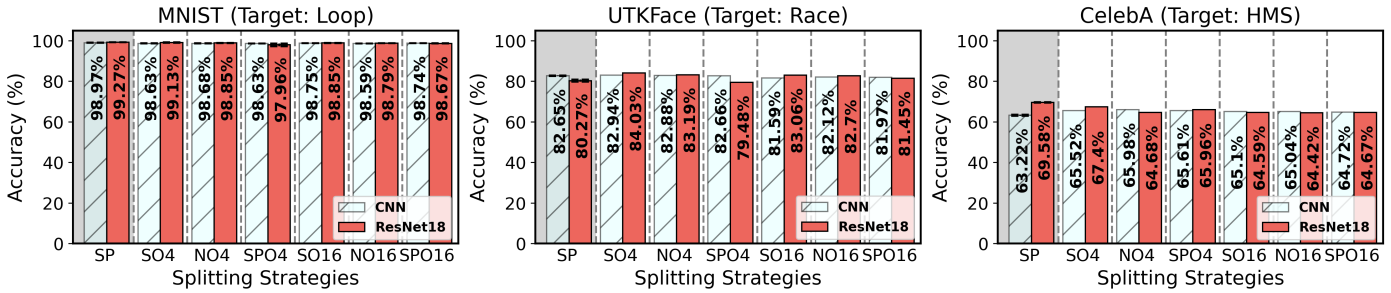

Fig. 7: Target classification performance using Bundle-Net across all splitting strategies (SO4, NO4, SPO4, SO16, NO16, SPO16). The first two bars labeled **SP** represent the accuracy of split learning.

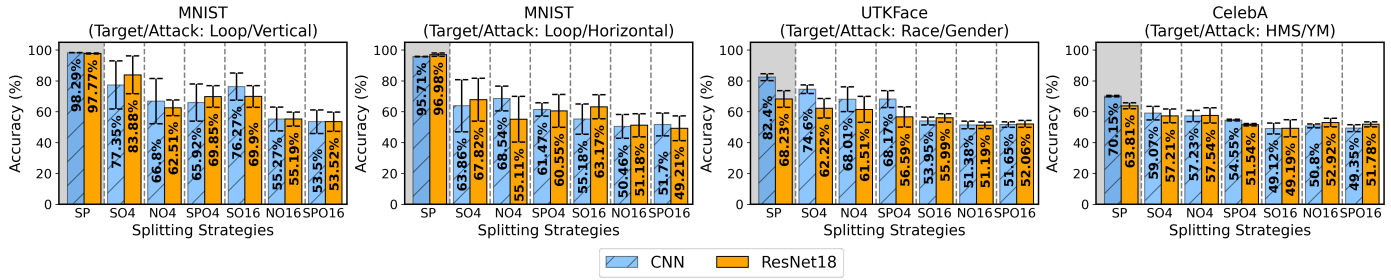

Fig. 8: Accuracy of property inference attacks on Bundle-Net using test data from all datasets and data splitting strategies (SO-4, NO-4, SPO-4, SO-16, NO-16, SPO-16). The first two bars labeled **SP** represent the attack accuracy of split learning.

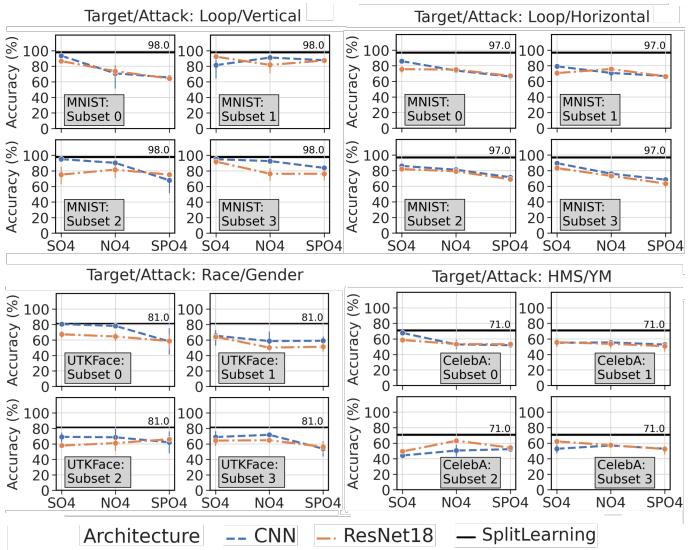

Fig. 9: Attack accuracy on BundleNet's sub-models, *i.e.,* individual data partitions. The dashed and dotted lines represent the attack accuracy on CNN and ResNet18 models using BundleNet learning, respectively. The solid line shows the attack accuracy of split learning.

the training of the server's sub-model, *i.e.,* $\mathcal{F}_s$. During model training, the adversary freezes its sub-model ($\mathcal{F}_s^i$) and uses the shadow dataset to train a set of shadow sub-models, $\{\overline{\mathcal{F}_s}^j : \forall j \in T, j \neq i\}$, aiming to replace all the missing intermediate sub-models, which are owned by other servers. Therefore, the attacker has access to $\mathcal{F}_s^i$ and the set of shadow models, which constitute $\overline{\mathcal{F}_s}$; a sub-model that mimics $\mathcal{F}_s$. Following Section IV-B, the attacker uses $\overline{\mathcal{F}_s}$ for training the attack model to process the received chunk of the client's data and infer the expected properties.

2) Bundle-Net training for classification: We first evaluate the target classification accuracy using Bundle-Net architecture over the six different data splitting approaches (*i.e.,* SO-4, NO-4, SPO-4, SO-16, NO-16, and SPO-16), following the same evaluation metrics as we used to assess the performance of split learning for a fair comparison. Fig. 7 suggests that the Bundle-Net architecture achieves comparable classification performance to split learning. This holds true across all three datasets and split strategies, although the accuracy slightly diminishes as the number of adopted features decreases. For instance, when employing ResNet18 for UTKFace classification, the accuracy of the attack drops from 84.03% for SO-4 to 83.19% for NO-4, and further decreases to 79.4% for SPO-4. Similarly, the splitting strategies (SO-16, NO-16, and SPO-16) exhibit corresponding decreases in accuracy, ranging from 83.06% to 81.45%. Furthermore, when using the ResNet18 model, the accuracy decreases from 99.27% for split learning to a range of 99.13% for SO-4 and 97.96% for SPO-4 on the MNIST dataset. On the CelebA dataset, the accuracy decreases from 69.58% for split learning to 67.4% for SO-4 and further to 65.96% for SPO-4. The CNN models also demonstrate similar trends, although with less pronounced differences. These results highlight the feasibility of Bundle-Net learning across datasets.

3) Property inference attack using Bundle-Net: Fig. 8 summarizes the performance of the property inference attack against the Bundle-Net architecture using test data across various data splitting strategies (*i.e.,* SO, NO, and SPO) and chunk numbers (*i.e.,* 4 and 16). While the attack performance on the training data follows the same trend, we do not present them due to space limitations. When considering the MNIST dataset with the CNN architecture, utilizing Bundle-Net training significantly reduces the average attack accuracy on the "Vertical" attribute. It drops from 98.29±0.03% in split learning to a range between 77.35±21.13% and 53.5±10.3%, depending on the specific data splitting strategy. Similarly, for

inferring the "Horizontal" attribute, the average attack accuracy decreases from 97.71±0.05% in split learning to a range of 68.54±10.84% and 49.21±10.79% across all strategies. We observed a similar trend in the UTKFace and CelebA datasets. Specifically, on the UTKFace dataset, the attack accuracy decreases from 82.5±2.63% for CNN in split learning to a range of 74.6±3.75% to 51.65±2.34% across all strategies with Bundle-Net learning. The accuracy of property inference attacks on ResNet18 models consistently drops using Bundle-Net learning. Fig. 9 visually demonstrates how Bundle-Net reduces the success rate of property inference attacks for each sub-model of the server, using the representation of sub-features under different data splitting strategies (SO-4, NO-4, and SPO-4). These results highlight the effectiveness of Bundle-Net in mitigating the risk of privacy leakage.

VI. Conclusion

In this work, we exposed the privacy vulnerability of the novel split learning architecture against an honest-but-curious adversary. In particular, we have demonstrated that a passive adversary can identify attributes of the client's private data without compromising the training process or manipulating the client's model. To address this privacy leakage, we introduced Bundle-Net architecture and demonstrated its efficacy in reducing the information leakage and practicality of the property inference attack. In the future, we will consider an architecture-agnostic adversary, with no prior knowledge of the architecture, and investigate the impact of noisy and non-IID shadow datasets on the attack performance.

References

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems 27*. Curran Associates, Inc., 2014, pp. 2672–2680.

[2] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[3] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proceedings of ACM conference on computer and communications security*, 2015, pp. 1310–1321.

[4] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*, 2017, pp. 1273–1282.

[5] S. Truex, L. Liu, M. E. Gursoy, L. Yu, and W. Wei, "Demystifying membership inference attacks in machine learning as a service," *IEEE Transactions on Services Computing*, 2019.

[6] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning," in *2019 IEEE symposium on security and privacy*. IEEE, 2019, pp. 739–753.

[7] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi, "Beyond inferring class representatives: User-level privacy leakage from federated learning," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 2512–2520.

[8] M. Xu and X. Li, "Subject property inference attack in collaborative learning," in *International Conference on Intelligent Human-Machine Systems and Cybernetics*, vol. 1. IEEE, 2020, pp. 227–231.

[9] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, 2015, pp. 1322–1333.

[10] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the gan: information leakage from collaborative deep learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 603–618.

[11] O. Gupta and R. Raskar, "Distributed learning of deep neural network over multiple agents," *Journal of Network and Computer Applications*, vol. 116, pp. 1–8, 2018.

[12] P. Vepakomma, O. Gupta, T. Swedish, and R. Raskar, "Split learning for health: Distributed deep learning without sharing raw patient data," *arXiv preprint arXiv:1812.00564*, 2018.

[13] C. Thapa, M. Chamikara, and S. Camtepe, "Advancements of federated learning towards privacy preservation: from federated learning to split learning," in *Federated Learning Systems*. Springer, 2021, pp. 79–109.

[14] P. Vepakomma, T. Swedish, R. Raskar, O. Gupta, and A. Dubey, "No peek: A survey of private distributed deep learning," *arXiv preprint arXiv:1812.03288*, 2018.

[15] A. Singh, P. Vepakomma, O. Gupta, and R. Raskar, "Detailed comparison of communication efficiency of split learning and federated learning," *arXiv preprint arXiv:1909.09145*, 2019.

[16] K. Singhal, H. Sidahmed, Z. Garrett, S. Wu, J. Rush, and S. Prakash, "Federated reconstruction: Partially local federated learning," *NeurIPS*, vol. 34, pp. 11 220–11 232, 2021.

[17] D. Pasquini, G. Ateniese, and M. Bernaschi, "Unleashing the tiger: Inference attacks on split learning," in *Proceedings of ACM Conference on Computer and Communications Security*, 2021, pp. 2113–2129.

[18] V. Shejwalkar and A. Houmansadr, "Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning," *Internet Society*, p. 18, 2021.

[19] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 118–128.

[20] S. Abuadbba, K. Kim, M. Kim, C. Thapa, S. A. Camtepe, Y. Gao, H. Kim, and S. Nepal, "Can we use split learning on 1d cnn models for privacy preserving training?" in *Proceedings of the ACM AsiaCCS*, 2020.

[21] Z. He, T. Zhang, and R. B. Lee, "Attacking and protecting data privacy in edge–cloud collaborative inference systems," *IEEE Internet of Things Journal*, vol. 8, no. 12, pp. 9706–9716, 2020.

[22] R. Tourani, S. Srikanteswara, S. Misra, R. Chow, L. Yang, X. Liu, and Y. Zhang, "Democratizing the edge: A pervasive edge computing framework," *arXiv preprint arXiv:2007.00641*, vol. 1, no. 1, 2020.

[23] C. Niu, F. Wu, S. Tang, S. Ma, and G. Chen, "Toward verifiable and privacy preserving machine learning prediction," *IEEE TDSC*, vol. 19, no. 03, pp. 1703–1721, 2022.

[24] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 3–18.

[25] A. S. Rakin, Y. Luo, X. Xu, and D. Fan, "Deep-dup: An adversarial weight duplication attack framework to crush deep neural network in multi-tenant fpga," in *30th USENIX Security Symposium*, 2021.

[26] K. Ganju, Q. Wang, W. Yang, C. A. Gunter, and N. Borisov, "Property inference attacks on fully connected neural networks using permutation invariant representations," in *Proceedings of ACM conference on computer and communications security*, 2018, pp. 619–633.

[27] M. P. Parisot, B. Pejo, and D. Spagnuelo, "Property inference attacks on convolutional neural networks: Influence and implications of target model's complexity," *arXiv preprint arXiv:2104.13061*, 2021.

[28] M. Mohri, G. Sivek, and A. T. Suresh, "Agnostic federated learning," in *International Conference on Machine Learning*. PMLR, 2019, pp. 4615–4625.

[29] L. Deng, "The MNIST database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[30] Z. Zhang, Y. Song, and H. Qi, "Age progression/regression by conditional adversarial autoencoder," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5810–5818.

[31] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 3730–3738.

[32] Y. Liu, R. Wen, X. He, A. Salem, Z. Zhang, M. Backes, E. De Cristofaro, M. Fritz, and Y. Zhang, "{ML-Doctor}: Holistic risk assessment of inference attacks against machine learning models," in *USENIX Security Symposium*, 2022, pp. 4525–4542.