Privacy-Preserving Split Learning via Patch Shuffling over Transformers

Dixi Yao, Liyao Xiang, Hengyuan Xu, Hangyu Ye, Yingqi Chen
John Hopcroft Center, Shanghai Jiao Tong University, Shanghai, China
dixi.yao@mail.utoronto.ca, {xiangliyao08, doby-xu, yhy792140335, yingqichen}@sjtu.edu.cn

***Abstract*—We focus on the privacy-preserving problem in split learning in this work. In vanilla split learning, a neural network is split to different devices to be trained, risking leaking the private training data in the process. We novelly propose a patch shuffling scheme on transformers to preserve training data privacy, yet without degrading overall model performance. Formal privacy guarantees are provided and we further introduce the batch shuffling and the spectral shuffling schemes to enhance the guarantee. We show through experiments that our methods successfully defend the black-box, white-box, and adaptive attacks in split learning, with superior performance over baselines, and are efficient to deploy with negligible overhead compared to the vanilla split learning.**

***Index Terms*—Data privacy, deep learning, transformer, split learning, shuffling**

I. Introduction

Recent years have witnessed a great surge in machine learning applications such as face identification [1], recommendation systems [2], and natural language processing [3]. Meanwhile, a new computational paradigm is arising that the data is kept local and the machine learning model runs across different hardwares [4]–[7]. It allows users to take advantage of computational power while preserving the privacy of their local data. Split learning [5] recently emerges as such a framework to support model training across the edge and the cloud. The edge takes local inputs, transforms them into intermediate features, or smashed data, and sends them to the cloud. The cloud trains over features without being aware of the original inputs.

However, the vanilla split learning recently raised wide concern of privacy-leaking. It has been shown that an adversary could infer the private input from the intermediate features [8], [9]. But the problem largely remains unresolved. Most of the existing works have been devoted to privacy-preserving inference [10], [11] rather than split learning, as inference only requires the features transmitted in the one-time forward propagation is privacy-preserving, while it is much more difficult to protect the training data in multiple rounds of forward and backward propagations. Hence, privacy should be guaranteed from the start of training to its end, not only on a fixed model. Additionally, in split learning, the label data is often privately owned by the cloud, posing greater challenges to privacy-preserving split learning. For an instance, while the facial image of a user is private to the edge device, its identity belongs to a proprietary enterprise database which is restricted from being revealed.

Particularly, addressing the privacy issue in split learning requires seeking a sweet spot in the tradeoffs in the learning task utility, data privacy, and computational efficiency. It means that the privacy-preserving approach has to be practical to run on a piece of thin edge devices. Cryptographic tools are not a fit for this scenario as it typically has a high demand on the computational and communication costs [12], which is unbearable on a moderately deep neural network. Differential privacy and other transform-based approaches applied to the training inputs [13], [14], or intermediate features [10], [11], usually sacrifice significant accuracy performance to achieve the privacy guarantee. It is not ideal in cases where accuracy loss is undesirable.

In this work, we propose a practical privacy-preserving split learning framework based on transformer models. Transformer is a cutting-edge neural network structure with superior performance [2], [3], [15], [16]. We exploit its robustness against distortions such as occlusion, shuffling, noise, etc. [17], [18], to derive a *patch shuffling* scheme to protect the training data privacy. The idea is to remove the position embedding layer from the transformer and randomly shuffle the patch tokens to prevent input reconstruction. As the position of each patch is random, it is almost impossible for the adversary to recover the inputs. But model accuracy is rarely affected due to the robustness property. The overhead at the edge is rather lightweight, as a user only needs to shuffle its data to be sent without getting involved in the backward loop.

We not only provide the privacy guarantee based on random shuffling, but also propose two more methods, *batch shuffling*, and *spectral shuffling*, to secure the training data. In patch shuffling, one may argue that the correlation between adjacent patches degrades its privacy guarantee. We show through both analysis and experiments that, such correlations could be brought down to almost zero if we expand the range of possible permutations, or transform features into another domain. For the former, batch shuffling introduces shuffling, occlusion, and mixture to a batch of, rather than a single instance of the data, significantly enlarging the search space for an adversary to obtain the correct permutation order. For the latter, features are transformed into the spectral domain

Liyao Xiang (xiangliyao08@sjtu.edu.cn) is the corresponding author.

This work was partially supported by National Key R&D Program of China under Grant 2021ZD0112801, NSF China (62272306, 61902245, 62032020, 62136006). Authors would like to appreciate the Student Innovation Center of SJTU for providing GPUs.

before being shuffled, which destroys the correlation in the time domain. Interestingly, for the spectral shuffling, the transformer learns in the frequency domain but achieves an equivalent performance to that in the time domain. All three methods proposed provide slightly different tradeoffs between privacy and utility, but their performances are superior to the state-of-the-art. The privacy-preserving training process introduces negligible overhead to vanilla split learning.

Highlights of our contributions are as follows. We are among the first to propose patch shuffling as a privacy-preserving approach in split learning. A formal privacy guarantee is provided and is further enhanced by batch shuffling and spectral shuffling. Experiments on a variety of datasets and tasks have shown the superior performance of our schemes regarding accuracy, privacy, and efficiency. The code is now available on www.github.com/dixiyao/PatchShuffling.

II. Background and Related Work

A. Transformer Properties

Inspired by the great success of Transformers in natural language processing (NLP) [3], researchers adopt similar model structures in various fields including vision, recommendation system [2], electrocardiogram [19], etc. and have achieved superior performance over conventional neural networks. For example, a Vision Transformer (ViT) [15] model treats the linear projection of cropped, fixed-size patches of an image as patch tokens, and takes other task tokens, such as class tokens for classification to complete the specified learning task. The patch order in the original image is used for position embedding. Tokens-To-Token Vision Transformer (T2T) [16] improves over ViT by changing the patch embedding layer into a tokens-to-token module, which encodes the important local structure of each token.

An intriguing property of a transformer is that its basic operations — multilayer perceptron (MLP) and self-attention — are *permutation invariant*, suggesting the possibility of altering the order of the patch sequence without affecting the computation of class tokens. Moreover, transformers exhibit desirable properties for building privacy-preserving networks. Naseer *et al.* [17] showed that ViT is robust against different levels of corruption over images. Even if the position embedding, indicating the patch location information [20], is removed, and patches are randomly shuffled, ViT merely loses 4% accuracy on ImageNet [21]. Apart from that, even if half of the patches are occluded, ViT loses 4% accuracy, compared to over 20% accuracy decline when 5% of the patches are masked in the conventional neural networks. He *et al.* [18] exploited such a property to propose a self-learning framework called Masked AutoEncoder (MAE), by which masking 15% of the patches would still maintain the state-of-the-art accuracy. We build our privacy-preserving scheme based on these transformer properties.

B. Split Learning

As neural networks are growing larger and larger, a trend is to split them apart into submodules to run on different hardware. Hao *et al.* [4] proposed to deploy the first layer of a model on the edge device, and the rest on the cloud, keeping users' private data local while utilizing the cloud resources to complete learning tasks. However, their framework is only used for inference. As for training, split learning (SL) [5], [6] splits a neural network into multiple submodules, and jointly trains these submodules on different devices. A critical part is communication among different parties, which involves sending back and forth intermediate features, smashed data, error gradients, etc. As shown in the example of Fig. 1, a model is split into two parts residing at the edge and the cloud. Different from the edge-cloud inference, the edge sends the feature to the cloud in the forward loop, whereas it retrieves the error gradient from the cloud in the backward loop.

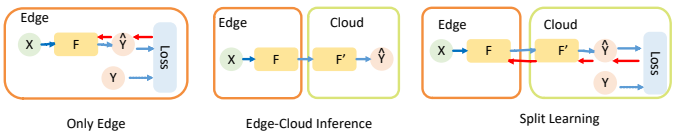


Fig. 1. Different learning frameworks: edge computing, edge-cloud inference, and split learning. X are inputs; Y are labels, F and F' denote (partial) models, and $\hat{Y}$ represents the prediction. Blue and red lines stand for forward and backward propagation, respectively.

Nevertheless, an unprotected split learning framework risks users' data privacy. An inversion attack can be launched to infer private information from smashed data [11]. It is also possible to fit the private data by optimization over the known model parameters [8]. Besides, label inference attack [22] presents a threat to the party who holds the labels, which could also be private. Hence, it is our goal to address the threat to private training data in split learning.

C. Privacy-Perserving Split Learning

Many efforts have been made to preserve data privacy in split learning, where the privacy of both the inputs and labels should not be leaked to any other party. However, most existing methods aim at the inference stage, and few can be applied to training. Traditional methods include cryptographic ones such as secure multi-party computation and homomorphic encryption. But these methods typically involve significant overhead in encryption, decryption, computation, and communication. Lee *et al.* [12] implemented a polynomial approximation over non-linear functions and encrypt the training process with fully homomorphic encryption. It demands 10 to 1000 times more computation power compared to the unprotected split learning, which is unacceptable to edge devices. The approximation computation also results in accuracy losses. Hence, a viable approach would leverage properties of neural networks; for example, Li *et al.* [23] adversarially learned the client submodule to produce intermediate features not containing any private input information; but sufficient to complete the task. However, the method only works when the learning converges and thus suffers potential leakage at the early stage of training. Dong *et al.* [13] inserted Gaussian noise to the smashed data following the convention of differential privacy. Ryoo *et al.* [14] adjusted

the image resolution to find the best tradeoff between utility and privacy. The latter two works have to sacrifice considerable accuracy to meet the privacy requirement. To the best of our knowledge, no method up to now could provide satisfying performance in accuracy, privacy, and computation efficiency in split learning.

III. Formulation and Threat Model

We present the problem formulation and threat model in this section.

A. Problem Formulation

In split learning, the edge client owns $X = \{x_1, \cdots, x_n\} \subset R^d$ (n samples of d-dimensional features), which is a part of the private dataset $\mathcal{D} = \{X, Y\}$, and the cloud server possesses k-class private labels $Y = \{y_1, \cdots, y_n\}, y_i \in \{1, \cdots, k\}, i \in \{1, \cdots, n\}$. Typically, the edge client is resource-restricted, while the server is powerful. The output of the edge is called smashed data or features in this work. We also cover cases where multiple edge clients collaboratively compute the smashed data to feed to the server model. but it is a trivial extension to the one-client-one-server case, and thus we focus on the latter.

In the split learning task, we define the edge model as F and the cloud model as F'. The goal of split learning is to minimize the accuracy loss as

$$\min_{F,F'} L_{task}(F'(F(X)), Y). \tag{1}$$

If we denote the privacy protection method as M, we can write the task objective as

$$\min_{F,F',M} L_{task}(F'(F(M(X))), Y). \tag{2}$$

At the same time, the model F and the mechanism M should be chosen to maximize the attacker loss L_{attack}, referring to the loss of reconstructing X from the smashed data $F(M(X))$. We will thoroughly discuss the L_{attack} in the following section.

B. Threat Model

We assume an honest-but-curious cloud server who completes the learning task as required but is curious about the edge client's private data. Depending on the access level of the server, we divide the attack into the following categories, which should cover most of the possible attacks.

Black-box attacks. The attacker merely accesses the smashed data [11]. It trains an inversion model G to infer inputs from the smashed data over a public dataset X_{pub}. The input of G is the smashed data; the output is the reconstructed input $\tilde{X}_{pub}$, and the goal is to minimize the mean square error (MSE) between $\tilde{X}_{pub}$ and X_{pub}:

$$\min_{G} L_{attack}\left(G(F(M(X_{pub}))), X_{pub}\right). \tag{3}$$

At the end of the training, the smashed data of X is fed into G to invert X.

White-box attacks directly perform gradient descent over its guess $\tilde{X}$ on the known model F to minimize the reconstruction loss between $F(M(\tilde{X}))$ and $F(M(X))$ [8], [24]. The optimization objective is as follows:

$$\min_{\tilde{X}} L_{attack}\left(F(M(\tilde{X})), F(M(X))\right). \tag{4}$$

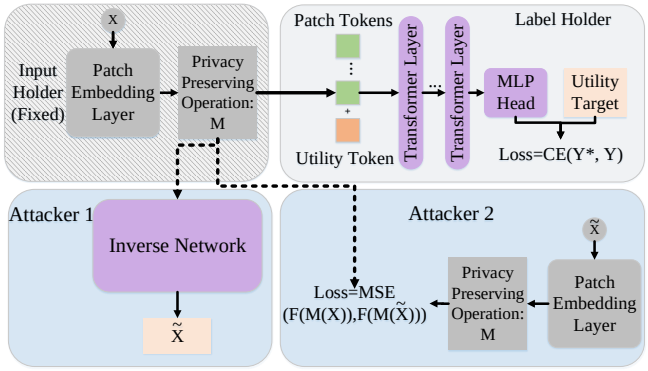


Fig. 2. The structure of our privacy-preserving split learning framework.

(a) Original Input (b) Output of Patch Shuffling (c) Output of Batch Shuffling

Fig. 3. The illustration of different patch privacy-preserving methods: each image is assumed to be cropped into 9 patches; index represents the correct position of each image in its original input; different colors represents different original inputs.

L_{attack} can be an MSE for an instance.

Adaptive attacks. In the training process, the attacker gains much more as it could access the smashed data from multiple rounds [25]. The model is similar to that of a black-box attack but takes multiple iterations of the smashed data. Letting the model be trained for e iterations, the attacker could recover X with the smashed data from $F^1(M(X)), \cdots, F^e(M(X))$ by

$$\min_{G} L_{attack}\left(G(F^1(M(X)), \cdots, F^e(M(X))), X\right). \tag{5}$$

The attacks are typical in the split learning scenario. We hereby design our privacy-preserving learning framework to defend against these attacks.

IV. Methodology

Our privacy-preserving split learning framework over transformers is shown in Fig. 2. On the edge, the input is embedded into patches before the linear projection, and we call them patch embeddings. Different from conventional transformers, position embedding is removed so as to eliminate the position information from the learning task, enabling patch shuffling without performance degradation. We apply a mechanism M over the patch embeddings. The output of M is the protected smashed data transmitted to the cloud. As there is a very small proportion of model parameters residing at the edge, the part of parameters can be fixed. The cloud takes the smashed data to complete the training. We also instantiated the attacks in Sec. III to our framework in Fig. 2. The attacks target at $F(M(X))$ to reconstruct X. In the following, we will give the details of our mechanism M, as well as its privacy guarantee.

A. Definition of Privacy

The goal of M is to constrain the likelihood that the original input can be inferred from the given smashed data. The key insight is to randomly shuffle the patches to maximize the uncertainty of the attacker in input reconstruction. We list patches of an image or a batch of images in a sequence represented by permutation σ. If we permute the red patches in Fig. 3(a) to the red ones in Fig. 3(b), the permutation σ of the latter is (1, 4, 8, 9, 7, 2, 3, 5, 6). That is, patch 2 is now in the place of patch 4 in (b). Let the collection of σ be S. To quantize the uncertainty, we first give the definition of *Neighboring Permutations*, which defines how 'similar' a pair of permutations are.

Definition 1. *(Neighbouring Permutations) We divide a single instance into N patches, and the permutations of these N patches constitute S. Any two permutation σ, $\sigma' \in S$ are defined to be neighboring.*

Now, we formally define σ-privacy as follows.

Definition 2. *(σ-privacy) Given private dataset X and a set of permutations S, a randomized mechanism $\mathcal{A} : f(X) \mapsto \mathcal{V}$ ($\mathcal{V}$ is the output space of f) is σ-private if for all $x \in X$, neighbouring permutations σ and σ' and any $z \in \mathcal{V}$, we have*

$$\Pr[\mathcal{A}(\sigma(f(x))) = z] = \Pr[\mathcal{A}(\sigma'(f(x))) = z]. \quad (6)$$

σ-privacy states that mechanism $\mathcal{A}$ is agnostic of the order of patches. And thus an adversary cannot distinguish the correct permutation from the wrong ones (its neighboring permutations) given the smashed data z. Hence, the adversary is uncertain about the original patch order based on the smashed data. This property is similar to d_σ-privacy [26], but we focus on the patch sequence while they emphasize the relative ordering of data. In d_σ-privacy, the privacy level is determined by sensitivity and the parameter α. As we sample permutations from a uniform distribution instead of the mallows model, the parameter $\alpha = 0$ in d_σ-privacy. That is, neighboring permutations have the same probability of occurring given the output.

B. Patch Shuffling

In patch shuffling, we remove the position embedding from the model without affecting the accuracy too much, yet preventing an adversary from recovering the original input. We randomly shuffle the patches in one instance. The patch embedding is defined as $f : R^d \rightarrow R^{N \times D}$, *i.e.,* each $f(x)$ has N patches, and the dimension of each patch token is D. We randomly sample σ from S and shuffle $f(x)$ in the unit of patch tokens. An instance of patch shuffling is given in Fig. 3(b) where each patch is shuffled within the image. The detail of patch shuffling is shown in Alg. 1. In practice, we add a transformer block following the shuffled patches and transmit the output of the transformer block to the cloud to further enhance privacy performance. The model weights of the additional transformer block are fixed prior.

Algorithm 1: Patch (Spectral) Shuffling

Input: $X = \{x_1, \cdots, x_n | \forall i, x_i \in R^d\}$
Output: $M(X)$

Initialize the patch embedding layer $f : R^d \rightarrow R^{N \times D}$, Batch size B, permutation set S of all permutations of sequence $1 : N$, $M(X) = \emptyset$, transformer block t;
for *each x_i in X* **do**
  $h = f(x_i)$;
  if *Spectral Shuffling* **then**
    $h = FFT(h)$ // FFT is a Fast Fourier Transform;
  end
  Sample σ from S with a uniform distribution;
  $h = \sigma(h)$;
  if *Patch Shuffling* **then**
    $h = t(h)$;
  end
  $M(X) = M(X) \cup \{h\}$;
end

In this case, as we sample permutations following a uniform distribution, each permutation including the original order occurs with the same probability of $1/N!$ (N=196) to produce z. Hence, each pair of neighboring permutations involving the original sequence would occur with the same probability. The adversary could only random guess which permutation is the correct one from all permutations, and thus fails to reconstruct the original instance. Therefore, we have the following proposition:

Proposition 1. *Patch shuffling (Alg. 1) is σ-private with uniformly random permutation on N patches of the instance.*

For example, given a permutation σ from the original instance as in Fig. 3(a), and another arbitrary permutation, σ', the two have the same probability to produce the sequence in Fig. 3(b) under Alg. 1. An adversary is thus unable to tell the correct instance from the wrong one.

Strictly speaking, even with the uniform sampling in Alg. 1, each sequence has a slightly different probability to occur given the smashed data. This may attribute to the correlation in patches of the same instance, *i.e.,* patches sharing the same edges are correlated, and therefore the position of every single patch is not independent. Hence, in the following sections, we derive two approaches to mitigate such a correlation.

C. Batch Shuffling

We further propose batch shuffling to enhance the difficulty of the adversary in inverting the inputs. Three basic operations are defined: shuffling, occlusion, and mixture. We first select a batch of data $X = \{x_1, \cdots, x_B\}$ and perform patch embedding. The hyperparameter $k \in (0, 1)$ is selected, meaning that for each instance, k of the patches stay where they are, and the rest are shifted to other instances in the same batch. For example, in Fig. 3(c), we keep patches 1, 2, 5, 6, and 7 in the red instance while randomly shifting the rest to other instances: patches 3, and 9 are exchanged with patches in the green instance, and patch 4, 8 are shuffled to the blue one. Occlusion is realized by removing $1 - k$ patches from the original place,

Algorithm 2: Batch Shuffling

Input: $X = \{x_1, \cdots, x_n | \forall i, x_i \in R^d\}$
Output: $M(X)$

Initialize the patch embedding layer $f : R^d \to R^{N \times D}$, Batch size B, hyperparameter k, N'=$N - \lfloor k \cdot N \rfloor$, permutation set S_1 of all permutations of sequence $1 : N$, permutation set S_2 of all permutations of sequence $1 : B \cdot N'$, $M(X) = \emptyset$, transformer block t;
Split X into $\lceil \frac{n}{B} \rceil$ batches: $X_1, \cdots, X_{\lceil \frac{n}{B} \rceil}$;
for $i = 1 : \lceil \frac{n}{B} \rceil$ **do**
  $X_i = \{x_1, \cdots, x_B\}$;
  $f(X_i) = \{f(x_1), \cdots, f(x_B)\}$;
  Sample $\{\sigma_1, \cdots, \sigma_B\}$ from S_1 and perform patch shuffling to get $\{\sigma_1(f(x_1)), \cdots, \sigma_B(f(x_B))\}$;
  Generate the patch sequence to be kept in each instance (parameterized by k): $\{u_1, \cdots, u_B\}$, each of which is a sequence of length $\lfloor k \cdot N \rfloor$;
  Concatenate the rest patches into p;
  Sample δ from S_2 to get $\delta(p)$;
  Split $\delta(p)$ into sequences of the same length $(N' \cdot D)$ $\{v_1, \cdots, v_B\}$;
  for j=$1 : B$ **do**
    Concatenate u_j and v_j to obtain m_j;
    $M(x_j) = t(m_j)$;
  end
  $M(X) = M(X) \cup M(X_i)$;
end

and mixture is realized by mixing up with patches from other instances. Finally, a patch shuffling is performed instance-wise to further shuffle the patches within the instance. The detailed procedure of batch shuffling is shown as Alg. 2. A result of the batch shuffling is shown in Fig. 3(c).

In batch shuffling, permutations are applied to a batch instead of a single instance. Take Fig. 3(c) as an example, in each instance, we choose k out of the total number of patches to keep, and the rest patches are shuffled among different instances. For the first step, the red instances of Fig. 3(a) and Fig. 3(b) have an equivalent probability to keep patches 1, 7, 6, 2, and 5 within the same instance. In the second step, the two instances (after the first step) have the same probability of turning into the first instance in Fig. 3(c). Hence the red instances of Fig. 3(a) and Fig. 3(b) are two neighboring permutations, and they share the same likelihood of producing the same smashed data. Therefore, we have

Proposition 2. *Batch shuffling (Alg. 2) is σ-private with permutations applied to $N \times B$ patches, where N is the number of patches in an instance and B is the batch size.*

It is obvious that the correlation between patches of the same instance is weakened by the interpolation of other instances. Moreover, there are much more potential permutations than patch shuffling, which significantly increases the reconstruction hardness of the attacker. Given z, if the adversary has extremely strong computational power, it would have to traverse all possible permutations, that totals

$$\left(\binom{N}{\lfloor N \cdot k \rfloor} \cdot \lfloor N \cdot k \rfloor! \right)^{B} \cdot (B \cdot N')! \tag{7}$$

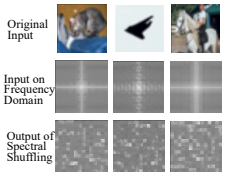


Fig. 4. The illustration of Spectral Shuffling: transform the image into frequency domain, crop the spectral image into 16×16 patches, and shuffle them.

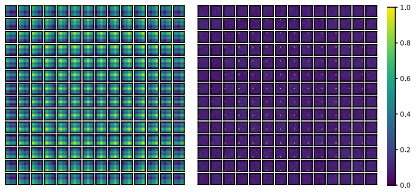


Fig. 5. Position embeddings of training ViT over Cifar10 images in time domain and frequency domain. Each color map in 14×14 grid represents the cosine similarity of position embedding of patch i and every other patches.

To seek those recovering the target instance. A total number of

$$\left(\binom{N}{\lfloor N \cdot k \rfloor} \cdot \lfloor N \cdot k \rfloor! \right)^{B-1} \cdot ((B-1) \cdot N')! \tag{8}$$

Permutations contain the correct target instance, which occupies a much smaller proportion than $1/N!$ of the patch shuffling.

D. Spectral Shuffling

We propose an alternative approach — spectral shuffling — to further eliminate the latent position correlation among patches. The procedure resembles patch shuffling and thus we present it in Alg. 1. An illustrative example can be found in Fig. 4. After the initial patch embedding, we turn all features into the frequency domain through Fast Fourier Transform (FFT). The transformed features are spectral representations of the instance other than the pixel (in RGB) representation. The spectral features are presented in the form of complex values, in which we regard the real and imaginary parts as individual channels and feed them into the following layers. Now each complex-valued feature denotes a value at a different frequency. Shuffling is conducted to mix up values at various frequencies.

Although the spectral images in Fig. 4 resemble an image in pixels; the relation among different patches has little meaning. This is because the patches in the spectral presentation are not continuous in nature, *i.e.,* values at different frequencies are likely not correlated. To further verify the point, we draw in Fig. 5 the cosine similarity of the position embeddings of every pair of patches in the original time-domain and the spectral domain, respectively. The darker color means less correlation. It can be observed that patches in the time domain are somewhat correlated, but the cosine similarity is almost all zeros in the spectral domain. The observation further supports that each shuffling sequence in the spectral domain is equally likely. We hence inherit the privacy guarantee in patch shuffling (Proposition 1. As a result, spectral shuffling is also *σ-private*).

V. Evaluation

A. Setup and implementation detail

1) Models and datasets: Our framework is built on `Pytorch` and `Torchvision`, and a T2T [16] model serves as

the transformer backbone for image datasets. Our neural network is trained on a model base pretrained on ImageNet [21]. The neural network structure is T2T Patch16 Depth24, where each input image is cropped to a patch size of 16, fed into a model consisting of 24 transformer blocks at the cloud. Particularly for spectral shuffling, we use the patch embedding layer in ViT [15] implemented by Pytorch image models (timm [1], a collection of SOTA computer vision models and pretrained weights.) Instead of tokens-to-token in T2T. For the tabular dataset, we adopt Dual Importance-aware Factorization Machines (DIFM) [27] with its default hyperparameters. The part preceding the prediction layer is placed at the edge, as shown in Fig. 8. All model weights on the edge are fixed prior.

We choose a facial image dataset CelebA [28], an object image dataset Cifar10 [29], and a tabular dataset Criteo [2] in the evaluation. CelebA contains 2,022,599 faces from 10,177 celebrities. A 40-attribute classification task is performed on the dataset. Cifar10 consists of 60000 natural images in 10 different classes. Criteo is a click-through-rate dataset for recommendation systems. It contains one month of ad click logs with 100 million records, 13 numerical, and 26 categorical features.

Our default hyperparameters are set as follows. The precision is 32-bit. For tasks on CelebA, LFW, and Cifar10, we train the models with the SGD optimizer. The learning rate (lr) of the final MLP classifier block and the transformer blocks is set to 0.05 and 5×10^{-4}, respectively, and a cosine scheduler is used with the minimal lr 2×10^{-4} and 2×10^{-6} correspondingly. The momentum is 0.9, and the weight decay is 5×10^{-4}. In implementing spectral shuffling, due to the difference in the representation space between the frequency domain and time domain, we use a learning rate of 0.1, and 0.001 for the final MLP classifier block and the transformer blocks, respectively. Models are trained for 60 epochs with a batch size of 50. In batch shuffling, $k = 0.4$ by default.

2) Baselines: For a fair comparison, we select a set of existing privacy-preserving methods applicable in split learning. In these baselines, labels are kept in the cloud. Conventional cryptographic approaches are not included as they require exorbitant computational and communication costs, which is infeasible to be applied at the edge in split learning. Hence our baselines include:

SL: The unprotected split learning, implemented as [5], without any protection for the smashed data.

Adv: The adversarial learning approach [23] trains the edge model against a simulated attacker G' to defend against the inversion attack, alongside its primary learning task. Specifically, the approach maximizes the reconstruction loss of the simulated attacker while minimizing the task loss:

$$\min_{F,F'} \max_{G'} L_{task}(F'(F(X)), Y) - L_{attack}(G'(F(X)), X) \quad (9)$$

However, the method is restricted by its high demand for the edge resource in training the simulated attacker. Moreover, if the real-world attacker is more powerful than the simulated attacker, the defense would break. Last but not the least, Adv does not guarantee training data privacy during the training process; privacy is guaranteed only when Eq. (9) reaches a saddle point.

Transform: The methods apply permutation or transformation over the smashed data to prevent inversion attacks. The most prominent of this type is adding Gaussian noise [13] to achieve differential privacy. Since differential privacy is gauged in different ways in papers, we use the noise $\sim \mathcal{N}(0, 4)$ as an example, denoted by **GN**. We also insert noise following the instruction of differential privacy [13], which yields $\epsilon \approx 6.68$ with a Gaussian noise standard deviation of 1.0. The method is called **DP**. Finally, Ryoo *et al.* propose a transformation technique **Blur** [14] generating extremely low-resolution images for preserving image privacy in neural networks. In our experiments, the default setting is to decrease the resolution to 1/14 of the original.

We implemented three methods: patch shuffling as **Our PS**, batch shuffling as **Our BS** and spectral shuffling as **Our PS+**. The number behind batch shuffling denotes k, *e.g.*, Our BS 40 means batch shuffling with $k = 0.4$.

3) Metrics: We evaluated our methods against baselines regarding utility, privacy, and efficiency.

Utility: We use accuracy (Acc) to evaluate the performance of the classification task. In particular, for face attribute classification on CelebA [28], we also use the Matthews correlation coefficient (MCC) due to the imbalanced label distribution:

$$\mathrm{MCC} = \frac{(\mathrm{TP} \cdot \mathrm{TN} - \mathrm{FP} \cdot \mathrm{FN})}{\sqrt{(\mathrm{TP}+\mathrm{FP})(\mathrm{TP}+\mathrm{FN})(\mathrm{TN}+\mathrm{FP})(\mathrm{TN}+\mathrm{FN})}}. \quad (10)$$

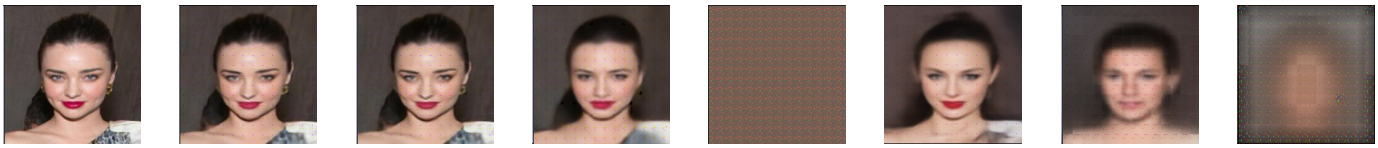
TP, TN, FP and FN represent true positive, true negative, false positive and false negative, respectively. The range of MCC is $[-1, 1]$ where 1, 0, -1 indicate the perfect, random, and the worst predictions, respectively.

Privacy: To evaluate privacy, we mainly gauge the capability of attacker in reconstructing the private inputs. For image data, we evaluate the performance using the following metrics.

- MSE: The mean square error between the original inputs and the reconstruction results.
- PSNR: Peak Signal to Noise Ratio [30], a common metric to evaluate image quality.
- SSIM: Structural Similarity [30], a metric evaluating the similarity between the original and the reconstructed images.
- F-SIM: We feed the original and the reconstructed images into a deep neural network and compare the cosine similarity between the features. For human face datasets CelebA and LFW, we use InceptionResNetV1 of FaceNet [1], pretrained on VggFace2 [31]. For Cifar10 [29], we use ResNet18 pretrained on ImageNet. In both networks, the feature fed into the final linear classifier is used.
- ID: For human face datasets, we also evaluate the identification success rate on InceptionResNetV1 [1]. The identification network is trained on the entire dataset

[1]https://github.com/rwightman/pytorch-image-models

[2]http://labs.criteo.com/downloads/download-terabyte-click-logs/



(a) Input (b) SL (c) Adv (d) Blur (e) DP (f) GN (g) Our BS (h) Our PS+

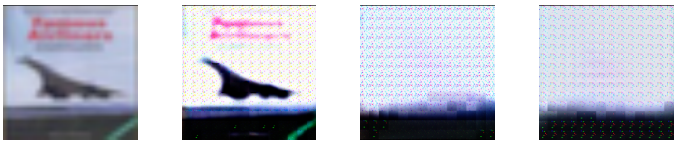
Fig. 6. Examples of reconstructed images by the black-box attack against different defense methods on CelebA.

TABLE I
THE COMPARISON OF UTILITY AND PRIVACY ON DIFFERENT METHODS UNDER THE BLACK-BOX ATTACK. ATTRIBUTE CLASSIFICATION TASK IS PERFORMED ON CELEBA. ↑ DENOTES DESIRABLE DIRECTIONS, AND ITALICS MEANS UNACCEPTABLE RESULTS.

Methods	Utility		Privacy				
	Acc ↑	MCC↑	MSE ↑	SSIM↓	PSNR ↓	F-SIM ↓	ID↓
SL	91.05	0.727	*0.014*	*0.670*	*18.64*	*0.925*	*0.909*
Adv	90.36	0.705	*0.014*	*0.669*	*18.69*	*0.925*	*0.909*
Blur	89.58	0.678	*0.02*	0.447	15.66	0.550	0.0451
DP	*80.67*	0.312	0.384	0.017	4.17	0.171	**0**
GN	87.35	0.601	*0.029*	0.424	*15.28*	0.483	0.0227
Our BS	**89.18**	**0.660**	0.109	0.222	9.62	0.248	0.0006
Our PS+	88.21	0.631	**0.372**	**0.004**	**4.33**	**0.170**	**0.0001**

TABLE II
THE COMPARISON OF UTILITY AND PRIVACY ON DIFFERENT METHODS UNDER THE BLACK-BOX ATTACK. CLASSIFICATION TASK IS CONDUCTED ON CIFAR10. ↑ DENOTES DESIRABLE DIRECTIONS, AND ITALICS MEANS UNACCEPTABLE RESULTS.

Methods	Utility	Privacy			
	Acc ↑	MSE ↑	PSNR ↓	SSIM ↓	F-SIM ↓
SL	98.36	*0.362*	*4.64*	0.367	0.678
Our PS	**96.99**	0.696	1.670	0.104	**0.498**
Our BS 75	96.16	**0.730**	**1.463**	**0.084**	0.513



(a) Input (b) SL (c) Our PS (d) Our BS 75

Fig. 7. Examples of reconstructed images by the black-box attack against different defense methods on Cifar10.

offline. It is a metric used to evaluate whether a user's identity has been leaked.

Efficiency: This metric decides how the algorithm is readily deployed to real-world edge devices. The following metrics are used.

- Macc Edge: MAC operation counts records the number of multiplication and addition operations for computation on a neural network.
- Mem Edge : Memory cost on the edge for completing the training task.

B. Privacy and Efficiency

We report the evaluation results in this section.

TABLE III
THE COMPARISON OF UTILITY AND PRIVACY ON DIFFERENT METHODS UNDER THE BLACK-BOX ATTACK. CLASSIFICATION TASK IS PERFORMED ON CRITEO. ↑ DENOTES DESIRABLE DIRECTIONS.

Methods	Utility: Acc ↑	Privacy: MSE ↑
SL	**77.81**	0.0012
Our PS	77.78	**0.0015**
GN	77.28	0.0012

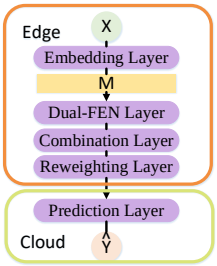


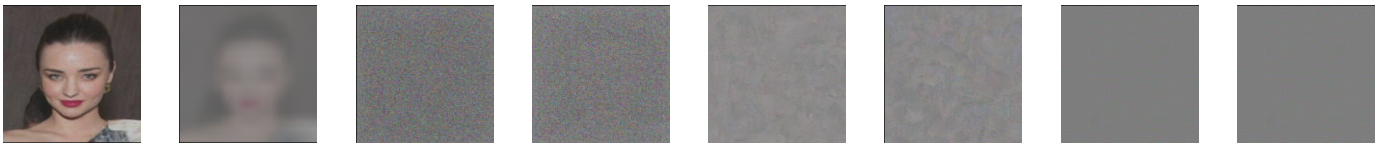
Fig. 8. The illustration of deployment of DIFM over the edge and the cloud.

1) Black-Box Attack: For the attack model, we adopt a similar structure to the edge model: an MAE decoder [18] is used, and is pretrained on ImageNet. We slightly modified the MAE decoder by adding a position embedding layer at the head of the decoder and a Tanh activation layer at the rear. The attack model is trained with an MSE loss and is optimized by AdamW optimizer with a learning rate of 1.5^{-5} and its default hyperparameters. We train the attack model on the original training set for 50 epochs until full convergence and test it on the testing set. Note that our attacker represents a worst-case adversary, as a real-world attacker hardly accesses the private training data. Hereby the privacy performance reported is worse than that in practice. Each experiment is repeated three times to report the average results.

The results of CelebA is shown in Table I and Fig. 6. The identification network we use to measure 'ID' achieves 1.57% error rate on the original dataset. We can see that Adv, Blur, GN, and our methods achieve good utility performance, but Adv, GN, and Blur do not maintain a privacy guarantee. While DP has a satisfying privacy level, it leads to poor utility. Both batch shuffling and spectral shuffling enjoy desirable tradeoffs but differ in details: spectral shuffling completely removes the facial information while batch shuffling allows an attacker to recover a face different from the original one. It suggests spectral shuffling completely destroys the input, whereas batch shuffling somehow retains some average input information, as all the shuffled patches are human faces. We still consider the latter defense a successful one as it would mislead the attacker into a wrongly reconstructed face image.

On Cifar10, Table II reports the utility and privacy results and Fig. 7 are examples of the reconstructed images by the attack. It can be observed that our methods work well on natural images regarding accuracy and privacy.

On Criteo, we apply Our PS to the core vector-wise part of



(a) SL/Adv (b) Blur (c) DP (d) GN (e) Our PS (f) Our BS (g) Our PS+ (h) Jigsaw to Our BS

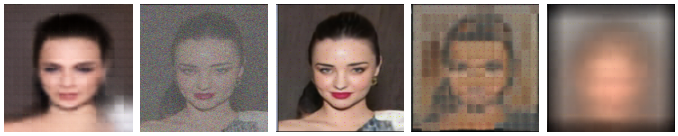
Fig. 9. Examples of reconstructed images by the white-box attack against different defense methods on CelebA. The original input is the same as in Fig. 6. Jigsaw means attacker first performs Jigsaw attack, and then launch the white-box attack.

TABLE IV
THE COMPARISON OF PRIVACY ON DIFFERENT METHODS UNDER THE WHITE-BOX ATTACK TO THE CLASSIFICATION TASK ON CELEBA. ↑ DENOTES DESIRABLE DIRECTIONS, AND ITALICS MEANS UNACCEPTABLE RESULTS.

Methods	MSE ↑	SSIM↓	PSNR ↓	F-SIM ↓
SL/ Adv	*0.011*	*0.647*	*19.74*	*0.602*
Blur	*0.038*	0.215	*14.15*	0.439
DP	0.798	0.0006	0.98	0.111
GN	0.826	0.0006	0.83	0.108
Our PS	0.930	0.0005	0.32	0.131
Our BS	0.914	0.0005	0.39	**0.082**
Our PS+	0.716	**0.0001**	1.45	0.123
	Jigsaw + WhiteBox Attack			
Our BS	0.871	0.0007	0.60	0.132

TABLE V
THE PRIVACY OF ADV OVER THE TRAINING PROCESS OF CELEBA. ↑MEANS DESIRABLE DIRECTIONS, AND THE ITALICS DENOTES UNACCEPTABLE RESULTS.

Training Iterations	MSE ↑	SSIM↓	PSNR ↓	F-SIM ↓	I ID ↓
1000	*0.014*	*0.669*	*18.69*	*0.925*	*0.909*
20000	0.083	0.290	10.85	*0.71*	0.0003
Converged	0.373	0.042	4.29	0.117	0.0001



(a) Our BS (b) GN (c) Adv 1000 (d) Adv 20000 (e) Adv Converged

Fig. 10. The adaptive attack to the same input in Fig. 6. Our BS displays the reconstruction results on 30 rounds of the smashed data collected in training. GN displays the result of averaging 60 noisy images with the Gaussian noise sampled from $\mathcal{N} \sim (0, 4)$. Adv does not defend the attack during training.

DIFM. The attacker model also has a transformer structure, containing a vector-wise part and an output MLP layer. The training hyperparameters are set as default in DIFM. For a fair comparison, we measure the mean absolute values between the two random input features, which is 0.003. We compare feasible baselines and display the results in Table III. For GN, Gaussian noise is randomly sampled from $\mathcal{N}(0, 0.001)$, which although does not hurt accuracy too much, its MSE under attack, almost remains the same with the unprotected SL. If we adopt $\mathcal{N}(0, 1)$ in GN, the reconstruction MSE would be 0.0014, which is still inferior to the privacy level provided by our method. Meanwhile, our method could achieve accuracy close to SL.

2) White-Box Attack: Since the white-box attack directly optimizes the guess value $\tilde{X}$ over the known model weights at the edge, its success rate is independent of the auxiliary data. We train the white-box attack for 100,000 iterations with Adam optimizer (lr = 0.001) to attack the defense on CelebA. Batch size is 16.

Table IV shows the attack results. Split learning and adversarial learning-based methods share the same privacy level as Adv has the highest attack success rate right at the start of training when the model weights are not optimized yet (the defense has not begun). The transform-based methods all enjoy good privacy performance as they introduce randomness, which is defensive to white-box attacks. Also, introducing randomness, our methods are the strongest, as recovering from randomly shuffled patches is extremely difficult. Even if the attacker is aware of the model weights, it is hard to learn the correct permutation out of that many possible sequences as in Eq. 7 ($N = 196$ here). Despite that, we complement another attack called Jigsaw solving [32] trying to infer the correct permutation, and adopt the permutation to launch the white-box attack. As shown in Table IV, the attacker fails to recover a sequence close to the original input. Fig. 9 shows the reconstructed instances by the white-box attack. It can be told that GN and Blur still leak out some outline of the input, while our methods can prevent the adversary from recovering anything.

3) Adaptive Attack: In an adaptive attack, an attacker can supervise the entire training process to launch the attack. Hence, we compare the inversion results at different training iterations in Table V and Fig. 10. As different training iterations play no difference to our methods compared to the black-box attack, we omit the part of results. However, such a difference is significant to Adv. It is observed that privacy can only be guaranteed upon convergence; when the model is not converged, an attacker can easily invert the input from the smashed data. Hereby we claim Adv cannot defend against adaptive attack.

Besides using the smashed data from one iteration, an attacker can also infer the private data from the smashed data collected in multiple training rounds. For instance, in DataMix [10], during the training process, attackers can average a large set of mixtures that contain the same raw image

TABLE VI
THE EFFICIENCY OF DIFFERENT METHODS IN THE SPLIT LEARNING. ↑ MEANS DESIRABLE DIRECTIONS.

Methods	Macc Edge (M)↓	Mem Edge (G)↓
SL / Transform	**3.10**	**0.97**
Adv	81.63	2.43
Our PS/BS	**3.10**	**0.97**
Our PS+	1.18	1.01

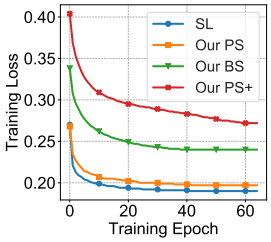


Fig. 11. Training losses in split learning for different methods.

to recover the input. The attack is particularly effective on noise insertion-based methods in averaging out the noise: If we train a model for e epochs with noise $\sim \mathcal{N}(0, \sigma)$. After averaging the smashed data, the noise will follow $\mathcal{N}(0, \frac{\sigma}{\sqrt{e}})$, which lowers the privacy level. Taking Fig. 10 as an example, if we apply the Gaussian noise once to the image, it will be reconstructed as the occlusive result in Fig. 9(d). However, if the attacker collected the same image 60 times in training, its reconstructed results would be as in Fig. 10(b).

For our method, we take batch shuffling as an example and evaluate it against the adaptive attack. We sampled 30 rounds of the smashed data for each image, every other epoch. Then we concatenate them and feed them into the MAE decoder. The reconstruction performance is evaluated as MSE 0.111, SSIM 0.263, PSNR 9.59, F-SIM 0.269, and ID 0.0008, which is approximately at the same level as the one under the black-box attack. This is because the space of possible permutation sequences is so large that multiple rounds of the smashed data hardly help. Its visualization effect is displayed in Fig. 10(a), which is an obviously misleading one from the original input in Fig. 6.

4) Efficiency: We test the efficiency on a single image of size 224×224 and provide the results in Table VI. Our methods of patch shuffling and batch shuffling share the same efficiency level as the unprotected split learning, while spectral shuffling incurs a bit less overhead as we place the first layer of the transformer block at the cloud. For Adv, the overhead is significant as training against a simulated attacker is involved at the edge. The convergence performance of our methods is given in Fig. 11, which is comparable to SL.

C. Ablation Study

1) Setting of k: An important hyperparameter is k in batch shuffling. By changing the value of k, we can choose different tradeoffs between privacy and utility. Hence we show the results with different ks in Table VII and examples in Fig. 12. The trend is that with a bigger k, one can achieve a better utility but lower privacy. We select $k = 0.4$ on CelebA as it denotes the best tradeoff.

2) Attackers Trained on Public Dataset: In evaluating defenses against the black-box attack, we train the attack model and evaluate the result on data coming from the same distribution. While, in most situations, the attacker has no access to data following the same distribution as the training

TABLE VII
THE UTILITY AND PRIVACY WITH DIFFERENT kS IN BATCH SHUFFLING. ↑ MEANS DESIRABLE DIRECTIONS.

k	Utility	Privacy				
	Acc ↑	MSE ↑	SSIM↓	PSNR ↓	F-SIM ↓	ID↓
0.5	90.29	0.071	0.320	11.47	0.36	0.0004
0.4	89.18	0.109	0.222	9.62	0.25	0.0006
0.25	88.54	0.128	0.231	8.94	0.21	0.0006
0.15	88.76	**0.184**	**0.178**	**7.35**	**0.18**	**0.0002**


(a) 0.5 (b) 0.4 (c) 0.25 (d) 0.15

Fig. 12. The example of black box attack results of the input in the Fig. 6 with setting different ks in Batch Shuffling

TABLE VIII
PRIVACY METRICS OF BLACK BOX ATTACK RESULTS OVER BATCH SHUFFLING. ↑ MEANS BETTER

Methods	MSE ↑	SSIM↓	PSNR ↓	F-SIM ↓	ID↓
Public dataset: CelebA. Private dataset: LFW					
SL	0.011	0.705	19.31	0.937	0.99
Our BS	0.152	0.203	8.22	0.269	0.013
Public dataset: LFW. Private dataset:CelebA					
SL	0.086	0.342	10.75	0.650	0.1274
Our BS	0.243	0.079	6.16	0.148	0


(a) Input (b) SL (c) Our BS

Fig. 13. Black-box attack to batch shuffling. Public dataset: CelebA. Private dataset: LFW.

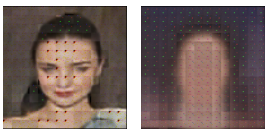

(a) SL (b) Our BS

Fig. 14. Public dataset: LFW. Private dataset: CelebA. Input is the same as in Fig. 6.

data. To mimic the attack, we let the attacker train its model over a public dataset and evaluate the privacy of the private dataset. Hence we use LFW [33] as the public dataset, which contains 13,233 faces from 5,479 different persons. The InceptionResNetV1 identification accuracy on LFW is 99%. The batch shuffling results are given in Table VIII. As Fig. 14 shows, even if the public dataset is smaller than the private dataset, and the data follows a different distribution, our method successfully defends against black-box attacks. We also show a complementary result in Fig. 13 where the role of CelebA and LFW is switched, too see the conclusion still holds.

3) Other Attacker Model : Besides using the transformer structure as the attacker model, we also test against attackers with different networks and we adopt representative convolution neural network (CNN) pix2pix [9]. Pix2pix is trained as

TABLE IX
THE COMPARISON OF PRIVACY ON BLACK-BOX ATTACK WITH PIX2PIX OVER CELEBA. ↑ DENOTES DESIRABLE DIRECTIONS, AND ITALICS MEANS UNACCEPTABLE RESULTS.

Methods	MSE ↑	SSIM↓	PSNR ↓	F-SIM ↓	ID ↓
SL	*0.042*	0.453	*13.77*	0.382	0.033
Blur	*0.052*	0.360	*12.88*	0.249	0.006
GN	0.267	0.209	5.82	0.131	**0.0002**
Our BS	**0.435**	**0.068**	**3.67**	0.115	0.0003
Our PS+	0.265	0.117	5.82	**0.117**	**0.0002**

(a) Input (b) SL

(c) BS (d) PS+

Fig. 15. Examples of reconstructed images by pix2pix on CelebA.

a black-box attacker model using its default hyperparameters. The privacy performance is shown in Table IX and example results are in Fig. 15. As pix2pix is a CNN-based GAN method, the network heavily depends on position information. It is clear that shuffling over patches completely destroys the pix2pix attacker's ability in reconstructing private data.

VI. CONCLUSION

We address the important issue of training data privacy in split learning. We designed a novel and practical patch shuffling scheme by drawing on the robustness property of the cutting-edge model transformers. We further propose batch shuffling and spectral shuffling to remove patch correlations, enhancing privacy guarantees. While being almost as efficient as vanilla split learning, our methods achieve competitive performance regarding utility and privacy.

REFERENCES

[1] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015.

[2] W. Lu, Y. Yu, Y. Chang, Z. Wang, C. Li, and B. Yuan, "A dual input-aware factorization machine for ctr prediction," in *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, 2021.

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, 2017.

[4] H. Dong, C. Wu, Z. Wei, and Y. Guo, "Dropping activation outputs with localized first-layer deep network for enhancing user privacy and data security," *IEEE Transactions on Information Forensics and Security*, 2017.

[5] O. Gupta and R. Raskar, "Distributed learning of deep neural network over multiple agents," *Journal of Network and Computer Applications*, 2018.

[6] C. Thapa, M. A. P. Chamikara, S. Camtepe, and L. Sun, "Splitfed: When federated learning meets split learning," *arXiv preprint arXiv:2004.12088*, 2020.

[7] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, 2020.

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

[9] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017.

[10] Z. Liu, Z. Wu, C. Gan, L. Zhu, and S. Han, "Datamix: Efficient privacy-preserving edge-cloud inference," in *European Conference on Computer Vision*, 2020.

[11] L. Xiang, H. Zhang, H. Ma, Y. Zhang, J. Ren, and Q. Zhang, "Interpretable complex-valued neural networks for privacy protection," in *International Conference on Learning Representations*, 2019.

[12] J.-W. Lee, H. Kang, Y. Lee, W. Choi, J. Eom, M. Deryabin, E. Lee, J. Lee, D. Yoo, Y.-S. Kim *et al.*, "Privacy-preserving machine learning with fully homomorphic encryption for deep neural network," 2022.

[13] J. Dong, A. Roth, and W. Su, "Gaussian differential privacy," *Journal of the Royal Statistical Society*, 2021.

[14] M. Ryoo, K. Kim, and H. Yang, "Extreme low resolution activity recognition with multi-siamese embedding learning," in *Proceedings of the AAAI conference on artificial intelligence*, 2018.

[15] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations*, 2020.

[16] L. Yuan, Y. Chen, T. Wang, W. Yu, Y. Shi, Z.-H. Jiang, F. E. Tay, J. Feng, and S. Yan, "Tokens-to-token vit: Training vision transformers from scratch on imagenet," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[17] M. M. Naseer, K. Ranasinghe, S. H. Khan, M. Hayat, F. Shahbaz Khan, and M.-H. Yang, "Intriguing properties of vision transformers," *Advances in Neural Information Processing Systems*, 2021.

[18] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," *arXiv preprint arXiv:2111.06377*, 2021.

[19] X. Li, C. Li, Y. Wei, Y. Sun, J. Wei, X. Li, and B. Qian, "Bat: Beat-aligned transformer for electrocardiogram classification," in *2021 IEEE International Conference on Data Mining (ICDM)*, 2021.

[20] Y.-A. Wang and Y.-N. Chen, "What do position embeddings learn? an empirical study of pre-trained language model positional encoding," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*.

[21] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, 2015.

[22] O. Li, J. Sun, X. Yang, W. Gao, H. Zhang, J. Xie, V. Smith, and C. Wang, "Label leakage and protection in two-party split learning," in *International Conference on Learning Representations*, 2022.

[23] T. Xiao, Y.-H. Tsai, K. Sohn, M. Chandraker, and M.-H. Yang, "Adversarial learning of privacy-preserving and task-oriented representations," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[24] Y. Zhang, R. Jia, H. Pei, W. Wang, B. Li, and D. Song, "The secret revealer: Generative model-inversion attacks against deep neural networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020.

[25] F. Tramer, N. Carlini, W. Brendel, and A. Madry, "On adaptive attacks to adversarial example defenses," in *Advances in Neural Information Processing Systems*, 2020.

[26] C. Meehan, A. R. Chowdhury, K. Chaudhuri, and S. Jha, "Privacy implications of shuffling," in *International Conference on Learning Representations*, 2022.

[27] W. Lu, Y. Yu, Y. Chang, Z. Wang, C. Li, and B. Yuan, "A dual input-aware factorization machine for ctr prediction," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, 2020.

[28] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.

[29] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[30] A. Hore and D. Ziou, "Image quality metrics: Psnr vs. ssim," in *2010 20th international conference on pattern recognition*, 2010.

[31] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman, "Vggface2: A dataset for recognising faces across pose and age," in *2018 13th IEEE international conference on automatic face & gesture recognition (FG 2018)*.

[32] M. Noroozi and P. Favaro, "Unsupervised learning of visual representations by solving jigsaw puzzles," in *European conference on computer vision*, 2016.

[33] G. B. Huang, M. Mattar, H. Lee, and E. Learned-Miller, "Learning to align from scratch," in *NIPS*, 2012.