Visual Transformer Meets CutMix for Improved Accuracy, Communication Efficiency, and Data Privacy in Split Learning

Sihun Baek[1] , **Jihong Park**[2] , **Praneeth Vepakomma**[3]
Ramesh Raskar[3] , **Mehdi Bennis**[4] , **and Seong-Lyun Kim**[1]
[1]Yonsei University, Seoul, South Korea
[2]Deakin University, Victoria, Australia
[3]MIT Media Lab, Massachusetts, the United States
[4]University of Oulu, Oulu, Finland
shbaek@ramo.yonsei.ac.kr, jihong.park@deakin.edu.au, vepakom@mit.edu
raskar@media.mit.edu, mehdi.bennis@oulu.fi, slkim@yonsei.ac.kr

Abstract

This article seeks for a distributed learning solution for the *visual transformer (ViT)* architectures. Compared to convolutional neural network (CNN) architectures, ViTs often have larger model sizes and are computationally expensive, making federated learning (FL) ill-suited. Split learning (SL) can detour this problem by splitting a model and communicating the hidden representations at the split-layer, also known as smashed data. Notwithstanding, the smashed data of ViT are as large as and as similar as the input data, negating the communication efficiency of SL while violating data privacy. To resolve these issues, we propose a new form of *CutSmashed data* by randomly punching and compressing the original smashed data, and develop a novel SL framework for ViT, coined *CutMixSL*. CutMixSL communicates CutSmashed data, thereby reducing communication costs and privacy leakage. Furthermore, CutMixSL inherently involves the CutMix data augmentation, improving accuracy and scalability. Simulations corroborate that CutMixSL outperforms other baselines including parallelized SL and SplitFed that integrates FL with SL.

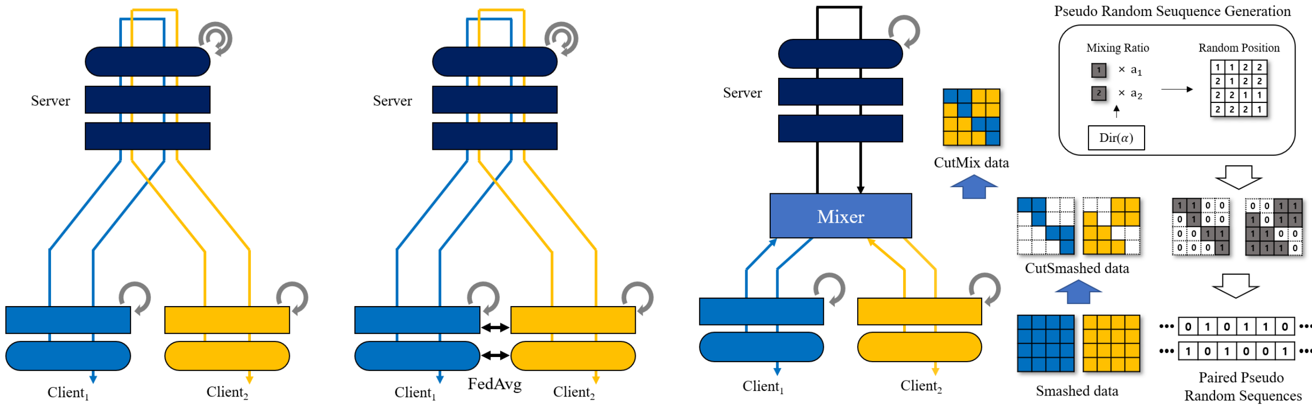
1 Introduction

Transformer architectures have revolutionized various application domains in deep learning, ranging from natural language processing (NLP) [Vaswani *et al.*, 2017] to speech recognition [Karita *et al.*, 2019]. Fueled by this success, recently there has been another paradigm shift in computer vision (CV) where the *visual transformer (ViT)* architecture has broken the performance record set by the *de facto* standard convolutional neural network (CNN) architectures [Dosovitskiy *et al.*, 2020a]. The core idea of ViT is to divide each input image sample into multiple *patches* as in the tokens in NLP, and to process the patches in parallel using the *attention* mechanism of Transformer. This process is in contrast to that of CNN which processes only neighboring pixels in sequence using the convolutional operations.

While the existing studies focus mostly on centralized ViT operations [Han *et al.*, 2022], in this article we delve into the problem of distributed learning with ViTs that are often computationally expensive, and require more training samples than CNNs [Khan *et al.*, 2021; Han *et al.*, 2022]. In the recent literature of distributed learning, federated learning (FL) is one promising solution that enables training a global model across edge devices such as phones, cameras, and the Internet of Things (IoT) devices [Li *et al.*, 2020]. The key idea is periodically averaging model parameters across edge devices or clients through a parameter server, without directly exchanging private data. However, model averaging requires every client to store and communicate the entire model, so is ill-suited for ViT due to its large model size.

Alternatively, each client can store only a fraction of the entire model, and offload the remaining segment onto the server under a model-split architecture [Gupta and Raskar, 2018]. *Split learning (SL)* follows this model-split parallelism, and at the split-layer, each client uploads the hidden layer activations in the forward propagation (FP), also known as *smashed data*, and downloads gradients in the back propagation (BP) [Vepakomma *et al.*, 2018]. Unfortunately, as opposed to CNN's smashed data spatially distorted by convolution operations, ViT's smashed data wihtout convolution look similar to their input data [yuan2021tokens], which may leak a non-negligible amount of private information on raw data to the server. Furthermore, ViT often lacks pooling layers, so the smashed data sizes are as large as the input data, negating the communication efficiency of SL.

To resolve the aforementioned issues, inspired from the *CutMix* data augmentation technique [Yun *et al.*, 2019], we propose a new type of *CutSmashed data*, and thereby develop a novel split learning framework for ViTs, coined *CutMixSL*. In CutMixSL, each client constructs the CutSmashed data by randomly masking the patches of the original smashed data. For instance, Fig. 1 illustrates two clients locally constructing the CutSmashed data by applying the mutually exclusive masks 010110 and 101001 to their original smashed data, respectively. Here, to guarantee the mutual exclusiveness and the subsequent operations, a common pseudo random generator is



(a) Baseline 1: Parllel SL. (b) Baseline 2: SFL. (c) Proposed: CutMixSL.

Figure 1: Schematic illustrations of: (a) parallel split learning (SL), (b) SplitFed (SFL), and (c) CutMixSL.

shared by all clients and the server, without sharing raw data. Then, the mixer at the server adds these uploaded CutSmashed data, resulting in *CutMix data* (i.e., with the mask 111111) that continues FP in the server and the rest of SL operations.

The multi-fold benefit of CutMixSL is summarized as follows.

- First, the clients in CutMixSL can only upload non-zero masked CutSmashed data, *reducing privacy leakage*.
- Second, CutMix data plays its original role as data augmentation complementing ViT's lack of useful inductive bias as opposed to CNNs, thereby *improving accuracy*.
- Lastly, CutMixSL is free from the standard paralell SL's imbalance problem between the server-side and client-side model updates, *achieving scalability* in temrs of the accuracy increasing with the number of clients.
- Simulations show that in CIFAR-10 classification, CutMixSL reduces the privacy leakage (measured by reconstruction mean-squared errors) by around $8\times$, decreases uplink communication payload sizes by $20-50\%$, and improves accuracy by up to 18.5% while achieving scalability up to (at least) 10 clients.

2 Related works

FL and SL are two popular distributed collaborative machine learning techniques keeping data within the data custodian without raw data sharing, considered as privacy-by-design techniques [Park *et al.*, 2021a]. FL has scalable performance with the number of clients [Konečný *et al.*, 2015], however, it has a limitation to only handle models of small size because of the constraints of memory, computing, and communication resources of the clients [Konečný *et al.*, 2016]. On the other hand, SL is considered as an enabler to exploit models of large size through splitting model into two segments that the sever and clients hold [Gupta and Raskar, 2018; Vepakomma *et al.*, 2018]. SplitFed (SFL) [Thapa *et al.*, 2020] is the first hybrid of FL and SL to achieve advantages from both, and its generalized version is proposed [Gao *et al.*, 2021] introducing groups on the server-side. Nevertheless, an additional communication payload is appended despite of its better generalization capability. [Xiao *et al.*, 2021], and [Oh *et al.*, 2022] address mixing activations during FP to deal with data privacy leakage, and scalability.

Transformer utilizes attention mechanisms to extract intrinsic features [Vaswani *et al.*, 2017], and is first applied to the field of natural language processing. BERT [Devlin *et al.*, 2018], and GPT-3 [Brown *et al.*, 2020] are popular examples of trasnformer-based models in NLP tasks. Another line of works of transformer-based models are found in computer vision (CV) tasks. One of representative transformer model in the CV domain is ViT [Dosovitskiy *et al.*, 2020b] applying a pure transformer directly to sequences of images patches to classify the full image, and transformers have been utilized in a variety of other vision tasks [Carion *et al.*, 2020; Zheng *et al.*, 2021]. [Park *et al.*, 2021b] proposed a split learning architecture with a vision transformer to diagnose COVID-19 infection by utilizing a transformer's robustness on task-agnostic training and its decomposable configuration. [Qu *et al.*, 2021] has analyzed the performance of ViT in federated learning for data heterogeneity. From a different perspective, [Hong *et al.*, 2021] proposed a federated dynamic transformer to deal with problems of utilizing transformer models in a Text to Speech (TTS) task.

3 Token-based Split Learning for ViT

In this paper, a novel split learning based on patches where transformers operate is proposed to solve the issue of heavy communication payload and data privacy leakage problem. We describe components of the proposed learning algorithm in order of training sequence. The overall procedure of the proposed is visualized as Figure 1c.

3.1 CutSmashed data for Communication Efficiency and Privacy Enhancement

To increase a communication efficiency, and to resolve the data privacy leakage problem, we pay attention to the way how transformers address data. Transformers divide data into multiple separate tokens and process the entire range of sequence in parallel. In this sense, we instinctively expect that cutting off a certain amount of patches of smashed data at random could reduce communication cost and strengthen the data privacy since adversaries are not able to regenerate the untransmitted region at the expense of performance. This partially uploaded smashed data is labelled as *CutSmashed Data*. CutSmashed data conceal its information by random removal of patches. While SL requires to upload the entire smashed data of each client to the server, our proposed does not need to upload the whole smashed data from each client.

Let there exist n clients with a set of $\mathbb{C} = \{1, 2, \cdots, n\}$ and a single server. $(\mathbf{x}_i, \mathbf{y}_i)$ denotes a batch of raw data-label tuples from i-th local dataset, $\mathbb{D}_i$. A neural network model is denoted with weights $\mathbf{w}_i$, which is splitted into two segments such that $\mathbf{w}_i = [\mathbf{w}_{c,i}, \mathbf{w}_s]^{\mathrm{T}}$ for $i \in \mathbb{C}$. We define f as a representation for mapping from the input data to the output. The smashed data is expressed as $\mathbf{s}_i := f_{\mathbf{w}_{c,i}}(\mathbf{x}_i)$. Here, transformers divide the raw data into M number of patches, and each patch is transformed into an embedding vector during FP. A smashed data is denoted as $\mathbf{s} = [e_1, ..., e_M] \in \mathcal{R}^{M \times d_m}$, where e_i is the i-th embedding vector and d_m is the dimension of the patch embedding.

Before uploading CutSmashed data, a piece of information which patch embeddings would be transmitted are shared between a server and clients. This prior information is called as *a pseudo random sequence*, and is denoted as $\mathbb{B}$. Then, CutSmashed data is expressed as $\mathbf{s}'_i = \mathbb{B} \odot \mathbf{s}_i$, where $\mathbb{B} = [m_j]_{M \times 1}$, and $m_j \in \{0, 1\}$ where 0 indicates not to transmit, and 1 to transmit. $\odot$ denotes element-wise multiplication operation. Its communication cost can be ignored, since $\mathbb{B}$ is treated as a binary number which could be converted to one integer-type data having a negligible communication payload. Figure 4b shows examples of CutSmashed data whose black regions indicate the cut off regions. The pseudo random mask over the smashed could be created regardless of the depth of the cut layer since it is determined based on the number of patches of the smashed data.

Figure 2 shows the performance of SL and SFL training with CutSmashed data instead of smashed data in regard to the average size of CutSmashed data to identify the effect of the size of masking. The cases when the positions to be cut off are fixed for all rounds, and randomly selected at each iteration are compared. In the both cases, the performance decreases more with a larger size of the cutoff. The model cannot generalized well since the masked regions discard meaningful features of original data distribution. Interestingly, when the mask positions changed randomly, the performance is improved up to a certain extent of the cut off size. This implies that randomly generated CutSmashed data with a moderate size rather helps to prevent the model from overfitting to the local dataset acting as regularization like Cutout [DeVries and Taylor, 2017]. While the original Cutout putting one square region of mask to input, ours generate multiple masks to random locations.

In terms of the leakage of data privacy, CutSmashed data is preferable than smashed data, since partial elements are con-

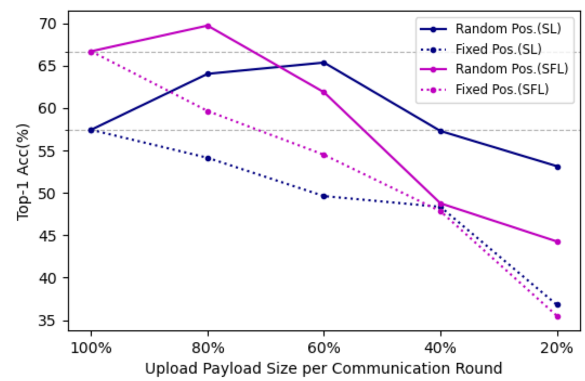


Figure 2: Performance of SL with CutSmashed data w.r.t. the average size of CutSmashed data.

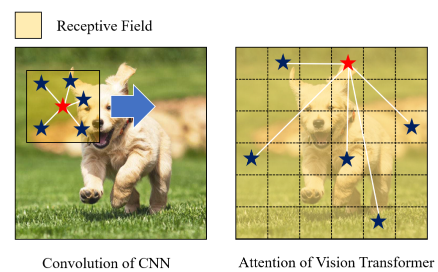


Figure 3: Operation of CNN and ViT.

cealed to the server. We elaborate on the data privacy leakage in the setting of a reconstruction attack where the attacker is willing to reconstruct an original data from uploaded smashed data. Hard to be restored by reconstruction attack implies raw data has strong privacy. In Section 4 evaluates the privacy leakage of CutSmashed data compared with the other techniques to be described in the next section.

3.2 Smashed Patch CutMix: Inter-client Mixup

The vision transformer is difficult to be generalized well in situations where there is insufficient data held by clients due to its low inductive bias [Baxter, 2000]. Inductive bias is a set of assumptions added as prior information to solve unknown machine learning problems. The weak inductive bias causes reliance on data augmentation and model regularization to gather sufficient training data [Steiner *et al.*, 2021]. Due to the data privacy, data augmentation between data held by different clients is limited in a distributed learning, making a severer problem for a transformer in a distributed learning. To overcome a limited inter-client data augmentation, and reduce the risk of the performance drop by uploading CutSmashed data, the blank parts of CutSmashed could be filled with patches of different clients' CutSmashed data.

A self-attention mechanism evaluates which patches they should pay more attention to. Combining patches from different smashed data corresponds to a new attention between data from different clients. We demonstrate this assumption by putting CutSmashed data together from different clients during FP like putting the puzzle together. Since the process resembles CutMix [Yun *et al.*, 2019] in that some region of data are cut and pasted into different data in the form of multiple patches with the same sizes, this operation is expressed as *Smashed Patch CutMix*, and its mixed smashed data as *CutMix data*. The central idea of *Smashed Patch CutMix* is to create a new smashed data to supplement the performance loss due to partial uploading with filling blanks of CutSmashed data with each other. As shown in Figure 1c, a virtual entity conducting the mix operation, *a mixer*, is conceptualized to compare with SL and SFL. A mixer could be a 3rd party, or could be divided and belong to the server and the clients. Since overlapped patches lose its clear representation, non-overlapping mix between different CutSmashed data should be guaranteed, and paired pseudo random sequences are generated for Smashed Patch CutMix.

We call k-way CutMix to represent that each k number of smashed data are mixed to generate one CutMix data. Mixing groups, $\mathbb{G} = \{\mathbf{g}_1, .., \mathbf{g}_l\}$, are generated randomly from $\mathbb{C}$ at each iteration. Then, the number of patches allocated to each CutSmashed data (i.e. mixing ratio), $\{a_1, .., a_k\}$, is determined by "Dirichlet-multinomial distribution" [Bishop *et al.*, 2007]. Here, the number of trials of the distribution is equal to M so that $\sum_{i=1}^{k} a_i = M$, and a probability vector is drawn from a Dirichlet distribution. Afterwards, patches at which positions of each smashed data are determined by the given mixing ratio. Here, the selected positions for patches of each client are not superposed with each other. Pseudo random sequence for i-th smashed data in a mixing group is denoted as $\mathbb{B}_{a_i}$, and $\mathbb{B}_{a_i}$ is determined uniformly and randomly. $\sum_{j=1}^{M} m_{a_i,j} = a_i$ where $\mathbb{B}_{a_i} = [m_{a_i,j}]_{M\times 1}$ for $i \in [k]$. $\sum_{i=1}^{k} \mathbb{B}_{a_i} = \mathbf{1}_{M\times 1}$, where $\mathbf{1}$ is a vector of all ones.

Each client uploads CutSmashed data according to the received pseudo random sequence generated and shared by a mixer. The mixer conducts Smashed Patch CutMix, and CutMix data and its corresponding label by the 2-way Smashed Patch CutMix is expressed as:

$$\tilde{\mathbf{s}}_{\{i,j\}} = \mathbf{s}'_i + \mathbf{s}'_j = \mathbb{B}_{a_i} \odot \mathbf{s}_i + \mathbb{B}_{a_j} \odot \mathbf{s}_j \tag{1}$$

$$\tilde{\mathbf{y}}_{\{i,j\}} = \frac{a_i}{M} \cdot \mathbf{y}_i + \frac{a_j}{M} \cdot \mathbf{y}_j, \tag{2}$$

where $\odot$ denotes the element-wise product.

The proposed is simple to implement with negligible extra communication cost for sharing pseudo random sequences. Intermixing CutSmashed data transmitted from each client's side enhances the server side's generalization capability acting as a data augmentation at feature space. Additionally, the impact of shuffling the order of the sequence on the performance is analyzed in detail in Appendix C.

3.3 Weight Update based on CutMix data

At the server, the upper model segment $\mathbf{w}_s$ propagates $\tilde{\mathbf{s}}_{\{i,j\}}$ uploaded by i-th and j-th client, and generates softmax output $f_{\mathbf{w}_s}(\mathbf{s}_{\{i,j\}})$. Then, the loss $\tilde{L}_{\{i,j\}}$ generated by the server model with CutMix data can be expressed as:

$$\tilde{L}_{\{i,j\}} = \frac{1}{b} \sum \mathrm{CE}\big(f_{\mathbf{w}_s}(\tilde{\mathbf{s}}_{\{i,j\}}), \tilde{\mathbf{y}}_{\{i,j\}}\big), \tag{3}$$

Algorithm 1 Operation of Mixer

function SEQUENCE GENERATION
  Sample $\{a_1, .., a_k\} \sim \mathrm{Dir}(\alpha)$ for k-way CutMix
  Generate $\{\mathbb{B}_{a_1}, .., \mathbb{B}_{a_k}\}$ uniformly at random
  return paired pseudo random sequences
end function

for $e \leftarrow 1$ to E **do**
  Generate a set of mixing groups, $\mathbb{G}$ from $\mathbb{C}$
  SEQUENCE GENERATION
  Send $\{\mathbb{B}_{a_1}, .., \mathbb{B}_{a_k}\}$ to clients based on $\mathbb{G}$
  Receive $\{(\mathbf{s}'_1, \mathbf{y}_1), .., (\mathbf{s}'_n, \mathbf{y}_n)\}$ from $\mathbb{C}$
  for $\mathbf{g} \in \mathbb{G}$ **do**
    $(\tilde{\mathbf{s}}_\mathbf{g}, \tilde{\mathbf{y}}_\mathbf{g}) \leftarrow (\sum_{j\in\mathbf{g}} \mathbf{s}'_j, \sum_{j\in\mathbf{g}} \frac{a_j}{M}\mathbf{y}_j)$
    Upload $\tilde{\mathbf{s}}_\mathbf{g}, \tilde{\mathbf{y}}_\mathbf{g}$ to the server
  end for
end for

where b is a batch size. The weights of the server and the clients are updated by BP as follows:

$$\begin{bmatrix} \mathbf{w}_s \\ \mathbf{w}_{c,i\in\mathbb{C}} \end{bmatrix} \leftarrow \begin{bmatrix} \mathbf{w}_s \\ \mathbf{w}_{c,i\in\mathbb{C}} \end{bmatrix} - \eta \begin{bmatrix} \sum_{\{i,j\}\in\mathbb{G}} (\nabla_{\mathbf{w}_s} \tilde{L}_{\{i,j\}}) \\ \nabla_{\mathbf{w}_{c,i}} \tilde{L}_{\{i,j\}} \end{bmatrix} \tag{4}$$

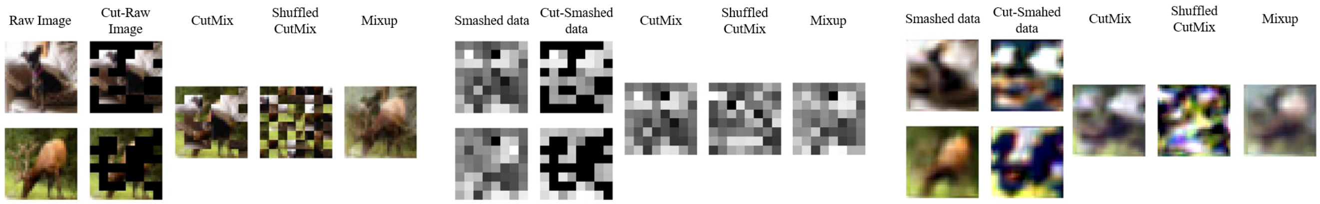
where $\mathbb{G}$ is a set of groups of clients whose smashed data are mixed, η is a learning rate, and $\nabla_{\mathbf{w}_s} \tilde{L}_{\{i,j\}}$ and $\nabla_{\mathbf{w}_{c,i}} \tilde{L}_{\{i,j\}}$ are the derivatives of the error with respect to $\mathbf{w}_s$ and $\mathbf{w}_{c,i}$, respectively. The server sends the gradients of $\tilde{L}_{i,j}$ with respect to the uploaded smashed data $\mathbf{s}'_i$ and $\mathbf{s}'_j$, $\nabla_{\mathbf{s}_i} \tilde{L}_{i,j}$ and $\nabla_{\mathbf{s}_j} \tilde{L}_{i,j}$, to the corresponding clients, and the clients calculates $\nabla_{\mathbf{w}_{c,i}} \tilde{L}_{\{i,j\}}$, which is given as:

$$\nabla_{\mathbf{w}_{c,i}} \tilde{L}_{i,j} = \frac{\partial \tilde{L}_{\{i,j\}}}{\partial \mathbf{s}'_i} \frac{\partial \mathbf{s}'_i}{\partial \mathbf{w}_{c,i}}. \tag{5}$$

The method above aims to resolve a server-clients update imbalance problem, hindering the scalability of parallel split learning [Oh *et al.*, 2022]. The more clients, it incurs the more imbalanced updates between the server and the clients. There are n times of update on the upper model segment, whereas each client updates its model once. Meanwhile, with our proposed method, the server executes its server model's update only once on a unit mixed smashed data, as shown in Figure 1c. Therefore, when all clients update their parameters with CutMix data, n times updates are in the server model reduce to $\frac{n}{k}$ times updates for k-way CutMix. The CutMixSL (k times) in Table 2 is the case when gradients from one CutMix data flow to only one client and the server updates n times in total like SL. Compared with CutMixSL in Table 2 where the server updates $\frac{n}{k}$ times, it shows that the reduced update of the server by CutMix data has a positive impact on a performance gain.

4 Evaluation

The total number of devices is ten in our setting. The total epoch is 600 with 5 warmup epochs, the learning rate is 0.001,



(a) Raw images. (b) Smashed data. (c) Reconstruction from smashed data.

Figure 4: Examples of data with different operations at different levels.

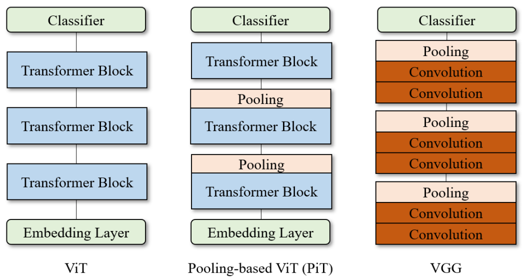
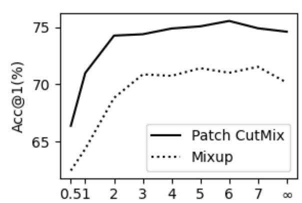


Figure 5: Simplified model architectures of ViT, PiT, and VGG.

Method	Models		
	ViT-Tiny	PiT-Tiny	VGG-16
Standalone	49.14	47.77	54.97
Parallel SL	57.05	52.28	62.62
Cutout	65.03	60.87	67.06
Mixup	71.02	65.92	**74.43**
Patch CutMix	**75.55**	**73.19**	72.23
Shuffled CutMix	72.78	57.59	33.50

Table 1: Performance w.r.t. mixing methods.



(a) Top-1 Accuracy w.r.t. α for mixing ratio.

Type	Mixing Ratio	
	Uniform	Dir($\alpha = 6$)
2-way	**70.93**	**75.55**
3-way	68.59	70.86
4-way	66.96	66.67
5-way	67.01	67.04

(b) Top-1 Accuracy w.r.t. a mixing group size.

Figure 6: Performance w.r.t. mixing ratio and group size.

the optimizer is adamW, a batch size is 128, and a scheduler is cosine annealing. Our benchmark is CIFAR 10 to reduce the computation overload compared to ImageNet, and the patch size is 4×4 in our settings. A transformer receives input as a 1D sequence of token embeddings, and a vision transformer divides an image into a sequence of M flattened 2D patches. ViT-Tiny [Touvron *et al.*, 2020] is used in our settings, consists of transformer encoder blocks, whose the depth of the transformer is 6 blocks to eliminate unnecessary computation burden to process CIFAR-10 dataset. Each client has an embedding layer, concatenates a class token and the patch tokens, adds positional embedding, and uploads its activations to the server. The server has the rest of the model. Each client has 5,000 images.

Impact of Mixing Methods To analyze the effectiveness of Smashed Patch CutMix on the transformer, we evaluated different types of mix operations on both transformer-based models and a CNN-based model. ViT-Tiny for a pure transformer, PiT-Tiny [Heo *et al.*, 2021] for a pooling-based transformer, and VGG-16 [Simonyan and Zisserman, 2014] for CNN are evaluated whose simplified model architectures are described in Figure 5. Recently, many ViT models with hierarchical representations are proposed such as Swin Transfromer [Liu *et al.*, 2021], and T2T-ViT [Yuan *et al.*, 2021] including PiT. VGG is a representative model of CNN composed of only convolutional layers, pooling layers and a classifier, and the cut-layer is after two convolutional layers and one pooling layer.

Table 1 shows the performances of Cutout (CutSmashed data), Smashed Patch CutMix (CutSmashed data + CutMix), Mixup (Smashed data + Mixup), and Shuffled CutMix (CutSmashed data + CutMix + Shuffling). Smashed Patch CutMix has 18.5% and 20.9% performance gain in ViT and PiT compared to parallel SL. It is higher than Mixup for ViT and PiT (4% and 8%, respectively), and 2% lower than Mixup for VGG-16. Masking such as CutMix is better at preserving local features than interpolation such as mixup without distorting data distribution [Harris *et al.*, 2020]. Transformers process data as divided separate patches, and even though randomly chosen patches to be cut and pasted destroy a global structure of image, transformers do not lose its inference ability because of parallel processing for an entire range of sequence by attention mechanism. In this sense, masking smashed data is more suitable than interpolation to transformers. On the other hand, CNN's core operation is a sliding convolutional filter, and one of assumptions CNN poses is a locality of pixel dependencies. Neighboring pixels which tend to be correlated, get uncorrelated by masking due to junctions of different clients' patches. Notwithstanding, the result shows that the impact of masking is not big enough to degrade the performance of CNN. We guess that the unchanged position of patches from the original location keeping its architecture of the data results in a positive impact to the performance gain.

Impact of Mixing Ratio & Group Size In Figure 6a, the performance of CutMixSL with respect to the parameter, α,

Method	# of clients				
	2	4	6	8	10
Parallel SL	52.91	55.55	56.81	56.55	57.05
SplitFed	53.96	58.39	63.07	65.59	66.70
CutMixSL (k times, uniform)	58.06	65.11	65.76	68.97	67.66
CutMixSL (uniform)	61.66	66.80	67.69	69.86	70.93
CutMixSL (α=6)	62.62	69.47	71.80	73.67	75.55
CutMixSFL (uniform)	65.82	72.31	74.31	78.88	80.14
CutMixSFL (α=6)	67.58	73.52	76.85	79.48	80.97

Table 2: Scalability. Top-1 accuracy w.r.t. the # of clients.

of Dirichlet-multinomial distribution, is evaluated to identify the sensitivity of the operation to the mixing ratio. The higher α, the higher the probability of even mixing ratio. The result shows that the higher probability of evenly mix gains more accuracy than the lower one. The top-1 accuracy when $\alpha \to \infty$, which is the number of sampled patches is fixed to even, is lower than the case for the lower α. It implies that having an unbalanced mixture with a moderate probability is preferable for a higher performance. Figure 6b evaluated a performance of the proposed in regard to the different size of mixing group with a uniform distributions and the dirichlet distribution with $\alpha = 6$. It is expected that the performance would be improved with the larger mixing group size from the standpoint of update imbalance problem, since the server updates less. Nevertheless, 2-way CutMix has a best performance gain, resulting from the fact that overly mixing with different classes causes the server cannot learn each class's distinct features.

Scalability In Table 2, while all evaluated methods are scalable up to at least 10 clients, each has a different performance gain at scale. The brackets in Table 2 represents the distribution that a mixing ratio is sampled from. Parallel SL has a weak scalable performance gain (4%) due to the update imbalance problem, whereas CutMixSL has a higher scalable performance gain than parallel SL (12.9%) and even SFL (12.7%), and an absolute accuracy is also higher than SFL. For CutMixSFL, which is CutMixSL with FedAvg on the lower models, mixing activations and FedAvg are overlapped, and CutMixSFL achieves 13.4% performance gain at scale and about 80% top-1 accuracy with ten clients.

Communication Efficiency. Compared to Parallel SL requiring entire smashed data, one of the advantages of the proposed technique is a large reduction of upload payload size with an enhanced accuracy. Figure 7 shows an uploading payload size per communication round and its performance of CutMixSL and its derivatives. The black line shows an accuracy of SL with CutSmashed data regarding λ, the ratio of the size of CutSmashed data to the one of the original smashed data, described in Figure 2. CutMixSL shows the highest performance and upload payload reduction up to 20 - 50%. The more mixing group size increases, the higher communication cost reduction can be achieved. In our settings, one unmatched client of 3-way sends smashed data, and two unmatched clients in 4-way CutMix operate 2-way CutMix.

Privacy Leakage Data privacy leakage increases with the mutual information between the CutSmashed data and its raw data. Due to the unknown data distributions, the mutual information is often approximated by the error when reconstructing the input data [Vepakomma *et al.*, 2020]. In this respect, following [Wang *et al.*, 2021], we train an autoencoder whose input is the CutSmashed data and the output is the original raw data. The trained autoencoder's loss, given as the mean squared error (MSE), can thereby treated as the amount of privacy leakage to the server.

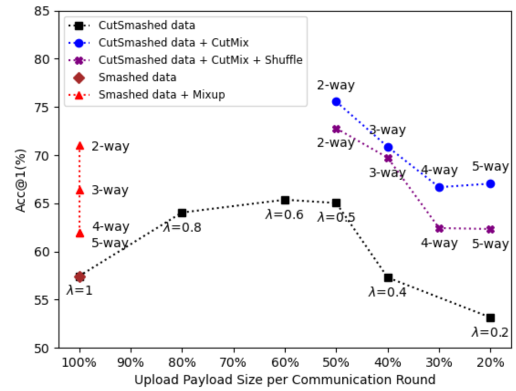


Figure 7: Upload payload size per communication round of CutMixSL and its derivatives.

Type	Train Dataset(10%)	Train Dataset(100%)
Smashed data	0.0091	0.0056
CutSmashed data	**0.0920**	**0.0829**
Mixup	0.0402	0.0351
Patch CutMix	0.0458	0.0434
Shuffled CutMix	**0.1233**	**0.1250**

Table 3: Privacy leakage measured by the reconstruction loss (MSE).

Table 3 shows the privacy leakage of 2-way CutMixSL and its variants on a test set when the autoencoder trains 10% of a train set, and when it trains 100% of a train set. As expected intuitively, CutSmashed data reduces privacy leakage by 15 $\times$ compared to Smashed data. For the mix operations, Smashed Patch CutMix is better than Mixup, and this gap is highly amplified when the order of its sequences is shuffled. Compared to the baseline, CutMixSL reduce data privacy by around 8 times. Masking data and shuffling are two principal factors for privacy enhancement. Figure 4c shows some examples of reconstructed images by the reconstruction attack for a qualitative comparison.

5 Conclusion

In this work, we proposed CutSmashed data to resolve data privacy leakage and improve communication efficiency of split learning motivated by a vision transformer processing data as a sequence of patches. Furthermore, CutMixSL is introduced with Smashed patch CutMix, smashed data augmentation, to deploy a transformer-based model in split learning. We analyzed the design elements' impact on the proposed operation, and confirmed that the proposed approach has advantages in data privacy, performance gain, and communication efficiency compared to Parallel SL and SplitFed.

Acknowledgments

This work was partly supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2021-0-00347, 6G Post-MAC (POsitioning- & Spectrum-aware intelligenT MAC for Computing & Communication Convergence)), and Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (1711134613, 2021-0-00270, Development of 5G MEC framework to improve food factory productivity, automate and optimize flexible packaging).

References

[Baxter, 2000] Jonathan Baxter. A model of inductive bias learning. *Journal of artificial intelligence research*, 12:149–198, 2000.

[Bishop *et al.*, 2007] Yvonne M Bishop, Stephen E Fienberg, and Paul W Holland. *Discrete multivariate analysis: theory and practice*. Springer Science & Business Media, 2007.

[Brown *et al.*, 2020] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[Carion *et al.*, 2020] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020.

[Devlin *et al.*, 2018] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[DeVries and Taylor, 2017] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

[Dosovitskiy *et al.*, 2020a] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[Dosovitskiy *et al.*, 2020b] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, abs/2010.11929, 2020.

[Gao *et al.*, 2021] Yansong Gao, Minki Kim, Chandra Thapa, Sharif Abuadbba, Zhi Zhang, Seyit Camtepe, Hyoungshick Kim, and Surya Nepal. Evaluation and optimization of distributed machine learning techniques for internet of things. *IEEE Transactions on Computers*, 2021.

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

[Han *et al.*, 2022] Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, et al. A survey on vision transformer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[Harris *et al.*, 2020] Ethan Harris, Antonia Marcu, Matthew Painter, Mahesan Niranjan, Adam Prügel-Bennett, and Jonathon S. Hare. Understanding and enhancing mixed sample data augmentation. *CoRR*, abs/2002.12047, 2020.

[Heo *et al.*, 2021] Byeongho Heo, Sangdoo Yun, Dongyoon Han, Sanghyuk Chun, Junsuk Choe, and Seong Joon Oh. Rethinking spatial dimensions of vision transformers. *CoRR*, abs/2103.16302, 2021.

[Hong *et al.*, 2021] Zhenhou Hong, Jianzong Wang, Xiaoyang Qu, Jie Liu, Chendong Zhao, and Jing Xiao. Federated learning with dynamic transformer for text to speech. *arXiv preprint arXiv:2107.08795*, 2021.

[Karita *et al.*, 2019] Shigeki Karita, Nanxin Chen, Tomoki Hayashi, Takaaki Hori, Hirofumi Inaguma, Ziyan Jiang, Masao Someki, Nelson Enrique Yalta Soplin, Ryuichi Yamamoto, Xiaofei Wang, et al. A comparative study on transformer vs rnn in speech applications. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 449–456. IEEE, 2019.

[Khan *et al.*, 2021] Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. Transformers in vision: A survey. *ACM Computing Surveys (CSUR)*, 2021.

[Konečnỳ *et al.*, 2015] Jakub Konečnỳ, Brendan McMahan, and Daniel Ramage. Federated optimization: Distributed optimization beyond the datacenter. *arXiv preprint arXiv:1511.03575*, 2015.

[Konečnỳ *et al.*, 2016] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[Li *et al.*, 2020] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020.

[Liu *et al.*, 2021] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF Inter-*

national Conference on Computer Vision, pages 10012–10022, 2021.

[Naseer *et al.*, 2021] Muhammad Muzammal Naseer, Kanchana Ranasinghe, Salman H Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang. Intriguing properties of vision transformers. *Advances in Neural Information Processing Systems*, 34, 2021.

[Oh *et al.*, 2022] Seungeun Oh, Jihong Park, Praneeth Vepakomma, Sihun Baek, Ramesh Raskar, Mehdi Bennis, and Seong-Lyun Kim. Locfedmix-sl: Localize, federate, and mix for improved scalability, convergence, and latency in split learning. In *Proceedings of the ACM Web Conference 2022*, pages 3347–3357, 2022.

[Pal *et al.*, 2021] Shraman Pal, Mansi Uniyal, Jihong Park, Praneeth Vepakomma, Ramesh Raskar, Mehdi Bennis, Moongu Jeon, and Jinho Choi. Server-side local gradient averaging and learning rate acceleration for scalable split learning. *arXiv preprint arXiv:2112.05929*, 2021.

[Park *et al.*, 2021a] Jihong Park, Sumudu Samarakoon, Anis Elgabli, Joongheon Kim, Mehdi Bennis, Seong-Lyun Kim, and Mérouane Debbah. Communication-efficient and distributed learning over wireless networks: Principles and applications. *Proceedings of the IEEE*, 109(5):796–819, 2021.

[Park *et al.*, 2021b] Sangjoon Park, Gwanghyun Kim, Jeongsol Kim, Boah Kim, and Jong Chul Ye. Federated split task-agnostic vision transformer for covid-19 cxr diagnosis. *Advances in Neural Information Processing Systems*, 34, 2021.

[Qu *et al.*, 2021] Liangqiong Qu, Yuyin Zhou, Paul Pu Liang, Yingda Xia, Feifei Wang, Li Fei-Fei, Ehsan Adeli, and Daniel Rubin. Rethinking architecture design for tackling data heterogeneity in federated learning. *arXiv preprint arXiv:2106.06047*, 2021.

[Simonyan and Zisserman, 2014] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[Steiner *et al.*, 2021] Andreas Steiner, Alexander Kolesnikov, Xiaohua Zhai, Ross Wightman, Jakob Uszkoreit, and Lucas Beyer. How to train your vit? data, augmentation, and regularization in vision transformers. *CoRR*, abs/2106.10270, 2021.

[Thapa *et al.*, 2020] Chandra Thapa, Mahawaga Arachchige Pathum Chamikara, and Seyit Camtepe. Splitfed: When federated learning meets split learning. *CoRR*, abs/2004.12088, 2020.

[Touvron *et al.*, 2020] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. *CoRR*, abs/2012.12877, 2020.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

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

[Vepakomma *et al.*, 2020] Praneeth Vepakomma, Abhishek Singh, Otkrist Gupta, and Ramesh Raskar. Nopeek: Information leakage reduction to share activations in distributed deep learning. In *2020 International Conference on Data Mining Workshops (ICDMW)*, pages 933–942. IEEE, 2020.

[Verma *et al.*, 2019] Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. Manifold mixup: Better representations by interpolating hidden states. In *International Conference on Machine Learning*, pages 6438–6447. PMLR, 2019.

[Wang *et al.*, 2021] Yulin Wang, Zanlin Ni, Shiji Song, Le Yang, and Gao Huang. Revisiting locally supervised learning: An alternative to end-to-end training. *arXiv preprint arXiv:2101.10832*, 2021.

[Xiao *et al.*, 2021] Danyang Xiao, Chengang Yang, and Weigang Wu. Mixing activations and labels in distributed training for split learning. *IEEE Transactions on Parallel and Distributed Systems*, 2021.

[Yuan *et al.*, 2021] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zi-Hang Jiang, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 558–567, 2021.

[Yun *et al.*, 2019] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6023–6032, 2019.

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

[Zheng *et al.*, 2021] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip HS Torr, et al. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6881–6890, 2021.

A Pseudo Algorithm of CutMixSL

Algorithm 2 CutMixSL

function SEQUENCE GENERATION
  Sample $\{a_1, .., a_k\} \sim \text{Dir}(\alpha)$ for k-way CutMix
  Generate $\{\mathbb{B}_{a_1}, .., \mathbb{B}_{a_k}\}$ uniformly at random
  return paired pseudo random sequences
end function

for $e \leftarrow 1$ to E **do**
  /*Runs on mixer*/
  Generate a set of mixing groups, $\mathbb{G}$ from $\mathbb{C}$
  SEQUENCE GENERATION
  Send $\{\mathbb{B}_{a_1}, .., \mathbb{B}_{a_k}\}$ to clients based on $\mathbb{G}$
  Receive $\{(\mathbf{s}'_1, \mathbf{y}_1), .., (\mathbf{s}'_n, \mathbf{y}_n)\}$ from $\mathbb{C}$
  for $\mathbf{g} \in \mathbb{G}$ **do**
    $(\tilde{\mathbf{s}}_\mathbf{g}, \tilde{\mathbf{y}}_\mathbf{g}) \leftarrow (\sum_{j\in\mathbf{g}} \mathbf{s}'_j, \sum_{j\in\mathbf{g}} \frac{a_j}{M}\mathbf{y}_j)$
    Upload $\tilde{\mathbf{s}}_\mathbf{g}, \tilde{\mathbf{y}}_\mathbf{g}$ to the server
  end for

  /*Runs on clients*/
  for each client $i \in \mathbb{C}$ in parallel **do**
    Receive $\mathbb{B}_i$ from mixer
    $\mathbf{s}_i \leftarrow f_{\mathbf{w}_{c,i}}(\mathbf{x}_i); \mathbf{s}'_i \leftarrow \mathbb{B}_i \odot \mathbf{s}_i$
    Upload $(\mathbf{s}'_i, \mathbf{y}_i)$ to mixer
    Receive $\nabla_{\mathbf{s}_i}\tilde{L}_\mathbf{g}$; Calculate gradient $\nabla_{\mathbf{w}_{c,i}}\tilde{L}_\mathbf{g}$
    Weight Update $\mathbf{w}_{c,i} \leftarrow \mathbf{w}_{c,i} - \eta\nabla_{\mathbf{w}_{c,i}}\tilde{L}_\mathbf{g}$
  end for

  /*Runs on server*/
  for $\mathbf{g} \in \mathbb{G}$ **do**
    Receive $(\tilde{\mathbf{s}}_\mathbf{g}, \tilde{\mathbf{y}}_\mathbf{g})$ from mixer
    $\tilde{L}_\mathbf{g} \leftarrow \frac{1}{b}\sum_{\{\tilde{\mathbf{s}}_\mathbf{g}, \tilde{\mathbf{y}}_\mathbf{g}\}} \text{CE}(f_{\mathbf{w}_s}(\tilde{\mathbf{s}}_\mathbf{g}), \tilde{\mathbf{y}}_\mathbf{g})$
    Calculate gradient $\nabla_{\mathbf{w}_s}\tilde{L}_\mathbf{g}$
    Weight Update $\mathbf{w}_s \leftarrow \mathbf{w}_s - \eta\nabla_{\mathbf{w}_s}\tilde{L}_\mathbf{g}$
    Download $\nabla_{\mathbf{s}_i}\tilde{L}_\mathbf{g}$ to client $i \in \mathbf{g}$
  end for
end for

B Mixing Methods

Data augmentation can generate new samples through mixing different samples, and there are generally two types of data augmentation: masking and interpolation. The typical techniques of masking are CutMix [Yun *et al.*, 2019] and Cutout [DeVries and Taylor, 2017], and the one of interpolation is Mixup [Zhang *et al.*, 2017].

Mixup is an entire interpolation of two given raw data, x_i and x_j with a certain ratio λ, and is expressed as follows:

$$\mathbf{x}_{mixup} = \lambda\mathbf{x}_i + (1-\lambda)\mathbf{x}_j. \tag{6}$$

In [Verma *et al.*, 2019], Manifold Mixup, which is Mixup in the intermediate feature space of the deep neural network (DNN), has been proposed. Manifold Mixup is expressed as follows:

$$\mathbf{s}_{mixup} = \lambda\mathbf{s}_i + (1-\lambda)\mathbf{s}_j. \tag{7}$$

The corresponding label for Mixup and Manifold Mixup is expressed as follows:

$$\mathbf{y}_{mixup} = \lambda\mathbf{y}_i + (1-\lambda)\mathbf{y}_j. \tag{8}$$

As a masking technique, Cutout [DeVries and Taylor, 2017] masks out square regions of input, and is expressed as follows:

$$\mathbf{x}_{cutout} = \mathbf{M} \odot \mathbf{x}, \tag{9}$$

where $\mathbf{M} \in \{0,1\}^{W\times H}$ is a binary mask indicating which pixel is to be dropped out. $\mathbf{M}$ fills 0 inside the bounding box coordinates, $\mathbf{B} = (r_x, r_y, r_w, r_h)$, indicating the cropping region of the image.

As one step further, CutMix drops a unit square region of random size, and fills in the blanks with a different raw image, and is expressed as following:

$$\begin{aligned}\mathbf{x}_{cutmix} &= \mathbf{M} \odot \mathbf{x}_i + (1-\mathbf{M})\mathbf{x}_j \\ \mathbf{y}_{cutmix} &= \lambda\mathbf{y}_i + (1-\lambda)\mathbf{y}_j,\end{aligned} \tag{10}$$

where $\lambda = \frac{r_w r_h}{WH}$, is a mixing ratio.

Smashed Patch CutMix, we proposed, is operated in the feature space and is expressed as follows:

$$\begin{aligned}\mathbf{s}_{cutmix} &= \mathbf{M} \odot \mathbf{s}_i + (1-\mathbf{M})\mathbf{s}_j \\ \mathbf{y}_{cutmix} &= \lambda\mathbf{y}_i + (1-\lambda)\mathbf{y}_j\end{aligned} \tag{11}$$

The proposed masks random number of patches with other client's patches, which is similar to the CutMix except for the number of patches to be replaced; ours use multiple fixed-size patches.

Shuffled CutMix conducts shuffling after mixing smashed data by Smashed patch CutMix, and is expressed as follows:

$$\mathbf{s}_{shuffle} = \mathbf{S}(\mathbf{s}_{cutmix}), \tag{12}$$

where $\mathbf{S}$ is a shuffle operation on $\mathbf{s}_{cutmix} = [e_1, ..., e_M] \in \mathcal{R}^{M\times d_m}$. e_i is the i-th embedding vector, d_m is the size of a vector of a patch, and $\mathbf{S}$ shuffles the sequence of the embedding vectors of $\mathbf{s}_{cutmix}$. For CNN, activations can be reshaped to a 2D dimension like activaitons in a vision transformer by dividing it with a given patch size, and aligning it in parallel.

C Add-On Experiments

Shuffling Transformers process data as a long-range sequence in parallel by attention mechanism, and are not unaffected by sequence order. This high permutation invariance also applies to ViTs [Naseer *et al.*, 2021], and could bring benefit on reducing privacy leakage by reconstruction attacks in SL since shuffling patches can destroy images' overall structure as shown in Figure 4a. While all patches retain their own positions during mixing, we can extend it to a shuffled version of Smashed Patch CutMix, *Shuffled CutMix*, in short, for a further privacy enhancement.

The last row of Table 1 shows that the influence of shuffling an order of patches of CutMix data. ViT keeps its performance in spite of shuffling with a slight performance loss by its capability of high permutation invariance. However, PiT has a pooling operation blurring shuffled patches and lose their own distinct features, and the decline of a performance is more serious for CNN due to a consequence of convolutional filters in addition to pooling. Nevertheless, a considerable profit by shuffling is an improvement of reconstruction mitigation as shown in Table 3. It reduces the privacy leakage by around 22 $\times$ compared to the the baseline (SL with smashed data).

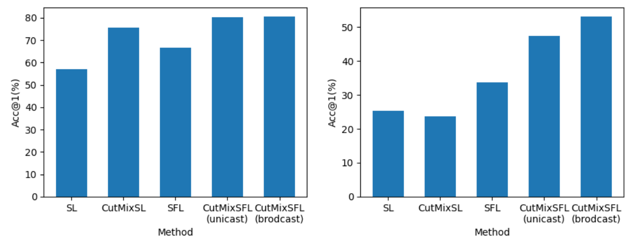
FedAvg FedAvg on clients can be utilized to enhance the performance gain and to be tolerant to data distribution shifts. When data are non-IID to clients, each client contains one or two dominant classes' data, training its one-sided classes predominantly. A mix operation like Smashed Patch CutMix can alleviate unbalanced training in cooperation with FedAvg. In Figure 8b, dirichlet distribution with concentration parameter, μ, 0.1 is used to formulate the non-IID case. The result indicates that the degenerated performance of SL and SFL by non-IID condition can be less worsened by Smashed Patch CutMix and FedAvg.

During BP through CutMix data, the flow of gradients to each client can be calculated from the perspective of the server and and the perspective of clients. [Pal *et al.*, 2021] uses local gradient averaging for broadcasting the gradients to clients. Likewise, the gradients from CutMix data can flow as unicast and broadcast.

$$\nabla_{\mathbf{w}_{c,i}} \tilde{L}_{i,j} \approx \begin{cases} \frac{\partial \tilde{L}_{\{i,j\}}}{\partial \mathbf{s}'_i} \frac{\partial \mathbf{s}'_i}{\partial \mathbf{w}_{c,i}}; & \text{for unicast} \\ \frac{\partial \tilde{L}_{\{i,j\}}}{\partial \mathbf{s}_{\{i,j\}}} \frac{\partial \tilde{\mathbf{s}}_{\{i,j\}}}{\partial \mathbf{w}_{c,i}}; & \text{for broadcast} \end{cases}. \tag{13}$$

The unicast case is that the gradients is calculated in the perspective of clients, and the server knows paired pseudo random sequences indicating which patches each client uploaded. The server sends individual gradients, $\nabla_{\tilde{s}_i} \tilde{L}_{i,j}$ and $\nabla_{\tilde{s}_j} \tilde{L}_{i,j}$, to i-th and j-th clients, respectively, according to the corresponding portion in CutMix data by unicast. For the broadcast case, the server is assumed not to know the pseudo random sequences, thereby the gradient with respect to the whole region of CutMix data is sent during BP, and it is named as *CutMix Gradient*. It broadcasts combined gradients, $\nabla_{\tilde{s}_{\{i,j\}}} \tilde{L}_{i,j}$, to the clients whose CutMix data are generated from. The communication payload for back propagation are the same for both cases. CutMix Gradient could be used when the server does not know the pseudo random sequences.

Although the difference of the performances is negligible in the IID condition, the performance of CutMixSFL with CutMix gradients is 6% higher than the one with the gradients with respect to CutSmashed data in the non-IID condition. It could be interpreted that CutMix Gradients by the mixed smashed data is more effective in that each model trains data of scarce classes indirectly through CutMix gradient with the help of FedAvg.



(a) IID. (b) non-IID($\mu = 0.1$).

Figure 8: Performance w.r.t. IID and non-IID data distribtuion.

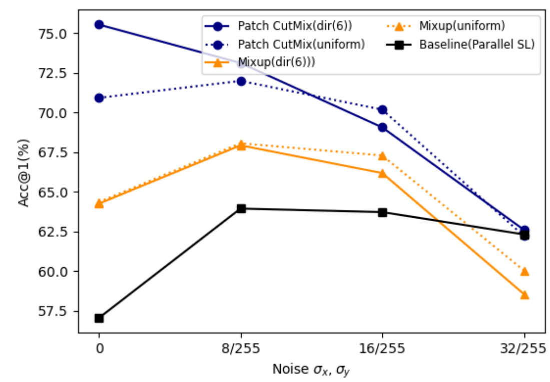
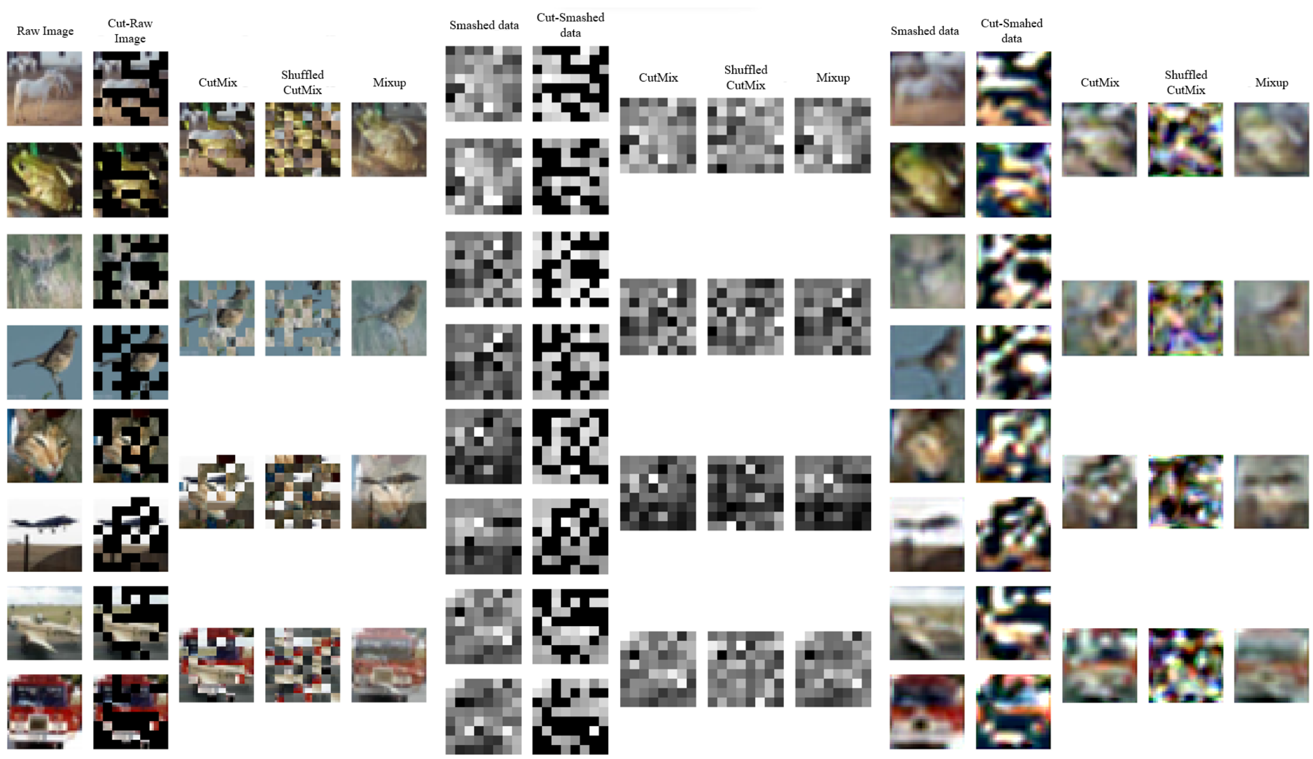


Figure 9: Utility according to the scale of gaussian noise added to the smashed data and its label.

Noise Injection Many privacy-preserving data mining techniques involve noise injection, such as differential privacy, to randomly distort and mask data reducing the distance correlation between an output of a mechanism and a raw input data. In Figure 9, the utility of mixing operations are evaluated when additive white gaussian noises are integrated to the smashed data and its corresponding label for data privacy. It shows a comparison of utility of Smashed Patch CutMix, Smashed Mixup, and the baseline (Parallel SL) according to the scale, σ_x and σ_y of the noise distribution on the raw data and its label, respectively. The higher σ preserves stronger privacy at the larger cost of utility. The result indicates the proposed has a higher utility compared to the others, even though the gap is decreased with a higher σ. One interesting founding is that two cases except for the proposed show utility gains when a small amount of noise is injected. It could be explained that two methods get a positive effect by regularization making the model less certain of its predictions, while the proposed does not so since it already gets enough regularization and data augmentation effect.

D Additional Visualization of images

Additional examples of mix operations on raw images, smashed data and its corresponding reconstruction images are shown in the next page. All the images shown in Figure 10 are based on mixtures of two images.



(a) Raw images. (b) Smashed data. (c) Reconstruction from smashed data.

Figure 10: Additional examples of data with different operations at raw images, smashed data, and reconstructed images.