Multi-Task Distributed Learning Using Vision Transformer With Random Patch Permutation

Sangjoon Park and Jong Chul Ye, Fellow, IEEE

Abstract — The widespread application of artificial intelligence in health research is currently hampered by limitations in data availability. Distributed learning methods such as federated learning (FL) and split learning (SL) are introduced to solve this problem as well as data management and ownership issues with their different strengths and weaknesses. The recent proposal of federated split task-agnostic (FESTA) learning tries to reconcile the distinct merits of FL and SL by enabling the multi-task collaboration among participants through Vision Transformer (ViT) architecture, but they suffer from higher communication overhead. To address this, here we present a multi-task distributed learning using ViT with random patch permutation, dubbed p-FESTA. Instead of using a CNN-based head as in FESTA, p-FESTA adopts a simple patch embedder with random permutation, improving the multi-task learning performance without sacrificing privacy. Experimental results confirm that the proposed method significantly enhances the benefit of multi-task collaboration, communication efficiency, and privacy preservation, shedding light on practical multi-task distributed learning in the field of medical imaging.

Index Terms — Federated learning, split learning, multi-task learning, vision transformer, privacy preservation.

I. INTRODUCTION

Artificial intelligence (AI) has been gaining unprecedented popularity thanks to its potential to revolutionize various fields of data science. Specifically, the deep neural network has attained expert-level performances in the various applications of medical imaging [1], [2].

To enable the AI models to offer precise decision support with robustness, an enormous amount of data are indispensable. However, data collected from volunteer participation of only a few institutions cannot fully meet the amount to guarantee robust performances. Even for the large public datasets, it may inevitably include unquantifiable biases stemming from the limited geographic regions and patient demographics such as ethnicities and races, resulting in performance instability in real-world applications. Especially for the newly emerging disease like Coronavirus disease 19 (COVID-19), this limitation can be exacerbated as it is hard to promptly build a large, well-curated dataset with sufficient diversity.

Therefore, the ability to collaborate between multiple institutions is critical for the successful application of AI in medical imaging, but the rigorous regulations and the ethical restrictions for sharing patient data are other obstacles to multi-institutional collaborative work. Several formal regulations and guidelines, such as the United States Health Insurance Portability and Accountability Act (HIPAA) [3] and the European General Data Protection Regulation (GDPR) [4], state the strict regulations regarding the storage and sharing of patient data.

Accordingly, distributed learning methods, which perform learning tasks at edge devices in a distributed fashion, can be effectively utilized in healthcare research [5], [6]. Specifically, distributed learning was introduced to enable the model training with data that reside on the source devices without sharing. Federated learning (FL) is one of these methods that enables distributed clients to collaboratively learn a shared model without sharing their training data [7]. However, it still holds several limitations in that it is heavily dependent on the client-side computation resources for parallel computation and not completely free from privacy concerns with gradient inversion attack [8], [9] or the membership inference attack [10], [11], [12], as the entire model is trained in client-side and transmitted between the server and clients. Another distributed learning method, split learning (SL) [13], which splits the network into parts between clients and the server, is a promising method that puts low computational loads at the edge devices. However, it has the disadvantage of high communication overhead between the clients and server [14], shows significantly slower convergence compared with FL, and shows suboptimal performance given the significantly skewed data distribution between clients [15]. In addition, it also has the limitation in privacy preservation as the private data can be recovered by the malicious attacker with

This work was supported in part by the M.D.-Ph.D./Medical Scientist Training Program through the Korea Health Industry Development Institute (KHIDI), funded by the Ministry of Health and Welfare, South Korea; in part by the National Research Foundation of Korea under Grant NRF-2020R1A2B5B03001980; and in part by the Korea Advanced Institute of Science and Technology (KAIST), Daejeon 34141, South Korea (e-mail: depecher@kaist.ac.kr).

Jong Chul Ye is with the Kim Jaechul Graduate School of AI, Korea Advanced Institute of Science and Technology (KAIST), Daejeon 34141, South Korea (e-mail: jong.ye@kaist.ac.kr).

Digital Object Identifier 10.1109/TMI.2022.3218783
the feature-space attack [16], [17]. The privacy enhancing methods like differential privacy can be used along with these distributed learning methods [18], but it imposes the trade-off between the privacy and the performances.

Inspired by the modular decomposition structure of Vision Transformer (ViT), a novel distributed learning method dubbed Federated Split Task-Agnostic learning (FeSTA) was recently proposed for distributed multi-task collaboration using ViT architecture [19]. The FeSTA framework, equipped with the shared task-agnostic ViT body on the server-side and multiple task-specific convolutional neural network (CNN) heads and tails on the clients-side, was able to balance the merit of FL and SL, thereby improving the performances of individual tasks under distributed multi-task collaboration setting at a level even better than the single-task expert model trained in a data-centralized manner.

Nevertheless, there remain several critical limitations with the FeSTA framework. First, the communication overhead is higher than that of SL and FL, as the model should continuously share features and gradients as well as head and tail parts of the network, which may impose difficulties in practical implementation. Second, we found that the large size head and tail parts in the original FeSTA tend to reduce the role of the shared body, resulting in a small improvement compared to single-task learning despite the ViT’s potential for multi-task learning (MTL). Finally, the FeSTA framework was not free from the privacy issue, as the features transmitted to the server body can be hijacked and reverted to the original data by the outside malicious attackers or “honest-but-curious” server in the same manner in SL.

To alleviate these drawbacks, here we introduce p-Festa framework, a Federated Split Task-Agnostic learning with permuting pure ViT, which empowers communication efficient MTL with privacy-preservation. Although the overall composition of p-Festa is similar to that of FeSTA, instead of using a CNN based head, p-Festa adopts a fixed task agnostic patch embedder, enforcing the self-attention within the transformer architecture to improve the MTL performance. For privacy preservation, we introduce a feature-space permutation module that randomly shuffles the order of all patch features ahead of sending them to the server, to prevent either an outside attacker or an “honest but curious” server from reverting features into original data containing privacy.

The contributions of the proposed p-Festa can be summarized as follows:

- Privacy is enhanced with a simple but effective feature-space permutation module leveraging the intrinsic property of ViT.
- Communication cost is substantially reduced by saving features to be used throughout the entire learning process.
- Benefit with MTL is enhanced by employing a simple, fixed patch embedder and enforcing the multi-task body to do the heavy lifting.

II. RELATED WORKS

A. Vision Transformer (ViT)

ViT [20], a recently introduced deep learning model equipped with an exquisite attention mechanism inspired by its successful application in natural language processing (NLP), has demonstrated impressive performances across many vision tasks. The multi-head self-attention in ViT can flexibly attend to a sequence of patches of the image to encode the cue, enabling the model to be robust to nuisances like occlusion, spatial permutation, and adversarial perturbation, and thereby having the model be more shape-biased like a human than CNN-based model [21].

In addition, the modular design of the ViT is straightforward, implying that the components can be easily decomposed into parts: head to project the images patches into embeddings, transformer body to encode the embeddings, and tail to yield task-specific output. This easily decomposable design offers the possibility in the application for MTL. Recall that the motivation of MTL originated from attempts to mitigate the data insufficiency problem where the numbers of data for individual tasks are limited. MTL can offer the advantage of improving data efficiency, reducing overfitting through shared representation, and faster convergence by leveraging auxiliary knowledge.

Specifically, MTL with transformer-based models has emerged as a popular approach to improve the performances of the closely related task in NLP [22], [23]. In this approach, a shared transformer learns several related tasks simultaneously, like sentence classification and word prediction, and the task-specific modules yield the outcome for each task. As shown in previous literature [23], the model trained with MTL strategy generally shows improved performances in a wide range of tasks. Even though not well been studied as in language, the decomposable design of ViT has unleashed the application of MTL to visual transformer models. In an early approach [24], the ViT was divided into the task-specific head, tail, and shared transformer structures across the tasks, and it was possible to attain a similar generalization performance with fewer training steps, by sharing the transformer model among the related tasks.

B. Federated Split Task-Agnostic (FeSTA) Learning

As described in Fig. 1A and Fig. 2A, the main motivation of the existing FeSTA framework was to devise a framework to maximally exploit the distinct strengths of FL and SL methods and to improve the performances of individual tasks with multi-task collaboration between clients performing various tasks.

Let $\mathcal{C} = \bigcup_{k=1}^{K} \mathcal{C}_k$ be a group of client sets with different tasks, where $K$ denotes the number of tasks and the client set $\mathcal{C}_k$ includes one or more clients having different data samples for the $k$-th task, i.e. $\mathcal{C}_k = \{c_1^k, c_2^k, \ldots, c_{N_k}^k : N_k \geq 1\}$. Clients in each client set for the $k$-th task have their own task-specific model architecture for head $\mathcal{H}_c$ and a tail $\mathcal{T}_c$, while the server-side transformer body $\mathcal{B}$ is shared.

For training, the server and each client initialize the weights of each sub-network with random initialization or from the pre-trained parameters. For learning round $i = 1, 2, \ldots, R$, individual clients do the forward pass on their task-specific head $\mathcal{H}_c$ using the local training data $\{(x^{(i)}_c, y^{(i)}_c)\}_{i=1}^{N_c}$, and send the intermediate feature $h^{(i)}_c$ to the server; $h^{(i)}_c = \mathcal{H}_c(x^{(i)}_c)$. The transformer body $\mathcal{B}$, then receives the intermediate features
Fig. 1. The model configuration of (A) the original FeSTA and (B) the proposed p-FeSTA frameworks. Different from the FeSTA, the feature-space permutation module is employed in the p-FeSTA for privacy preservation as well as the communication efficiency.

Fig. 2. The learning process of (A) the original FeSTA and (B) the proposed p-FeSTA frameworks. Different from the FeSTA in which both the head and tail are updated throughout the learning process, the patch embedder is fixed throughout the learning process and the permuted features are transmitted to the server only in the initial feature preparation step.

from all clients and gets features $b^{(i)}_c$ in parallel with the forward pass, to send them back to each client $c$: $b^{(i)}_c = B(h^{(i)}_c)$. With the features $b^{(i)}_c$, the task-specific tail in client yields the output $\hat{y}^{(i)}_c = T_c(b^{(i)}_c)$, and forward pass finishes. Back-propagation is performed exactly the opposite way, in order of tail, body, and head. First, loss is calculated in tail as: $\ell_c(y^{(i)}_c, T_c(B(H_c(x^{(i)}_c))))$, where $\ell_c(y, \hat{y})$ denotes the $c$ task-specific loss between the target $y$ and the estimate $\hat{y}$. Then, the gradients are passed from the tail, body to head in reverse order to forward propagation, using the chain rule.

For multi-task body update, the optimization is performed by fixing the head and tails. For the task-specific head and tail updates, the optimization problem is solved by fixing the Transformer body. In addition, per every “UnifyingRounds”, the server aggregates, averages and distributes the head and tail parameters between clients participating in the same task, as in FedAvg [25].

In the previous study, the FeSTA along with the MTL was shown to ameliorate the individual performances of the clients in collaboration, while resolving the data governance and ownership issue as well as eliminating the need to transmit the huge weights of the transformer body [19].

III. METHOD

A. p-FeSTA

Nonetheless, the FeSTA framework still has several drawbacks. First, the communication cost is higher since the features and gradients should be continuously exchanged between the server and clients like in SL but the head and tail weights should also be aggregated and distributed as in FL. Accordingly, the total communication costs are inevitably higher than SL, and even higher than FL depending on the network size. Second, as shown in the ablation study without the transformer body, the CNN head and tail themselves already have strong representation capacity, which may diminish the role of the transformer body between head and tail. Third, privacy concerns may arise as there is no privacy-preserving method from the privacy attack on the feature transmitted from client to server.

The proposed p-FeSTA is a framework devised to mitigate these shortcomings. As shown in Fig. 1B and Fig. 2B, the overall composition of p-FeSTA is similar to that of FeSTA, which decomposes networks into head $H$, body $B$ and tail $T$. However, unlike previous FeSTA, we do not use the CNN head tailored for each task. Having the CNN head to
be powerful enough to play a major role in the task hinders the shared transformer from being an important component as there remains little room to improve with this additional module. Instead, we adopted a simple patch embedder similar to the standard ViT, but with the fixed weights in a task-agnostic manner, enforcing the self-attention within the transformer architecture to do the heavy lift.

Unfortunately, the use of patch embedding in a standard ViT may be prone to outside attackers that attempt to invert the patch embedder to obtain private data. To address this, here we propose a novel feature-space permutation module as depicted in Fig. 1B to prevent either an outside attacker or an “honest but curious” server from reverting features into original data containing privacy. Specifically, this feature-space permutation module randomly shuffles the order of all patch features ahead of sending them to the server, and stores the key to reverse the permutation on the client-side. Then, the transformer body $B$ in the server does a forward pass with the permutated patch features and sends the encoded features back to the clients. Finally, the client reverses the permutation with the saved key and yields the final output by passing the reverted features to the task-specific tail $T_{\text{t}}$. The back-propagation is performed in the exact opposite way, where the same feature-space permutation module to forward-propagation is utilized.

The availability of feature-space permutation module attributes to an intriguing property of ViT that all the components composing the transformer body, such as multi-head self-attention, feed-forward network, and layer normalization, is fundamentally “permutation invariant” [21]. They are processed independently in a patch-based manner and the order of the patch does not affect the outcome, and therefore, the transformer body can be trained without any performance degradation. In addition, as the orders of patches are completely shuffled, it is infeasible for a malicious attacker to successfully revert the private data. How the feature-space permutation module can provide privacy protection from the malicious attacker will be described in detail in the following section.

B. Protecting Privacy With Feature-Space Permutation Module

For FL, privacy is improved by the ephemeral and focused nature of the federated aggregation, averaging, and distribution of the model updates, assuming that the model updates are considered to be less informative than the original data. However, recent studies have thrown doubt to the false sense of security, showing that private data can be uncovered faithfully only with these local model updates [26], [27], [28]. In detail, given the access to the global model $W$ and the client’s model update $\Delta W$, the attacker can optimize the input image from a prior to produce a gradient that matches the client’s model update as illustrated in Fig. 3A. However, this type of attack is infeasible for the proposed $p$-FeESTA method, since only the tail part of the entire model is aggregated and distributed by the server to the clients. For instance, for COVID-19 classification, the aggregated task-specific tail is a simple linear classifier, with which private data are hard to be uncovered.

SL protects privacy in a different way. As the name suggests, it split the entire model into client-side and server-side sub-networks and does not send the models between the server and clients. Instead, the features and gradients are transmitted back and forth between the server and clients, and it can be the prey of the malicious attacker [16]. As described in Fig. 3B, when clients send the intermediate features $f$ to the server, the attackers may hijack these in the feature space, and instead of running the remainder of the SL model, they train three components: attacker model $\hat{F}$, decoder $G$, and discriminator $D$ with their own public data. The $D$ is trained to discriminate between the hijacked feature $f$ and the feature $\hat{f}$ encoded by $\hat{F}$, which enforces $\hat{f}$ to be in the same feature space as $f$. Simultaneously, $G$ learns to decode $\hat{f}$ into the image with minimal error. Then, well-trained $G$ can also be used to decode the hijacked feature $f$ to original data faithfully.

The feature space hijacking is also possible for our $p$-FeESTA. To make matter worse, the head part of our model is relatively simple and can be easy prey for an attacker. This is why we introduce a novel feature-space permutation...
module to protect privacy as described in Fig. 4A. The feature-space permutation module randomly shuffles the order of all patch features. In the implementation, the permutations of each data of each client are all different without any regularity as shown in Fig. 4A, resulting in innumerable patterns for all data. By employing these random permutations, even if a malicious attacker or server steals the intermediate features, two unknown learnable parameters that are required to uncover private data, namely those of patch embedder and the position embedding, are difficult to be inferred without knowing the original order of patches.

Specifically, possible attack scenario when the feature-space permutation module exists is illustrated in Fig. 4B. Suppose the permuted features are hijacked during the communication and the attacker has sufficient public data similar to the private one. Since those are permuted patch embeddings from the unknown client model, the attacker should solve two problems: (1) training the attacker model \( \hat{F} \) to embed the public data to the embedding space same as the hijacked feature before the random permutation and (2) training the jigsaw solver that can solve the randomly shuffled patch embeddings into the correct original order in the feature space, not in the image space.

Suppose that \( \hat{f}_{pub} \) and \( f_{pub} \) are the permuted and original versions of features embedded from public image \( x_{pub} \) by the attacker model \( \hat{F} \), \( m \) is the number of public images available for the attacker, and that \( f_{priv} \) is permuted feature hijacked by the attacker during transmission embedded by unknown client model \( F \), and \( n \) is the number of hijacked features.

Then, training the attacker model \( \hat{F} \), discriminator \( D \), and the decoder \( G \) can be formally represented by optimizing the following two objectives:

\[
\begin{align*}
&\min_{\hat{F}} \max_{D} \sum_{i=1}^{m} \sum_{j=1}^{n} \left[ \log(1 - D(J(\hat{f}_{priv}^{(i)}))) + \log D(J(\hat{f}_{pub}^{(i)}))) \right] \\
&\min_{G} \sum_{i=1}^{m} L_{decoder}(G(J(\hat{f}_{pub}^{(i)})), x_{pub}^{(i)})
\end{align*}
\]

where \( L_{decoder} \) denotes reconstruction loss for decoder such as L1 or L2 loss.

Meanwhile, solving the optimization for the jigsaw solver \( J \) can be represented as:

\[
\min_{J} \sum_{i=1}^{m} L_{jigsaw}(J(\hat{f}_{pub}^{(i)}), f_{pub}^{(i)})
\]

where \( L_{jigsaw} \) denotes feature similarity loss to optimize the jigsaw solver \( J \).

Note that to solve Eq. (1) and Eq. (2), it is required to know the exact jigsaw solver \( J \), which is the target of optimization in Eq. (3). Conversely, to optimize the Eq. (3), as the jigsaw solver should be trained in the exactly same feature space as the hijacked features, knowing the correct solution for the attacker model \( \hat{F} \) is necessary, which can be obtained after solving the optimization problem of Eq. (1). These mean that the optimization for both objectives needs each other, making the problems underdetermined and practically not feasible to solve. We experimentally demonstrated this assertion in Section IV-H.

### C. Training Procedure

The learning process of the p-FeSTA is akin to the original FeSTA, but dissimilar in several aspects. Instead of task-specific head \( H_k \) for each task \( k \), the task non-specific patch embedder \( H \) prepares the patch embeddings \( h_c \) for each client \( c \) at the beginning and sends them to the server after passing them through the feature-space permutation module. The server then saves the received patch embeddings \( h_c \) on its side and uses them throughout the remainder of the learning process in order to update the body \( B \) and tail \( T_k \) parts of the model. Consequently, the overall communication costs can be significantly reduced compared to the original FeSTA, as the communications to send the intermediate feature \( h_c \) or to update the head \( H \) are no more required.

As can be seen, the head part of the model, the patch embedder, cannot be updated in this configuration. However,
fixing the parameters of the patch embedder did not bring any performance exacerbation thanks to the simple structure to just embed the image patches into the same vector spaces. Having them trainable rather slightly decreased the performances by resulting in the discrepancy in embedding between tasks. The experimental results will be provided in the ablation study of Section IV-G. The detailed process of the proposed p-FeSTA is formally described in Algorithm 1.

D. Comparison With Other Distributed Learning Methods

Fig. 5 illustrates the detailed configurations and the comparison between the proposed and the existing distributed learning methods.

In the FL, the goal is to obtain a model in a central server while training data remains unmoved over the edge devices of multiple clients [7]. The central server distributes the global model to each client, and the clients then perform training iterations with their own data in parallel, to return the results of local computations to the server. The server then aggregates and averages local updates, and distributes the updated global model to each client. This process is repeatedly performed until the model converges. As the entire model parameters should be aggregated and distributed in the federated learning (gray area in Fig. 5A), it quickly consumes the network bandwidth if the highly expressive model is adopted.

In the SL, the model is split into several sub-networks trained separately on the server-side and client-sides. Specifically, the first sub-network performs a forward pass on the client-side with local data and sends the smashed features to the second sub-network located in the server (gray arrow in Fig. 5B). The server then performs the forward propagation on the server-side sub-network to pass back the subsequent features to the third sub-network on the client-side (green arrow in Fig. 5B). The third sub-network on the client-side can yield the final outcome and the loss can be calculated with the labels on clients. Backpropagation through split sub-networks on clients and server-sides is conducted in the exact opposite manner to the forward propagation (red arrow in Fig. 5B). Allocating small-sized sub-networks at the client-side reduces the computational load of local clients that usually lacks computational resources in practical implementation. Privacy is offered in the SL by inserting a black box that clients have no access to the server-side model and vice versa. However, relay-based training incurs the inefficient usage of computational resources resulting in a substantial increase in the training time [14]. In addition, as the client-side model is not averaged between the clients, the performance can be devastated if the clients have the data with skewed distribution.

The FeSTA learning, depicted in Fig. 5C, reconciles the distinct strengths of the FL and SL. It divides the entire network into the client-side and server-side sub-networks enabling privacy protection through the black box as well as the multi-task collaboration similar to the SL. In addition, it facilitates the learning with clients having data with skewed distribution by averaging head and tail parameters like FL, which is difficult to attain with the SL. However, it also holds the drawbacks of each method as privacy threats arise in the vulnerabilities of each method. Moreover, the communication overheads significantly increase with the feature/gradient transmission similar to in SL along with the head/tail parameter transmission like FL.

The p-FeSTA learning solves these limitations with the introduction of the feature-space permutation module and by fixing the patch embedder as in Fig. 5D. As the features

---

**Algorithm 1 Proposed p-FeSTA Algorithm**

```plaintext
/* Run on Main Server */
Function ServerMain:
ClientInit and Server initializes body weight
for tasks k ∈ {1, 2, ..., K} do in parallel
for clients c ∈ C_k do in parallel
    h_c ← ClientHead(c)
    memory ← {h_1, h_2, ..., h_n} ← h_c // Save all embedding h_i in memory
for rounds i = 1, 2, ..., R do
    for tasks k ∈ {1, 2, ..., K} do in parallel
        for clients c ∈ C_k do in parallel
            if i = 1 or (i - 1) ∈ UnifyingRounds then
                Set client w_T,c ← w_T,k
                h_c(i) ← memory ← {h_1, h_2, ..., h_n} // Load embedding h_c from memory
                h_c(i) ← B(h_c(i))
            nL_c(i) ← ClientTail(c, b_c(i)) & Backpropagation
            w_T,c(i+1) ← ClientUpdate(c)
            Update body w_B(i+1) ← w_B(i) - \frac{\eta}{K} \sum_{k=1}^{K} \sum_{c} \frac{\partial L_c(i)}{\partial w_B} N_k \delta w_B
            if i ∈ UnifyingRounds then
                for tasks k ∈ {1, 2, ..., K} do
                    Update w_T,k ← \frac{1}{N_k} \sum_c w_T,c(i+1)
            /* Run on Client k */
            Function ClientInit:
                Client initializes head/tail weights
            Function ClientHead(c):
                x_c ← All data on client c
                b_c ← H(x_c)
                Randomly permute patch embedding h_c // each data is differently permuted
                return h_c
            Function ClientTail(c, b_c):
                y_c ← Current batch of label from client c
                L_c ← l_c(y_c, T_c(h_c)) & Backprop.
                return \frac{\partial L_c}{\partial b_c}
            Function ClientUpdate(c):
                Backprop. tail, body & w_T,c ← w_T,c - \eta \frac{\partial L_c}{\partial w_T,c}
                return w_T,c
```
transmitted from client to server are permuted for privacy and the patch embedder does not need to be optimized, the features can be transmitted only once at the start of training (blue arrow in Fig. 5D) and used throughout the learning process. This halves the overall communication costs, since the features and gradients are transmitted only between the body and the tail (green and red arrows in Fig. 5D) as well as the aggregated and distributed part of the network is only the tail (gray arrow in Fig. 5D), which means that it also significantly reduces the communication overheads to average the clients’ parameters in this configuration.

IV. RESULTS

A. Implementation Details

As for the head part, we used the task-agnostic patch embedder consisting of the convolution layer with a kernel size of $16 \times 16$ and stride of 16, input channel of 3, and output channel of 768. For the server-side body, the transformer encoder of the ViT-Base model, consisting of 12 encoder layers and 12 attention heads, was used. The selection study for the numbers of encoder layers and attention heads are provided in Section IV-I. In addition, we utilized the feature compressor that consists of a single linear projection to reduce the dimension of the feature and gradients transmitted between the server and clients to lessen the communication overheads. The detailed model architecture is illustrated in Fig. 6. For fair comparisons of the communication costs and performances, the same feature compressor module was also employed for all the other distributed learning methods.

For the tail part, the network architectures specialized to yield the task-specific output were adopted. For the COVID-19 classification task, we used a simpler linear classifier. For severity prediction, the mapping module with five up-sizing convolution layers was adopted as proposed in [29]. For pneumothorax segmentation, the decoder part of U-Net [30] was used.

We simulated the distributed MTL between the institutions participating in three different CXR tasks: classification, severity prediction of COVID-19, and pneumothorax segmentation. As in [19], the model was first initialized with pre-trained weights for the CheXpert dataset. We minimized the binary cross-entropy (BCE) losses for each class for the classification task. The severity of COVID-19 was predicted and evaluated in an array-based manner as suggested in [31]. Specifically, BCE losses for each six location arrays of the lung were used for the optimization in severity prediction. Finally, for the pneumothorax segmentation, we minimized the binary cross-entropy loss combined with dice and focal losses. The SGD optimizer was used for the classification and severity prediction tasks, while the Adam optimizer was utilized for the segmentation task, with a learning rate of 0.001 and a warm-up constant learning rate scheduler for all tasks. The batch size was 4 per client, and the warm-up step was 500. The total training round was 6,000 for all clients, and the tail weights are averaged every 100 local iterations. To adjust the scale of gradients, the 1:2:10 gradient
scaling was applied for classification, severity prediction, and segmentation, respectively. Since 2 clients participate in each task with the batch size of 4 per client, the batch size of 8 was used for each task for the data centralized learning, and other hyperparameters have remained the same in all learning methods for a fair comparison.

All learning methods were performed in both the STL and MTL settings, except for the FL in which averaging of the entire model weights between the clients involving different tasks is not feasible.

To evaluate the efficacy of the feature-space permutation module, we simulated the privacy attack supposing the optimal settings for the attackers. We supposed that the attackers hijacked the entire permuted features from a client and know the ratios of permutation, the architecture and dimension of the patch embedder as well as the positional embedding, and retain a sufficient amount of public data that is similar to the public data to train the attacker-side networks. For the configuration illustrated in Fig. 4B, the architecture same as the original embedder $F$ was used as the attacker model $\hat{F}$, and the three-layered discriminator and the four-layered generator from the DCGAN [32] were employed as the discriminator $D$ and the decoder $G$, and the standard transformer with 12 encoder layers and 12 attention heads were utilized as the feature-level jigsaw solver as it can effectively handle the relationships between the embedded patches. For the learning objectives, the modified version of GAN loss [33] was used to train the discriminator as formulated in Eq. 1, the combined $L_1$ and $L_2$ losses were used as the decoder loss $L_{\text{decoder}}$, and the $L_1$ loss was utilized as the $L_{\text{jigsaw}}$. The batch size of 1 was used, and the model was trained for 5 epochs with a learning rate of 0.0002.

The FL, SL, FESTA and p-FESTA was simulated on the modification of Flower (licensed under an Apache-2.0 license) [34] framework. All experiments were performed with Python 3.8 and Pytorch 1.8 on Nvidia RTX 3090, 2080 Ti.

### B. Practical Simulation for Multi-Task Collaboration

One of the paramount motivations for FL in medical imaging is to make a robust model leveraging the dispersed and small-sized datasets from multiple institutions while avoiding data governance. Therefore, we assume the FL scenario in which the data of several clients are scanty.

For COVID-19 classification and severity prediction, we used both publicly available datasets and private data collected from local institutions. Overall, 1093 CXR from a local hospital (KNUH, client #1) and 875 from public data (BIMCV [35], client #2) were used for training and 556 CXRs from another hospital (CNUH) were used as the external test set in COVID-19 classification task as shown in Table I.

Similarly, for the COVID-19 severity prediction task, 286 CXRs from a local hospital (YNU, client #3) and 4,265 from public data (Brixia [36], client #4) were used as the training, and 81 CXRs data of another hospital (CNUH) were used as the external test set as provided in Table II. For pneumothorax segmentation, we used the Society for Imaging Informatics in Medicine and the American College of Radiology (SIIM-ACR) Pneumothorax Segmentation Challenge [37] dataset consisting of 10,679 CXR images. The randomly selected 1,000 CXR images were used as the test set, and the remaining 9679 CXR images were randomly split with a 1:1 ratio (4840 and 4839 CXRs) into two clients (client #5 and client #6) to emulate the participation of two hospitals as in Table III. For practical simulation of collaboration between hospitals, we allocated non-overlapping data samples to each client except for the pneumothorax segmentation task where the exact sources of the data can not be estimated. Overall, six clients participated in the MTL scenario, two clients per task.

| Data Partitioning for COVID-19 Classification |
|---------------------------------------------|
| Class       | CNUH (Test) | KNUH (Client #1) | BIMCV (Client #2) |
| Normal      | 417         | 400              | 93               |
| Other infection | 58         | 400              | -                |
| COVID-19    | 81          | 293              | 782              |
| Total       | 556         | 1093             | 875              |

| Data Partitioning for COVID-19 Severity Prediction |
|-----------------------------------------------|
| Severity | CNUH (Test) | YNU (Client #3) | Brixia (Client #4) |
| 1        | 26          | 63              | 261               |
| 2        | 11          | 59              | 443               |
| 3        | 8           | 25              | 414               |
| 4        | 7           | 35              | 866               |
| 5        | 12          | 18              | 745               |
| 6        | 17          | 86              | 1536              |
| Total    | 81          | 286             | 4265              |
For this study, Institutional Review Board approvals of each participating hospital were obtained and informed consent was waived.

Considering the sizes and compositions of each client, collaboration for the COVID-19 classification task can be regarded as the collaboration between all clients having small data with a substantial imbalance in data distribution. Likewise, the collaboration for COVID-19 severity can be considered to be the simulation of an imbalance in data size between the participants, one client has scanty data while the other client has relatively sufficient data, in addition to the differences in data composition. Finally, the clients for pneumothorax segmentation emulate the situation in which each participating client has relatively sufficient and homogeneous data with similar sizes.

When viewed in terms of the relevance between tasks, the COVID-19 classification and severity prediction task can be considered to be highly correlated tasks, while the pneumothorax segmentation task may be regarded as a less relevant task.

### C. Performance Metrics

To evaluate the classification performance, the area under the receiver operating characteristic curve (AUC) was used. The AUCs are separately calculated for the three labels, and the average of the AUCs is also reported to assess overall classification performance. For the severity prediction task, the mean squared error (MSE) of prediction was used as in the previous work [29]. To evaluate the segmentation accuracy, the Dice coefficient was used as the main evaluation metric to measure the intersection of the segmentation results and ground truth annotations, while the intersection over union (IoU), pixel-level AUC, and the F-score were also reported. The segmentation metrics are calculated per image and the mean values are reported. Since the test set for the segmentation includes non-pneumothorax cases with the empty label, the metrics were defined to be 1 when both label and predictions are empty as proposed in the original evaluation criteria for the SIIM-ACR Pneumothorax Segmentation Challenge [37]. The reconstructed results from the privacy attack were quantitatively evaluated with the mean squared error (MSE) and the structural similarity index measure (SSIM). The same evaluation methods were used for the results from all distributed learning methods and the ablation and selection studies, to provide a clear comparison. All experiments were performed repeatedly with three different seeds to exclude the coincidence of getting over- or underestimated results.

### D. Comparison Results

Table IV shows a comparison of the proposed \( p \)-FeSTA with data centralized learning, other distributed learning, and original FeSTA learning methods. For a fair comparison, all other methods underwent the same pre-training step as the proposed method. The same model architectures were used for the data centralized learning and all other distributed learning methods except for the original FeSTA methods, in which the task-specific CNN head is a key part of the method. For original FeSTA methods, DenseNet-121 equipped with PCAM operation [38] tailored for CXR classification were used as the head instead of the simple patch embedder as in the previous work [29]. For the FL, the entire network is aggregated and distributed without split. For the SL, the network is split with the same configuration to \( p \)-FeSTA but the client-side sub-network is not averaged. For the data-centralized baseline, we supposed that the central server has all data for three tasks, and the MTL is performed by splitting the network the same as in the \( p \)-FeSTA but in a data-centralized manner.

In the STL setting, the models trained with \( p \)-FeSTA showed the performance similar to the data centralized learning for three tasks, surpassing those of the FL and the SL. The improvements with the \( p \)-FeSTA over the FL and the SL were noticeable especially for the classification and severity prediction tasks, where the data insufficiency and imbalance problems are prominent. Note that the slightly better performance of the single-task model trained with the original FeSTA than the \( p \)-FeSTA for the task of severity quantification, which is even better than data centralized learning, may attribute to the more expressive task-specific CNN head tailored for CXR tasks.

In the MTL setting, the model obtained with the MTL among three tasks using \( p \)-FeSTA significantly outperforms the single-task counterparts and all other learning methods.

---

**TABLE IV**

PERFORMANCE COMPARISON WITH OTHER DISTRIBUTED LEARNING METHODS

| Methods               | Classification | Severity | Segmentation |
|-----------------------|----------------|----------|--------------|
|                       | Average        | Normal   | Others       | COVID-19 | MSE  | Dice |
| Single-task learning  |                |          |              |          |      |      |
| Data centralized      | 0.671 (0.051)  | 0.735 (0.071) | **0.777 (0.045)** | 0.500 (0.051) | 1.592 (0.081) | 0.793 (0.005) |
| Federated learning    | 0.601 (0.036)  | 0.597 (0.146) | 0.483 (0.068) | 0.722 (0.023) | 2.159 (0.188) | 0.789 (0.001) |
| Split learning        | 0.546 (0.024)  | 0.522 (0.067) | 0.534 (0.050) | 0.583 (0.013) | 2.546 (0.414) | 0.790 (0.000) |
| FeSTa                 | **0.718 (0.047)** | 0.680 (0.088) | 0.677 (0.032) | **0.795 (0.036)** | **1.318 (0.125)** | 0.801 (0.011) |
| \( p \)-FeSTa         | 0.696 (0.022)  | 0.739 (0.093) | 0.557 (0.118) | 0.790 (0.045) | 1.717 (0.148) | 0.803 (0.004) |
| Multi-task learning   |                |          |              |          |      |      |
| Data centralized      | **0.915 (0.036)** | 0.948 (0.013) | 0.823 (0.023) | **0.875 (0.080)** | 1.391 (0.165) | 0.799 (0.002) |
| Split learning        | 0.697 (0.017)  | 0.834 (0.010) | 0.520 (0.040) | 0.737 (0.007) | 1.765 (0.155) | 0.797 (0.003) |
| FeSTa                 | 0.780 (0.019)  | 0.785 (0.009) | 0.793 (0.100) | 0.761 (0.034) | 1.416 (0.048) | 0.796 (0.013) |
| \( p \)-FeSTa         | 0.884 (0.008)  | 0.906 (0.004) | **0.890 (0.011)** | 0.857 (0.014) | **1.361 (0.057)** | **0.808 (0.003)** |

Values are presented in the mean (standard deviation) of three repeats with different seeds.
Note that the performance gain with the MTL over the STL is more prominent in the $p$-FESTA compared with other methods. When compared with the MTL model obtained with the previous FESTA method, $p$-FESTA showed similar or slightly better performance in severity prediction and segmentation, but substantially outperformed the previous one in the classification task, providing generally superior performance. Even when compared with the data-centralized baseline in the MTL setting, $p$-FESTA offered similar performances for the classification and severity prediction, and even slightly outperformed for the segmentation task.

The fact that the benefit of the MTL is formidable especially in classification and severity prediction is intriguing, as they are the tasks in which scanty data with skewed distribution are problematic. On the contrary, the performance improvement was modest for pneumothorax segmentation where each participating client have a relatively large number of data with even distribution. Moreover, the close relevance between COVID-19 classification and severity prediction might have further enhanced the benefit of MTL to those tasks, compared with the relatively less relevant task of pneumothorax segmentation.

E. Convergence Analysis

To compare the convergence with different distributed learning methods with respect to the communication overheads, we performed the convergence analysis by calculating the losses and evaluation metrics for each task every 500 communication rounds, either in training or test sets.

As shown in Fig. 7, the MTL with $p$-FESTA (red) offered faster and stable convergence, showing less fluctuation in the losses and metrics, as well as better performance after convergence than other methods. Note that the benefit is more prominent in the test set than the train set. For instance, methods such as the single-task FL (orange) or the SL (green) show lower losses and better metrics in the train set, but their performances are substantially lower in the test set, suggesting the sign of severe overfitting to the training data. Overall, the convergence plots clarify the benefit of the MTL with $p$-FESTA in terms of the communication efficiency and performance, and the stability of convergence.

F. Communication Costs Between Server and Clients

In this section, we provide the estimated communication costs between the server and clients. Given the number of data as $D$, the batch size as $B$, the rounds between aggregation and distribution by the server as $n$, and the transmission of features, gradients, and the head, body, tail parameters as $F$, $G$, and $P_h$, $P_b$, $P_t$, the communication costs $T$ of each distributed learning strategy for a total of $R$ rounds between the server and one client can be formulated as follow:

$$T_{\text{FL}} = \frac{2R}{n}(P_h + P_b + P_t),$$ (4)

$$T_{\text{SL}} = 2BR(F + G),$$ (5)

$$T_{\text{FESTA}} = 2BR(F + G) + \frac{2R}{n}(P_h + P_t),$$ (6)

$$T_{p\text{-FESTA}} = DF + BR(F + G) + \frac{2RP_t}{n},$$ (7)

where the constant 2 is multiplied to account for the both-way transmissions between server and client. Note that the cost for features and gradients transmissions are not multiplied by 2 in $p$-FESTA to reflect no transmission of features and gradients to the head during the learning process.

Numerically, the communication cost for each distributed learning method in our experimental setting can be calculated as in Table V. As one of the critical drawbacks, the communication cost of FESTA is larger than SL and even higher than FL. On the contrary, the proposed $p$-FESTA substantially lessens the communication burden by saving the head features at the beginning and using them throughout the entire learning process on the server-side. In our experimental setting, the total communication overhead of the proposed $p$-FESTA is less than half of the previous FESTA, and also significantly lower than SL as well as FL.

G. Ablation Studies

To investigate the roles of each component, we performed the ablation study as in Table VI. As each component can differently affect the performances of individual tasks, we assessed the performances of all three tasks under the MTL setting.

1) Feature-Space Permutation Module: We first ablated the feature-space permutation module to verify whether our method is indeed “permutation invariant”. For this proposition to be true, the performance should be the same regardless of the presence of permutation module. As expected, the performances with and without the permutation module were similar regardless of the presence of the position embedding, proving that the permutation does not affect the performance of the transformer model whether or not the position information is provided. Overall, these results suggest that the transformer architecture used in our method is indeed permutation invariant.

2) Position Embedding: In the proposed method, the position embedding takes two roles, first provides position information to yield the final output in the tail, and second adds an additional unknown parameter to prevent an attacker from easily uncovering patch features into image patches. We performed
Fig. 7. Comparison of the convergence plots per communication rounds in (left) training and (right) test sets. The MTL with p-FESTA (red) offered remarkable benefits in terms of communication efficiency and performance, and the stability of convergence compared with other distributed learning methods.

TABLE VI
ABSTRACTION STUDIES FOR THE PROPOSED p-FESTA

| Ablation setting | Classification | Severity | Segmentation |
|------------------|----------------|----------|--------------|
|                  | AUC            | MSE      | Dice         |
|                  | Average | Normal | Others | COVID-19 | Average | Normal | Others | COVID-19 | Average | Normal | Others | COVID-19 |
| Position embedding |        |        |        |          |        |        |        |          |        |        |        |          |
| permutation fixed embedder |        |        |        |          |        |        |        |          |        |        |        |          |
| X                 | 0.826 (0.023) | 0.849 (0.029) | 0.765 (0.045) | 0.865 (0.011) | 2.196 (0.517) | 0.791 (0.002) |
| X                 | 0.827 (0.028) | 0.831 (0.035) | 0.786 (0.049) | 0.862 (0.007) | 1.942 (0.112) | 0.798 (0.004) |
| ✓                 | 0.890 (0.010) | 0.909 (0.002) | 0.904 (0.023) | 0.858 (0.008) | 1.461 (0.064) | 0.809 (0.002) |
| ✓                 | 0.890 (0.001) | 0.909 (0.014) | 0.895 (0.005) | 0.866 (0.013) | 1.545 (0.356) | 0.789 (0.000) |
| ✓                 | 0.884 (0.008) | 0.906 (0.004) | 0.890 (0.011) | 0.837 (0.014) | 1.361 (0.057) | 0.808 (0.003) |

Values are presented in the mean (standard deviation) of three repeats with different seeds.

the ablation study to confirm that the position embedding is necessary for optimal performance in addition to privacy preservation. The model trained without the position embedding showed lower performances than those with the position embedding for all three tasks, suggesting that the position embedding is indispensable for optimal performance as well as privacy preservation.

3) Fixed Embedder: Fixing the patch embedder is one of the key components of our method, which significantly reduces the communication overheads as well as enables genuine multi-task collaboration between different tasks by embedding patches into the same feature space. Therefore, we performed an ablation to verify whether fixing the embedder parameters does not harm the performance. Compared with the proposed method with a fixed embedder, the same model trained using the learnable embedding showed similar or even slightly worse performance for severity prediction and segmentation, which may attribute to the overfitting to training data. Therefore, we concluded that the patch embedder can be fixed during all the learning rounds without concerns of performance degradation.

H. Experiments on Privacy Protection

Figure 8 and Table VII show the qualitative and quantitative results of reconstruction from the privacy attack in different
settings. Given the optimal setting for attackers, the private CXR data can be plausibly reconstructed if the attackers hijack both the parameters of the embedder (e.g., CNN head in FeSTA) and the intermediate feature, which can occur in the setting for the previous FeSTA (MSE of $0.012 \pm 0.002$ and SIIM of $0.638 \pm 0.018$). In the setting where either embedder is unknown or the random permutation is employed, the private CXR data can be reconstructed to some degree, albeit not perfect, that some personal information could be inferred with the recovered image (MSE of $0.056 \pm 0.003$, SIIM of $0.412 \pm 0.008$, respectively). For instance, it can be inferred from the upper case that the subject has no definite lung collapse, and is neither skinny nor obese. Similarly, the fact that there exists consolidation or collapse in the right lower lung can be inferred from the reconstruction of the lower case. These results imply that there may be a risk of some degree of privacy leakage. However, when using the feature-space permutation along with the unknown embedder concealed to the attackers, it was nearly impossible to infer any privacy as provided in the figure and the metrics (MSE of $0.241 \pm 0.014$ and SSIM of $0.068 \pm 0.018$), which is the setting employed in the $p$-FeSTA. These results confirmed our claim that solving the two optimization problems simultaneously without knowing each other’s solution can be considered an under-determined problem, and therefore practically impossible to solve.

I. Selection Study for the Encoder Layers and Attention Heads

Table VIII provides the results of the selection studies for the numbers of encoder layers and the attention heads. For a fair comparison, all experiments were performed with the same settings, including the pre-training and hyperparameters. As shown in the table, the standard ViT-Base model equipped with 12 encoder layers and 12 attention heads showed the best performances, suggesting that those with smaller numbers of heads or layers may not have sufficient capacity for multi-task collaboration and those with more numbers of heads or layers may be prone to overfitting.

J. Detailed Analysis for the Segmentation Performance

In Table IX, we provided other evaluation metrics for segmentation to compare the performances with the different methods in detail.

Beside the Dice coefficient, the IoU, pixel-level AUC, and F-score are all better with the $p$-FeSTA than other methods, both in the STL settings and the MTL, while the superiority was more prominent for the MTL.

Fig. 9 depicts the visual results of the pneumothorax segmentation by each method. The MTL with $p$-FeSTA offered the most accurate segmentation results, while other methods showed over- or under-segmented results. Although the performance gains were noted with the multi-task collaboration...
for the data centralized learning or the SL over their single-task counterparts, the MTL with \( p\)-FeSTA provided the best performance among the methods.

V. DISCUSSION

In this work, we introduced an improved federated task-agnostic learning framework with permutating pure ViT, dubbed \( p\)-FeSTA, which resolves the major drawbacks of our previous FeSTA framework, leveraging the intrinsic properties of the ViT. The newly proposed \( p\)-FeSTA substantially reduces the communication overheads between server and clients as well as offers a good performance which is more prominent in the MTL setting than the STL with the authentic multi-task collaboration in the same embedding space while offering better privacy preservation.

As the client-side model is not averaged in the SL, the performance can deteriorate if the clients have the data with skewed distribution, as shown in the classification task of our experiment where it has shown the worst performance. However, with the \( p\)-FeSTA, since the client-side models have either fixed or been averaged during the training as in the FL, it was possible to successfully train the model with the skewed data, coming close to the performances of the data-centralized learning.

One of the most tackling problems of the previous FeSTA method was the communication overhead between server and clients since it requires the feature and gradient transmission the same as in SL as well as the server-side aggregation and distribution of the heads and tails parameters for each client. This configuration increases the communication cost to be inevitably larger than SL and even larger than FL according to the network sizes of each model component. To mitigate the problem, we configured the head part to be a simpler

---

**TABLE VIII**

| Methods                      | Classification AUC | Severity | Segmentation |
|------------------------------|--------------------|----------|-------------|
|                              | Average | Normal | Others | COVID-19 | MSE | Dice |
| **Number of encoder layers** |         |        |        |          |     |      |
| 6 layers 12 heads            | 0.842 (0.009) | 0.817 (0.020) | 0.864 (0.024) | 0.845 (0.031) | 1.631 (0.092) | 0.800 (0.009) |
| 12 layers 12 heads           | **0.884 (0.008)** | **0.906 (0.004)** | **0.890 (0.011)** | 0.857 (0.014) | **1.361 (0.057)** | **0.808 (0.003)** |
| 24 layers 12 heads           | 0.855 (0.005) | 0.849 (0.013) | 0.855 (0.024) | **0.859 (0.046)** | 1.698 (0.118) | 0.799 (0.010) |
| **Number of attention heads** |         |        |        |          |     |      |
| 12 layers 6 heads            | 0.779 (0.027) | 0.724 (0.042) | 0.793 (0.039) | 0.818 (0.038) | 2.014 (0.246) | 0.794 (0.008) |
| 12 layers 12 heads           | **0.884 (0.008)** | **0.906 (0.004)** | **0.890 (0.011)** | **0.887 (0.014)** | **1.361 (0.057)** | **0.808 (0.003)** |
| 12 layers 16 heads           | 0.804 (0.031) | 0.779 (0.028) | 0.829 (0.021) | 0.803 (0.046) | 1.677 (0.378) | 0.801 (0.011) |

Values are presented in the mean (standard deviation) of three repeats with different seeds.

---

**TABLE IX**

| Methods                      | IoU   | AUC   | F-score  |
|------------------------------|-------|-------|----------|
| **Single-task learning**     |       |       |          |
| Data centralized             | 0.792 (0.003) | 0.986 (0.004) | 0.793 (0.005) |
| Federated learning           | 0.789 (0.001) | 0.984 (0.000) | 0.789 (0.001) |
| Split learning               | 0.797 (0.007) | 0.989 (0.002) | 0.801 (0.011) |
| FeSTA (STL)                  | 0.796 (0.007) | 0.991 (0.001) | 0.800 (0.010) |
| \( p\)-FeSTA (STL)           | **0.799 (0.003)** | **0.992 (0.001)** | **0.803 (0.004)** |
| **Multi-task learning**      |       |       |          |
| Data centralized             | 0.796 (0.002) | 0.991 (0.001) | 0.799 (0.002) |
| Split learning               | 0.799 (0.009) | 0.991 (0.001) | 0.803 (0.012) |
| FeSTA (MTL)                  | 0.794 (0.004) | 0.989 (0.002) | 0.796 (0.006) |
| \( p\)-FeSTA (MTL)           | **0.803 (0.003)** | **0.992 (0.001)** | **0.808 (0.003)** |

AUCs are calculated per pixel.
Values are presented in the mean (standard deviation) of three repeats with different seeds.
structure like a patch embedder so that the pre-trained head can sufficiently show the best performance without further training that requires communications back and forth between server and clients. Consequently, the features from the head could be stored on the server-side at the beginning and used throughout the entire learning process, reducing the overall communication cost to approximately half of other distributed learning methods.

Moreover, having the head part be a common patch embedder provides another advantage of embedding the image features of different tasks to be in the same embedding space. This result in the increasing role of following multi-features of different tasks to be in the same embedding space provides another advantage of embedding the image learning methods.

TABLE X

| Comparison                             | Federated learning | Split learning | FESTA | p-FESTA |
|----------------------------------------|--------------------|----------------|-------|---------|
| Learning with clients having skewed data| Feasible with model averaging | Poor performance | Feasible with model averaging | Feasible with model averaging |
| Shared model parameter                 | Entire model       | -              | Head / Tail | Tail |
| Shared feature / gradient              | Large              | Head / tail    | Head / tail | Tail |
| Overall communication cost             | -                  | Large          | Very large | Small |
| Multi-task collaborative learning       | Not possible       | Possible, small benefit | Possible, small benefit | Possible, large benefit |
| Specialized module for privacy         | -                  | -              | Feature-space permutation module | - |

Nevertheless, our study is not free of limitations. First, even though we have simulated the practical collaboration between hospitals with our p-FESTA implemented based on the user-friendly Flower framework, there remain a number of considerations in the practical implementation, like the robustness to the other tackling factor such as straggler-resilience [8], [44], the stability of network connection, heterogeneous computing resource of clients participating in multi-task collaboration, and so on. Considering that connection instability and the lack of computation resources become the common problems in online learning, these should be addressed technically ahead of real-world implementation.

Second, although we have demonstrated the efficacy of the feature-space permutation module for privacy preservation by comparing the reconstruction results with and without the feature-space permutation module, it should be noted that only the transformer-based architecture was utilized as the jigsaw solver. The reconstruction results may be improved to some degree if the network more tailored for the jigsaw puzzle solving is employed. Finally, we did not consider other types of attacks for distributed learning, such as model poisoning and data poisoning [45], [46], [47], which is beyond the scope of this work, or other types of privacy threat like the membership inference [10], [11], [12]. For defense against these types of malicious attacks, the existing methods [48], [49], [50] can be utilized along with our framework. Future work might verify the robustness to these types of malicious attacks.

VI. CONCLUSION

In this paper, we proposed the novel p-FESTA framework with pure ViT, which elicits the synergy of the MTL among heterogeneous tasks as well as reduces the communication overhead significantly compared to the existing FEISTA. In addition, we also enhanced the privacy using the feature-space permutation module in a way specific to ViT. We believe that our work is a step toward facilitating distributed learning among the institutions wanting to participate in different tasks, mitigating the major drawbacks of the existing methods.

ACKNOWLEDGMENT

The authors are grateful to Gwanghyun Kim (Seoul National University) for his help in the implementation of the framework for the experiments.

REFERENCES

[1] A. Hosny, C. Parmar, J. Quackenbush, L. H. Schwartz, and H. J. W. L. Aerts, “Artificial intelligence in radiology,” Nature Rev. Cancer, vol. 18, no. 8, pp. 500–510, May 2018.
[2] M. K. K. Niazi, A. V. Parwani, and M. N. Gurcan, “Digital pathology and artificial intelligence,” *Lancet Oncol.*, vol. 20, no. 5, pp. e253–e261, 2019.

[3] P. F. Edemekong, P. Annamalai, and M. J. Haydel, “Health insurance portability and accountability act,” in *StatPears*. Treasure Island, FL, USA: StatPears Publishing, Jan./Feb. 2012.

[4] C. J. Hoofnagle, B. van der Sloot, and F. Z. Borgesius, “The European union general data protection regulation: What it is and what it means,” *Inf. Commun. Technol. Law.*, vol. 28, no. 1, pp. 65–98, Jan. 2019.

[5] W. Li et al., “Privacy-preserving federated brain tumour segmentation,” in *Proc. Int. Workshop Mach. Learn. Med. Imag.*, Shenzhen, China. Berlin, Germany: Springer, 2019, pp. 133–141.

[6] N. Rieke et al., “The future of digital health with federated learning,” *NPJ Digit. Med.*, vol. 3, no. 1, pp. 1–7, 2020.

[7] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, “Federated learning: Strategies for improving communication efficiency,” 2016, *arXiv*:1610.05492.

[8] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, “Federated learning: Challenges, methods, and future directions,” *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 50–60, May 2020.

[9] P. M. Mammen, “Federated learning: Opportunities and challenges,” 2021, *arXiv*:2101.05428.

[10] U. Gupta, D. Stripelis, P. K. Lam, P. Thompson, J. L. Ambite, and R. Shokri, “Comprehensive privacy analysis of deep learning models,” in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2017, pp. 3–18.

[11] M. Nasr, R. Shokri, and A. Houmansadr, “Membership inference attacks against centralized and federated learning,” in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2019, pp. 739–753.

[12] P. Vepakomma, O. Gupta, T. Swedish, and R. Raskar, “Split learning for health: Distributed deep learning without sharing raw patient data,” 2018, *arXiv*:1812.05064.

[13] C. Thapa, M. A. P. Chamikara, S. Camtepe, and L. Sun, “SplitFed: When federated learning meets split learning,” 2020, *arXiv*:2004.12088.

[14] Y. Gao et al., “End-to-end evaluation of federated learning and split learning for Internet of Things,” 2020, *arXiv*:2003.13376.

[15] G. Gawron and P. Stubbings, “Feature space hijacking attacks against differentially private split learning,” 2022, *arXiv*:2201.04018.

[16] D. Pasquini, G. Ateniese, and M. Bernaschi, “Unleashing the tiger: Inference attacks on split learning,” in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2021, pp. 2113–2129.

[17] B. Jayaraman and D. Evans, “Evaluating differentially private machine learning in practice,” in *Proc. 28th USENIX Symp. Secur. (USENIX Security)*, 2019, pp. 1895–1912.

[18] S. Park, G. Kim, J. Kim, B. Kim, and J. Chul Ye, “Federated split vision transformer for COVID-19 CXR diagnosis using task-agnostic training,” 2021, *arXiv*:2111.01338.

[19] A. Dosovitskiy et al., “An image is worth 16 x 16 words: Transformers for image recognition at scale,” 2020, *arXiv*:2010.11929.

[20] M. M. Naseer, K. Ranasinghe, S. H. Khan, M. Hayat, F. S. Khan, and M.-H. Yang, “Intriguing properties of vision transformers,” in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 1–13.

[21] S. Chen, Y. Zhang, and Q. Yang, “Multi-task learning in natural language processing: An overview,” 2021, *arXiv*:2109.09138.

[22] S. Liu, F. He, H. Chen, and J. Gao, “Multi-task deep neural networks for natural language understanding,” 2019, *arXiv*:1901.11504.

[23] H. Chen et al., “Pre-trained image processing transformer,” 2020, *arXiv*:2012.03064.

[24] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, “Communication-efficient learning of deep networks from decentralized data,” in *Proc. 29th Int. Conf. Artif. Intell. Statist.*, 2017, pp. 1273–1282.

[25] J. Geiger, S. Farschman, H. Dreiseitl, and M. Moeller, “Inverting gradients—how easy is it to break privacy in federated learning?” in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 16937–16947.

[26] L. Zhu, Z. Liu, and S. Han, “Deep leakage from gradients,” in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–11.

[27] B. Zhao, K. R. Mopuri, and H. Bilen, “IDLG: Improved deep leakage from gradients,” 2020, *arXiv*:2001.02610.

[28] S. Park et al., “Multi-task vision transformer using low-level chest X-ray feature corpus for COVID-19 diagnosis and severity quantification,” *Med. Image Anal.*, vol. 75, Jan. 2022, Art. no. 102299.

[29] O. Ronneberger, P. Fischer, and T. Brox, “U-Net: Convolutional networks for biomedical image segmentation,” in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Intervent.*, Munich, Germany. Berlin, Germany: Springer, 2015, pp. 234–241.

[30] D. Tousset et al., “Clinical and chest radiography features determine patient outcomes in young and middle-aged adults with COVID-19,” *Radiology*, vol. 297, no. 1, pp. E197–E206, Oct. 2020.

[31] A. Radford, L. Metz, and S. Chintala, “Unsupervised representation learning with deep convolutional generative adversarial networks,” 2015, *arXiv*:1511.06434.

[32] I. Goodfellow et al., “Generative adversarial nets,” in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 27, 2014, pp. 1–9.

[33] D. J. Beutel et al., “Flower: A friendly federated learning research framework,” 2020, *arXiv*:2007.14390.

[34] S. Park et al., “Multi-task vision transformer using low-level chest X-ray presentations by solving jigsaw puzzles,” in *CVPR Workshops*, 2020, pp. 69–84.

[35] S. Sharma and K. Shen, “Image disguising for privacy-preserving deep learning,” in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2018, pp. 2291–2293.

[36] M. Noroozi and P. Favaro, “Unsupervised learning of visual representations by solving jigsaw puzzles,” in *Proc. Eur. Conf. Comput. Vis.*, Amsterdam, The Netherlands. Berlin, Germany: Springer, 2016, pp. 69–84.

[37] A. Reisizadeh, I. Tziotis, H. Hassani, A. Mokhtari, and R. Pedarsani, “Straggler-resilient federated learning: Leveraging the interplay between statistical accuracy and system heterogeneity,” 2020, *arXiv*:2012.14453.

[38] L. Lyu, H. Yu, and Q. Yang, “Threats to federated learning: A survey,” 2020, *arXiv*:2003.02133.

[39] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu, “Data poisoning attacks against federated learning systems,” in *Proc. Eur. Symp. Res. Comput. Secur.*, Guildford, U.K. Berlin, Germany: Springer, 2020, pp. 480–501.

[40] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, “How to backdoor federated learning,” in *Proc. Int. Conf. Artif. Intell. Statist.*, 2020, pp. 2938–2948.

[41] L. Lyu et al., “Privacy and robustness in federated learning: Attacks and defenses,” 2020, *arXiv*:2012.06337.

[42] Y. Zhao, J. Chen, J. Zhang, D. Wu, J. Teng, and S. Yu, “PGDAN: A novel poisoning defense method in federated learning using generative adversarial network,” in *Proc. Int. Conf. Algorithms Archit. Parallel Process.*, Melbourne, VIC, Australia. Berlin, Germany: Springer, 2019, pp. 595–609.

[43] S. Li, Y. Cheng, W. Wang, Y. Liu, and T. Chen, “Learning to detect malicious clients for robust federated learning,” 2020, *arXiv*:2002.00211.