FedBERT: When Federated Learning Meets Pre-Training

YUANYISHU TIAN, Huazhong University of Science and Technology, China
YAO WAN, Huazhong University of Science and Technology, China
LINGJUAN LYU, Sony AI, Japan
DEZHONG YAO*, Huazhong University of Science and Technology, China
HAI JIN, Huazhong University of Science and Technology, China
LICHAO SUN*, Lehigh University, USA

The fast growth of *pre-trained models* (PTMs) has brought natural language processing to a new era, which has become a dominant technique for various *natural language processing* (NLP) applications. Every user can download the weights of PTMs, then fine-tune the weights for a task on the local side. However, the pre-training of a model relies heavily on accessing a large-scale of training data and requires a vast amount of computing resources. These strict requirements make it impossible for any single client to pre-train such a model. In order to grant clients with limited computing capability to participate in pre-training a large model, we propose a new learning approach FedBERT that takes advantage of the federated learning and split learning approaches, resorting to pre-training BERT in a federated way. FedBERT can prevent sharing the raw data information and obtain excellent performance. Extensive experiments on seven GLUE tasks demonstrate that FedBERT can maintain its effectiveness without communicating to the sensitive local data of clients.

CCS Concepts: • **Computing methodologies → Machine learning**; **Natural language processing**; **Distributed computing methodologies**.

Additional Key Words and Phrases: Federated learning, pre-training, BERT, NLP

1 INTRODUCTION

Natural language processing (NLP) techniques have been widely used in many real-life applications, such as natural language inference, natural language understanding, named entity recognition, question answering and generation [9, 10, 12, 30, 53, 57]. Many works [9, 10, 24] point out that most deep learning-based NLP tasks are driven by a large amount of labeled data, resulting in limited task performance in many scenarios due to the

* Corresponding Author

Authors' addresses: Yuanyishu Tian, yystian@hust.edu.cn, National Engineering Research Center for Big Data Technology and System, Services Computing Technology and System Lab, Cluster and Grid Computing Lab, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China, 430074; Yao Wan, wanyao@hust.edu.cn, National Engineering Research Center for Big Data Technology and System, Services Computing Technology and System Lab, Cluster and Grid Computing Lab, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China, 430074; Lingjuan Lyu, Lingjuan.Lv@sony.com, Sony AI, 1-7-1 Konan Minato-ku, Tokyo, Japan; Dezhong Yao, dyao@hust.edu.cn, National Engineering Research Center for Big Data Technology and System, Services Computing Technology and System Lab, Cluster and Grid Computing Lab, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China, 430074; Hai Jin, hjin@hust.edu.cn, National Engineering Research Center for Big Data Technology and System, Services Computing Technology and System Lab, Cluster and Grid Computing Lab, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China, 430074; Lichao Sun, lis221@lehigh.edu, Lehigh University, USA.

2157-6904/2022/1-ART1 $15.00
https://doi.org/10.1145/3510033

labeled data limitation. To address this restriction, *pre-trained models* (PTMs) use a self-supervised representation learning method to automatically capture much linguistic information and patterns from large unlabeled text corpora, which has become a dominant technique in this area. All pre-trained models follow the same learning paradigm: (1) first train on unlabeled text data via a task that resembles language modeling; (2) fine-tune on a smaller amount of downstream data, labeled for a specific task. Pre-trained models usually achieve higher accuracy than other traditional models on NLP GLUE benchmarks [61].

However, for such a universal language learning approach, two main challenges are expected to be solved [46]. Firstly, the pre-training methods heavily require an available large-scale dataset, which are only affordable for a few giant high-tech companies, leading to *Artificial Intelligence* (AI) hegemony [70]. Moreover, in practice, most sensitive data are isolated and forbidden to be directly shared, e.g., medical health records and personal finance data, due to privacy concerns and regulatory constraints. It is difficult for individuals or small companies to pre-train models on sensitive dataset. Secondly, a large amount of computational resources are required for pre-training a NLP model. As shown in Table 1, the most miniature pre-trained model, i.e., BERT$_{Small}$, takes more than four days with one NVIDIA V100 GPU.

Unfortunately, the *Internet-of-Things* (IoT) devices (e.g., robots, drone swarms, and Raspberry Pi) may suffer from processing ability, low bandwidth and power, or limited storage capacity. Moreover, it is difficult for those small business clients to access a strong computing infrastructure. As shown in Table 1, the hardware requirement for training the BERT$_{Small}$ model is at least one V100 card, which costs about 11 thousand dollars. This is quite expensive for some small companies, which has data privacy concern and are unwilling to use cloud service. This requirement is unfriendly for devices or users with low computing resources to finish such consumptive pre-trained model learning. In order to break the AI hegemony and support resource-constrained devices contributing to a pre-training task for NLP, this paper proposes two available solutions for the above challenges.

Solution 1. Due to privacy concerns, most clients or users save their own private data on local devices. Therefore, it is difficult for each client to pre-train a language model using insufficient training data, even with sufficient computing resources. To fully utilize the vast amount of distributed, diverse and privately-owned data, *federated learning* (FL) provide a feasible solution [33] to train a global model across multiple datasets without raw data sharing. FL is a collaborative and privacy-aware learning paradigm, which learns a global model by aggregating the models trained on local devices [37, 64]. Through FL, each client would not worry about their private data exposed to other clients, but they can collaboratively build a pre-trained model together.

Solution 2. To overcome the challenge of limited computing resources of the devices, we argue that a deep neural network can be trained in a split way [60]. *Split learning* (SL) makes the clients with low computation feasible to train BERT within adequate training time, such as mobile devices or small clients without GPU resources. However, existing SL approaches work on feed-forward neural networks and cannot be used to deal with sequential data [2]. In order to train on sequentially partitioned data, BERT should be split in a way that preserves latent dependencies between modules. As shown in Fig. 1, the pre-trained model, i.e., BERT network, is composed of multiple bidirectional Transformer [59] layers for unsupervised language representation learning,

Table 1. Comparison of the training efficiency of different pre-trained models [9]

Model	**Train / Infer FLOPs**	**Params**	**Train Time (days)**	**Hardware**
ELMo	3.3e18 / 2.6e10	96M	14	3 GTX 1080 GPUs
GPT	4.0e19 / 3.0e10	117M	25	8 P6000 GPUs
BERT$_{Small}$	1.4e18 / 3.7e9	14M	4	1 V100 GPU
BERT$_{Base}$	6.4e19 / 2.9e10	110M	4	16 TPUv3s

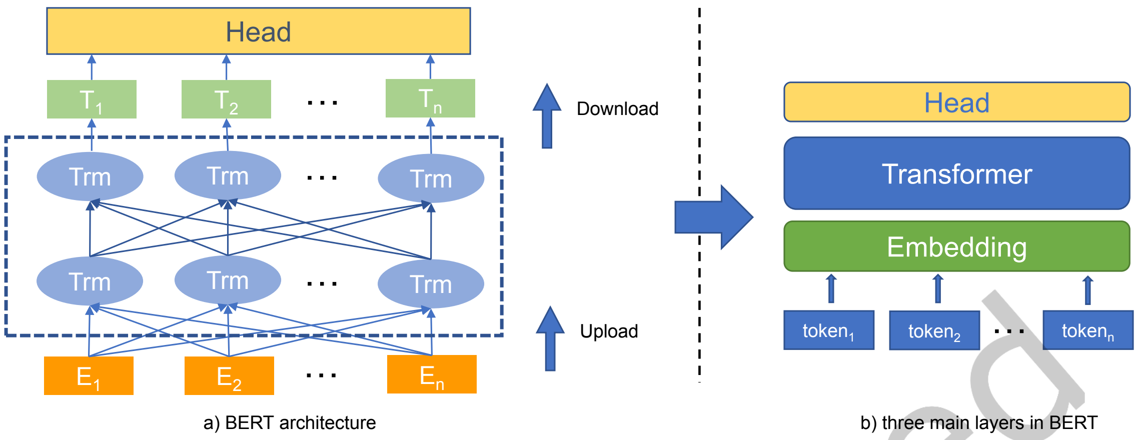


Fig. 1. An overview of the architecture of BERT, which is composed of an Embedding layer, Transformer layer, and Head layer

which predict the masked words according to the contextual information. We split the BERT model into three main layers: Embedding layer, Transformer layer, and Head layer, which can decouple latent dependencies. Based on the complexity analysis and computing analysis of a deep neural network, the Transformer layer requires high-computing resources that could be allocated on a powerful server, and the other two resource-constrained sub-networks, i.e., Embedding layer and Head layer, can be trained on clients. After pre-training the model, each client can fine-tune specific NLP tasks privately. Furthermore, by using federated learning, the split training approach can incrementally incorporate datasets from multiple devices to improve the robustness and generalization of pre-trained models [7].

This paper proposes a novel federated learning framework for pre-training language models. Following the FL framework, the pre-training task is distributed among multiple clients and executed in parallel. All clients and cloud collaboratively train a pre-trained model for NLP with advanced federated learning and split learning techniques. First, each client will train and update their Embedding layers and pass the embedded information to the server. Then, the server will aggregate the information from the clients and update the Transformer layers during training and communication processes. The training will end once the pre-trained model converges. Last, each client can fine-tune the task by their own choice. The primary contributions of this paper are summarized as follows:

- We are the first that proposes a novel federated learning framework for language models pre-training. The proposed framework can well support all devices or clients without high computation resources and a large amount of data to contribute to a pre-training task.
- In our proposed framework, we also leverage split training techniques for pre-training the BERT model in a federated learning setting. Each client can fine-tune the local NLP task individually after pre-training the global model.
- Extensive experiments on GLUE [61] validate the effectiveness of our proposed FedBERT with an acceptable performance decrease. Moreover, a small version of GPT is pre-trained to illustrate the general effect of FedBERT.
- Some open questions and analyses about the performance of the federated framework are discussed, both in theory and experiment.

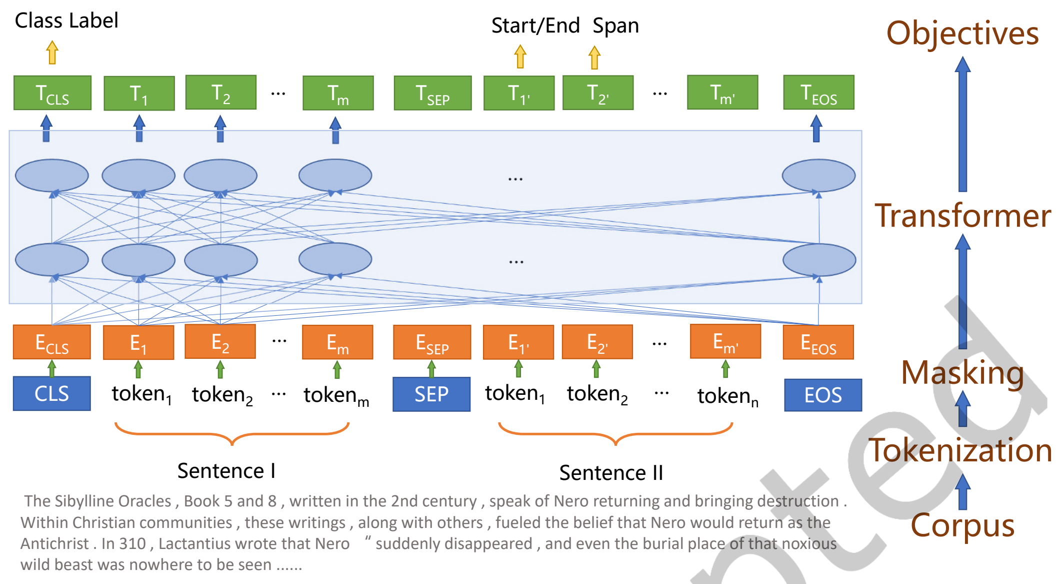

Class Label
Start/End Span
T_{CLS} T_1 T_2 ... T_m T_{SEP} $T_{1'}$ $T_{2'}$ ... $T_{m'}$ T_{EOS}
E_{CLS} E_1 E_2 ... E_m E_{SEP} $E_{1'}$ $E_{2'}$ ... $E_{m'}$ E_{EOS}
CLS $token_1$ $token_2$ ... $token_m$ SEP $token_1$ $token_2$ ... $token_n$ EOS
Sentence I
Sentence II
The Sibylline Oracles , Book 5 and 8 , written in the 2nd century , speak of Nero returning and bringing destruction . Within Christian communities , these writings , along with others , fueled the belief that Nero would return as the Antichrist . In 310 , Lactantius wrote that Nero " suddenly disappeared , and even the burial place of that noxious wild beast was nowhere to be seen ......
Objectives
Transformer
Masking
Tokenization
Corpus

Fig. 2. The process of pre-training BERT based on masked language modeling

Organization. The rest of this paper is organized as follows. Section 2 presents the preliminaries and problem formulation about this work, including the pre-trained models for NLP, federated learning, and split learning. In Section 3, we elaborate each module of our proposed FedBERT. Section 4 raises several research questions and shows the details of extensive experiments. Further, Section 5 analyzes the corresponding results and answers the research questions. In Section 6, we investigate the related works from the perspectives of federated learning, split learning, and pre-trained models for NLP. At last, we conclude this paper and present some future directions in Section 7.

2 PRELIMINARIES

This section sets out the preliminary knowledge of our work, including model pre-training for NLP, federated learning, and split learning.

2.1 Model Pre-Training for NLP

For NLP tasks, pre-training a portable text representation on a large-scale unlabeled corpus has attracted substantial attention. Previously, many works focused on learning the word embedding, e.g., Word2Vec [40] and GloVe [44]. Recently, self-supervised learning from masked language modeling has become a dominant technique for natural language understanding and generation [9, 10, 12, 30, 53, 57].

For example, the OpenAI GPT [45] pre-trains a language model by iteratively predicting the next token, and it has achieved promising performance in text generation. The BERT model [10] designs a bi-directional Transformer to reconstruct the masked words based on the contextual information. The pre-trained models can be fine-tuned on specific domain data so as to be adapted to different tasks. In this paper, we select BERT as the representative pre-trained model for investigation without loss of generality.

BERT Pre-training. Given a corpus, each segment (i.e., sentence) is firstly tokenized into a series of tokens via a tokenizer (e.g., Byte Pair Encoding, BPE [50]). Before pre-training BERT, it takes the concatenation of two segments as input, denoted as $x_1, x_2, \dots, x_N$ and $y_1, y_2, \dots, y_M$, where N and M denote the length of two segments, respectively. The two segments are always connected by a special separator token [SEP]. The first and last token of each sequence is always padded with a special classification token [CLS] and ending token [EOS], respectively. Finally, the input of each training sample will be represented as: $x = \texttt{[CLS]}, x_1, x_2, \dots, x_N, \texttt{[SEP]}, y_1, y_2, \dots, y_M, \texttt{[EOS]}$.

During BERT pre-training, two objectives are designed for self-supervised learning, i.e., *masked language modeling* (MLM) and *next sentence prediction* (NSP). In the masked language modeling, the tokens of the input sentence are randomly sampled and replaced with the special token [MASK]. In practice, BERT uniformly selects 15% of the input tokens for possible replacement. Among the selected tokens, 80% are replaced with [MASK], 10% are unchanged, and the left 10% are randomly replaced by selected tokens from the vocabulary. Mathematically, the masked language modeling aims to reconstruct the masked tokens based on the contextual tokens. In other words, the training objective can be formulated as:

$$\mathcal{L}_{MLM} = -\sum_{\hat{x} \in \mathcal{M}(x)} \log P(\hat{x} | f(x \backslash \mathcal{M}(x))) \tag{1}$$

where notations $\mathcal{M}(x)$ and $x \backslash \mathcal{M}(x)$ denote the masked token from x and the rest, respectively. The $P(\hat{x} | f(x \backslash \mathcal{M}(x)))$ denotes the probability of the predicted tokens over the vocabulary, where $f(x)$ is the representation function of masked sentence. In particular, BERT adopts the Transformer for representing the contextual tokens, which is based on the self-attention mechanism.

For the next sentence prediction, it is modeled as a binary classification task to predict whether two segments are consecutive. Training data of positive examples and negative examples are constructed based on the following rules: (1) if two sentences are consecutive, it will be considered as a positive example; (2) otherwise, those paired segments from different documents are considered as negative examples. The training objective of the next sentence prediction can be formularized as follows:

$$\mathcal{L}_{NSP} = -\log P(c | x, y) \tag{2}$$

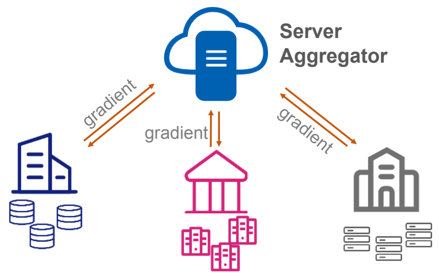
where $c = 1$ if x and y are consecutive segments from corpus, otherwise $c = 0$.

RoBERTa. Since the vanilla BERT is highly sensitive to the parameters of model training, Liu et al. [30] proposed a robust BERT (called RoBERTa) by tuning the parameters and investigating the performance brought by different training objectives. From their investigation, they find that the performance can be substantially improved by training the model longer, with bigger batches over more data, and longer sequences. In addition, they find that the next sentence prediction objective is unhelpful to improve the model performance. Therefore, we investigate RoBERTa on our experiments part instead. To be convenient for the problem formulation, we denote the whole model parameters of RoBERTa as $\mathbf{W}$.

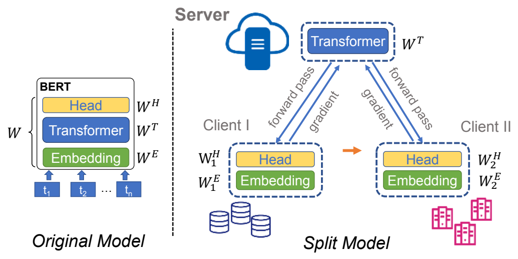
Note that either BERT or RoBERTa is heavily dependent on a large amount of training data. This will hinder the training of an adequate model. Furthermore, the fundamental Transformer architecture is always computational costly, hindering the large-scale pre-training in specific domains for small clients.

2.2 Federated Learning

Various intelligent devices generate a large amount of data daily, such as smartphones, tablets, and wearable devices. By launching the AI algorithm on devices, each device becomes smarter for particular tasks. However, with the increasing arisen of telecommunication fraud, junk mail, and advertising harassment, privacy concerns get more attention. Privacy protection acts are done in many countries and regions, like *General Data Protection Regulation* (GBPR) in Europe. There is no doubt that these acts will slow data-intensive AI development, since most data are privately stored in the local device. It is necessary to break the data island to acquire more comprehensive



(a) The framework of federated learning



(b) The framework of split learning

Fig. 3. An overview of federated learning and split learning

insights for tasks in reality. In answer to the aforementioned needs, federated learning is proposed by Google [36] to utilize data in mobile phones to train a more accurate global model. In contrast to the traditional training paradigm, federated learning only uses shared models to obtain the optimized results, as shown in Fig. 3a. Training task is delegated on devices, and intermediate information like gradients or model weights are exchanged and aggregated. Hence, raw data or labels are avoided from being exposed to others directly.

Assuming that we have N clients, where each client C_k has its own private dataset $\mathcal{D}_k$. A common training scheme of FL is that each client sends its local trained model to the server and the server sends back the aggregated model to all clients at the beginning of each training epoch. One of the most popular and efficient algorithm is FedAvg [36], which updates the global model $\mathbf{W}$ by averaging local model updates:

$$\mathbf{W}_t = \sum_{k=1}^{N} \frac{n_k}{n} \mathbf{W}_{k,t} \tag{3}$$

where n_k is the number of training samples in client k, and n is the total number of samples in all the N clients.

2.3 Split Learning

Similar to federated learning, split learning is also a privacy-aware framework. As shown in Fig. 3b, the deep neural network of BERT $\mathbf{W}$ is split into three parts, denoted as server-side network $\mathbf{W}^T$ and client-side network $\mathbf{W}^E$, $\mathbf{W}^H$. The network $\mathbf{W}^E$ and $\mathbf{W}^H$ are trained on the client side, and network $\mathbf{W}^T$ is trained on the server side. The raw data is processed and the intermediate information is exchanged. We denote the pre-training task as F and the original BERT model is $\mathbf{W}$. For a given input X, the output of the single training model is given by:

$$F(\mathbf{X}) = \mathbf{W}X \tag{4}$$

For the split learning approach, the output of the split BERT model is calculated as:

$$F(\mathbf{X}) = \mathbf{W}^H(\mathbf{W}^T(\mathbf{W}^E X)) \tag{5}$$

Due to the partition of the network, several parts of the computing task can be taken on the server side. However, it brings the issue of consistency of two parts between clients. The initial solution is to share the layers on the client side in sequence. Further, the layers deployed on the server side are updated by each client sequentially [16]. From beginning to end, layers on server side are the only transcript, and the bottom part is shared by clients. Thus the consistency is guaranteed.

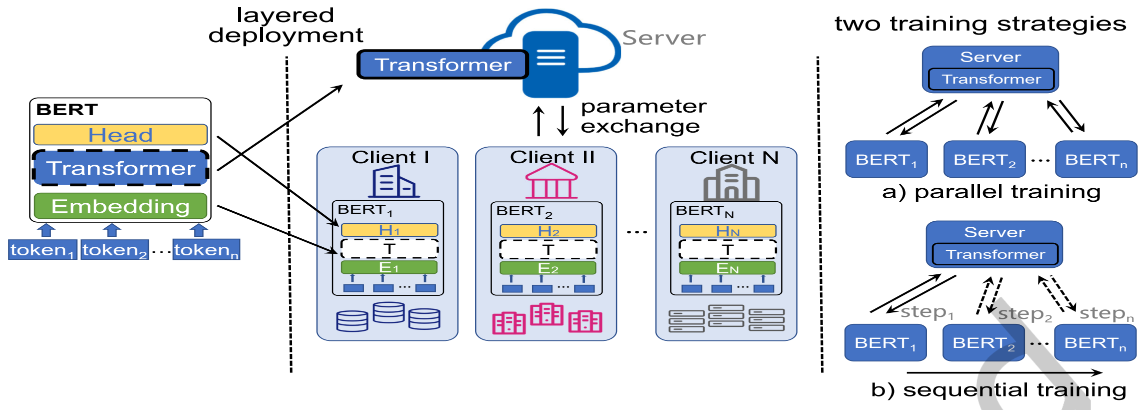


Fig. 4. Overview of FEDBERT

3 FEDBERT

In this section, we first introduce the proposed FEDBERT framework, then explain the two training methods for the proposed model. At last, the variants of our split learning approach are discussed.

3.1 An Overview

Fig. 4 describes the framework of our proposed FEDBERT. The BERT model is split into three parts: head layer, Transformer layer, and embedding layer. As the Transformer layer is the most computationally heavy part, it can be deployed in a powerful central server when there is no acceleration hardware in the local client. Each client just uses their own data to train the head layer and embedding layer locally. The central server receives the gradient of forward and back propagation from clients and updates the Transformer layer. Therefore a local BERT is composed by the two layers stored locally and the shared Transformers in server. This split computing model saves huge amount of computing cost for local clients, making it possible for mobile devices to join pre-training BERT task.

Federated split learning (FSL) combines the strength of FL, which is parallel processing among distributed clients, and the strength of SL, which is a neural network split during training. Our pre-training task can be formulated as: given a server S, n clients $C_1, C_2, \ldots, C_n$, the federated split learning for pre-trained model is to establish a collaborative learning between the server and the clients. The k-th client trains the embedding layer $\mathbf{W}_k^E$ and the head layer $\mathbf{W}_k^H$, and the server is to update the Transformer layer $\mathbf{W}^T$ for all clients.

Based on the above settings, our proposed framework FEDBERT can adopt two federated training methods according to the client's computing resource. One training method is the naive federated learning approach without network split for resource-rich clients. Particularly, we use FedAvg [37] to achieve naive federated learning. Second, for the resource-constrained devices, we use federated split learning to train our split BERT models.

3.2 FEDBERT via Federated Learning (FedAvg)

Nowadays, a large amount of data is constantly being generated on the client side and users want to keep their training data private. Traditional deep learning approaches need to combine all data together at one server, which may violate the laws on user privacy and data confidentiality. Federated learning, which trains model on

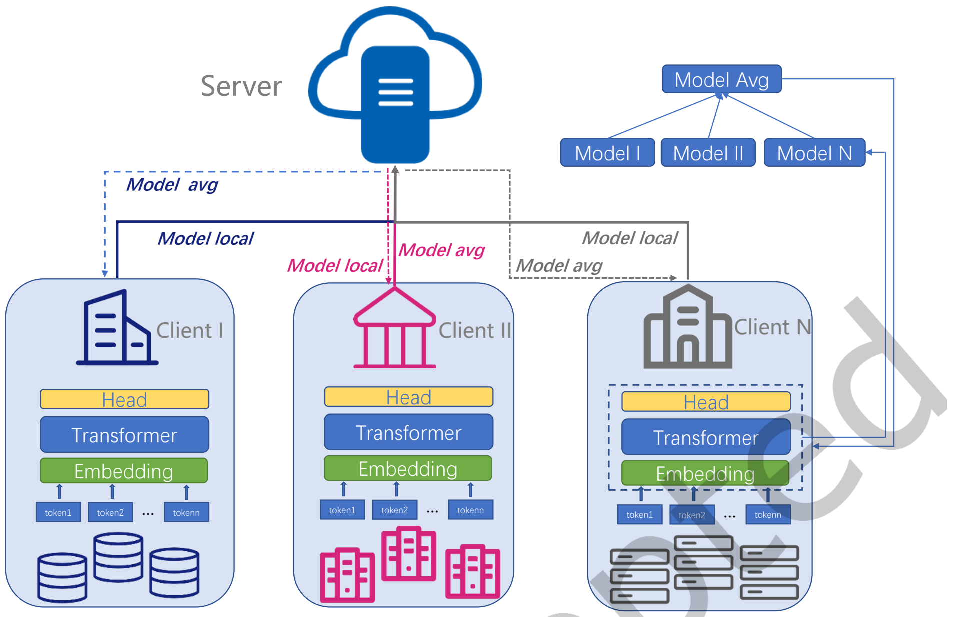


Fig. 5. Pre-training BERT in a federated way

each client without data sharing, has attracted a growing interest [5, 26, 29]. FL allows multiple data owners to collaboratively contribute a shared prediction model while keeping the local data decentralized and private.

For the pre-train BERT task, we first use the FedAvg [37] method to achieve the goal without clients' resource constrain. It assumes that each client has the capability to train the whole BERT model without model splitting. The FedAvg based pre-training workflow for BERT is shown in Fig. 5. Each client trains its local BERT model using its own data and sends the BERT model parameters to the server. The server aggregates the updates and sends back the global BERT model to all clients in each round. In general, there are N clients generating consecutive local BERT model $\mathbf{W}_k$. At training epoch t, N clients send its local model optimized $\mathbf{W}_{k,t}$ to the server. The global model $\mathcal{W}_t$ is calculated by Eq. 3 and sent back to all clients. The FedAvg based training approach for BERT pre-training is summarized in Algorithm 1.

3.3 FedBERT via Federated Split Learning

Requiring each client to build and maintain a powerful computing infrastructure is a strong constraint [26, 64]. As a result, it becomes difficult to deploy such powerful but computational expensive language representation models into computation constrained devices including mobile phones and edge devices. To release this constrain, we propose a federated split learning approach to solve the pre-training task on constrained device.

As shown in Fig. 1, the BERT can be split into three blocks and the Transformer layer has large computational and memory requirements. So the Transformer layer is deployed in a powerful central server. During the pre-training process, each client's embedding layer and head layer exchange parameters with server Transformer layer. The client, with training sample, performs forward propagation from embedding layer to Transformer layer at server. The server receives forward propagation from client and obtains the output of Transformer layer. Next the server sends back the output to client as input of head layer. The client computes the final output at the

Algorithm 1 FEDBERT pre-training via FedAvg

```
Require: A Server $S$; Clients set $C$; The faction of clients selected in each round $F$; The number of local epoch $E$
    and local batch set $B$; The learning rate $\eta$

    Server:
1:  initializes $\mathbf{W}_0$
2:  for round $t = 0, 1, \ldots$ do
3:      distributes $\mathbf{W}_t$ to all clients
4:      $C_t \leftarrow$ (randomly select $F \times C$ clients)
5:      for all client $k \in C_t$ do in parallel
6:          $\mathbf{W}_{k,t+1} \leftarrow$ ClientUpdate($\mathbf{W}_t$)          // distribute $\mathbf{W}_t$ to client and receive update
7:      end for
8:      Compute global model $\mathbf{W}_{t+1} = \sum_{k=1}^{|C_t|} \frac{n_k}{n} \mathbf{W}_{k,t+1}$
9:  end for

    Client $k$:
    ClientUpdate($\mathbf{W}_t$):
10: $\mathbf{W}_{k,t} \leftarrow \mathbf{W}_t$          // $\mathbf{W}_t$ is from server
11: for each local epoch $e$ from 1 to $E$ do
12:     for batch $b \in B$ do
13:         Compute forward propagation (calculate loss), backward propagation (calculate gradients $\nabla W_{k,b,t}$)
14:         $\mathbf{W}_{k,t+1} \leftarrow \mathbf{W}_{k,t} - \eta \nabla W_{k,b,t}$
15:     end for
16: end for
```

last layer. Then the gradient is calculated and back propagated along the reverse path. The pre-training process of two clients are illustrated in Fig. 6.

Following the proposed federated split learning framework, there are two types of pre-training strategies for FEDBERT: (1) parallel training strategy, and (2) sequential training strategy, as shown in Fig. 4. In parallel training method, all clients train the BERT at the same time. The server receives the local BERT updates and aggregates those updates. The sequential training method allows each client to use their own data to train their BERT models one by one. The clients in these two training methods only need to train embedding layers and head layers. The Transformer layers are only trained at server side. The forward and backward propagation between each layer of the neural network is managed by split learning approach.

Below, we give a formulation description for federated split learning approach. The three layers (i.e., Embedding layer, Head layer, and Transformer layer) in federated split learning are denoted as $\mathbf{W}^E$, $\mathbf{W}^H$, and $\mathbf{W}^T$. Given a server S, n clients $C_1, C_2, \ldots, C_n$, the federated split learning for pre-trained model is to establish a collaborative learning between the server and the clients, where the k-th client trains the embedding layer $\mathbf{W}_k^E$ and the head layer $\mathbf{W}_k^H$, and the server updates the Transformer layer $\mathbf{W}^T$ for all clients. Mathematically, the learning approach can be defined as follows:

$$\overline{\mathbf{W}^E} = \sum_{k=1}^{N} \frac{n_k}{n} (\mathbf{W}_k^E), \qquad \overline{\mathbf{W}^T} = \sum_{k=1}^{N} \frac{n_k}{n} (\mathbf{W}_k^T), \qquad \overline{\mathbf{W}^H} = \sum_{k=1}^{N} \frac{n_k}{n} (\mathbf{W}_k^H), \tag{6}$$
$$F(\mathbf{X}) = \overline{\mathbf{W}^H}(\overline{\mathbf{W}^T}(\overline{\mathbf{W}^E} X))$$

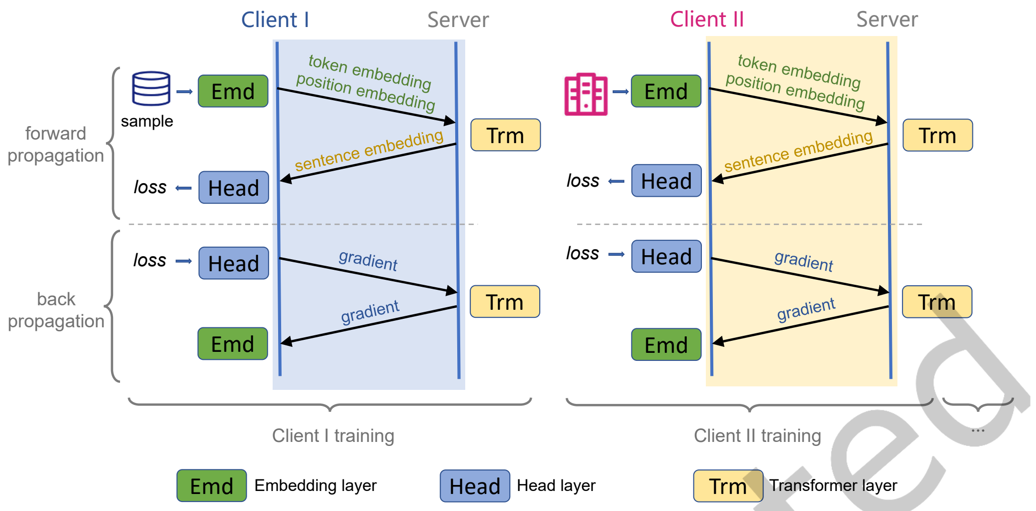


Fig. 6. The workflow of pre-training process

The aggregated outputs of three layers are computed as $\overline{\mathbf{W}^E}$, $\overline{\mathbf{W}^T}$, and $\overline{\mathbf{W}^H}$. Based on these aggregated outputs, we rebuild the network architecture of BERT and calculate the final output $F(\mathbf{X})$.

For these two training strategies, the server maintains different size of Transformer blocks. For parallel training method, each client communicates with server individually. In order to maintain the consistency of the model during each training epoch, the server stores N Transformer blocks for each client. After all clients finish their local training, the server will aggregate N clients' Transformer blocks to a global Transformer block. Then the global Transformer block is shared to all the clients. For sequential training method, the server only needs to maintain a Transformer block. Each client will update the Transformer block one by one in each training epoch. In the following section we will introduce these two strategies in detail.

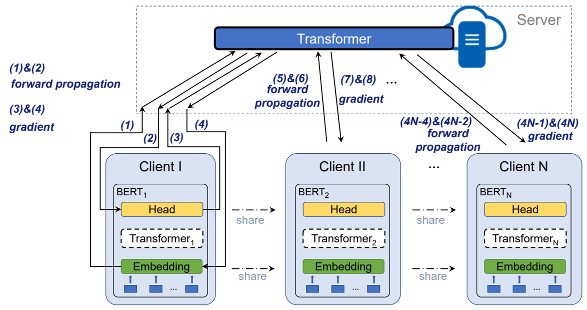
3.3.1 Parallel Training Strategy. In parallel learning method, each client will start the training process individually. In each client, raw data is encoded by embedding layer, and then the corresponding token and position embeddings are sent to the server for further process. The server receives embeddings and output sentence embedding via Transformer blocks. The sentence embedding is sent back to all clients and computes loss via head layers. At server side, an independent Transformer blocks $\mathbf{W}_k^T$ is maintained for the client k. The client k only updates the parameters to $\mathbf{W}_k^T$ in each training epoch. Thus, there are $N + 1$ Transformer blocks in server side. The $N + 1$-th Transformer block is the aggregated result.

For pre-training task, the language model head consists of full connection layer and normalization layer. Up to this time, the forward propagation is finished. Back propagation begins with head layers on client side. And then output gradient of head layers is uploaded to server side to update weights of Transformer blocks. Finally, output gradient of Transformer blocks is sent back to update weights of embedding layers on client. The whole update is done in a split way. For embedding and head sharing mode, after a certain period of local epochs, the following client asks embedding and head layers from the former client, and then starts local training itself. This process takes place in turns until the last client. On the contrary, for two average modes, all clients execute local training for a certain period of epoch. Then the specified corresponding layers are uploaded and averaged on server. All clients continue local training based on the averaged layers and repeat the iteration until achieving maximum epoch. The parallel training strategy is summarized in Algorithm 2.

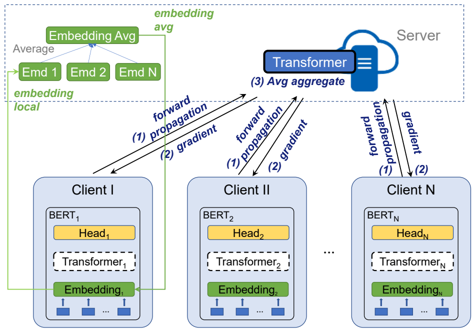
Algorithm 2 FedBERT pre-training via parallel federated split learning

```
Require: A Server $S$; Clients set $C$; The faction of clients selected in each round $F$; The number of local epoch $E$
    and local batch set $B$; The learning rate $\eta$

    Server:
1:  initializes $\mathbf{W}_0^T$
2:  for round t = 0, 1, ... do
3:      $C_t \leftarrow$ (random select $F \times C$ clients)
4:      copy $\mathbf{W}_t^T$ into $|C_t|$ copies $\{\mathbf{W}_{k,t}^T\}|_{k=1}^{|C_t|}$ for clients
5:      for all client $k \in C_t$ do in parallel
6:          ClientUpdate()
7:      end for
8:      Compute global model $\mathbf{W}_{t+1}^T = \sum_{k=1}^{|C_t|} \frac{n_k}{n} \mathbf{W}_{k,t+1}^T$
9:  end for

    ServerTrmFordprop($\mathcal{P}_{ef}$, $k$):
10: Compute forward propagation $\mathcal{P}_{tf}$ on Transformer layer $\mathbf{W}_{k,t}^T$
11: Send $\mathcal{P}_{tf}$

    ServerTrmBackprop($\mathcal{P}_{hb}$, $k$):
12: Compute backward propagation $\mathcal{P}_{tb}$ on Transformer layer $\mathbf{W}_{k,t}^T$ (calculate gradients $\nabla W_{k,b,t}^T$)
13: $\mathbf{W}_{k,t+1}^T \leftarrow \mathbf{W}_{k,t}^T - \eta \nabla W_{k,b,t}^T$
14: Send $\mathcal{P}_{tb}$

    Client $k$:
    ClientUpdate():
15: initializes $\mathbf{W}_0^E$, $\mathbf{W}_0^H$
16: for each local epoch $e$ from 1 to $E$ do
17:     for batch $b \in B$ do
18:         Compute forward propagation $\mathcal{P}_{ef}$ on embedding layer $\mathbf{W}_t^E$
19:         $\mathcal{P}_{tf} \leftarrow$ ServerTrmFordprop($\mathcal{P}_{ef}$, $k$)
20:         Compute forward propagation (calculate loss) on head layer $\mathbf{W}_t^H$
21:         Compute back propagation $\mathcal{P}_{hb}$ on head layer $\mathbf{W}_t^H$ (calculate gradients $\nabla W_{k,b,t}^H$)
22:         $\mathcal{P}_{tb} \leftarrow$ ServerTrmBackprop($\mathcal{P}_{hb}$, $k$)
23:         Compute back propagation $\mathcal{P}_{eb}$ on embedding layer $\mathbf{W}_t^E$
24:         $\mathbf{W}_{k,t+1}^E \leftarrow \mathbf{W}_{k,t}^E - \eta \nabla W_{k,b,t}^E$
25:         $\mathbf{W}_{k,t+1}^H \leftarrow \mathbf{W}_{k,t}^H - \eta \nabla W_{k,b,t}^H$
26:     end for
27: end for
```

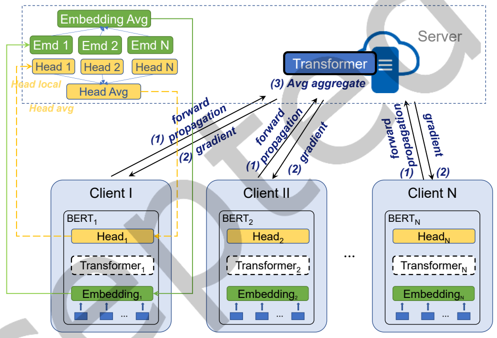
3.3.2 Sequential Training Strategy. In sequential learning method, each client will start the training process sequentially. Simpler than the parallel method, only client will update the forward propagation and back propagation at a time. Same as the parallel training strategy, each client encodes raw data by embedding layer, and then sends the corresponding token and position embeddings to the server. In general, each client runs an embedding



(a) Embedding and head layers sharing (sequential training)



(b) Embedding layers average (parallel training)



(c) Head and embedding average (parallel training)

Fig. 7. Three different types of federated pre-training via split learning

layer $\mathbf{W}_k^E$ and a head layer $\mathbf{W}_k^H$ in local and server runs a Transformer layer $\mathbf{W}^T$. During training, only one client k communicates with server and updates the layers $\mathbf{W}_k^E$, $\mathbf{W}^T$, and $\mathbf{W}_k^H$ using client's data.

Instead of maintaining $N + 1$ transformer blocks in server for parallel training strategy, sequential training only needs to maintain one Transformer block $\mathbf{W}^T$. Each client will exchange its parameters with $\mathbf{W}^T$ in turn. After the end of one global epoch, the Transformer block $\mathbf{W}^T$ learns information from N clients. As all clients sequentially update the Transformer block in server, there is no aggregation time. The sequential training strategy is summarized in Algorithm 3.

3.4 Variants of Federated Split Learning

Federated split learning circumvents challenges of data privacy and AI hegemony. Another issue is that few institutions have adequate infrastructure to afford the heavy computing tasks. For instance, hospitals own much valuable and privacy-sensitive data. They can unite their data and train a powerful federated model for professional tasks. However, there may lack enough computational resource in hospitals as heavy computing tasks are not major aim for information construction in hospitals. Therefore, the whole model can be trained in a split way [16, 60].

For all Transformer-based model like BERT, the Transformer blocks dominant computing operations, which can be migrated to supercomputing clusters. Clients only execute embedding layers and head layers. As depicted in Fig. 7, three types of federated pre-training protocols via split learning are proposed. Respectively, Fig. 7a shows situation when embedding and head layers are shared, Fig. 7b shows the embedding average pattern,

Algorithm 3 FedBERT pre-training via sequential federated split learning

```
Require: A Server $S$; Clients set $C$; The faction of clients selected in each round $F$; The number of local epoch $E$
    and local batch set $B$; The learning rate $\eta$

    Server:
 1: initializes $\mathbf{W}_0^T$
 2: for round t = 0, 1, ... do
 3:    $C_t \leftarrow$ (random select $F \times C$ clients)
 4:    copy $\mathbf{W}_t^T$ into $|C_t|$ copies $\{\mathbf{W}_{k,t}^T\}|_{k=1}^{|C_t|}$ for clients
 5:    for client $k \in C_t$ do
 6:       ClientUpdate()
 7:    end for
 8:    Compute global model $\mathbf{W}_{t+1}^T = \sum_{k=1}^{|C_t|} \frac{n_k}{n} \mathbf{W}_{k,t+1}^T$
 9: end for

    ServerTrmFordprop($\mathcal{P}_{ef}$, $k$):
10: Compute forward propagation $\mathcal{P}_{tf}$ on Transformer layer $\mathbf{W}_{k,t}^T$
11: Send $\mathcal{P}_{tf}$

    ServerTrmBackprop($\mathcal{P}_{hb}$, $k$):
12: Compute backward propagation $\mathcal{P}_{tb}$ on Transformer layer $\mathbf{W}_{k,t}^T$ (calculate gradients $\nabla W_{k,b,t}^T$)
13: $\mathbf{W}_{k,t+1}^T \leftarrow \mathbf{W}_{k,t}^T - \eta \nabla W_{k,b,t}^T$
14: Send $\mathcal{P}_{tb}$

    Client $k$:
    ClientUpdate():
15: initializes $\mathbf{W}_0^E$, $\mathbf{W}_0^H$
16: for each local epoch $e$ from 1 to $E$ do
17:    for batch $b \in B$ do
18:       Compute forward propagation $\mathcal{P}_{ef}$ on embedding layer $\mathbf{W}_t^E$
19:       $\mathcal{P}_{tf} \leftarrow$ ServerTrmFordprop($\mathcal{P}_{ef}$, $k$)
20:       Compute forward propagation (calculate loss) on head layer $\mathbf{W}_t^H$
21:       Compute back propagation $\mathcal{P}_{hb}$ on head layer $\mathbf{W}_t^H$ (calculate gradients $\nabla W_{k,b,t}^H$)
22:       $\mathcal{P}_{tb} \leftarrow$ ServerTrmBackprop($\mathcal{P}_{hb}$, $k$)
23:       Compute back propagation $\mathcal{P}_{eb}$ on embedding layer $\mathbf{W}_t^E$
24:       $\mathbf{W}_{k,t+1}^E \leftarrow \mathbf{W}_{k,t}^E - \eta \nabla W_{k,b,t}^E$
25:       $\mathbf{W}_{k,t+1}^H \leftarrow \mathbf{W}_{k,t}^H - \eta \nabla W_{k,b,t}^H$
26:    end for
27: end for
```

and Fig. 7c shows situation when both head and embedding layers are averaged. For pattern (a), each client encodes raw data and inferences masked tokens using the prior client's embedding and head layers in turn refer to [16]. Although it is not in strict conformance with FegAvg [37], it still protects the privacy of raw data, and information is exchanged by embedding and head layer sharing to strengthen the model. Pattern (b) averages

Table 2. Cost Analysis per epoch of the three proposed training methods

Model	**Communication cost**		**Training time**	
	per Client	**Server**	**per Client**	**Server**
FedAvg	$2\lvert\mathbf{W}\rvert$	$2\lvert\mathbf{W}\rvert$	$\mathcal{T}(\mathbf{W})$	$\mathcal{T}_a(\mathbf{W})$
Parallel FSL	$4pL$	$N \cdot 4pL$	$\mathcal{T}(\mathbf{W}^E) + \mathcal{T}(\mathbf{W}^H)$	$\mathcal{T}(\mathbf{W}^T) + \mathcal{T}_a(\mathbf{W}^T)$
Sequential FSL	$4pL$	$N \cdot 4pL$	$\mathcal{T}(\mathbf{W}^E) + \mathcal{T}(\mathbf{W}^H)$	$N \cdot \mathcal{T}(\mathbf{W}^T)$

embedding layers after a definite epoch and pattern (c) averages both embedding and head layers. For all three patterns, there is only one set of Transformer blocks in server during forward and back propagation. They obey the same training protocol, expect the information processing mode of embedding and head layers.

3.5 Cost Analysis and Discussion

In this section, we analyze the communication cost and training time for the FedAvg, parallel FSL, and sequential FSL approaches. The communication cost is the total amount of data transmitted by device. The training time describes the computational complexity of the model. Given N clients, we assume that p is the number of training batches per epoch, $|\mathbf{W}|$ is the parameter size of BERT model, L is the parameter size of propagation between the cut layers, $\mathcal{T}(\mathbf{W})$ is the model training time for one global epoch, $\mathcal{T}_a(\mathbf{W})$ is the federated averaging time for model $\mathbf{W}$. Here we consider the faction of clients F is 1, which means all clients' model are available in each epoch. So for the split learning approach, the original computation cost of model BERT can be expressed as three parts: $\mathcal{T}(\mathbf{W}) = \mathcal{T}(\mathbf{W}^E) + \mathcal{T}(\mathbf{W}^T) + \mathcal{T}(\mathbf{W}^H)$.

Table 2 provides the communication cost and the training time of each client and server for one training epoch. For the communication cost, although parallel FSL and sequential FSL transmit the same amount of data, the client in parallel FSL transmits data in parallel with the server. So the data exchange between parallel FSL is N times faster than sequential FSL. For the training complexity, as the model size of $\mathbf{W}^T$ is larger than $\mathbf{W}^E$ and $\mathbf{W}^H$, the training time $\mathcal{T}(\mathbf{W}^T)$ is longer than $\mathcal{T}(\mathbf{W}^E) + \mathcal{T}(\mathbf{W}^H)$. As all clients sequentially update the Transformer block in server, there is no aggregation time. Comparing with client in FedAvg, the client in FSL can use less computing resources for training: $\mathcal{T}(\mathbf{W}^E) + \mathcal{T}(\mathbf{W}^H) \ll \mathcal{T}(\mathbf{W}^T) < \mathcal{T}(\mathbf{W})$.

For the performance, sequential FSL has higher accuracy than parallel FSL. This is because the sequential FSL learns the knowledge from the entire dataset and parallel FSL only obtains the knowledge from the participate clients. The sequential FSL is suitable for the clients who want higher accuracy and not sensitive to training speed.

4 EMPIRICAL EVALUATION

In this section, we conduct experiments led by the following research questions:

- **RQ1:** Is our proposed FedBERT effective in pre-training BERT over a large-scale corpus?
- **RQ2:** What is the effectiveness of each split module of FedBERT during BERT pre-training?
- **RQ3:** Does the average period matter?
- **RQ4:** What is the performance of FedBERT when varying the number of clients?

4.1 Datasets

We conduct pre-training tasks based on Fairseq[1] and evaluate models on the *General Language Understanding Evaluation* (GLUE) benchmark [61], which is a collection of diverse natural language understanding tasks. In

[1] https://github.com/pytorch/fairseq

particular, our experiments are conducted on seven downstream tasks in GLUE: SST-2, MRPC, STS-B, QQP, MNLI, QNLI, RTE. The descriptions of each task are presented as follows.

- **SST-2** *(Stanford Sentiment Treebank)* [52] is a binary single-sentence classification task consisting of sentences extracted from movie reviews with human annotations of their sentiments.
- **MRPC** *(Microsoft Research Paraphrase Corpus)* [11] consists of sentence pairs annotated for whether the sentences in the pair are semantically equivalent.
- **STS-B** *(The Semantic Textual Similarity Benchmark)* [8] is a collection of sentence pairs annotated with a score denoting how similar the two sentences are in terms of semantic meaning.
- **QQP** *(Quora Question Pairs)*[2] is a binary classification task where the goal is to determine if two questions asked on Quora are semantically equivalent.
- **MNLI** *(Multi-Genre Natural Language Inference)* [65] is an entailment classification task that predicts whether one sentence is an entailment, contradiction, or neutral with respect to another one.
- **QNLI** *(Question Natural Language Inference)* [61] is a version of the Stanford Question Answering Dataset [47] which has been converted to a binary classification task.
- **RTE** *(Recognizing Textual Entailment)* [4] is a binary entailment task similar to MNLI, but with much less training data.

Among these seven NLP tasks, MNLI, QQP, QNLI, STS-B, MRPC, and RTE are sentence pair classification tasks, and SST-2 task is a single sentence classification task. Thus, the embedding of `[CLS]` is adopted as sentence embedding, as shown in Fig. 2. Because the STS-B task measures the similarity between two sentences by scoring, we use Pearson's correlation coefficient as the performance metric. The others are evaluated by accuracy.

4.2 Experimental Setup

Experiments are conducted on 4 nodes with 4 NVIDIA V100 GPUs, 32GB RAM per node and one node with 8 NVIDIA V100 GPUs, 16GB RAM. To demonstrate the effectiveness of our proposed methods, we select three language models RoBERTa_small, RoBERTa_base, and GPT2 to achieve our pre-training task. The parameters of architecture are shown in Table 3. To illustrate the effectiveness of our proposed FedBERT, all pre-trained models are evaluated on GLUE benchmark. There are 11 tasks in GLUE benchmark totally. Following [9], seven tasks about single sentence classification and sentence-pair classification are chosen in our setting. Moreover, a small version of GPT-2 with 12 decoder layers is pre-trained to further illustrate general significance of the proposed method. The performance of the pre-trained model is evaluated by perplexity (PPL). Smaller PPL indicates better performance. We implement the aforementioned training patterns, including centralized, averaging the whole model (naive FedAvg), split learning with sharing head and embedding layers, split learning with averaging only embedding layers, and split learning with averaging both head and embedding layers. We use WikiText103[3] as the training corpus. Hyperparameters of pre-training for all the aforementioned pre-training tasks are given in Table 4.

4.2.1 *Training Protocols.*

The training protocols of several patterns are demonstrated as follows.

- **Centralized** pre-training uses the whole dataset with batch size 256 and 125,000 updates for RoBERTa. The GPT is trained with 50000 updates. The best checkpoint is adopted for evaluation.
- **Naive FedAvg** pre-training equally distributes dataset to 10 clients for RoBERTa and 5 clients for GPT respectively. For RoBERTa, we set the average period as 50 epochs. The latest model of all clients are averaged on server every 50 epochs. Then all clients begin the new round based on the averaged model until maximum epoch 500. For GPT, the number of clients is 5, and the average period is 5 epoch. Totally,

[2] https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs
[3] https://www.salesforce.com/products/einstein/ai-research/the-wikitext-dependency-language-modeling-dataset/

Table 3. Parameters of model structure [9, 10, 46]

Parameters	Small	Base	GPT2
Layers(L)	12	12	12
Hidden Size (H)	256	768	768
FFN* Size (4H)	1024	3072	3072
Attention Heads (A)	4	12	12
Total Parameters	23M	110M	117M

Table 4. Hyperparameters of pre-training tasks

Parameters	Pre-training
Batch Size	256
Total Updates	125000
Warmup Updates	10000
Peak Learning Rate	0.0005
LR Scheduler	polynomial_decay
Adam β_1	0.9
Adam β_2	0.98
Adam ϵ	1e-06
Weight Decay	0.1
Dropout	0.1
Attention Dropout	0.1
#Clients	10
Average Period	50 epochs

all clients are trained with 20 epochs, thus average 4 times. The averaged model of the final epoch from all clients is adopted for evaluation. This method is described in Algorithm 1.

- **Split Training with HE Sharing** equally distributes dataset to 10 clients for RoBERTa and 5 clients for GPT respectively. The local maximum epoch is 500 epochs for RoBERTa. Clients are trained in turns. The first client is trained based on its local data in a split way. After 500 epochs, the first client finishes its local training and shares the *head and embedding* (HE) layers to the second client. Then the second client starts local training based on the head and embedding layers of the first one. It is the same procedure for GPT except that the maximum epoch is set as 20 epochs. The best checkpoint of last client is chosen for evaluation. This method is described in Algorithm 3 when head and embedding layers are set to share. Fig. 7a illustrates this method.
- **Split Training with Embedding Avg** distributes equally dataset to 10 clients for RoBERTa and 5 clients for GPT respectively. The average period is set as 50 epochs. The embedding layers are averaged on the server every 50 epochs. The Transformer blocks and head layers are updated as normal. Clients use the averaged layers to train a new round until maximum epoch. It is the same procedure for GPT except that the maximum epoch is set as 20 epochs. Fig. 7b illustrates this method.
- **Split Training with HE Avg** is similar to split training with embedding avg. The only difference is that this pattern averages both head and embedding layers. Fig. 7c illustrates this method.
- **Split Training with Parallel FedAvg** is implemented based on naive FedAvg. For Naive FedAvg, the whole model is trained on client side. After the specific epochs, all model are averaged on server, and the averaged model is used as the initial model for the next epoch. On the contrary, as for parallel FedAvg, Transformers are deployed on server and the server keeps a copy of Transformer for each client. The local models will continue to learn until the next participates in aggregation. Thus all clients can train in parallel. This method is described in Algorithm 2.
- **Single Training** uses the data of a single client and trains the model in a centralized way.

Moreover, in order to explore the influence of the number of clients and average period, naive FedAvg and split training with embedding layers average in 20 clients setting, 100 epoch period are added respectively. In our experiments, the data is distributed equally as in the federated learning. Thus weighted average is not executed, which is necessary when the number of samples are unequal. Note that here we assume all clients are included

Table 5. Hyperparameters of fine-tuning tasks of GLUE. STS-B is a regression task

Hyperparameters	**RTE**	**MRPC**	**MNLI**	**QNLI**	**QQP**	**STS-B**	**SST-2**
Number of Classes	2	2	3	2	2	-	2
Batch Size	256	256	512	512	512	256	512
Total Updates	2036	2296	123873	33112	113272	3598	20935
Warmup Updates	122	137	7432	1986	28318	214	1256
Peak Learning Rate	2e-05	1e-05	1e-05	1e-05	1e-05	2e-05	1e-05
LR Scheduler	polynomial_decay						
Adam β_1	0.9						
Adam β_2	0.98						
Adam ϵ	1e-06						
Weight Decay	0.1						
Dropout	0.1						
Attention Dropout	0.1						
Maximum Epoch	10						

during training to fully utilize the data. The client selection strategy and non-iid issue are beyond the scope of this paper. Moreover, the number of clients is 10 for RoBERTa and 5 for GPT, and the period of average is 50 epochs for RoBERTa and 5 for GPT if not specified.

4.2.2 Implementation Details. For the RoBERTa$_{\texttt{small}}$ and RoBERTa$_{\texttt{base}}$ models, we follow the hyperparameters from the settings [30] for the most part. As our training target is to evaluate the efficiency of different FSL approaches, we only use a small training dataset provided by Fairseq. So the performance of the fine-tuning tasks may not be comparable with benchmarks. But for fair comparison, we use the same settings for the different approaches. Based on the training settings in Table 3 and Table 4, the pre-training task based on RoBERTa$_{\texttt{base}}$ takes about a week on 8 V100 GPU cards.

After pre-training in various patterns, the RoBERTa$_{\texttt{small}}$ and RoBERTa$_{\texttt{base}}$ models are fine-tuned and evaluated on 7 datasets. For fine-tuning tasks, all final pre-trained models are evaluated based on the same setting. Table 5 summarizes the hyperparameters. The best model is used by fine-tuning for the centralized pre-training and embedding sharing pattern. The other three modes related to weights average use the averaged model of the last epoch to evaluate their effect. To simplify the comparison between different modes, SST-2, QQP, MRPC, RTE, MNLI, and QNLI are measured by *accuracy*, and STS-B is measured by *Pearson Corr*. Furthermore, MNLI task includes two results corresponding to the matched and mismatched accuracy. The GPT is evaluated on WikiText-103 by PPL after pre-trained.

5 EXPERIMENTAL RESULTS AND ANALYSIS

In this section, we present the experimental results with a comprehensive analysis to answer the aforementioned research questions.

We used two different sizes of pre-train models to demonstrate the effectiveness of our proposed method. The evaluation results of model RoBERTa$_{\texttt{small}}$ and RoBERTa$_{\texttt{base}}$ are illustrated in Table 6 and Table 7. The result of GPT2$_{\texttt{small}}$ is shown in Table 8. From these tables, we notice that the centralized model has the best performance as it is trained on the whole dataset. Our proposed split federated learning approaches can achieve the same performance as classical federate learning method FedAvg.

Table 6. The comparison of pre-trained RoBERTa$_{\mathtt{small}}$ via federated learning approaches on the GLUE benchmark

Model	RTE	MRPC	MNLI	QNLI	QQP	STS-B	SST-2
Centralized	58.5	76.5	75.3/76.6	86.7	86.6	81.8	89.2
FedAvg	52.7	72.3	71.0/71.8	83.7	84.7	73.4	86.1
Head-Emd Share	59.2	77.5	73.4/75.2	85.4	85.9	78.3	88.9
EmdAvg	51.3	73.8	70.0/70.9	81.5	84.1	70.8	83.9
Head-Emd Avg	57.0	71.3	70.9/72.6	84.1	85.2	74.6	85.8
Parallel FedAvg	58.8	71.6	71.2/72.8	82.8	84.7	75.4	85.0
Single Client	57.4	70.3	70.0/71.4	82.4	84.6	70.2	84.7

Table 7. The comparison of pre-trained RoBERTa$_{\mathrm{base}}$ via federated learning approaches on the GLUE benchmark

Model	RTE	MRPC	MNLI	QNLI	QQP	STS-B	SST-2
Centralized	62.82	85.54	81.09/81.48	88.74	89.78	84.19	90.40
FedAvg	61.01	78.92	76.47/76.79	85.1	89.36	81.18	90.14
Head-Emd Share	57.04	78.92	76.82/77.69	84.35	88.42	78.45	90.37
EmdAvg	58.12	72.30	71.27/71.41	81.71	84.59	68.17	85.21
Head-Emd Avg	55.96	73.04	71.72/71.91	81.44	86.90	70.64	86.24
Single Client	58.84	73.53	71.72/71.91	81.79	86.11	68.81	87.16

Table 8. The comparison of pre-trained GPT2$_{\mathtt{small}}$ under the metric of perplexity. Smaller is better.

Model	PPL
Centralized	25.25
FedAvg	536572.25
Head-Emd Share	24.69
EmdAvg	39.03
Head-Emd Avg	39.52
Single Client	38.71

5.1 RQ1: Effectiveness of Federated Learning for Pre-Training

Table 6, Table 7 present the performance of pre-trained RoBERTa models. The centralized training model is a baseline. It has the best performance except for the RTE and MRPC task for RoBERTa$_{\mathtt{small}}$ model. This is because the dataset for the RTE and MRPC task are small and the centralized model is more tolerant to fall into the local optimum comparing with the distributed model. As split learning with head and embedding layers sharing is equal to the centralized training logically, the evaluation results should approximate the centralized one. However, we found that the learning rate decayed to 0 midway and the training process terminated for RoBERTa. Thus part of client data is not used for training, which leads to slightly worse results on these five tasks. For GPT, the share pattern gets a negligible decrease of 0.56 on PPL compared to centralized training. As a distributed learning pattern, federated learning is worse than a centralized model in theory. Among several federated split learning models, Head-Emd Avg achieves the best performance, while EmdAvg is worst generally. As for split learning with embedding average, all tasks under-perform the centralized and sharing pattern. The worst one is STS-B, which decreases 11% and 7.5% on *Person Corr*, compared with the centralized pattern and sharing pattern respectively. The rest tasks have a decrease around or below 5%. FedAvg is the suboptimal model among three FSL models by average. On MNLI, QNLI, QQP, STS-B, and SST-2 task, FedAvg under-performs about 1% than Head-Emd Avg, while it outperforms EmdAvg on MNLI, QNLI, QQP, STS-B, and SST-2. For parallel federated split learning based on FedAvg, each client keeps a corresponding copy of Transformer layers on server, thus all clients can train in parallel. After the specific epochs, all Transformer copies on server and head and embedding layers on client are averaged. The latest averaged model is adopted when new epoch starts. Logically, parallel FSL on FedAvg and naive FedAvg are equal. The difference is that server needs to keep multi Transformer transcripts for parallel model, while all training processes are done on clients for naive FedAvg. The evaluation results are shown in Table 6 and Table 7, parallel FedAvg and naive FedAvg achieve the same performance on all tasks except RTE. It is foreseeable that the performance of federated pre-training under-performs the centralized one. However, this consensus is based on the assumption that the aforementioned models are trained on the same dataset. There is no doubt that raw data is not exchanged by our proposed FEDBERT, which is meaningful in reality. On the one hand, this advantage can attract more clients to contribute private data in training. In addition, it enables general model pre-training in privacy-sensitive domains and for thin clients.

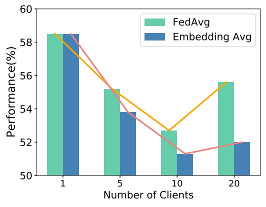
5.2 RQ2: Analysis of Each Split Module in BERT Pre-Training

BERT is split into three parts and processed in a different way in our experiments. Different training patterns lead to different performance. In detail, pre-training via FedAvg averages the whole model including embedding layers, Transformer blocks, and head layers. FEDBERT via split learning deploys the resource-consuming Transformer blocks on server, and the rest of layers updated on local clients are processed in various ways, including average and sharing. In this section, the performance of these variants are analyzed.

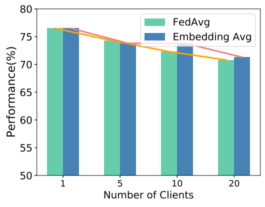
From Table 6 and Table 7, all patterns are based on the same setting. It can be observed that among four split learning training methods, Head-Emd share achieves the best performance. We hypothesize that the reason is that sharing head and embedding layers is logically equivalent to the centralized pattern. The FedAvg achieves the same performance to Head-Emd Avg except for RTE task. On MRPC, MNLI-Matched, and SST-2 tasks, FedAvg is better, while HE Avg outperforms on other tasks. Among all tasks, EmdAvg is the worst model except that it outperforms slightly than FedAvg and Head-Emd Avg on MRPC. We assume that Transformer block is an independent part. As for embedding layers and head layers trained on the same local data, they should better be updated in the same way in order to seek better performance. As for GPT, the performance of EmdAvg and Head-Emd Avg patterns are similar, which are lower than single within 1 on PPL. However, the naive FedAvg seems not applicable for GPT pre-training, which dose not converge in our experiment.

5.3 RQ3: Impact of Number of Clients

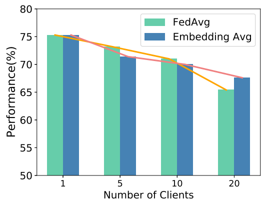
In federated settings, the whole training dataset is distributed to several clients. Thus, the number of clients may influence the final performance of pre-trained model. In this section, we made three groups of experiments based on FedAvg and EmdAvg to explore the effectiveness of this factor. The number of clients are set as 5, 10, 20 respectively. The evaluation results of centralized training are compared. Fig. 8 depicts the evaluation results on 7 GLUE tasks. For MNLI, it is split into matched and mismatched items. Overall, except RTE, other 6 tasks reveal a decreasing trend along with the increase of clients. Moreover, FedAvg is better than EmdAvg on almost all tasks and the number of clients settings, which is discussed in RQ2. As expected, the Centralized way presents the best performance. It implies that delicate optimization methods for distributed scenarios are required to improve performance. However, it is notable that this conclusion is based on the assumption that the same dataset is adopted, while a federated training pattern is able to gather more data than a single client.



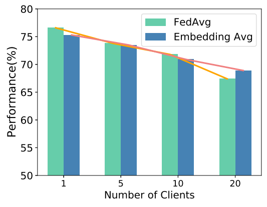
(a) RTE



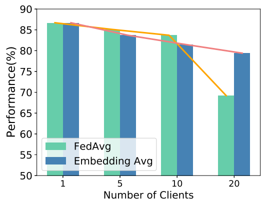
(b) MRPC



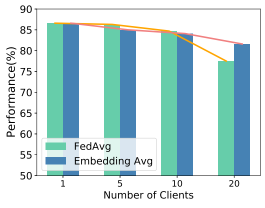
(c) MNLI-Matched



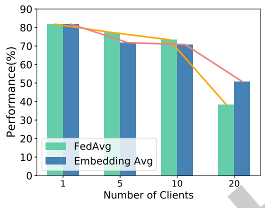
(d) MNLI-Mismatched



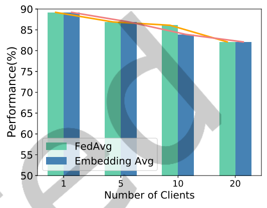
(e) QNLI



(f) QQP



(g) STS-B



(h) SST-2

Fig. 8. The performance of different pre-training patterns on various number of clients settings

5.4 RQ4: Effectiveness of Average Period

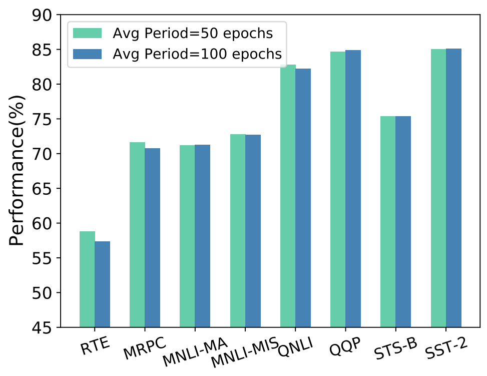
To explore the effectiveness of average period, we pre-trained FedAvg and Embedding Avg models with average period set as 100 epochs compared with 50 epochs. The evaluation results are shown in Fig. 9. For FedAvg pattern, model averaged every 100 epochs outperforms 50 epochs in MNLI-MA, QQP, and SST-2, and under-performs in RTE, MRPC, MNLI-MIS, and QNLI. The results of STS-B are both 75.4. The biggest difference value 1.4 appears in RTE task, the results of which are 58.8 and 57.4 respectively. The performance of other tasks are almost the same. As for Embedding Avg pattern, model averaged every 50 epochs outperforms 100 in RTE, MNLI, QNLI, and QQP tasks. The model averaged per 100 epochs achieves better performance in MRPC, STS-B, and SST-2 tasks. The biggest difference value 2.5 appears in RTE task, the results of which are 52.0 and 49.5 respectively. Among all the evaluated tasks, RTE is the most sensitive task to the average period. Even so, the difference is not significant between two training patterns. Overall, the average period does not make much influence on the performance of the pre-trained models.

6 RELATED WORK

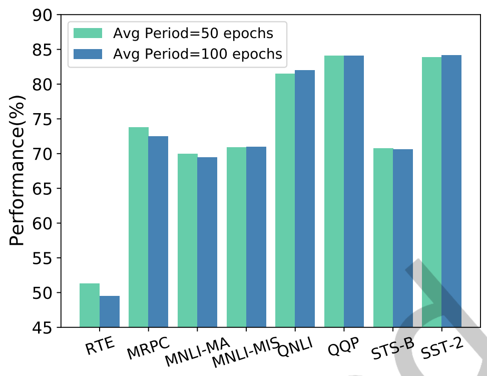
In this section, we survey the related work from the perspectives of federated learning, split learning, and model pre-training in NLP.

6.1 Federated Learning

Federated learning (FL) distributes machine learning or deep learning model training to the low computational edges from which data originates, emerged as a promising alternative ML paradigm [21, 36, 37, 68]. FL enables a multitude of participants to construct a joint ML model without sharing their private training data [6, 36, 37, 41]. Many research works in CV/NLP fields have been proposed in federated settings [18]. Specifically for the CV task, federated learning has been applied to many applications such as medical image analysis [22, 28, 48, 67], object detection [29], and landmark classification [19]. For NLP, FedNER [14] is proposed to handle medical NER



(a) FedAvg



(b) Embedding Avg

Fig. 9. Comparison between avg period of 50 epochs and 100 epochs on pre-training via FedAvg and Embedding Avg. MNLI-MA means matched item in MNLI task, and MNLI-MIS means mismatched item in MNLI task.

task under federated setting. A federated topic model framework [20] aims to collaboratively learn a shared topic model while maintaining privacy protection from multiple clients.

Recently, more and more works start to handle unbalanced, *non-independent and identically distributed* (non-I.I.D.) data, which naturally arise in the real world [27, 63, 69]. Due to the special properties of FL, FL has been widely applied into a wide range of real applications such as next word prediction [36, 38], visual object detection for safety [29], entity resolution [17], and medical prediction [66].

Moreover, in order to provide a formal privacy guarantee, recent works have investigated how to use *differential privacy* (DP) [15], *local differential privacy* (LDP) [49, 55] or *distributed differential privacy* (DDP) combined with cryptosystems [32] in federated learning. However, existing approaches can not be applied practically on deep learning [34, 62]. Some of them [49, 55] only focus on small tasks and simple datasets and do not support deep learning approaches yet. The other works [56] studied LDP on federated learning problem. However, as we discussed in the experiments part, both of them hardly achieve good performance with a limited privacy budget.

6.2 Split Learning

Split learning (SL) is a distributed and collaborative training approach, which is first proposed to solve the privacy-aware healthy data training problem [16, 60]. Different from the strategy of training an entire model on one device, SL splits a deep neural network into sub-network blocks and distributes them to multiple devices. The computational load of model training is distributed to multiple devices. Comparing with FL approach, the split learning approach can be deployed on resource-constrained devices [36].

When the sub-neural networks are deployed on multiple devices, SL may raise communication concern. Singh et al. [51] gave a comparison of communication efficiency between split learning and federated learning. SplitFed [58] united the split learning and federated learning together to improve the communication efficiency and enhance privacy. Moreover, Gao et al. [13] evaluated the communication overhead of Internet-of-Things applications, which was based on split learning framework.

Recently, SL has gained more attention and applied in many domains. Abuadbba et al. [2] showed how to apply split learning approach for 1-dimensional CNN applications. Similarly, Abedi et al. [1] evaluated the applicability of split learning for RNN applications. Praneeth et al. [35] proposed a variant of split learning named NoPeek, which attempted to reduce the potential leakage via communicated activation. This method reduced the distance correlation of the raw data but still maintained good evaluation performance. Lastly, a split learning based real-time object detection framework on edge devices was proposed in [35]. The framework reduced the data transmission between devices by using a small model after knowledge distillation.

6.3 Model Pre-Training

Our work is also related to model pre-training for NLP, which has brought dramatic improvements on natural language understanding [9, 10, 30, 57] and generation [12, 53]. The *pre-trained models* (PTMs) can be categorized into two classes: the pre-trained word embeddings, and the pre-trained contextual encoders. For pre-trained word embeddings, Mikolov et al. [40] proposed Word2Vec with two variants of *Continuous Bag-of-Words* (CBOW) and *Skip-Gram* (SG) based on pairwise ranking. Consequently, Pennington et al. [44] proposed GloVe which is computed by global word-word co-occurrence statistics from a large corpus. Based on them, several works have also extended the word embedding into the embeddings of sentence and document, such as paragraph vector [23] and Context2Vec [39].

In contrast to those works that learned the embeddings of word or sentence, another line of works aim to learn the contextual embedding for each word or the graph relationship within the sentence [42, 43, 71]. Recently, many works proposed to achieve this goal via pre-training, e.g., OpenAI GPT [45] and BERT [10]. These *pre-trained models* (PTMs) are mainly based on the Transformer [59] architecture, composed of multiple self-attention layers. To pre-train these models, a self-supervised learning approach is designed via the masking mechanism. That is, some words or sentences are masked, and the goal of model learning is to reconstruct them. The PTMs have advanced the performance of many specific domains, e.g., BioBERT [24] in medical texts, SciBERT in scientific articles [3], etc.

Recently, the model pre-training techniques have also been extended to the multi-media area of NLP and computer vision [25, 31, 54], to understand the multi-modal image and natural language. To preserve the privacy when pre-training a large model, this paper extends the pre-training into a federated learning paradigm. Without loss of generality, we select RoBERTa [30] as a representative model.

7 CONCLUSION AND FUTURE WORK

This paper has proposed a novel pre-training approach FedBERT that integrates federated learning and split learning to address the problem of pre-training large-scale language model on resource-limited devices and clients. Under this architecture, the computational expensive layer is deployed on the powerful server with additional communication cost. Our approach has the advantage of resource-constrained resource requirement and privacy preserving as well as keeps the state-of-the-art performance. Extensive experiments on seven GLUE tasks verify the effectiveness of our proposed FedBERT. It shows that the federated split learning approach, such as *Head-Emd Share* model, can achieve the same performance as classical federated setting. Based on this federated framework, users can protect sensitive data from sharing while using pre-trained models.

In our future work, we plan to extend our FedBERT to other PTMs and apply it to more domain specific tasks. In addition, we also aim to investigate the minimal requirement of each pre-training device to running different FSL approaches.

ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China under Grant No.62072204, and the Fundamental Research Funds for the Central Universities under Grant HUST:2020kfyXJJS019. Yao Wan is supported in part by National Natural Science Foundation of China under grand No. 62102157.

REFERENCES

[1] Ali Abedi and Shehroz S. Khan. 2020. FedSL: Federated Split Learning on Distributed Sequential Data in Recurrent Neural Networks. *arXiv preprint arXiv:2011.03180* (2020).

[2] Sharif Abuadbba, Kyuyeon Kim, Minki Kim, Chandra Thapa, Seyit A Camtepe, Yansong Gao, Hyoungshick Kim, and Surya Nepal. 2020. Can We Use Split Learning on 1D CNN Models for Privacy Preserving Training?. In *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*. ACM, 305–318.

[3] Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A pretrained language model for scientific text. *arXiv preprint arXiv:1903.10676* (2019).

[4] Luisa Bentivogli, Bernardo Magnini, Ido Dagan, Hoa Trang Dang, and Danilo Giampiccolo. 2009. The Fifth PASCAL Recognizing Textual Entailment Challenge. In *Proceedings of the 2nd Text Analysis Conference*. NIST.

[5] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloé Kiddon, Jakub Konecný, Stefano Mazzocchi, Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, and Jason Roselander. 2019. Towards Federated Learning at Scale: System Design. In *Proceedings of Machine Learning and Systems (MLSys'19)*. mlsys.org.

[6] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical Secure Aggregation for Privacy-Preserving Machine Learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 1175–1191.

[7] Iker Ceballos, Vivek Sharma, Eduardo Mugica, Abhishek Singh, Alberto Roman, Praneeth Vepakomma, and Ramesh Raskar. 2020. SplitNN-driven Vertical Partitioning. *arXiv preprint arXiv:2008.04137* (2020).

[8] Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. ACL.

[9] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. In *Proceedings of International Conference on Learning Representations*. OpenReview.net.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805* (2019).

[11] William B. Dolan and Chris Brockett. 2005. Automatically Constructing a Corpus of Sentential Paraphrases. In *Proceedings of the 3rd International Workshop on Paraphrasing*. Asian Federation of Natural Language Processing.

[12] Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. *arXiv preprint arXiv:1905.03197* (2019).

[13] Yansong Gao, Minki Kim, Sharif Abuadbba, Yeonjae Kim, Chandra Thapa, Kyuyeon Kim, Seyit A. Camtepe, Hyoungshick Kim, and Surya Nepal. 2020. End-to-end evaluation of federated learning and split learning for Internet of Things. *arXiv preprint arXiv:2003.13376* (2020).

[14] Suyu Ge, Fangzhao Wu, Chuhan Wu, Tao Qi, Yongfeng Huang, and Xing Xie. 2020. Fedner: Medical named entity recognition with federated learning. *arXiv preprint arXiv:2003.09288* (2020).

[15] Robin C. Geyer, Tassilo Klein, and Moin Nabi. 2017. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557* (2017).

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

[17] Stephen Hardy, Wilko Henecka, Hamish Ivey-Law, Richard Nock, Giorgio Patrini, Guillaume Smith, and Brian Thorne. 2017. Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. *arXiv preprint arXiv:1711.10677* (2017).

[18] Chaoyang He, Songze Li, Jinhyun So, Mi Zhang, Hongyi Wang, Xiaoyang Wang, Praneeth Vepakomma, Abhishek Singh, Hang Qiu, Li Shen, Peilin Zhao, Yan Kang, Yang Liu, Ramesh Raskar, Qiang Yang, Murali Annavaram, and Salman Avestimehr. 2020. FedML: A Research Library and Benchmark for Federated Machine Learning. *arXiv preprint arXiv:2007.13518* (2020).

[19] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. 2020. Federated visual classification with real-world data distribution. In *Procedings of the 16th European Conference on Computer Vision*. Springer, 76–92.

[20] Di Jiang, Yuanfeng Song, Yongxin Tong, Xueyang Wu, Weiwei Zhao, Qian Xu, and Qiang Yang. 2019. Federated topic modeling. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. ACM, 1071–1080.

[21] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaïd Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konecný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Mariana Raykova, Hang Qi, Daniel Ramage, Ramesh Raskar, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr,

Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. 2019. Advances and Open Problems in Federated Learning. *arXiv preprint arXiv:1912.04977* (2019).

[22] Georgios A. Kaissis, Marcus R. Makowski, Daniel Rückert, and Rickmer F. Braren. 2020. Secure, privacy-preserving and federated machine learning in medical imaging. *Nature Machine Intelligence* 2, 6 (2020), 305–311.

[23] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International conference on machine learning*. PMLR, 1188–1196.

[24] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics* 36, 4 (2020), 1234–1240.

[25] Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. 2019. Visualbert: A simple and performant baseline for vision and language. *arXiv preprint arXiv:1908.03557* (2019).

[26] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. 2020. Federated Learning: Challenges, Methods, and Future Directions. *IEEE Signal Processing Magazine* 37, 3 (2020), 50–60.

[27] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. 2019. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189* (2019).

[28] Quande Liu, Cheng Chen, Jing Qin, Qi Dou, and Pheng-Ann Heng. 2021. FedDG: Federated Domain Generalization on Medical Image Segmentation via Episodic Learning in Continuous Frequency Space. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. 1013–1023.

[29] Yang Liu, Anbu Huang, Yun Luo, He Huang, Youzhi Liu, Yuanyuan Chen, Lican Feng, Tianjian Chen, Han Yu, and Qiang Yang. 2020. Fedvision: An online visual object detection platform powered by federated learning. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence*. AAAI Press, 13172–13179.

[30] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692* (2019).

[31] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. 2019. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *arXiv preprint arXiv:1908.02265* (2019).

[32] Lingjuan Lyu. 2020. Lightweight crypto-assisted distributed differential privacy for privacy-preserving distributed learning. In *Proceedings of 2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–8.

[33] Lingjuan Lyu, Han Yu, Xingjun Ma, Lichao Sun, Jun Zhao, Qiang Yang, and Philip S. Yu. 2020. Privacy and Robustness in Federated Learning: Attacks and Defenses. *arXiv preprint arXiv:2012.06337* (2020).

[34] Lingjuan Lyu, Han Yu, and Qiang Yang. 2020. Threats to Federated Learning: A Survey. *CoRR* abs/2003.02133 (2020). arXiv:2003.02133

[35] Yoshitomo Matsubara and Marco Levorato. 2020. Neural Compression and Filtering for Edge-assisted Real-time Object Detection in Challenged Networks. *arXiv preprint arXiv:2007.15818* (2020).

[36] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*. PMLR, 1273–1282.

[37] H. Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas. 2016. Federated learning of deep networks using model averaging. *arXiv preprint arXiv:1602.05629* (2016).

[38] H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. 2018. Learning Differentially Private Recurrent Language Models. In *Proceedings of the 6th International Conference on Learning Representations*. OpenReview.net.

[39] Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional LSTM. In *Proceedings of the 20th SIGNLL conference on computational natural language learning*. ACL, 51–61.

[40] Tomás Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Proceedings of 27th Annual Conference on Neural Information Processing Systems*. 3111–3119.

[41] Hao Peng, Haoran Li, Yangqiu Song, Vincent Zheng, and Jianxin Li. 2021. Differentially Private Federated Knowledge Graphs Embedding. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. ACM, 1416–1425.

[42] Hao Peng, Renyu Yang, Zheng Wang, Jianxin Li, Lifang He, Philip Yu, Albert Zomaya, and Raj Ranjan. 2021. LIME: Low-Cost Incremental Learning for Dynamic Heterogeneous Information Networks. *IEEE Trans. Comput.* (2021).

[43] Hao Peng, Ruitong Zhang, Yingtong Dou, Renyu Yang, Jingyi Zhang, and Philip S. Yu. 2021. Reinforced Neighborhood Selection Guided Multi-Relational Graph Neural Networks. *arXiv preprint arXiv:2104.07886* (2021).

[44] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. ACL, 1532–1543.

[45] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding with unsupervised learning. *Technical report, OpenAI* (2018).

[46] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.

[47] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100, 000+ Questions for Machine Comprehension of Text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. ACL, 2383–2392.

[48] Nicola Rieke, Jonny Hancox, Wenqi Li, Fausto Milletari, Holger R. Roth, Shadi Albarqouni, Spyridon Bakas, Mathieu N. Galtier, Bennett A. Landman, Klaus Maier-Hein, Sébastien Ourselin, Micah Sheller, Ronald M. Summers, Andrew Trask, Daguang Xu, Maximilian Baust, and M. Jorge Cardoso. 2020. The future of digital health with federated learning. *NPJ digital medicine* 3, 1 (2020), 1–7.

[49] Mohamed Seif, Ravi Tandon, and Ming Li. 2020. Wireless federated learning with local differential privacy. *arXiv preprint arXiv:2002.05151* (2020).

[50] Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909* (2015).

[51] Abhishek Singh, Praneeth Vepakomma, Otkrist Gupta, and Ramesh Raskar. 2019. Detailed comparison of communication efficiency of split learning and federated learning. *arXiv preprint arXiv:1909.09145* (2019).

[52] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. ACL, 1631–1642.

[53] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. Mass: Masked sequence to sequence pre-training for language generation. *arXiv preprint arXiv:1905.02450* (2019).

[54] Chen Sun, Austin Myers, Carl Vondrick, Kevin Murphy, and Cordelia Schmid. 2019. Videobert: A joint model for video and language representation learning. In *Proceedings of the 17th IEEE/CVF International Conference on Computer Vision*. IEEE, 7464–7473.

[55] Lichao Sun and Lingjuan Lyu. 2020. Federated Model Distillation with Noise-Free Differential Privacy. *arXiv preprint arXiv:2009.05537* (2020).

[56] Lichao Sun, Jianwei Qian, Xun Chen, and Philip S. Yu. 2020. Ldp-fl: Practical private aggregation in federated learning with local differential privacy. *arXiv preprint arXiv:2007.15789* (2020).

[57] Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. 2019. Ernie: Enhanced representation through knowledge integration. *arXiv preprint arXiv:1904.09223* (2019).

[58] Chandra Thapa, Mahawaga Arachchige Pathum Chamikara, Seyit Camtepe, and Lichao Sun. 2020. Splitfed: When federated learning meets split learning. *arXiv preprint arXiv:2004.12088* (2020).

[59] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

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

[61] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *Proceeding of the 7th International Conference on Learning Representations*. OpenReview.net.

[62] Hongyi Wang, Kartik Sreenivasan, Shashank Rajput, Harit Vishwakarma, Saurabh Agarwal, Jy yong Sohn, Kangwook Lee, and Dimitris S. Papailiopoulos. 2020. Attack of the Tails: Yes, You Really Can Backdoor Federated Learning. In *Advances in Neural Information Processing Systems*.

[63] Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris S. Papailiopoulos, and Yasaman Khazaeni. 2020. Federated Learning with Matched Averaging. In *Proceedings of 8th International Conference on Learning Representations*. OpenReview.net.

[64] Jianyu Wang, Zachary Charles, Zheng Xu, Gauri Joshi, H. Brendan McMahan, Blaise Agüera y Arcas, Maruan Al-Shedivat, Galen Andrew, Salman Avestimehr, Katharine Daly, Deepesh Data, Suhas N. Diggavi, Hubert Eichner, Advait Gadhikar, Zachary Garrett, Antonious M. Girgis, Filip Hanzely, Andrew Hard, Chaoyang He, Samuel Horvath, Zhouyuan Huo, Alex Ingerman, Martin Jaggi, Tara Javidi, Peter Kairouz, Satyen Kale, Sai Praneeth Karimireddy, Jakub Konečný, Sanmi Koyejo, Tian Li, Luyang Liu, Mehryar Mohri, Hang Qi, Sashank J. Reddi, Peter Richtárik, Karan Singhal, Virginia Smith, Mahdi Soltanolkotabi, Weikang Song, Ananda Theertha Suresh, Sebastian U. Stich, Ameet Talwalkar, Hongyi Wang, Blake E. Woodworth, Shanshan Wu, Felix X. Yu, Honglin Yuan, Manzil Zaheer, Mi Zhang, Tong Zhang, Chunxiang Zheng, Chen Zhu, and Wennan Zhu. 2021. A field guide to federated optimization. *arXiv preprint arXiv:2107.06917* (2021).

[65] Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2018. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. ACL, 1112–1122.

[66] Xiaohang Xu, Hao Peng, Md Zakirul Alam Bhuiyan, Zhifeng Hao, Lianzhong Liu, Lichao Sun, and Lifang He. 2021. Privacy-Preserving Federated Depression Detection from Multi-Source Mobile Health Data. *IEEE Transactions on Industrial Informatics* (2021).

[67] Dong Yang, Ziyue Xu, Wenqi Li, Andriy Myronenko, Holger R. Roth, Stephanie Harmon, Sheng Xu, Baris Turkbey, Evrim Turkbey, Xiaosong Wang, Wentao Zhu, Gianpaolo Carrafiello, Francesca Patella, Maurizio Cariati, Hirofumi Obinata, Hitoshi Mori, Kaku Tamura, Peng An, Bradford J. Wood, and Daguang Xu. 2021. Federated semi-supervised learning for COVID region segmentation in chest CT using multi-national data from China, Italy, Japan. *Medical image analysis* 70 (2021), 101992.

[68] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. 2019. Federated Machine Learning: Concept and Applications. *ACM Transactions on Intelligent Systems and Technology (TIST)* 10, 2 (2019).
[69] Dezhong Yao, Wanning Pan, Yutong Dai, Yao Wan, Xiaofeng Ding, Hai Jin, Zheng Xu, and Lichao Sun. 2021. Local-Global Knowledge Distillation in Heterogeneous Federated Learning with Non-IID Data. *arXiv preprint arXiv:2107.00051* (2021).
[70] Dezhong Yao, Wanning Pan, Yao Wan, Hai Jin, and Lichao Sun. 2021. FedHM: Efficient Federated Learning for Heterogeneous Models via Low-rank Factorization. *arXiv preprint arXiv:2111.14655* (2021).
[71] Ke ZHANG, Carl Yang, Xiaoxiao Li, Lichao Sun, and Siu Ming Yiu. 2021. Subgraph Federated Learning with Missing Neighbor Generation. In *Advances in Neural Information Processing Systems*.