




Article

Efficient Federated Learning with Pre-Trained Large Language Model Using Several Adapter Mechanisms

Gyunyeop Kim , Joon Yoo *  and Sangwoo Kang * 

School of Computing, Gachon University, 1342, Seongnam-daero, Sujeong-gu, Seongnam-si 13120, Republic of Korea; gyop0817@gachon.ac.kr

* Correspondence: joon.yoo@gachon.ac.kr (J.Y.); swkang@gachon.ac.kr (S.K.)

Abstract: Recent advancements in deep learning have led to various challenges, one of which is the issue of data privacy in training data. To address this issue, federated learning, a technique that merges models trained by clients on servers, has emerged as an attractive solution. However, federated learning faces challenges related to data heterogeneity and system heterogeneity. Recent observations suggest that incorporating pre-trained models into federated learning can mitigate some of these challenges. Nonetheless, the main drawback of pre-trained models lies in their typically large model size, leading to excessive data transmission when clients send these models to the server. Additionally, federated learning involves multiple global steps, which means transmitting a large language model to multiple clients results in too much data exchange. In this paper, we propose a novel approach to address this challenge using adapters. Adapters demonstrate training efficiency by training a small capacity adapter layer alongside a large language model. This unique characteristic reduces the volume of data transmission, offering a practical solution to the problem. The evaluation results demonstrate that the proposed method achieves a reduction in training time of approximately 20–40% and a transmission speed improvement of over 98% compared to previous approaches.

Keywords: federated learning; deep learning; transfer learning; adapter transformer

MSC: 68T50



Citation: Kim, G.; Yoo, J.; Kang, S. Efficient Federated Learning with Pre-Trained Large Language Model Using Several Adapter Mechanisms. *Mathematics* **2023**, *11*, 4479. <https://doi.org/10.3390/math11214479>

Academic Editors: Florin Leon, Mircea Hulea and Marius Gavrilescu

Received: 27 September 2023
Revised: 22 October 2023
Accepted: 26 October 2023
Published: 29 October 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Deep learning has emerged as a powerful and evolutionary technology, improving the quality of life across various fields. The demand for a proficient understanding of deep learning models has surged, driven by the availability of vast datasets. However, this pursuit has raised concerns about data privacy during the training process. One attractive method to mitigate these costs is federated learning [1]. In federated learning, each client trains its own model and shares only the trained model's parameters with the server, without actually sending raw data. Since the server only receives the model parameters, it cannot access the clients' data directly. Therefore, federated learning enables training that ensures the privacy of client data. While federated learning mitigates privacy risks, it is not without its drawbacks. In comparison to conventional learning methods, it often experiences performance degradation due to client heterogeneity. And data transfer costs are required for model aggregation. In this paper, we propose a novel methodology aimed at ameliorating the transmission challenges, with a specific focus on mitigating large data transmission needs. Our approach involves the use of adapters, which serve to enhance the efficiency of data transmission, thus addressing a key limitation of federated learning.

In general, federated learning, which combines the results from multiple clients, often exhibits a lower performance compared to the traditional centralized learning methods. This performance degradation can be attributed to the heterogeneity of the systems and data in federated learning. Since clients train models using their own devices (systems)

and individual data, variations in device performance can lead to differences in learning speed. In extreme cases, due to some clients' performance limitations, they cannot even participate in the learning process. Furthermore, in federated learning each client trains models using its own diverse data, where the quantity and quality of data held by each client may differ. For example, in tasks that require various labels, certain clients may lack data for some labels, causing problems.

Many methodologies have been proposed to solve the heterogeneity of federated learning. Many methodologies have improved the aggregation method and training method of federated learning. There were methods for resolving heterogeneity based on model weight, such as FedProx [2] and FedDyn [3]. Feature-based heterogeneity solutions, such as FedUFO [4] and MOON [5], have been proposed. Some tried to solve the heterogeneity problem by improving the global model performance, and the APFL algorithm [6] was proposed through this. Also, incorporating pre-trained models into federated learning has proven to mitigate the performance degradation caused by data heterogeneity [7].

In [7], it was experimentally confirmed that the pre-trained model solves various problems of federated learning without using any special aggregation method. Pre-trained models are models trained on general and large-scale datasets. In the field of natural language processing, the methodology of fine-tuning these pre-trained language models for downstream tasks through transfer learning has shown state-of-the-art performance in most areas. Moreover, in federated learning pre-trained models consistently outperform non-pre-trained deep learning models [7].

In the field of natural language processing, pre-training generally utilizes large-scale language models. The recent rapid development of deep learning is closely related to the increase in model capacity. Each year, the size of the model is increasing, which leads to an increase in performance. For example, the BERT-base [8] model proposed in 2018, a large-scale language model commonly used in natural language processing, has a parameter number of about 340 M. The T5 model [9] proposed in 2019 has a size of 11 B. In addition, the GPT-3-base [10] and Megatron-Tuning [11] models proposed in 2020 have parameter numbers of up to 175 B and 530 B, respectively. In federated learning, however, sending large-scale language models can be burdensome, since the trained parameters need to be transmitted over the network. In federated learning, during each global epoch, the trained models need to be downloaded from all clients, and then the models are aggregated and uploaded. However, uploading/downloading large-scale language models during each global epoch poses challenges in terms of time and network resources.

In this paper, we propose a novel methodology to address these issues and save network transmission time in federated learning. The proposed method applies Adapters [12], which were introduced for efficient transfer learning, to federated learning. This allows federated learning to proceed with less model transmission. We conduct experiments in the areas of natural language processing and computer vision to demonstrate how the proposed methodology can significantly reduce the network transmission time compared to existing approaches.

The main contributions of our paper are three-fold: First, we identify that pre-trained models can mitigate the data heterogeneity problem in federated learning but render a new challenge of large data transmission requirements. Second, we introduce the adapter mechanism, which involves training large language models using smaller-sized adapters. This approach effectively addresses the problem of excessive data transmission issues in federated learning that uses transformer-based pre-trained models. Finally, we conduct extensive experiments on diverse federated learning datasets in both natural language processing and computer vision domains, to demonstrate the efficiency and performance of our proposal. The evaluation results highlight a significant reduction in training time of approximately 20–40% and a remarkable improvement in transmission speed, surpassing 98% compared to previous approaches.

The structure of the remainder of this paper is outlined as follows. Section 2 provides an overview of the related work in the field. Section 3 elaborates on the details and design of the proposed approach. Section 4 presents the results of the evaluation conducted. Finally, Section 6 concludes the paper.

2. Related Work

2.1. Federated Learning

Deep learning has emerged as a powerful and evolutionary technology, revolutionizing various research fields across the spectrum. By leveraging large-scale neural networks and vast amounts of data, deep learning has made significant contributions to diverse domains, ranging from healthcare and finance to computer vision and natural language processing. While it is easy to collect massive amounts of data for some tasks to effectively train deep learning models, there are often difficulties in collecting data for certain tasks due to security concerns. In the case of medical or conversational data, data privacy is crucial, requiring extensive security measures and de-identification processes for data collection.

Solving security and privacy issues incurs significant costs, thus increasing the cost of collecting privacy-preserving data. One attractive method to mitigate these costs is federated learning [1]. The process of federated training is divided into clients with private data and servers for model aggregation. The client trains the local model through private learning data. Then, the trained model parameter is transmitted to the server. The server aggregates the received model parameters and overwrites them on the global model. This method can protect privacy by not sending raw data directly to the server. Federated learning, which gained significant attention after its initial introduction in [1], was officially introduced in a 2017 Google AI Blog [13] and has been successfully applied in technologies such as the Mobile G Keyboard. Federated learning is a methodology in deep learning that enables data decentralization by utilizing multiple local clients and a central server to train a global model. In this approach, each local client possesses its own data and trains each local model, which is then transmitted to the central server and aggregated by the central server to form the global model.

However, in federated learning some problems occurred instead of protecting data privacy. One of them is performance degradation due to model aggregation. In the process of aggregating the model, the performance was degraded by various factors, such as the data heterogeneity and system heterogeneity of each client. Until recently, various aggregation methodologies and training methodologies have been proposed as a way to solve this problem. In general, the most basic aggregation method is FedAvg [1], which averages and aggregates the value of each local model. Since then, FedProx [2] and FedNova [14], which are aggregation methodologies, have been proposed to solve data heterogeneity. FedProx adds near-field terms to the local objective function to limit local updates to be closer to the global model. FedNova uses momentum to accurately weight local models when updating global models. FedDyn [3] modifies the local goal with a dynamic normalizer consisting of linear terms based on primary conditions and Euclidean distance terms so that the local minimum matches the global minimum. FedUFO [4] shares client models with each other to sort features and log outputs. In addition, some tried to solve the heterogeneity problem by improving the global model performance, and, through this, the APFL algorithm [6] was proposed. There have been attempts to solve heterogeneity in various ways. Since then, [7] has tried to solve the problem using a pre-trained model. According to [7], it was experimentally confirmed that various heterogeneity problems were solved despite the use of FedAvg when using the pre-trained model in federated learning.

2.2. Adapter

Recently, transfer learning-based methodologies have shown state-of-the-art performances in natural language processing. Transfer learning approaches pre-train large-scale language models on readily available and commonly collected massive datasets, such

as the wiki dataset. In turn, these pre-trained language models are fine-tuned for downstream tasks.

Most pre-trained language models require large model capacity. BERT, a representative pre-trained language model with transfer learning, has a significantly higher capacity than LSTM; newer models, like GPT3 [10] and T5 [9], require even larger capacities than BERT. While these large-scale language models exhibit high performances, they demand considerable resources and time for training.

To address these challenges, Dosovitskiy et al. [12] proposed the fine-tuning methodology using adapters. Instead of training all the parameters of a pre-trained language model, they demonstrated that training only the proposed adapter layers for downstream tasks can achieve a comparable performance. Many pre-trained language models form stacked transformer blocks, where self-attention and feed-forward networks are interconnected. In AdapterFusion [15], adapter layers with lower capacities were inserted between each transformer block of the pre-trained model. Furthermore, it was shown that training only these adapter layers can yield a similar performance to previous methods. Subsequently, structurally enhanced adapters, such as the Houlsby adapter [12] and LoRA [16], were proposed to achieve a high performance with a smaller adapter capacity.

3. Methodology

This paper proposes to use adapters in federated learning, to improve transmission efficiency during the process of federated learning with large transformer-based pre-trained language models. Ref. [7] has shown that using a large language model can solve various problems with federated learning. Various problems caused by heterogeneity were alleviated and the performance of the global model was improved. However, federated learning is trained using model transmission of servers and clients. At this time, using a large model causes network overload from transmission. The proposed method of this paper uses adapters to solve the transmission problem. The reason is that the adapter can train the large language model with fewer parameters. Experiments on NLP and CV are conducted to confirm the efficiency and performance of the methodology. They also measure the amount of reduction in transmission.

One of the biggest issues of federated learning is performance degradation due to data heterogeneity and system heterogeneity. The study [7] showed that the pre-trained large language model could improve the problem. However, pre-trained large language models require a high model capacity. When a large-capacity deep learning model is used in federated learning, a very large amount of transmission is required. Federated learning requires the parameters of clients to be uploaded/downloaded at each global step, so using a large capacity model increases the amount of transmission exponentially.

For example, popular transformer models in natural language processing and computer vision, such as BERT-base [8] and ViT-base [17], have sizes of 440 MB and 330 MB, respectively. In federated learning, it is necessary to transmit and receive the trained model parameters to and from each client at every global epoch. Therefore, transmitting and downloading large model parameters multiple times becomes problematic in federated learning. For instance, if 10 clients perform federated learning for 30 global epochs using BERT-base, it would result in a total transmission of approximately 264,000 MB, or 263 GB.

$$T = 2E_g N_c C \quad (1)$$

The amount of transmission in federated learning is calculated as shown in Equation (1). Federated learning should upload/download to all clients at every global step. Therefore, the total transmission amount T of federated learning is calculated by multiplying the global epoch E_g , the number of clients N_c , and the capacity C of the model transmission size and then doubling the result (upload and download).

Hence, we propose using adapters to reduce the size of the model parameters that need to be transmitted.

The overall structure of the proposed methodology is illustrated in Figure 1, which consists of three main steps. The first step is the preparation and downloading of the pre-trained model. In the first step, the pre-trained model is downloaded so that each client has the same model structure and parameter value before starting the first global step of federated learning. The second step is the client model training and upload. In the second step, clients train the local model. After that, each client uploads the trained adapter and classification head to the server. The final step is the aggregation and download of the models. In the final step, the learned models are merged. The learned model parameter value transmitted in the second step is aggregated to the global model. Each client then downloads the aggregated global model to start the next global step. Each step is described in Sections 3.2–3.4.

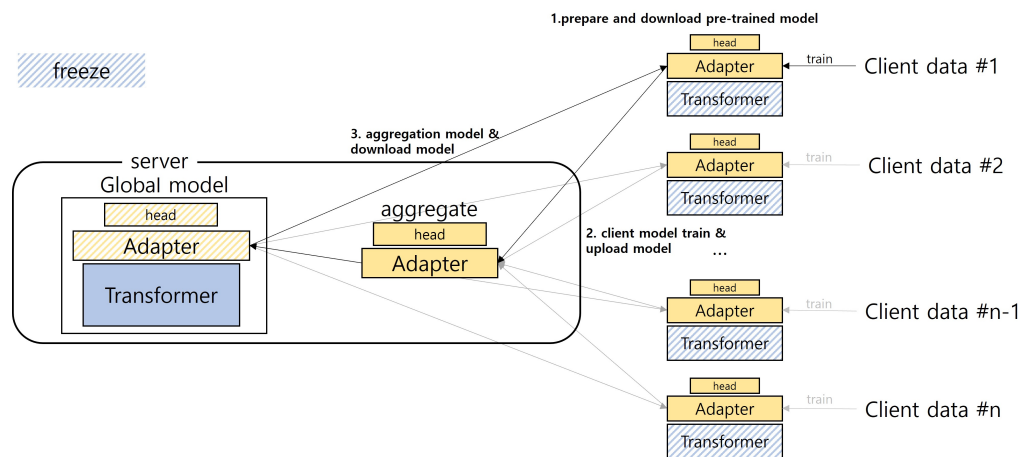


Figure 1. Overall architecture of federated learning with adapter.

3.1. Pre-Trained Model with Adapter

In this section, we discuss the deep learning model used in this paper. Firstly, the paper employs a pre-trained large language model. A pre-trained language model refers to a model that has been trained on a large-scale dataset and is typically divided into pre-trained transformer layers and embedding layers. The pre-trained large language model is fine-tuned to fit the downstream task. The classification head is added to classify the label for the purpose of each downstream task. The classification head is typically implemented as a one-layer feed-forward network. The classification head derives the probability for each label in the downstream task. For example, next word prediction predicts the probability that each word in every word will be used as the next word. Image classification predicts probabilities for all candidate categories.

Ref. [12] confirms that learning with adapter layers can achieve a similar performance. In this paper, we reduce the amount of training parameters by using adapter layers. The adapter mechanism trains only the adapter layer and classification head while freezing the pre-trained language model. As a result, the adapter mechanism can achieve a similar performance even with a small amount of training resources. The adapter layer, trained using adapter models, such as LoRA [16], Houslby [12], and Preffier [15], is an additional layer used to train the language model. In other words, the model to be used for federated learning in this paper consists of an embedding layer, transformer layer, adapter layer, and classification head. And according to [12], the model training process freezes the embedding and transformer layers.

3.2. Prepare Pre-Trained Language Model

In this section, we provide detailed explanations on the first steps shown in Figure 1. The first step is the preparation of the model for client training. In our proposed federated learning approach, each client performs downstream task training using a pre-trained large language model with adapters. Federated learning use the global model and the local

model. The global model is a deep learning model owned by the server, and the local model is a deep learning model owned by each client. Federated learning is when each client trains the local model through their dataset and aggregates it to the global model at the server. Therefore, each local model and global model have the same model structure. You must also have the same parameter value at the beginning of the training of the local model.

In summary, in the proposed methodology all clients receive the full model (pre-trained large language model) from the server before starting federated learning. Note that this is a one-time download. This ensures that all clients begin with the same pre-trained parameters for federated learning.

3.3. Train Local Model

The next step involves clients training the local model and uploading it to the server. Each client fine-tunes the local model for the downstream task. After the local learning epoch, each client uploads the model parameter value to the server. However, using the pre-trained large language model in conventional federated learning requires a large amount of transmission capacity to be uploaded to the server, because federated learning involves servers and clients transmitting the full model. Typically, pre-trained large language models have hundreds of MB or several GB of capacity, which requires too much transmission. In addition, the server is required to receive trained model parameter values from all clients, which creates a network bottleneck.

In this paper, learning is conducted using adapters to reduce transmission. For learning with adapters, such as that explained in Section 3.1, the pre-trained model is frozen. There is no change in the model parameter values of the transformer layer and embedding layer because only the adapter layer and classification head are learned. Only the adapter layer and the model parameter value of the classification head change. In this step, clients train the local model as much as the local epoch through each of their datasets. As a result of clients learning their respective datasets from the local model, only the adapter layer and classification head are learned. Therefore, clients send only the adapter layer and classification head to the server. Compared to the transformer layer and embedding layer, the capacities of the adapter layer and the classification layer are very small, which can increase transmission efficiency.

3.4. Aggregation into Global Model

Lastly, the server aggregates the trained parameters, and the clients download them. The final step aggregates the adapter layer uploaded by clients in the second step and the model parameter value of the classification head layer. The aggregated model parameter value is overwritten on the global model. The above process completes the global model learning in one global step. In this process, the federated learning aggregation method uses FedAvg [1]. FedAvg is the most basic method of averaging model parameter values for each local model. At this point, the server aggregates only adapter layers and classification heads, because the transformer layer and embedding layer did not change the values at client's training step.

After one global step is completed, all clients must synchronize their model parameter value before starting learning for the next global step. Therefore, all clients download the global model learned on the server. The downloaded parts are the adapter layer and classification head, because the transformer layer and embedding layer did not change the values at aggregation step. Therefore, each client downloads only the model parameter values of the adapter layer and the classification head layer, which can increase transmission efficiency.

The proposed methodology solves the increased network transmission problem when using pre-trained large language models in federated learning. Instead of training the transformer layers and embedding layers, which takes most of the model capacity in the pre-trained language model, the proposed approach trains and transmits only the smaller-

sized classification head and adapter layers. In result, the proposed method reduces the network transmission and the number of parameters to be trained and potentially decreases the overall training time.

4. Experiments

In this section, we conduct experiments on two types of datasets, namely, natural language processing and computer vision, to demonstrate the efficiency and performance of the proposed methodology.

4.1. Datasets and Downstream Task

For the experiments in natural language processing, the federated Stack Overflow dataset [18] and federated Shakespeare dataset [18], which are federated learning datasets for natural language processing, are used for training. The federated Stack Overflow dataset consists of question posts and their corresponding answers uploaded on Stack Overflow. The Shakespeare dataset contains various phrases from Shakespeare's literary works. We use the titles and contents of the posts in these datasets as input and then measure the performance of the next word prediction task [7]. Next, word prediction predicts the $n + 1$ th word when up to n words are entered. The accuracy of the predicted word is measured for performance evaluation of the next word prediction task.

For the experiments in computer vision, the EMNIST dataset [19] and CIFAR100 dataset [20], which are computer vision federated learning datasets, are used for training. EMNIST is a dataset of handwritten characters and digits, while CIFAR100 is an image classification dataset with 100 classes. The performance of image classification is measured using these datasets. All datasets were downloaded using the TensorFlow Federated API [18].

4.2. Experiment Setup

We detail the experiment setup in this section. We conduct federated learning using large language models. For natural language processing, we use a pre-trained language model in the form of a transformer decoder for the next word prediction task. We utilize the gpt2-base [21] pre-trained language model, which has a parameter size of approximately 490 MB.

For computer vision image classification, we use a large encoder-based language model called ViT [17]. We use ViT-base [17] for training, which has a parameter size of approximately 330 MB. Additionally, we perform the experiment using three different types of adapters. For the experiment, we use the Pfeiffer adapter [15], Houshy adapter [12], and LoRA [16].

The baseline uses the methodology proposed by [7]. Ref. [7] confirmed that using the pre-trained language model to conduct federated learning showed a high performance. The baseline uses a large language model as a global model and a local model for federated learning. Before starting federated learning, all clients download the model parameter value of the pre-trained large language model. After that, in each global step, clients pull-train the large language model and upload/download the large language model.

We also employ the experiments from [1] for comparison with traditional federated learning. The deep learning model for CV performance measurement is a 3-layer CNN (5×5 kernel size) with ReLU activation and max pooling (2×2 kernel size). And the classification layer is a 2-layer feed-forward network. The first layer has 1000 dimensions. In addition, the embedded layer and the 2-layer LSTM or RNN in 768 dimensions are used as models for measuring the performance of NLP, and the 1-layer feed-forward network for the classification head is used at the end.

The federated learning setup for this experiment is as follows. We conduct federated learning on both IID and non-IID datasets. For non-IID training, we use a total of 9 clients datasets for 9 clients. For IID training, we convert a total of 27 client datasets into 9 clients by grouping them in sets of 3. The experiments in this paper were conducted in a local

environment; thus, the real transmission rate was not measured. The experiments were performed on a total of 4 RTX3090 GPUs, with 3 GPUs used for parallel training on 3 clients. The server independently uses 1 GPU for aggregating client model parameters and conducting tests for performance evaluation.

The hyperparameters used in the experiment are shown in Table 1. Since each experiment was conducted with different models and environments, we experimented with various learning rates and considered the highest performance achieved by each model as its performance. The learning rates used in the experiment are 0.005, 0.001, 0.0005, and 0.0001, and the differences between each learning rate are mentioned in Section 4.3.2.

Table 1. Hyperparameters used in the experiments.

Hyperparameter	Value
Global epoch (NLP)	30
Global epoch (CV)	50
Local step	5
Number of clients	9
Optimizer	Adam
Epsilon	0.0005
Batch size (NLP)	16
Batch size (CV)	12
Pre-trained model (NLP)	gpt2-base
Pre-trained model (CV)	ViT-base
GPU	RTX 3090 × 4

4.3. Experimental Results

4.3.1. Accuracy

We examine the accuracy of the next word prediction to verify the performance of the pre-trained model. The results of the Stack Overflow dataset are shown in Table 2. We conduct next word prediction using the gpt2 model, employ federated learning for training, and compare the performance with and without adapters. Traditional federated learning methods using an RNN and LSTM without using the pre-trained large language model showed performance of about 13.26 and 14.66. In contrast, when the pre-trained large language model was used the performance improvement was more than 10 compared to traditional combined learning. When adapters were not used, an accuracy of 26.97 was observed. When adapters were used, the performances with the Pfeiffer, LoRA, and Housby adapters were 26.69, 25.87, and 26.58, respectively. In summary, not using adapters results in a slight accuracy improvement compared to using adapters, but the performance is comparable.

Table 2. Next word prediction accuracy of Stack Overflow non-IID dataset.

Method	Language Model	lr	Pre-Trained	Adapter	Accuracy
McMahan et al. [1]	RNN	0.005	X	X	13.26
	LSTM	0.005	X	X	14.66
Nguyen et al. [7]	gpt2-base	0.0005	O	X	26.97
Proposed	gpt2-base	0.0001	O	Pfeiffer adapter	26.69
	gpt2-base	0.0001	O	LoRA adapter	25.87
	gpt2-base	0.0001	O	Housby adapter	26.58

We next examine the performance of image classification in computer vision (CV), and the results on CIFAR100 are presented in Table 3. In this experiment, ViT and adapters were used. Traditional federated learning methods using a CNN without using the pre-trained large language model showed performance of about 19.5. The Cifar-100 dataset is difficult to solve with a small model, and when using the baseline ViT without adapters

an accuracy of 61.51 was achieved. When federated learning was performed with adapters, the performances were 64.09, 60.91, and 64.19, respectively. The Pfeiffer and Houlsby adapters showed better performances than the baseline, while the LoRA adapter had a similar performance to the baseline. The accuracy graphs from using the pre-trained large language model for each epoch are shown in Figures 2 and 3.

Table 3. Image classification accuracy of CIFAR100 non-IID dataset.

Method	Language Model	lr	Pre-Trained	Adapter	Accuracy
McMahan et al. [1]	CNN	0.001	X	X	19.5
Nguyen et al. [7]	ViT-base	0.0005	O	X	61.51
Proposed	ViT-base	0.005	O	Pfeiffer adapter	64.09
	ViT-base	0.005	O	LoRA adapter	60.91
	ViT-base	0.005	O	Houlsby adapter	64.19

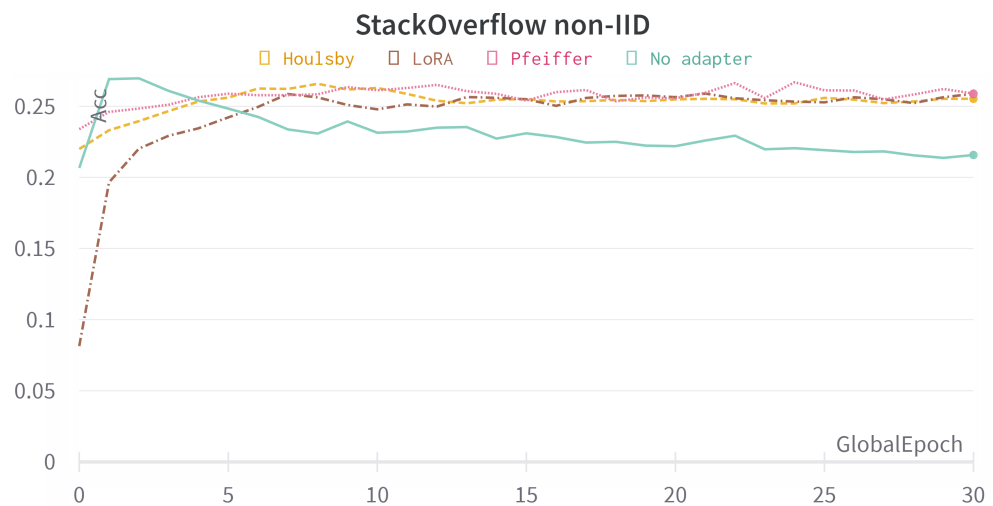


Figure 2. Test accuracy graph of Stack Overflow IID dataset using GPT2-base.

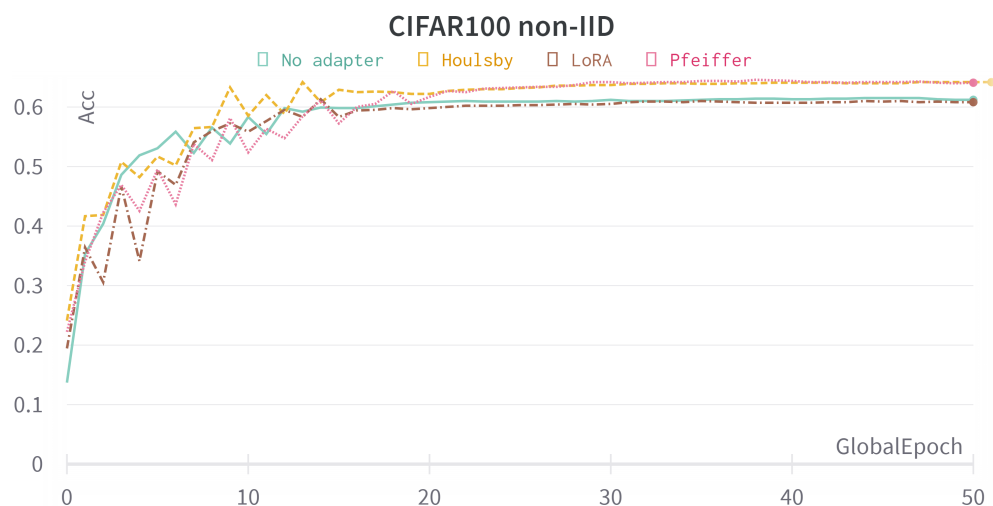


Figure 3. Test accuracy graph of CIFAR100 IID dataset using ViT-base.

To ensure the reliability of the experimental results, additional experiments were conducted on the Shakespeare dataset in NLP and the EMNIST dataset in CV. The results and accuracy graphs for these experiments are presented in Tables 4 and 5 and Figures 4 and 5. Overall, these experiments showed similar trends to the results in Tables 2 and 3 and Figures 2 and 3.

Table 4. Next word prediction accuracy of Shakespeare non-IID dataset.

Method	Language Model	lr	Pre-Trained	Adapter	Accuracy
McMahan et al. [1]	RNN	0.005	X	X	16.49
	LSTM	0.001	X	X	17.48
Nguyen et al. [7]	gpt2-base	0.0001	O	X	28.46
Proposed	gpt2-base	0.0005	O	Pfeiffer adapter	29.18
	gpt2-base	0.0005	O	LoRA adapter	27.75
	gpt2-base	0.0005	O	Houlsby adapter	28.29

Table 5. Image classification accuracy of EMNSIT non-IID dataset.

Method	Language Model	lr	Pre-Trained	Adapter	Accuracy
McMahan et al. [1]	CNN	0.005	X	X	94.64
Nguyen et al. [7]	ViT-base	0.0005	O	X	99.17
Proposed	ViT-base	0.005	O	Pfeiffer adapter	100
	ViT-base	0.001	O	LoRA adapter	95.83
	ViT-base	0.005	O	Houlsby adapter	100

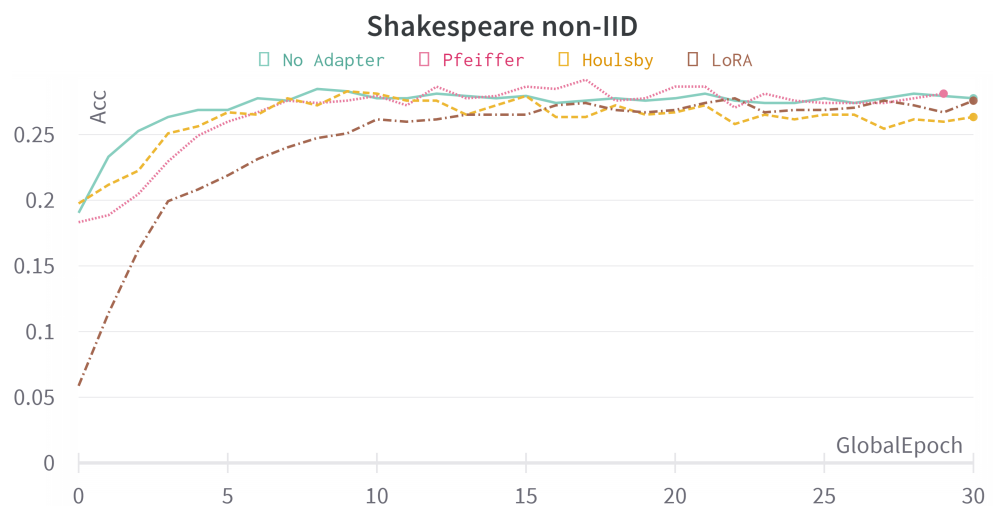


Figure 4. Test accuracy graph of Shakespeare non-IID dataset.

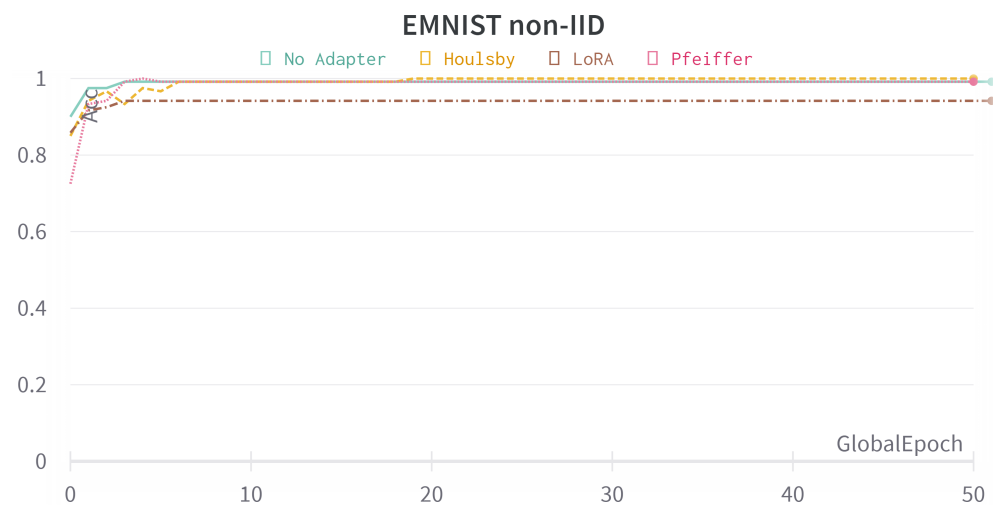


Figure 5. Test accuracy graph of EMNIST non-IID dataset.

The results of experiments on IID datasets are shown in Tables 6 and 7, which present the results assuming that each client has datasets from three individuals. Overall, the IID datasets showed better performances compared to the non-IID datasets in Tables 2 and 3. In the Stack Overflow dataset, when using the pre-trained large language model the performance improvement was more than 10 compared to traditional federated learning using an RNN and LSTM. And the highest performance was achieved with the Pfeiffer adapter, with an accuracy of 28.71. Similarly, in the CIFAR100 dataset the highest performance was achieved with the Pfeiffer adapter, with an accuracy of 79.0. Also, when the model was not pre-trained it showed a low performance. In summary, we observe that using adapters generally improves the performance in IID datasets. Even in cases where the performance degraded, it still showed a comparable performance to the baseline. The accuracy graphs from using a pre-trained large language model for each epoch are shown in Figures 6 and 7.

Table 6. Next word prediction accuracy of Stack Overflow IID dataset.

Method	Language Model	lr	Pre-Trained	Adapter	Accuracy
McMahan et al. [1]	RNN	0.001	X	X	14.94
	LSTM	0.005	X	X	16.96
Nguyen et al. [7]	gpt2-base	0.001	O	X	28.57
Proposed	gpt2-base	0.0001	O	Pfeiffer adapter	28.71
	gpt2-base	0.0005	O	LoRA adapter	28.16
	gpt2-base	0.0001	O	Houlsby adapter	28.04

Table 7. Image classification accuracy of CIFAR100 IID dataset.

Method	Language Model	lr	Pre-Trained	Adapter	Accuracy
McMahan et al. [1]	CNN	0.0001	X	X	24.9
Nguyen et al. [7]	ViT-base	0.0005	O	X	74.8
Proposed	ViT-base	0.001	O	Pfeiffer adapter	79.0
	ViT-base	0.001	O	LoRA adapter	76.3
	ViT-base	0.005	O	Houlsby adapter	76.9

We conduct additional experiments on the Shakespeare dataset and EMNIST dataset, to further investigate non-IID datasets. The results of these experiments are presented in Tables 8 and 9 and Figures 8 and 9. Overall, these experiments showed similar trends to the results in Tables 6 and 7 and Figures 6 and 7.

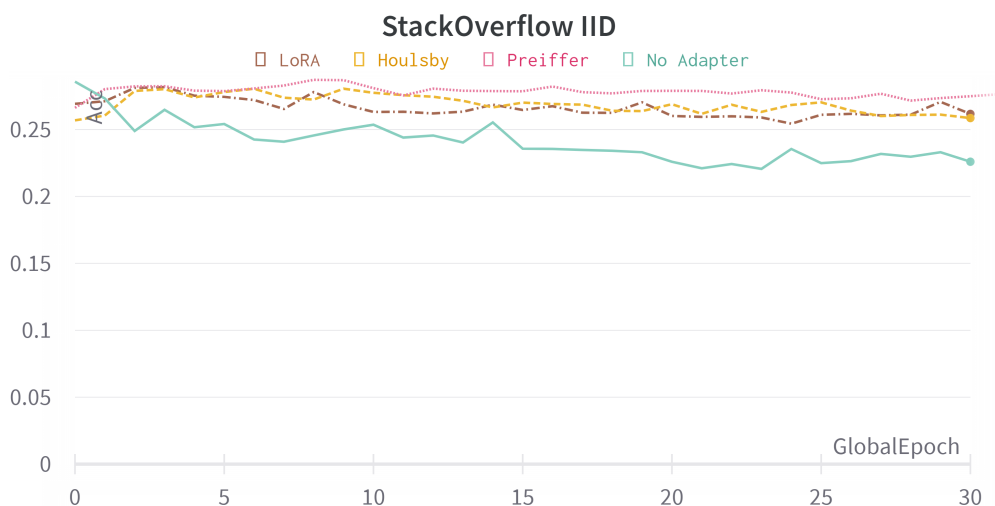


Figure 6. Test accuracy graph of Stack Overflow IID dataset.

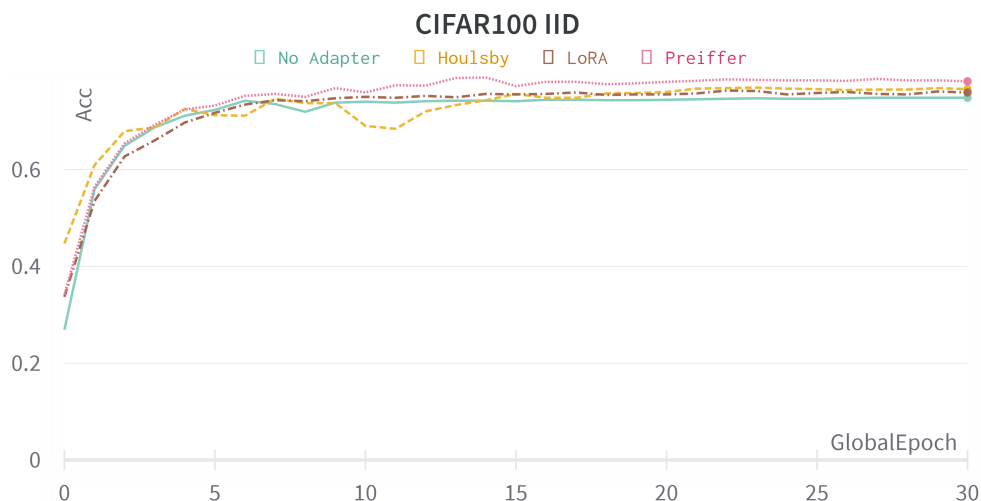


Figure 7. Test accuracy graph of CIFAR100 IID dataset.

Table 8. Next word prediction accuracy of Shakespeare IID dataset.

Method	Language Model	lr	Pre-Trained	Adapter	Accuracy
McMahan et al. [1]	RNN	0.005	X	X	19.83
	LSTM	0.005	X	X	19.79
Nguyen et al. [7]	gpt2-base	0.0001	O	X	31.10
Proposed	gpt2-base	0.0005	O	Pfeiffer adapter	30.6
	gpt2-base	0.001	O	LoRA adapter	30.73
	gpt2-base	0.0005	O	Houlsby adapter	31.01

Table 9. Image classification accuracy of EMNIST IID dataset.

Method	Language Model	lr	Pre-Trained	Adapter	Accuracy
McMahan et al. [1]	CNN	0.0005	X	X	98.21
Nguyen et al. [7]	ViT-base	0.0001	O	X	99.10
Proposed	ViT-base	0.005	O	Pfeiffer adapter	100
	ViT-base	0.005	O	LoRA adapter	99.10
	ViT-base	0.005	O	Houlsby adapter	100

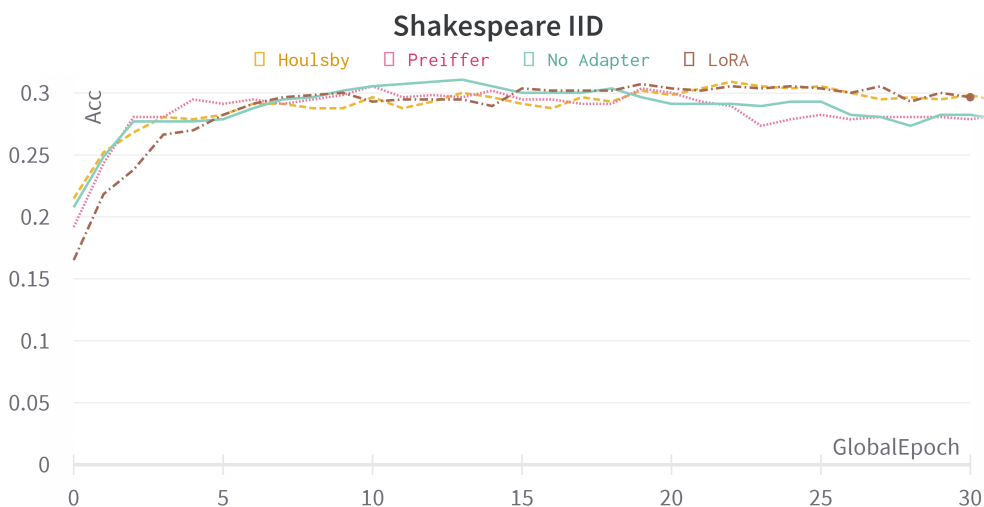


Figure 8. Test accuracy graph of Shakespeare IID dataset.

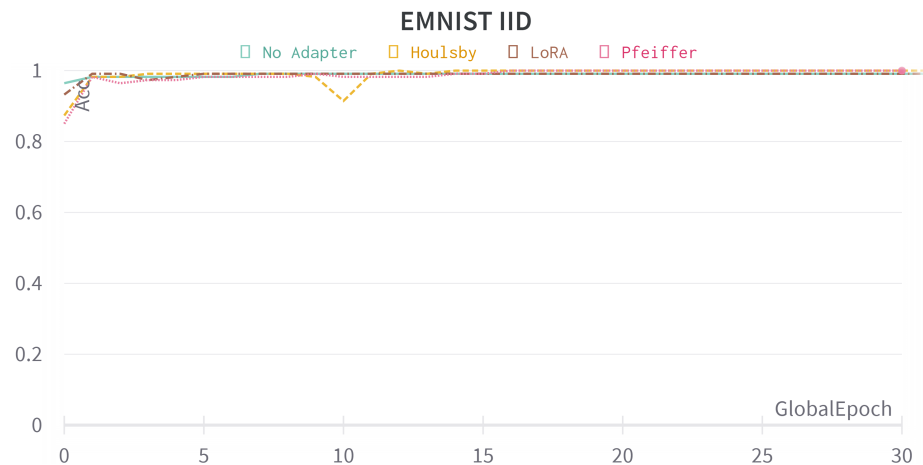


Figure 9. Test accuracy graph of EMNIST IID dataset.

4.3.2. Learning Rate

In this section, we conduct experiments with various learning rates. Figures 10 and 11 present the performance tables of the Stack Overflow dataset and CIFAR100 non-IID experiments, respectively, based on different learning rates. In Stack Overflow, which is an NLP dataset, there was not a significant change in performance according to the learning rate. However, in the CV dataset, CIFAR100, using adapters generally showed better performance at higher learning rates, while in smaller datasets, it showed a better performance at lower learning rates. This trend was observed to some extent in the Shakespeare dataset, but in EMNIST, where most adapters achieved an accuracy of 100, no significant differences were observed based on the learning rate. The corresponding performance and accuracy graphs of these experiments are provided in Figures 12 and 13.

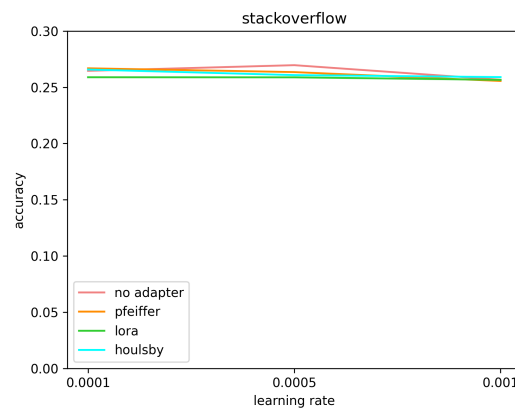


Figure 10. Test accuracy graph of each learning rate in Stack Overflow non-IID dataset.

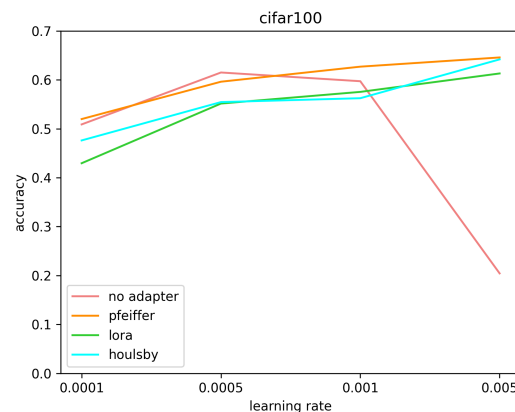


Figure 11. Test accuracy graph of each learning rate in CIFAR100 non-IID dataset.

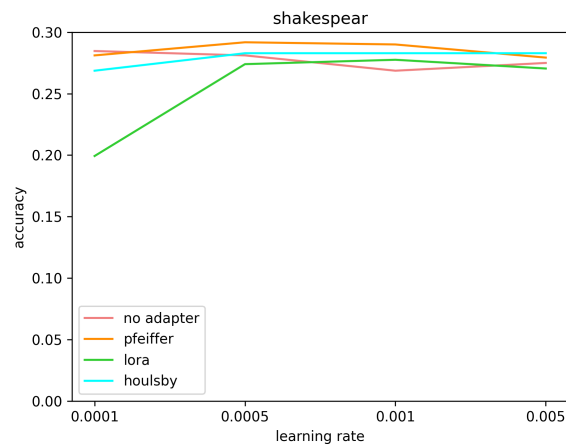


Figure 12. Test accuracy graph of each learning rate in Shakespeare non-IID dataset.

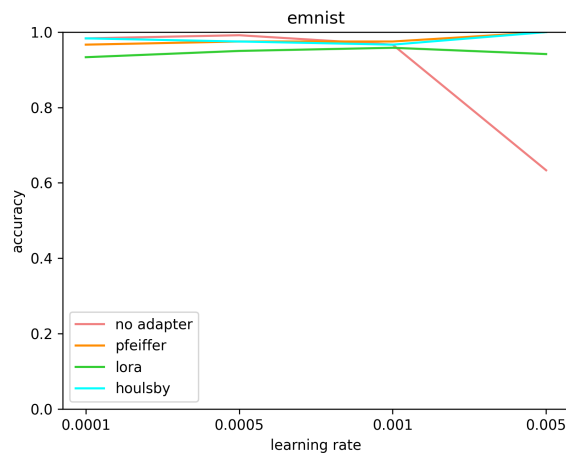


Figure 13. Test accuracy graph of each learning rate in EMNIST non-IID dataset.

4.4. Time Efficiency

4.4.1. Training Time

When using adapters in training a large language model, optimization is achieved with fewer parameters. This means there are fewer weights to compute gradients and to update. Therefore, fine-tuning only the adapters consumes less time compared to fine-tuning the entire model. This experiment measures the training time in federated learning, excluding the transmission time. This experiment shows the reduced local model training time due to the use of adapters. Note that in our experiments we measure the training time without considering the transmission time, since the experiments were conducted on a local machine. This allowed us to measure the efficiency in terms of pure training time.

The results are presented in Tables 10 and 11. In this experiment, only the case of using the same language model was compared because the structure and size of the model greatly affect the training speed. Table 10 measures the time taken for federated learning on the Stack Overflow dataset, for 30 epochs at the server. The results showed that using adapters allowed for a faster training time. For the Stack Overflow dataset, using the adapter methodology resulted in an approximately 20% reduction in training time compared to the baseline. The training speed results for the Shakespeare dataset are shown in Table 11. In this dataset, it was found that using adapters could save up to 40% of the training time.

Table 10. Training time in local environment with Stack Overflow non-IID dataset.

Method	Language Model	Pre-Trained	Adapter	Training Time	Time Compared to Baseline
Nguyen et al. [7]	gpt2-base	O	X	2 h 10 m 59 s	1
Proposed	gpt2-base	O	Pfeiffer adapter	1 h 38 m 43 s	0.75
	gpt2-base	O	LoRA adapter	1 h 45 m 28 s	0.80
	gpt2-base	O	Houlsby adapter	1 h 43 m 54 s	0.79

Table 11. Training time in local environment with Shakespeare non-IID dataset.

Method	Language Model	Pre-Trained	Adapter	Training Time	Time Compared to Baseline
Nguyen et al. [7]	gpt2-base	O	X	12 m 20 s	1
Proposed	gpt2-base	O	Pfeiffer adapter	7 m 11 s	0.58
	gpt2-base	O	LoRA adapter	7 m 43 s	0.65
	gpt2-base	O	Houlsby adapter	7 m 42 s	0.65

4.4.2. Transmission Time

Although we did not conduct experiments to measure the actual transmission time, the transmission efficiency can be computed based on the model's size. Table 12 presents the transmission sizes of the pre-trained models and adapters used in this paper.

Table 12. Transmission size for each model.

Model Name	Size
RNN	219.34 MB
LSTM	241.87 MB
Nguyen et al. [7] (gpt2-base)	487.82 MB
gpt2+Pfeiffer adapter	3.41 MB
gpt2+LoRA adapter	1.12 MB
gpt2+Houlsby adapter	6.82 MB
CNN	18.36 MB
Nguyen et al. [7] (ViT-base)	330.96 MB
ViT+Pfeiffer adapter	3.41 MB
ViT+LoRA adapter	1.12 MB
ViT+Houlsby adapter	6.82 MB

Firstly, gpt2-base, which is the NLP pre-trained model, had a large size of 487 MB, while each of the three adapters had sizes of 3.41 MB, 1.12 MB, and 6.82MB, respectively, which are significantly smaller. In the federated learning methodology used in this paper, after the initial download of the pre-trained large language model, only the adapters need to be transmitted at each global epoch, resulting in an efficient transmission time. In addition, the model sizes of the RNN and LSTM are 219 MB and 241 MB, respectively. Because the embedding layer of the NLP shows a very large model size, the proposed methodology in NLP shows better time efficiency than traditional federated learning that uses RNNs and LSTM. A similar trend is shown in the CV model ViT-base. Compared to the full model size of the large language model, the model size of the adapter is very small. Since the CNN model does not include an embedding layer, the model size is significantly small at 18 MB.

Based on the above Table 12, we measure the efficiency of transmission. Equation (1) shows the calculation of the amount of transmission in the federated learning methodology. In an experiment using gpt2-base, about 262,980 MB of transmission is required if federated learning is conducted using the Table 1 environment. However, when using an adapter, the transmission size decreases, as shown in Table 12. However, the transmission size is reduced when the proposed methodology is used. The transmission amount of the proposed methodology is calculated using the following Equation (2).

$$T = C_p N_c + (2E_g - 1) N_c C \quad (2)$$

C_p is the model capacity of the large pre-trained loan model. E_g is the global peer, and N_c is the number of clients. C is the size of the model transmission. As in Section 3.2, C_p and N_c are multiplied to calculate the amount of transmission that clients initially use to download the pre-trained model. Then, it is multiplied by the number of up/downloads for the global epoch, excluding the initial download by the transfer model capacity C . If the LoRA adapter is used, only about 4977 MB of transmission is required. If the server could perform 10 MBps of upload/download speed, it would take approximately 7 h and 18 min for gpt2-base transmission but only approximately 8 min and 17 s for LoRA adapter transmission. In addition, the amount of transmission in traditional federated learning using LSTM calculated through (1) is 130,610 MB, or 128 GB. If the server could perform 10 MBps of upload/download speed, it would take approximately 3 h and 37 min. Using the proposed method in NLP can show better time efficiency than traditional federated learning using LSTM.

We can expect the same time efficiency for the CV pre-trained model. In the conventional federated learning methodology, if ViT-base is used for federated learning, then a total of 297,000 MB of transmission would be required. However, when conducting federated learning using the LoRA adapter, only approximately 3967 MB of transmission would be needed. This means that with a capability of 10 MBps of upload/download speed the conventional federated learning methodology would require 8 h and 15 min of transmission time, while using adapters would only require approximately 6 min and 36 s, enabling efficient federated learning. In addition, transmission in traditional federated learning using CNN is 16,524 MB, or 16 GB, which takes approximately 27 min and 32 s. Therefore, the proposed methodology can save time than traditional federated learning.

5. Discussion and Limitations

The evaluation of the proposed method in this paper revolves around two primary issues. Firstly, it addresses the question of whether the use of the adapter mechanism can effectively decrease both the data transmission and learning time. Secondly, it investigates whether the reduction in training time and data transmission does not lead to performance degradation. These aspects are rigorously examined through experiments conducted using the federated learning datasets of both computer vision and natural language processing.

This paper validates the reduction in training time and data transmission detailed in Section 4.4.2. Adapters significantly reduce the size of the model to be transferred by up to 98%. We mathematically calculated the decrease in transmission when the model size is reduced, which causes the reduction in transmission time during the training time. As a result, the reduction in transmission time shows is about 98%. Furthermore, the use of adapters reduces the training time of the local model by minimizing the number of layers that need training. In this paper, we experimentally check the reduction in training time when the transmission time is excluded in the local environment through Section 4.4.1. This shows that the reduction in training time, excluding the transmission time, can be about 20%. This confirms that the proposed methodology may show a reduction in training time and transmission time. In addition, Section 4.3.1 shows that performance degradation does not occur despite the reduced training time. Experimental results of NLP generally show a slight performance degradation. Experiments with CV generally show performance improvements. We confirm from the experimental results

that the proposed method represents a reduction in training time, and this does not lead to significant performance degradation.

In this paper, experiments on CV tasks were not described, which prevented us from confirming the reduction in training time for these applications. Despite our efforts in conducting the experiments, we were unable to observe a decrease in the training time, regardless of whether the adapter was used or not. This limitation arose due to the speed discrepancy on the CPU-based image pre-processing and training speed using the GPU. Unfortunately, our computational resources did not permit us to bridge the gap effectively. In addition, we note that the experiments were conducted solely in a local environment. We did not perform transmission speed experiments on an actual network due to these limitations. Instead, we calculated the reduction in transmission speed based on our experiments in the local settings. Addressing these constraints in future studies will provide a more comprehensive understanding of the proposed methodology's applicability and effectiveness across diverse scenarios.

Furthermore, note that our methodology operates exclusively during the fine-tuning process and is not applicable in the pre-training phase. This limitation arises from the need to individually train the pre-trained large language model. Consequently, even though pre-training demands the most extensive dataset, the proposed method cannot be employed during this crucial phase. Additionally, while this paper successfully reduces the training time, there is a slight performance degradation observed in the NLP tasks. For future work, there is a need for research focused on reducing both the training time and data transmission during the full training of large language models in federated learning. Simultaneously, efforts should be directed towards eliminating performance degradation in NLP tasks through advancements in the adapter mechanism research.

6. Conclusions

In this paper, we addressed the problem of the increased transmission time caused by the pre-trained large language model in federated learning. To overcome this issue, we proposed and experimented with a federated learning approach using adapters, which previously have been suggested as an efficient fine-tuning method. As a result, the transmission time was reduced by about 98% compared to the methodology using the pre-trained large language model without adapters. In addition, the training time was also reduced by 20–40% as the number of parameters to be learned decreased. Nevertheless, the predictive performance was similar. Through this, it was confirmed that time-efficient federated learning is possible without performance degradation when an adapter is used in federated learning using a large language model, such as [7]. Also, the proposed methodology showed lower transmission sizes than traditional federated learning without a large language model. In addition, because the proposed methodology uses a large language model, it showed a higher predictive performance than traditional federated learning. Through this, it was confirmed that the proposed methodology can induce performance improvements with the same or lower transmission amount as traditional federated learning.

The significance of our proposed method lies in its ability to improve the transmission efficiency of federated learning. Therefore, it enables the use of a large language model, such as ChatGPT, powered by the GPT-3 model, in real-world federated learning environments. While large language models, such as ChatGPT, have shown impressive performances, it is practically challenging to use GPT-3 in an actual federated learning environment. This is mainly due to the substantial time and transmission costs incurred by clients with limited computational resources when learning and transmitting the large GPT-3 model. In contrast, the proposed methodology offers an attractive solution to significantly reduce the transmission costs. Furthermore, our experiments showed that the training time was partially mitigated. In summary, our proposal stands as a key enabler, facilitating the use of large models in a real-world federated learning environment.

Author Contributions: Conceptualization, G.K. and S.K.; methodology, G.K.; software, G.K.; validation, G.K.; investigation, G.K. and S.K.; resources, G.K. and S.K.; data curation, G.K.; writing—original draft preparation, G.K. and J.Y.; writing—review and editing, J.Y.; visualization, J.Y.; supervision, S.K. and J.Y.; project administration, S.K.; funding acquisition, S.K. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (2022R1A2C1005316 and 2021R1F1A1063640) and in part by the Gachon University research fund of 2023 (GCU-202300660001).

Data Availability Statement: Publicly available datasets were analyzed in this study. This data can be found here: Tensorflow Federated api [18] (<https://www.tensorflow.org/federated>).

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

LSTM	Long Short-Term Memory
NLP	Natural Language Processing
CV	Computer Vision
IID	Independent Identically Distributed
Non-IID	Non-Independent Identically Distributed

References

1. McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; Arcas, B.A.y. Communication-Efficient Learning of Deep Networks from Decentralized Data. In Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, Fort Lauderdale, FL, USA, 20–22 April 2017; Volume 54, pp. 1273–1282.
2. Li, T.; Sahu, A.K.; Zaheer, M.; Sanjabi, M.; Talwalkar, A.; Smith, V. Federated Optimization in Heterogeneous Networks. *Proc. Mach. Learn. Syst.* **2020**, *2*, 429–450.
3. Acar, D.A.E.; Zhao, Y.; Matas, R.; Mattina, M.; Whatmough, P.; Saligrama, V. Federated Learning Based on Dynamic Regularization. In Proceedings of the International Conference on Learning Representations, Vienna, Austria, 4 May 2021.
4. Zhang, L.; Luo, Y.; Bai, Y.; Du, B.; Duan, L.Y. Federated Learning for Non-IID Data via Unified Feature Learning and Optimization Objective Alignment. In Proceedings of the 2021 IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, BC, Canada, 11 October 2021; pp. 4400–4408. [[CrossRef](#)]
5. Li, Q.; He, B.; Song, D. Model-Contrastive Federated Learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021; pp. 10713–10722.
6. Deng, Y.; Kamani, M.M.; Mahdavi, M. Adaptive Personalized Federated Learning. *arXiv* **2020**, arXiv:2003.13461.
7. Nguyen, J.; Wang, J.; Malik, K.; Sanjabi, M.; Rabbat, M.G. Where to Begin? On the Impact of Pre-Training and Initialization in Federated Learning. *arXiv* **2022**, arXiv:2210.08090.
8. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), Minneapolis, MN, USA, 2–7 June 2019; pp. 4171–4186. [[CrossRef](#)]
9. Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; Liu, P.J. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *J. Mach. Learn. Res.* **2020**, *21*, 1–67.
10. Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language Models are Few-Shot Learners. In *Proceedings of the Advances in Neural Information Processing Systems*; Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2020; Volume 33, pp. 1877–1901.
11. Shoeybi, M.; Patwary, M.; Puri, R.; LeGresley, P.; Casper, J.; Catanzaro, B. Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism. *arXiv* **2019**, arXiv:1909.08053.
12. Houlsby, N.; Giurgiu, A.; Jastrzebski, S.; Morrone, B.; De Laroussilhe, Q.; Gesmundo, A.; Attariyan, M.; Gelly, S. Parameter-Efficient Transfer Learning for NLP. In Proceedings of the 36th International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; Volume 97, pp. 2790–2799.
13. Federated Learning: Collaborative Machine Learning without Centralized Training Data. Available online: <https://ai.googleblog.com/2017/04/federated-learning-collaborative.html> (accessed on 4 May 2023).
14. Wang, J.; Liu, Q.; Liang, H.; Joshi, G.; Poor, H.V. Tackling the Objective Inconsistency Problem in Heterogeneous Federated Optimization. In *Proceedings of the Advances in Neural Information Processing Systems*; Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2020; Volume 33, pp. 7611–7623.

15. Pfeiffer, J.; Kamath, A.; Rücklé, A.; Cho, K.; Gurevych, I. AdapterFusion: Non-destructive task composition for transfer learning. In Proceedings of the EACL 2021–16th Conference of the European Chapter of the Association for Computational Linguistics, Proceedings of the Conference, Association for Computational Linguistics (ACL), Virtual Event, 19–23 April 2021; pp. 487–503.
16. Hu, E.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, L.; Chen, W. LoRA: Low-Rank Adaptation of Large Language Models. *arXiv* **2021**, arXiv:2106.09685.
17. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An Image is Worth 16 × 16 Words: Transformers for Image Recognition at Scale. In Proceedings of the International Conference on Learning Representations, Virtual Event, Austria, 3–7 May 2021.
18. Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G.S.; Davis, A.; Dean, J.; Devin, M.; et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. 2015. Available online: <http://tensorflow.org/> (accessed on 13 December 2022).
19. Cohen, G.; Afshar, S.; Tapson, J.; van Schaik, A. EMNIST: Extending MNIST to handwritten letters. In Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN), Anchorage, AK, USA, 14–19 May 2017; pp. 2921–2926. [[CrossRef](#)]
20. Krizhevsky, A. Learning Multiple Layers of Features from Tiny Images. Master’s Thesis, University of Toronto, Toronto, ON, Canada, 2009; pp. 32–33.
21. Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I. Language Models are Unsupervised Multitask Learners. *OpenAI Blog* **2019**, *1*, 9.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.