Communication-Efficient Adaptive Federated Learning

Xiaoxin Su, 2250271010

Abstract

Federated learning is a new distributed machine learning paradigm that enables multiple clients to collaboratively train machine learning models without exposing their raw data through the coordination of parameter servers (PS). However, federated learning still faces many challenges in practical deployment. For example, frequent communication between the PS and clients consumes enormous bandwidth resources, and the optimization algorithms based on SGD need adaptability. Current researchers have designed various algorithms, such as compressing model updates through quantization or sparsification to reduce communication costs and adaptive optimizers for federated learning, such as FedAdam, to increase training adaptability. However, these algorithms can only solve one of the problems individually. In this paper, a novel communication-efficient federated learning algorithm (FedCAMS) is proposed to address both of these challenges simultaneously and with theoretically guaranteed convergence.

The paper does not provide the source code, so I refer to the description of the experiment in the paper and use the ResNet-18 and ConvMixer-256-8 models to classify the CIFAR-10 dataset. I implemented various adaptive optimization algorithms in the same framework for a fair comparison, and the FedAMS algorithm designed in the paper achieves the optimal performance, the same as the original results in the paper. In addition, I applied different compression algorithms to the FedCAMS algorithm and proved that the algorithm could train well-performing models even when the model updates are compressed.

Keywords： Federated Learning, Compression, Adaptive Optimizer.

1 Introduction

To alleviate growing concerns over data privacy breaches, Google in 2016 devised the federated learning (FL) paradigm, which enables model training without touching raw data samples residing on decentralized clients[1]. In federated learning setting, data is distributed across many clients, which can help us train various machine learning models. However, these data cannot be shared directly with PS or other clients due to privacy concerns. A parameter server (PS) in FL assists all clients in model aggregation. In each global iteration, PS will distribute the latest model to selected clients before they update the model by doing several stochastic gradient descent (SGD) using their local dataset. The PS then collects and aggregates updated model updates (i.e., changes in model parameters). Multiple global iterations are performed by involving different clients until the model converges[2-3].

Although FL has attracted much attention due to its privacy-preserving properties, it still faces significant challenges in practice:

- Large communication overhead: PS and clients in FL are usually connected through a wireless

network, so bandwidth resources are limited. Frequent synchronization of large-scale models between PS and clients can lead to severe communication overhead.

- Lack of adaptability: Most of the existing FL algorithms use SGD optimizer for model training, but SGD is not suitable for training large-scale networks such as BERT[4], GPT-3[5], GAN[6] or ViT[7].

Researchers currently devise different algorithms to solve one of the above problems individually but cannot handle them all at once. To reduce the communication overhead, some works compress model updates using quantization[8-9] and sparsification[10-11]. In terms of adaptive optimizers, various optimization algorithms such as FedAdam, FedYogi, etc.[12] have been conceived by introducing adaptive gradient algorithms into the federated learning framework. However, the design of adaptive optimization of federated learning frameworks with compression still lacks research.

This paper uses a compressed federated adaptive gradient optimization algorithm to address the above challenges. The authors first designed an algorithm, FedAMS, a variant of FedAdam, and theoretically demonstrated the improved convergence of the algorithm compared to FedAdam. Based on FedAMS, the authors designed a federated communication-compression AMSGrad algorithm with maximized stability called FedCAMS, which achieves both efficient communication and adaptive optimization.

Since the authors did not provide the paper's source code, I needed to build the system from scratch to reproduce the algorithm. I first built a simulation framework for federated learning so that multiple clients could collaboratively train the model under saved data. After building the system, I aggregated the model updates of the clients on the PS of the system as a pseudo-gradient and performed adaptive updates according to the algorithm described in the paper. In addition, in order to ensure the fairness of training, I implemented various other adaptive optimization algorithms in the same framework for comparison. The experimental results are similar to those in the original paper, which proves the high efficiency of the designed algorithm.

2 Related works

2.1 Optimizers for training models

Stochastic gradient descent (SGD) algorithm[13] is the most commonly used optimization algorithm for training models in machine learning, which minimizes a given loss function by gradually optimizing the parameters of the model using first-order gradients over multiple iterations. Although SGD is easy to implement, it is sensitive to the parameters of the model and has a slow convergence rate. Therefore, there is related research to design various adaptive algorithms based on SGD. AdaGrad[14], and RMSProp[15] speed up training by adaptively adjusting the learning rate of each model parameter based on its value, making the learning rate different for each parameter. Adam[16] and its variant AMSGrad[17] further optimize the training process of the model by incorporating momentum based on the above optimizers.

2.2 Federated Learning

In Federated Learning (FL)[18-19], decentralized clients collaboratively train machine learning models via *model averaging, i.e.,* iteratively averaging locally trained models towards global optimum[1]. It runs Stochastic Gradient Descent (SGD) in parallel on a subset of devices and then averages the resulting model updates via a central server once in a while. The convergence rates of FedAvg have been analyzed in[2] with strongly convex loss functions and[3] with non-convex loss functions, respectively. Transmitting model updates in a large scale network incurs heavy traffic overhead. Model compression is an effective way to optimize FL communication, which is divided into two strategies: *quantization* and *sparsification*.

Sparsification algorithms select only a small number of significant model updates for transmission. The DGC algorithm[10] discards 99.9% of gradients in communication, achieving a very high compression rate. STC[11] compresses uploaded and downloaded data simultaneously through sparse and ternary methods. DC2[20] further explores the delay-traffic trade off through adaptively coupling compression control and network latency.

Quantization algorithms map model updates to a small set of discrete values. For instance, the QSGD algorithm[9] generates a random number based on each model update and maps each update to a centroid. The PQ algorithm[21] splits model updates into intervals with a number of centroids, and each model update is randomly quantified to a centroid in an unbiased manner. Wen *et al.* optimized communication by quantizing model updates to ternary values[8].

In this paper, the authors define the compression error, so that any algorithm that satisfies this definition can be applied to FedCAMS. In addition, the authors theoretically proved the convergence of adaptive optimization with compression based on this definition, filling the research gap in this area.

3 Method

In FL, there are m clients, a dataset distribution of client i as $\mathcal{D}_i$, where $\mathcal{D}_i \neq \mathcal{D}_{i'}$ for any $i \neq i'$. Its local loss function as $F_i(\mathbf{x}) = \mathbb{E}_{\xi \sim \mathcal{D}_i} F_i(\mathbf{x}, \xi)$, where $\mathbf{x}$ denotes a model of dimension d and ξ is a data sample. Thus, the goal of FL is to train a model to optimize the global loss function:

$$\min_{\mathbf{x} \in \mathbb{R}^d} f(\mathbf{x}) = \frac{1}{m} \sum_{i=1}^{m} F_i(\mathbf{x}). \tag{1}$$

For model training, a batch data sample needs to be draw to compute the unbiased stochastic gradient $\mathbf{g}_t^i = \nabla F_i(\mathbf{x}_t, \xi_i)$ at client i in global iteration t.

FedAvg is the most commonly used optimization algorithm in FL to solve the problem in (1). $\mathbf{x}_t$ is the global model for the $t-th$ iteration. PS first randomly selects a set $\mathcal{S}_t$ of n clients to participate in the training of this iteration. The selected clients download the latest model $\mathbf{x}_t$ and perform K epoches training using local data to obtain the local model $\mathbf{x}_{t,K}$. After that, client i uploads the model updates $\Delta_t^i = \mathbf{x}_{t,K} - \mathbf{x}_t$ to PS for aggregation to update the global model $\mathbf{x}_{t+1} = \mathbf{x}_t + \Delta_t$, where $\Delta_t = \frac{1}{n} \sum_{i \in \mathcal{S}_t} \Delta_t^i$.

FedAdam is an adaptive FL optimization algorithm that modifies the single-step SGD of FedAvg in global updates to a single-step adaptive gradient optimization. FedAdam obtains Δ_t after aggregating the local updates from the clients, and its rules for global updates are as follows:

$$\mathbf{m}_t = \beta_1 \mathbf{m}_{t-1} + (1-\beta_1)\Delta_t, \tag{2}$$

$$\mathbf{v}_t = \beta_2 \mathbf{v}_{t-1} + (1-\beta_2)\Delta_t^2, \tag{3}$$

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \eta \frac{\mathbf{m}_t}{\sqrt{\mathbf{v}_t} + \varepsilon}. \tag{4}$$

There are related works to design other adaptive optimization algorithms by modifying the variance term $\mathbf{v}_t$, such as FedYogi, FedAdagrad, FedAMSGrad, etc.

3.1 Federated AMSGrad with Max Stabilization

The authors proposed federated AMSGrad algorithm with max stabilization (FedAMS), which adds a max stabilization step before PS aggregation. The specific details of the FedAMS optimizer are summarized in Procedure 1.

Procedure 1 FedAMS

Input: initital point $\mathbf{x}_1$, local step size η_l, global step size η, β_1, β_2, ε

$\mathbf{m}_0 \leftarrow 0, \mathbf{v}_0 \leftarrow 0$

for *t=1 to T* **do**

  Random sample a subset $\mathcal{S}_t$ of clients

  Server sends $\mathbf{x}_t$ to the subset $\mathcal{S}_t$ of clients

  $\mathbf{x}_{t,0}^i = \mathbf{x}_t$

  for *each client* $i \in \mathcal{S}_t$ *in parallel* **do**

    for *k=0,...,K-1* **do**

      Compute local stochastic gradient: $\mathbf{g}_{t,k}^i = \nabla F_i(\mathbf{x}_{t,k}^i; \xi_{t,k}^i)$

      $\mathbf{x}_{t,k+1}^i = \mathbf{x}_{t,k}^i - \eta_l \mathbf{g}_{t,k}^i$

    end

    $\Delta_t^i = \mathbf{x}_{t,K}^i - \mathbf{x}_t$

  end

  Server aggregates local update:$\Delta_t = \frac{1}{|\mathcal{S}_t|} \sum_{i \in \mathcal{S}_t} \Delta_t^i$

  Update $\mathbf{m}_t = \beta_1 \mathbf{m}_{t-1} + (1-\beta_1)\Delta_t$

  Update $\mathbf{v}_t = \beta_2 \mathbf{v}_{t-1} + (1-\beta_2)\Delta_t^2$

  $\hat{\mathbf{v}}_t = \max(\hat{\mathbf{v}}_{t-1}, \mathbf{v}_t, \varepsilon)$, update $\mathbf{x}_{t+1} = \mathbf{x}_t + \eta \frac{\mathbf{m}_t}{\sqrt{\hat{\mathbf{v}}_t}}$ ▷ Option 1

  $\hat{\mathbf{v}}_t = \max(\hat{\mathbf{v}}_{t-1}, \mathbf{v}_t)$, update $\mathbf{x}_{t+1} = \mathbf{x}_t + \eta \frac{\mathbf{m}_t}{\sqrt{\hat{\mathbf{v}}_t} + \varepsilon}$ ▷ Option 2

end

From Procedure 1, we can see that the initial training phase of FedAMS is the same as that of FedAvg, the difference is how Δ_t is used to update the global model at the end of the $t-th$ global iteration. The aggregated data Δ_t is used as a pseudo-gradient to calculate the momentum $\mathbf{m}_t$ and variance $\mathbf{v}_t$ according to (2), (3).

FedAMS is designed to provide two ways to update the global model as follows:

$$Option\ 1: \hat{\mathbf{v}}_t = \max(\hat{\mathbf{v}}_{t-1}, \mathbf{v}_t, \varepsilon), \mathbf{x}_{t+1} = \mathbf{x}_t + \eta \frac{\mathbf{m}_t}{\sqrt{\hat{\mathbf{v}}_t}}. \tag{5}$$

$$Option\ 2: \hat{\mathbf{v}}_t = \max(\hat{\mathbf{v}}_{t-1}, \mathbf{v}_t), \mathbf{x}_{t+1} = \mathbf{x}_t + \eta \frac{\mathbf{m}_t}{\sqrt{\hat{\mathbf{v}}_t} + \varepsilon}. \tag{6}$$

In option 1, by adding the ε term to the max function, FedAMS can directly use $\mathbf{v}_t$ as the denominator to update the model. Since the unsteady behavior (small values in $\mathbf{v}_t$) in the denominator occurs in only a few dimensions, the max stabilization strategy in option 1 affects only a few dimensions. In contrast, the traditional additive strategy affects all dimensions in the variance $\mathbf{v}_t$ and thus suffers more in terms of convergence.

3.2 Federated Communication-Compressed AMSGrad

In order to optimize communication during FedAMS training, this paper designs the federated communication compression AMSGrad with max stabilization (FedCAMS). The specific details of the FedCAMS algorithm are described in Procedure 2, whose difference from FedAMS is that the transmitted data Δ_t^i is compressed, and the PS received data becomes $\hat{\Delta}_t^i$. In addition, to ensure that the trained model can converge, an error compensation mechanism is applied in the algorithm. Specifically, the data to be compressed by client i before uploading is $\Delta_t^i + \mathbf{e}_t^i$, and the error term $\mathbf{e}_{t+1}^i = \Delta_t^i + \mathbf{e}_t^i - \hat{\Delta}_t^i$ is updated correspondingly after compression.

Procedure 2 FedCAMS

Input: initital point $\mathbf{x}_1$, local step size η_l, global step size η, β_1, β_2, ε, compressor $\mathcal{C}(.)$

$\mathbf{m}_0 \leftarrow 0, \mathbf{v}_0 \leftarrow 0, \mathbf{e}_1^i = 0$

for *t=1 to T* **do**

  Random sample a subset $\mathcal{S}_t$ of clients

  Server sends $\mathbf{x}_t$ to the subset $\mathcal{S}_t$ of clients

  $\mathbf{x}_{t,0}^i = \mathbf{x}_t$

  for *each client* $i \in \mathcal{S}_t$ *in parallel* **do**

    for *k=0,...,K-1* **do**

      Compute local stochastic gradient: $\mathbf{g}_{t,k}^i = \nabla F_i(\mathbf{x}_{t,k}^i; \xi_{t,k}^i)$

      $\mathbf{x}_{t,k+1}^i = \mathbf{x}_{t,k}^i - \eta_l \mathbf{g}_{t,k}^i$

    end

    $\Delta_t^i = \mathbf{x}_{t,K}^i - \mathbf{x}_t$ Compute $\hat{\Delta}_t^i = \mathcal{C}(\Delta_t^i + \mathbf{e}_t^i)$, send $\hat{\Delta}_t^i$ to the server and update $\mathbf{e}_{t+1}^i = \Delta_t^i + \mathbf{e}_t^i - \hat{\Delta}_t^i$

  end

  for *each client* $j \notin \mathcal{S}_t$ *in parallel* **do**

    client j maintains the stale compression error $\mathbf{e}_{t+1}^j = \mathbf{e}_t^j$

  end

  Server aggregates local update:$\hat{\Delta}_t = \frac{1}{|\mathcal{S}_t|} \sum_{i \in \mathcal{S}_t} \hat{\Delta}_t^i$

  Server updates $\mathbf{x}_{t+1}$ using $\hat{\Delta}_t$ in the same way as in Procedure 1.

end

In federated learning adaptive optimizers, one needs to consider the combination of error compensation mechanisms for data compression. The convergence property of existing adaptive algorithms under nonconvex loss functions relies on the construction of the Lyapunov function. When error compensation is performed, the original structure of the Lyapunov function needs to be modified, and a new auxiliary sequence is introduced to eliminate the accumulation of compression errors. By using an error compensation mechanism in the adaptive

optimizer, a variety of biased compressors can be used, resulting in more efficient compression.

Further, this paper also incorporates partial clients' participation in the compressed adaptive optimization algorithm. Clients that do not participate in the training retain their error terms unchanged. Such a setup would make FedCAMS more relevant to realistic scenarios.

4 Convergence Analysis

In order to analyze the convergence of the algorithm, the paper made the following assumptions.

Assumption 4.1. Each loss function on the i-th client is L-smooth, i.e., $\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^d$,

$$|F_i(\mathbf{x}) - F_i(\mathbf{y}) - < \nabla F_i(\mathbf{y}), \mathbf{x} - \mathbf{y} > | \leqslant \frac{L}{2}\|\mathbf{x} - \mathbf{y}\|^2.$$

Assumption 4.2. Each loss function on the i-th client $F_i(\mathbf{x})$ has G-bounded stochastic gradient on l_2, i.e., for all ξ, we have $\|\nabla f_i(\mathbf{x}, \xi)\|^2 \leqslant G$.

Assumption 4.3. Each stochastic gradient on the i-th client has a bounded local vatiance, i.e., for all $\mathbf{x}, i \in [m]$, we have $\mathbb{E}[\|\nabla f_i(\mathbf{x}, \xi) - \nabla F_i(\mathbf{x})\|^2] \leqslant \sigma_l^2$, and the loss function on each client has a global variance bound, $\frac{1}{m}\sum_{i=1}^m \|\nabla F_i(\mathbf{x} - \nabla f(\mathbf{x}))\|^2 \leqslant \sigma_g^2$.

4.1 Convergence Analysis for FedAMS

In the paper, FedAMS is firstly analyzed. In each round of global iteration, PS will randomly select n clients to participate in training. The probability of being sampled to participate in model update are $\mathbb{P}\{i \in \mathcal{S}_t\} = \frac{n}{m}$ and $\mathbb{P}\{i, j \in \mathcal{S}_t\} = \frac{n(n-1)}{m(m-1)}$.

Theorem 4.4. *Under Assumptions 4.1-4.3, if the local learning rate η_l satisfies the following condition: $\eta_l \leqslant \min\{\frac{1}{8KL}, \frac{n(m-1)\varepsilon}{24m(n-1)K\sqrt{\beta_2 K^2 G^2 + \varepsilon}[3\eta L + C_1^2 \eta L + 2\sqrt{2(1-\beta_2)}G]}\}$, then the iterates of FedAMS in Procedure 1 under partial participation scheme satisfy*

$$\min_{t\in[T]} \mathbb{E}[\|\nabla f(\mathbf{x}_t)\|^2] \leqslant 8\sqrt{\beta_2 \eta_l^2 K^2 G^2 + \varepsilon}[\frac{f_0 - f_*}{\eta\eta_l KT} + \frac{\Psi}{T} + \Phi], \tag{7}$$

where $\Psi = \frac{C_1 G^2 d}{\sqrt{\varepsilon}} + \frac{2C_1^2 \eta\eta_l KLG^2 d}{\varepsilon}$ *and* $\Phi = \frac{5\eta_l^2 KL^2}{\sqrt{2\varepsilon}}(\sigma_l^2 + 6K\sigma_g^2) + [(3 + C_1^2)\eta L + 2\sqrt{2(1-\beta_2)}C]\frac{\eta_l}{2n\varepsilon}\sigma_l^2 + [(3 + C_1^2)\eta L + 2\sqrt{2(1-\beta_2)}G]\frac{\eta_l(m-n)}{2n(m-1)\varepsilon}[15K^2L^2\eta^2(\sigma_l^2 + 6K\sigma_g^2) + 3K\sigma_g^2]$ *and* $C_1 = \frac{\beta_1}{1-\beta_1}$.

4.2 Convergence Analysis for FedCAMS

Firstly, the definition of compressor is introduced to facilitate subsequent analysis.

Assumption 4.5. Consider a biased operator $\mathcal{C} : \mathbb{R}^d \to \mathbb{R}^d : \forall \mathbf{x} \in \mathbb{R}^d$, there exists constant $0 \leqslant q \leqslant 1$ such that

$$\mathbb{E}[\|\mathcal{C}(\mathbf{x}) - \mathbf{x}\|] \leqslant q\|\mathbf{x}\|, \forall \mathbf{x} \in \mathbb{R}^d. \tag{8}$$

The above definition of a compressor is very common, and various biased compression algorithms satisfy this condition. The $Top - k$ algorithm is to select the k elements with the largest absolute value in the model

updates to upload. Under this compression algorithm, $q = \sqrt{1 - \frac{k}{d}}$. $Scaledsign$ is another biased algorithm. After compression, each parameter only needs to be represented by 1 bit. The formula for this compressor is $\mathcal{C}_{sign}(\mathbf{x}) = \|\mathbf{x}\|_1 * sign(\mathbf{x})/d$. The compression algorithm corresponds to $q = \sqrt{1 - \|\mathbf{x}\|_1^2/d\|\mathbf{x}\|^2}$. In the subsequent experimental stage, I applied both algorithms to the FedCAMS framework to verify the effectiveness of the compression algorithm.

After defining the compressor, the convergence of FedCAMS is as follows.

Theorem 4.6. *Under Assumptions 4.1-4.4, if the local learning rate η_l satisfies the following condition: $\eta_l \leqslant \min\{\frac{1}{8KL}, \frac{\varepsilon}{KC_{\beta,q}[3\eta L + 2C_2\eta L + 2\sqrt{2(1-\beta_2)}G]}\}$, where $C_{\beta,q} = \sqrt{4\beta_2(1+q^2)^3(1-q^2)^{-2}K^2G^2 + \varepsilon}$, then the iterates of FedCAMS in Procedure 2 satisfy*

$$\min_{t\in[T]} \mathbb{E}[\|\nabla f(\mathbf{x}_t)\|^2] \leqslant 4\sqrt{\beta_2 \frac{4(1+q^2)^3}{(1-q^2)^2}\eta_l^2 K^2 G^2 + \varepsilon}[\frac{f_0 - f_*}{\eta\eta_l KT} + \frac{\Psi}{T} + \Phi], \tag{9}$$

where $\Psi = \frac{C_1 G^2 d}{\sqrt{\varepsilon}} + \frac{2C_1^2\eta\eta_l KLG^2 d}{\varepsilon}$ *and* $\Phi = \frac{5\eta_l^2 KL^2}{\sqrt{2\varepsilon}}(\sigma_l^2 + 6K\sigma_g^2) + [(3+2C_2)\eta L + 2\sqrt{2(1-\beta_2)}G]\frac{\eta_l}{2m\varepsilon}\sigma_l^2$, $C_1 = \frac{\beta_1}{1-\beta_1} + \frac{2q}{1-q^2}$ *and* $C_2 = \frac{\beta_1^2}{(1-\beta_1)^2} + \frac{4q^2}{(1-q^2)^2}$.

Both C_1 and C_2 in the converged results are affected by the compression constant q. A larger q represents stronger compression, and the convergence is slower due to more information loss.

5 Experiments

Since the paper does not provide source code, the entire framework was built from scratch by myself. I first built the basic framework of federated learning so that multiple clients could cooperate in training the model without uploading their raw data. After building the framework, I referred to the experimental process in the paper and implemented various optimization algorithms in the same framework for comparison. Afterward, I applied the compression algorithm mentioned in the paper to FedCAMS to prove its effectiveness in realizing communication-efficient adaptive federated learning.

5.1 Experimental Setup

When I reproduced the code, I used the CIFAR-10 dataset for testing. The data contains ten labels, and each label contains 6000 pictures. I use 50,000 images as the training set and the remaining images as the test set to test the performance of the model. During the experiment, I used two different models for training: (1) ResNet-18 is a commonly used convolutional neural network; (2) ConvMixer model uses patch embedding to preserve locality.

In the system, I assume that there are 100 clients, and in each round of training, ten clients will be randomly selected to participate in the training. The selected clients will perform three epochs locally, and the batch size used is 20. For hyperparameters such as learning rate, I refer to the values provided in the paper.

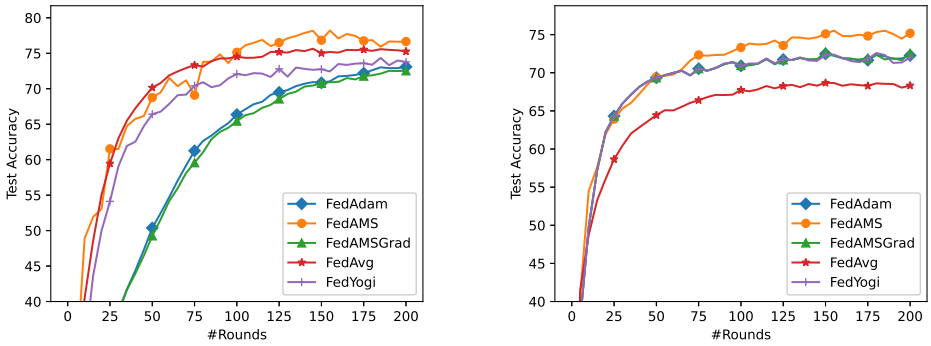
I denote Option 1 in the FedAMS framework as FedAMS and Option 2 as FedAMSGrad in my experiments.

In addition, several state-of-the-art adaptive federated learning optimization algorithm is used as a baseline for comparison, including FedAdam and FedYogi[12]. I also implemented the most standard federated learning algorithm FedAvg for comparison.

5.2 Experimental Results under IID Data Distribution

I first built an experimental environment where the data distribution is IID. In this scenario, each client contains 500 samples randomly sampled from the entire training set. Therefore, the data distribution of each client is consistent, and there are samples of 10 labels. This scene is also the environment realized in the original paper.

The comparison of FedAMS with other FL optimization algorithms is shown in Fig. 1. The results include the ResNet-18 and ConvMixer-256-8 models. The x-axis represents the number of global iterations, and the y-axis represents the accuracy of the model. On the ResNet-18 model, FedAMS performs best in terms of final test accuracy. On the other hand, FedAMSGrad and FedAdam get very similar results in test accuracy. On the ConvMixer-256-8 model, we observe that all adaptive federated optimization methods (FedAdam, FedYogi, FedAMSGrad, and FedAMS) achieve better performance than FedAvg in terms of test accuracy. FedAMS again achieves better results than the other baselines. These results demonstrate the effectiveness of FedAMS method under max stabilization.

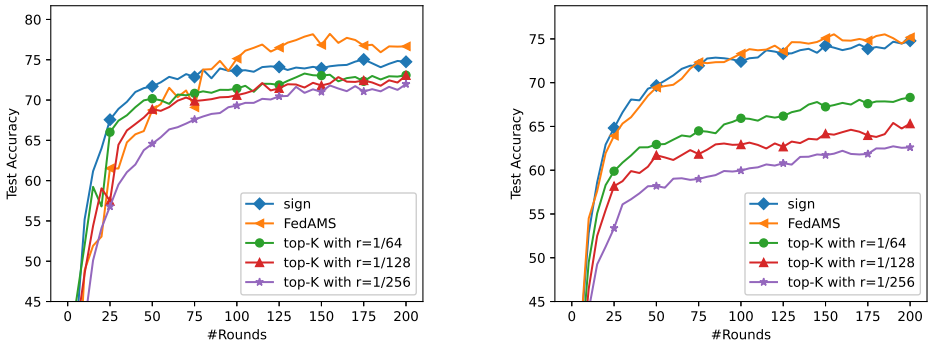


(a) ResNet-18 (b) ConvMixer-256-8

Figure 1: Comparison of accuracy of model trained by different adaptive optimization algorithms when data distribution is IID

Fig. 2 shows the convergence results of FedAMS and FedCAMS with different compression strategies when training the CIFAR-10 dataset with ResNet-18 and ConvMixer-256-8 models. The compression strategy includes scaled sign and top-k (k=1/64, 1/128, 1/256). FedAMS, without any communication compression, performs best in terms of training loss but requires high communication costs. In FedCAMS, the scaled sign compression algorithm achieves the best model performance, but its compression rate is also lower than other top-k algorithms.

It should be noted that the results I achieved in the experiment are basically the same as the experimental results in the paper. The relative accuracy between different adaptive algorithms matches the results in the paper. The only difference is that the achieved model accuracy is lower than the results in the paper. Since the authors did not provide detailed information on the division of data in their system, data enhancement, and model initialization, the accuracy of their paper cannot be achieved for the time being.

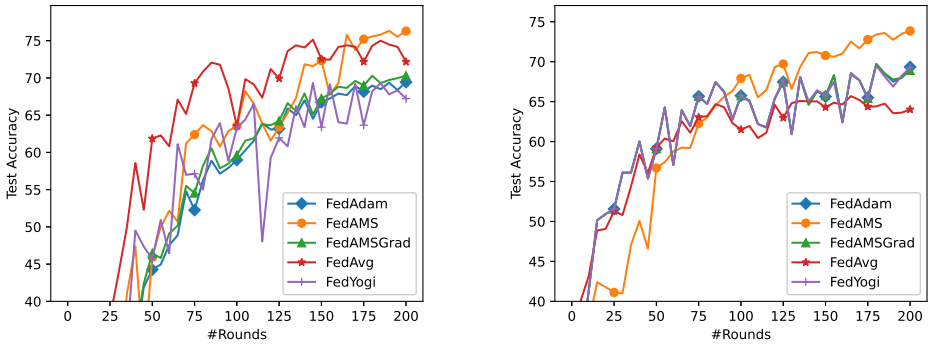


(a) ResNet-18 (b) ConvMixer-256-8

Figure 2: Comparison of accuracy of model trained by different compression when data distribution is IID

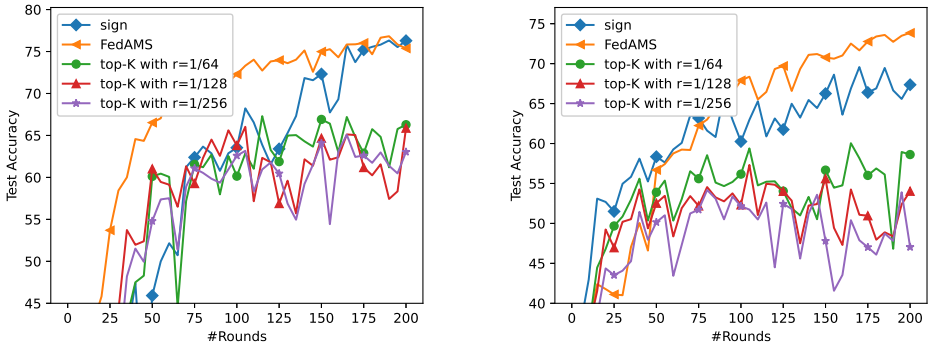
5.3 Experimental Results under non-IID Data Distribution

The scenario implemented in the paper includes only one data distribution, but the non-IID data distribution is more common in FL. Therefore, I extended the experiment in the paper and performed the same experiment when the data distribution is non-IID. In this data distribution scenario, each client only samples data from a training subset containing 5 labels, so the data distribution of different clients is different. The results of the experiment are shown in Fig. 3 and Fig. 4.



(a) ResNet-18 (b) ConvMixer-256-8

Figure 3: Comparison of accuracy of model trained by different adaptive optimization algorithms when data distribution is non-IID



(a) ResNet-18 (b) ConvMixer-256-8

Figure 4: Comparison of accuracy of model trained by different compression when data distribution is non-IID

By observing the experimental results, we found that the training trend of the models is roughly the same as the data distribution under IID. The training curve is more volatile due to the heterogeneity of data among clients.

Therefore, the extended experimental results prove that the designed FedAMS and FedCAMS algorithms can also achieve good performance in heterogeneous data distribution.

6 Conclusion and future work

Huge communication traffic and a lack of adaptive optimization algorithms are two difficulties in the practical application of federated learning. In this paper, an adaptive optimization algorithm with max stabilization was designed and theoretically proved the convergence. Furthermore, FedCAMS combines compression algorithms, partial client participation, and adaptive algorithms for efficient communication.

I implemented the training framework of federated learning and mastered how to use different optimization algorithms for model training in this framework. In addition, I also learned to use compression algorithms in the training process of FL to reduce the transmitted data traffic to improve the efficiency of its training. In follow-up research, I will further study the impact of compression rate on model convergence. I will theoretically explore how to set a reasonable compression rate to trade off communication efficiency and model accuracy and verify it experimentally.

References

[1] MCMAHAN B, MOORE E, RAMAGE D, et al. Communication-efficient learning of deep networks from decentralized data[C]//Artificial Intelligence and Statistics (AISTATS). 2017: 1273-1282.

[2] LI X, HUANG K, YANG W, et al. On the Convergence of FedAvg on Non-IID Data[C]//International Conference on Learning Representations (ICLR). 2020: 1-26.

[3] YANG H, FANG M, LIU J. Achieving Linear Speedup with Partial Worker Participation in Non-IID Federated Learning[C]//International Conference on Learning Representations (ICLR). 2021: 1-23.

[4] DEVLIN J, CHANG M W, LEE K, et al. Bert: Pre-training of deep bidirectional transformers for language understanding[J]. arXiv preprint arXiv:1810.04805, 2018.

[5] BROWN T, MANN B, RYDER N, et al. Language models are few-shot learners[C]//Advances in neural information processing systems (NIPS): vol. 33. 2020: 1877-1901.

[6] GOODFELLOW I, POUGET-ABADIE J, MIRZA M, et al. Generative adversarial networks[J]. Communications of the ACM, 2020, 63(11): 139-144.

[7] DOSOVITSKIY A, BEYER L, KOLESNIKOV A, et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale[C]//International Conference on Learning Representations (ICLR). 2020: 1-21.

[8] WEN W, XU C, YAN F, et al. TernGrad: Ternary Gradients to Reduce Communication in Distributed Deep Learning[C]//Annual Conference on Neural Information Processing Systems (NIPS). 2017: 1509-1519.

[9] ALISTARH D, GRUBIC D, LI J, et al. QSGD: Communication-efficient SGD via gradient quantization and encoding[C]//Advances in Neural Information Processing Systems (NIPS). 2017: 1709-1720.

[10] LIN Y, HAN S, MAO H, et al. Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training[C]//International Conference on Learning Representations (ICLR). 2018: 1-14.

[11] SATTLER F, WIEDEMANN S, MÜLLER K R, et al. Robust and Communication-Efficient Federated Learning From Non-i.i.d. Data[J]. IEEE Transactions on Neural Networks and Learning Systems (TNNLS), 2020, 31(9): 3400-3413.

[12] REDDI S J, CHARLES Z, ZAHEER M, et al. Adaptive Federated Optimization[C]//International Conference on Learning Representations (ICLR). 2021: 1-38.

[13] ROBBINS H, MONRO S. A stochastic approximation method[J]. The annals of mathematical statistics, 1951: 400-407.

[14] DUCHI J, HAZAN E, SINGER Y. Adaptive subgradient methods for online learning and stochastic optimization.[J]. Journal of machine learning research, 2011, 12(7).

[15] TIELEMAN T, HINTON G, et al. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude[J]. COURSERA: Neural networks for machine learning, 2012, 4(2): 26-31.

[16] KINGMA D P, BA J. Adam: A method for stochastic optimization[J]. arXiv preprint arXiv:1412.6980, 2014.

[17] REDDI S J, KALE S, KUMAR S. On the convergence of adam and beyond[J]. arXiv preprint arXiv:1904.09237, 2019.

[18] LIM W Y B, LUONG N C, HOANG D T, et al. Federated learning in mobile edge networks: A comprehensive survey[J]. IEEE Communications Surveys & Tutorials (COMST), 2020, 22(3): 2031-2063.

[19] YANG Q, LIU Y, CHEN T, et al. Federated machine learning: Concept and applications[J]. ACM Transactions on Intelligent Systems and Technology (TIST), 2019, 10(2): 1-19.

[20] ABDELMONIEM A M, CANINI M. DC2: Delay-aware Compression Control for Distributed Machine Learning[C]//International Conference on Computer Communications (INFOCOM). 2021: 1-10.

[21] SURESH A T, FELIX X Y, KUMAR S, et al. Distributed mean estimation with limited communication [C]//International Conference on Machine Learning (ICML). 2017: 3329-3337.