STRUCTURE AGNOSTIC FEDERATED LEARNING BY ADAPTABLE GRADIENT DESCENT

Anonymous authors
Paper under double-blind review

ABSTRACT

In deep learning, optimization algorithms or optimizers are often designed by hands, such as SGD, Adam, and AdaGrad. The goal of Meta-Learning is improving learning efficiency, while the key to fast learning is predicting the gradient accurately and quickly. Therefore, it is feasible for us to cast the design of an optimization algorithm as a learning problem instead of designing by hands. To be more specific, we can train a neural network called "meta-learner" to predict the gradient using previous learning tasks. Inspired by previous works, we apply LSTM to implement an optimization algorithm. In our work, we also integrated Meta-learning and Federated Learning to resolve medical data privacy issue —- because optimizer is designed as a neural network and share the same model structure, it can also be applied to federated learning which normally aggregates models with the same accessible structure. Therefore, we transfer and aggregate optimizers rather than optimizees, enabling us to train LSTM optimizer which could improve optimizee models with agnostic structures in different experiments. Since the parameters of each silo's optimizee are not leaked out, the privacy protection will be better.

1 INTRODUCTION

In Meta learning, there are two methods to improve the learning efficiency. One is finding better initialization, which means the model can learn initialization parameters by MAML(Finn et al., 2017) or Reptile(Nichol et al., 2018), while another is looking for better gradient descent.
Tasks in machine learning can be expressed as the problem of optimizing an objective function $f(\theta)$. And the goal is to find the final parameters of the neural network —- θ^* to minimize the objective function through some gradient descent algorithm. While traditional hand-designed optimizers can exploit the specific structure in their problems of interest, they are applied at the expense of potentially poor performance on problems outside of that scope. Such fact suggests that the gradient descent mechanism itself should also be treated as a machine learning optimization objective.
On the other hand, the availability of massive and representative datasets motivated the success of deep learning. However, collection of medical data distributed in different hospital databases is often prohibited by privacy concerns. When it comes to making the model more generalizable and error-prone, we need data from more than only a single health-care provider. In order to solve this problem, it's feasible to use federated learning during training. Many researchers(Liu et al., 2019)(Cui et al., 2019)(Liu & Miller, 2020)(Shao et al., 2020) have come up with great methods to deal with this kind of issue. For example, H. Brendan McMahan et al.(McMahan et al., 2017) proposed *FederatedAveraging* algorithm, which can train a classification or clustering model with training data distributed over a large number of clients. In this paper, considering that the optimizer is also designed as a neural network and share the same model structure, we did federated learning on optimizers rather than optimizees, enabling us to train LSTM optimizer which could train neural network models with different model structures and data. Since the parameters of each silo's optimizee are not leaked out, the privacy protection will be better. We did two experiments for every dataset we used. Experimental results show that LSTM optimizer can be applied to optimizee with different structures, and federated learning can indeed effectively integrate information.

2 METHOD

2.1 LOCAL META LEARNING

Before doing federated learning, we need to do local meta learning in every silo to get the parameters of every trained local LSTM optimizer and we can ignore the federated learning strategy in this part. In order to reach this goal, we just reproduce the LSTM optimizer in paper "Learning to Learn by Gradient Descent by Gradient Descent" (Andrychowicz et al., 2016). In the following part, we will discuss the details of designing LSTM optimizer.
In our work, we propose to replace hand-designed update rules with a learned update rule g, which is decided by a LSTM network. And its own set of parameters is Φ. During gradient descent, the optimizer is provided with performance of the neural network and proposes updates to increase the neural network's performance. This results in updates to optimizee f of the form:

$$\theta_{t+1} = \theta_t + g_t(\bigtriangledown f(\theta_t), \Phi)$$

When parameterizing the optimizer, we will write the final optimizee parameters $\theta^*(f, \Phi)$ as a function of the optimizer parameters Φ and the function in question. And the expected loss can be written as:

$$L(\Phi) = E_f[f(\theta^*(f, \Phi))]$$

Because the update rule is decided by a LSTM network m, we take the update step g_t to be the output of m, parameterized by Φ, whose state is denoted with h_t. For training the optimizer, it will also be convenient to make the objective depend on the entire trajectory of optimization, for some horizon T,

$$\begin{aligned} L(\Phi) &= E_f[\sum_{t=1}^{T} w_t f(\theta_t)] \\ \theta_{t+1} &= \theta t + gt, \\ \begin{bmatrix} g_t \\ h_{t+1} \end{bmatrix} &= m(\bigtriangledown_t, h_t, \Phi) \\ \bigtriangledown_t &= \bigtriangledown_\theta f(\theta_t) \end{aligned}$$

We can minimize the value of $L(\Phi)$ using gradient descent on Φ. The gradient estimate $\delta L(\Phi)/\delta\Phi$ can be computed by sampling a random function f and applying backpropagation to computing the gradient of the optimizer. When computing the optimizer's gradient, we make the assumption that the gradients of the optimizee do not depend on the optimizer parameters, i.e. $\delta \bigtriangledown_t /\delta\Phi = 0$. This assumption allows us to avoid computing second derivatives of f.

2.2 FEDERATED META LEARNING

In the above section, we have already discussed how to do local meta learning in every silo.
In this section, we will discuss in details how we combine meta learning and federated learning(Andrychowicz et al., 2016) together.
We will divide the training set into several silos and every silo corresponds to the optimizee. For each dataset, we did two experiments:
In **experiment one**, each silo's optimizee structure is set the same. And we train three different types of optimizers. Firstly, to examine the performance of federated learning process, we use the centralized dataset(the whole training set) to train a LSTM optimizer. Then we use Adam, RMSprop, and SGD as traditional optimizers to train optimizee respectively. At last, we initialize another LSTM optimzier to start federated learning. And the process of each global cycle is like this: The global LSTM opimizer is trained separately in each silo, and then the weigths of the trained LSTM optimizer are saved in a list. After obtaining the weights of LSTM under all silos, weighted average the weights to obtain global weights.Then let the global model load the global weights to obtain a new global model. This model is the optimizer to be trained in each silo in the next round of the global cycle. At the end of each round, test and use optimizee's loss, f1 as metrics to contrast the three different types of optimizers' performance.
In **experiment two**, the optimizee network structure of each silo is different, but the general training and testing process is the same. It is worth noticing that when testing at the end of each round, pay

attention to pre-training and fine-tuning the model with the training data of the silo, which will get better results.
To conclude, one complete round of federated learning is called one global cycle. At the beginning of one global cycle, we get and just need to store each LSTM meta-learner's parameters in a list. When we get the parameters of each silo's LSTM optimizer, we need to apply aggregation function to these parameters to get a new LSTM optimizer. Originally, we use federated averaging algorithm(McMahan et al., 2017) to do this job. However, such algorithm doesn't work well in the second experiment. This is because the original average algorithm cannot find the appropriate index to consider optimizees' different structures. In order to improve the performance of the LSTM optimizer, we improve the aggregation function and take into account the performance of different optimizees: After each global cycle, we will calculate the correlation by Pearson or Spearman of optimizee loss-step among all silos. After that, softmax function will be used to convert the correlation values into weights. Finally, the program will send back the global model back to each silo. We can repeat several global rounds until we get a satisfying result.
So far, we have talked about the whole process of one global cycle. We give the pseudo codes in **Algorithm 1**. And we will also talk about the Pearson correlation of loss-step in **Algorithm 2**. And these two algorithms can be found in **A.1**

3 RESULTS

We verify the performance of our proposed methods in three groups of experiments, conducting in MNIST, Fashion MNIST, and EICU. Due to space limitations, we only show the experimental results of EICU in the text. The experimental results of MNIST and FashionMNSIT are in the **appendix A.3 and A.4**. In all experiments, the trained optimizers use two-layer LSTMs with 20 hidden units in each layer. We compare our trained optimizers with standard optimizers used in Deep Learning: SGD, RMSprop and Adam. We will do two experiments for each dataset we used. The first experiment is called "same models between silos"-to ensure that the optimizees of different silos are the same, that is, unify the optimizee networks' parameters, layers set by different silos, we use *FederatedAveraging* algorithm to aggregate and transfer the optimizer network. And then we will compare the performance after every global cycle. The second experiment is called "different models between silos"-the optimizee parameters set by different silos are different, we use **Algorithm 1** to aggregate and transfer the optimizer network. And the performance will be compared similarly.

3.1 EXPERIMENTS IN EICU

In the training of this dataset, we selected 30,000 medical records and divided them into training set and test set in a 1:1 manner. Also, we process the dataset with Non-IID method, so as to better simulate the actual application scenarios. Therefore, we classify the data with the same hospital label as the same silo, and divide the training set and test set data into 5 silos. And in the actual federated learning process, we also add a pre-training technique——after all the optimizers of the 5 silo are trained and aggregated, the aggregated optimizers will be sent to the 5 silos' training set part for pre-training, and then the pre-trained model will be tested separately on the test set of 5 silos. The pre-training link not only conforms to the actual application scenario but also turns out to be very effective in the experiment. In EICU's experiments, the global federated cycle is set as 4 because further round only make little general improvement.

3.1.1 SAME MODELS BETWEEN SILOS

In experiment 1, as for the model structure of optimizee, we set layer size 20, number of layers 1. optimizee's loss-step curve will be used to evaluate the performance of different optimizers. In the EICU section, we choose the 4th cycle's testing result to compare the performance. First of all, two kinds of LSTM curves both show better performance than those of the hand-crafted optimizers. It presents that LSTM optimzier can surely promote the optimizee to converge faster to a better result. Then, the performance of LSTM-Federated is no worse than that of LSTM-Centralized, which suggests that the federated learning process aggregates the model information of each silo effectively.

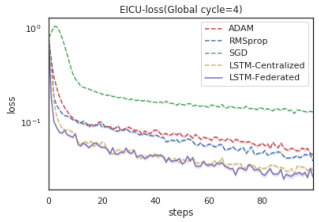
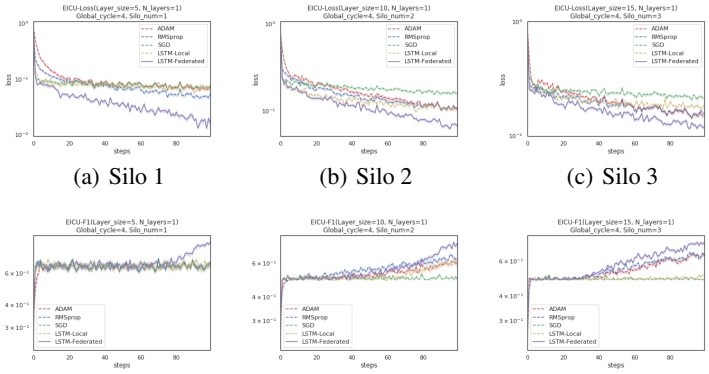


Figure 1: Comparisons among three different types of optimizers performance, i.e. the hand-crafted optimizers(ADAM, RMSprop, SGD), the optimizer trained by centralized data, the optimizer trained through global federated cycles

3.1.2 DIFFERENT MODELS BETWEEN SILOS

In experiment 2, 5 silos' optimizee model structures are respectively set as 5,10,15,20,25 and number of layers is set as 1. In this time, each silo's optimizee structure is different, and the model will be tested separately on 5 different silos. Therefore, there are 5 silos' testing results to be shown in the figure. In this experiment, we choose optimizee's 4th round loss and f1 score to evaluate different optimizers' performance. Due to space limitation, we only select the first 3 silos' experimental results. As a result, these two metrics of the 3 silos are shown respectively in the first and second line.



(a) Silo 1 (b) Silo 2 (c) Silo 3

(d) Silo 1 (e) Silo 2 (f) Silo 3

Figure 2: Comparisons among three different types of optimizers performance, i.e. the hand-crafted optimizers(ADAM, RMSprop, SGD), the optimizer trained through global federated cycles and the local optimizer trained only by 1 silo of data without federated learning process

Firstly, the convergence result of LSTM-Local is no better than those of hand-crafted optimizers, which suggests that the dispersion of dataset may hamper the performance of learned optimizer. However, LSTM-Federated turns out to be much better than the other two types of optimizers. It presents that learned optimizer can be applied well to optimizees of different structures and that federated learning process surely utilize effective aggregation algorithm.

4 CONCLUSION

In this paper, we do federated learning on optimizers rather than optimizees, enabling us to train LSTM optimizer which could optimize neural network models with different model structures and data. Since the parameters of each silo's optimizee are not leaked out, the privacy protection will be better. Experimental results show that the LSTM optimizer learns faster and better. Such fact also presents that learned optimizer can be applied to optimizee with different structures, and that federated learning can indeed effectively integrate information.

REFERENCES

Marcin Andrychowicz, Misha Denil, Sergio Gómez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 29*, pp. 3981–3989. Curran Associates, Inc., 2016. URL http://papers.nips.cc/paper/6461-learning-to-learn-by-gradient-descent-by-gradient-descent.pdf.

Jianfei Cui, He Zhu, Hao Deng, Ziwei Chen, and Dianbo Liu. Federated machine learning with anonymous random hybridization (fearh) on medical records. *arXiv preprint arXiv:2001.09751*, 2019.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1126–1135, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. URL http://proceedings.mlr.press/v70/finn17a.html.

Dianbo Liu and Tim Miller. Federated pretraining and fine tuning of bert using clinical notes from multiple silos. *arXiv preprint arXiv:2002.08562*, 2020.

Dianbo Liu, Timothy A Miller, and Kenneth D Mandl. Confederated machine learning on horizontally and vertically separated medical data for large-scale health system intelligence. *arXiv preprint arXiv:1910.02109*, 2019.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pp. 1273–1282. PMLR, 2017.

Alex Nichol, Joshua Achiam, and John Schulman. On First-Order Meta-Learning Algorithms. *arXiv e-prints*, art. arXiv:1803.02999, March 2018.

Rulin Shao, Hongyu He, Ziwei Chen, Hui Liu, and Dianbo Liu. Stochastic channel-based federated learning with neural network pruning for medical data privacy preservation: Model development and experimental validation. *JMIR Formative Research*, 4(12):e17265, 2020.

A APPENDIX

A.1 ALGORITHM

In **Algorithm 1**, the value of global cycle T is actually chosen through practice. For different dataset, we choose slightly different numbers to be the value of global cycle because of the balance between performance and computation cost. After certain rounds of global cycles, the traditional and the LSTM optimizer's loss curves will all basically converge and further global cycles will only make little general improvement.

In **Algorithm 2**, we normalize the correlation of loss-step with Softmax, which has the following form:

$$w_{ij} = \frac{e^{c_{ij}}}{\Sigma_{k=1}^{K} e^{c_{ik}}}$$

Algorithm 1 Federated Gradient-based meta learning

The K clients are indexed by k;
Initialize the k**th** optimizer network Φ_k with random weights ϕ_k
Initialize the k**th** optimizee network Θ_k with random weights θ_k
for Global cycle t $= 1 \rightarrow T$ **do**
  for each client k $= 1 \rightarrow K$ **do**
    Train client model Θ_k and Φ_k on local dataset D_k
    Freeze the optimizer weights Φ_k and reset the optimizee weights Θ_k
    Retrain the optimizee with the optimizer frozen on common test dataset and Get the loss-step l_k
  end for
  Update the frozen optimizer parameters with Pearson correlation of loss-step and Reset the optimizee parameters
end for

Algorithm 2 Pearson correlation of loss-step

The frozen optimizer parameters Φ_k(k=1,2,...K);
The loss-steps on common test dataset l_k(k=1,2,...K);
for each client i $= 1 \rightarrow K$ **do**
  for each client j $= 1 \rightarrow K$ **do**
    Compute the Pearson correlation c_{ij} between l_i and l_j
  end for
  Transform c_{ij} into normalized averaged weight w_{ij} with Softmax(j=1,2,...K)
  Update Φ_i with the averaged weights:
  $\Phi_i = \Sigma_{k=1}^{K} \Phi_k w_{ik}$
end for

A.2 DATASET INFORMATION

A.2.1 MNIST

The MNIST dataset comes from the National Institute of Standards and Technology (NIST). The training set consists of 250 handwritten numbers from different people, of which 50% are high school students and 50% are from the population Census Bureau (the Census Bureau) staff. The test set is also the same proportion of handwritten digital data.

A.2.2 EICU

The EICU dataset is an electronic medical record jointly produced by MIT and Harvard, which contains approximately 120,000 patient records. The entire dataset is presented in the form of a spreadsheet. A single patient record includes the patient's number, the patient's 1400 physiological characteristics and the patient's unit discharge status. The value of a certain physiological characteristic is only 0 and 1, indicating presence or absence. The unit discharge status also has only 0 and 1.

A.2.3 FASHION-MNIST

Fashion-MNIST is a dataset consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each exapmle is a 28x28 grayscale image, associated with a label from 10 classes. We intend Fashion-MNIST to serve as a direct drop-in replacement for the original MNIST dataset for benchmarking machine learning algorithms. It shares the same image size and structure of training and testing splits.

A.3 EXPERIMENTS IN MNIST

To preprocess the MNIST dataset, We split 60,000 images into training set and test set in a 1:1 manner. The first 30,000 pictures are equally divided into 5 silos and used for training separately.

After the optimizers of the 5 silos are trained and aggregated in some way, a global federated cycle is completed. There are 5 global federated cycles in the entire federated learning process. In each round of global federated testing, the test data is the second half of the overall dataset.

A.3.1 SAME MODELS BETWEEN SILOS

As for 5 silos' optimizee structures, we set layer size 20, number of layers 1. In this part, we use optimizee's loss-step curve as metric of the performance of different optimizers. And we choose the 5th cycle's testing result to compare different optimizers' performance.

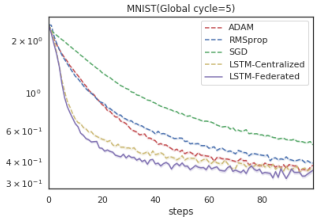
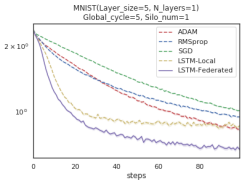


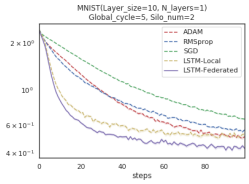
Figure 3: Comparisons among three different types of optimizers performance, i.e. the hand-crafted optimizers(ADAM, RMSprop, SGD), the optimizer trained by centralized data, the optimizer trained through global federated cycles

A.3.2 DIFFERENT MODELS BETWEEN SILOS

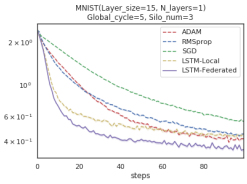
In the second experiment, 5 silos' optimizee model structures are respectively set as 5,10,15,20,25 and number of layers is set as 1. And we also show the 5th global cycle's testing loss. However, as each silo's optimizee structure is different, there are 5 silos' testing results to be shown.



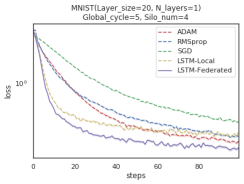
(a) Silo 1



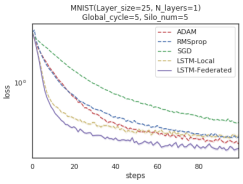
(b) Silo 2



(c) Silo 3



(d) Silo 4



(e) Silo 5

Figure 4: Comparisons among three different types of optimizers performance, i.e. the hand-crafted optimizers(ADAM, RMSprop, SGD), the optimizer trained through global federated cycles and the local optimizer trained only by 1 silo of data without federated learning process

A.4 EXPERIMENTS IN FASHION-MNIST

The process of Fashion-MNIST is the same us MNIST dataset. We split 60,000 images into training set and test set in a 1:1 manner. The first 30,000 pictures are equally divided into 5 silo and trained separately. After the models of the 5 silo are trained and aggregated in some way, a global federated cycle is completed. There are 5 global federated cycles in the entire federated learning process. In each round of global federated testing, the test data is the second half of the overall dataset.

A.4.1 SAME MODELS BETWEEN SILOS

In experiment 1, as for the model structure of optimizee, we set layer size 20, number of layers 1. optimizee's loss-step curve will be used to evaluate the performance of different optimizers. Like the MNIST section, we still choose the 5th cycle's testing result to compare the performance.

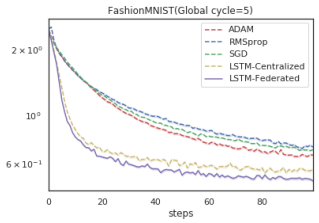
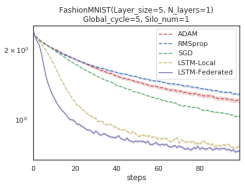


Figure 5: Comparisons among three different types of optimizers performance, i.e. the hand-crafted optimizers(ADAM, RMSprop, SGD), the optimizer trained by centralized data, the optimizer trained through global federated cycles

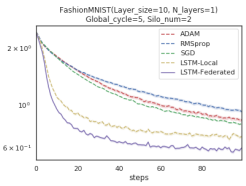
As you can see from the figure, the performance of LSTM optimizer is better than hand-crafted optimizers(Adam, RMSprop, SGD); to be more specific, LSTM optimzier can indeed make the objective function converge faster and converge to a smaller test error. Secondly, similar to section 3.1.1, the training effect of the LSTM optimizer finally trained through federated learning is better than the effect of using centralized data, which means that federated learning process does effectively integrate the model information of each silo.

A.4.2 DIFFERENT MODELS BETWEEN SILOS

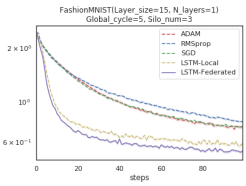
In experiment 2, 5 silos' optimizee model structures are respectively set as 5,10,15,20,25 and number of layers is set as 1. The 5th global cycle's testing loss is displayed. However, as each silo's optimizee structure is different, there are 5 silos' testing results to be shown.



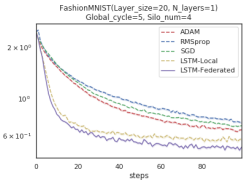
(a) Silo 1



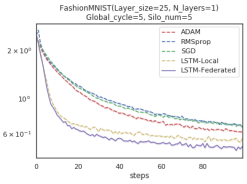
(b) Silo 2



(c) Silo 3



(d) Silo 4



(e) Silo 5

Figure 6: Comparisons among three different types of optimizers performance, i.e. the hand-crafted optimizers(ADAM, RMSprop, SGD), the optimizer trained through global federated cycles and the local optimizer trained only by 1 silo of data without federated learning process

According to the loss curve under each silo, first of all, the convergence result of LSTM-Local is better than that of the hand-crafted optimizers. What's more, the performance of LSTM-federated is significantly better than the other two, indicating that learned optimizer can be applied to optimizee with different structures, and federated learning can indeed effectively integrate information.