TOWARDS FASTER AND BETTER FEDERATED LEARNING: A FEATURE FUSION APPROACH

Xin Yao, Tianchi Huang, Chenglei Wu, Ruixiao Zhang, Lifeng Sun

Tsinghua University
Dept. of Computer Science and Technology
Beijing, China

ABSTRACT

Federated learning enables on-device training over distributed networks consisting of a massive amount of modern smart devices, such as smartphones and IoT devices. However, the leading optimization algorithm in such settings, i.e., *federated averaging*, suffers from heavy communication cost and inevitable performance drop, especially when the local data is distributed in a Non-IID way. In this paper, we propose a feature fusion method to address this problem. By aggregating the features from both the local and global models, we achieve a higher accuracy at less communication cost. Furthermore, the feature fusion modules offer better initialization for newly incoming clients and thus speed up the process of convergence. Experiments in popular federated learning scenarios show that our federated learning algorithm with feature fusion mechanism outperforms baselines in both accuracy and generalization ability while reducing the number of communication rounds by more than 60%.

***Index Terms*—** Federated Learning, Feature Fusion, Communication Cost, Generalization

1. INTRODUCTION

Mobile phones, wearable devices, and IoT devices play an important role in modern life. Intelligent applications on these devices are becoming popular, such as intelligent personal assistant, machine translation, keyboard input suggestion, etc. These applications usually use pre-trained models and perform forward inference on clients, which lacks flexibility and personalization. Meanwhile, smart edge devices are generating a tremendous amount of valuable yet privacy-sensitive data that has the potential to improve existing models. To take full advantage of the on-device data, traditional machine learning strategies require collecting data from clients, training a centralized model on the servers and then distributing the model to clients, which puts a heavy burden on the communication networks and is exposed to high privacy risks.

Recently, a series of work called *federated learning* (FL) [1, 2, 3] enables on-device training directly over the distributed networks. The leading algorithm is termed as *federated averaging* (FedAvg) [3], which performs distributed training on clients using their own local data and aggregates these models in a central server to avoid data sharing. In this way, FedAvg alleviates privacy concerns in a communication-efficient way. However, further studies [4, 5] point out that communication cost remains the main constraint in FL compared to other factors, e.g., the computation cost, and the accuracy of FedAvg would drop significantly if the models were trained with pathological Non-IID data.

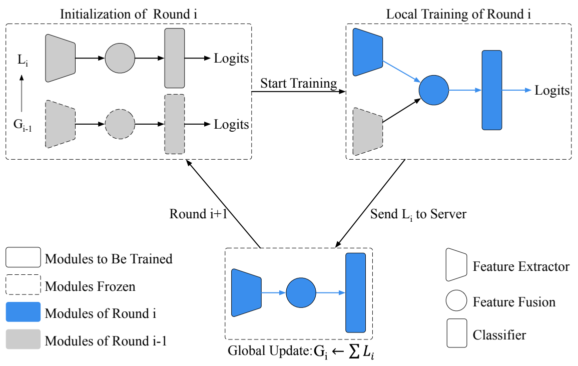
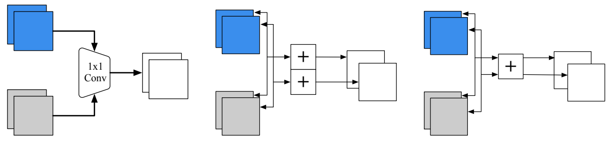


Fig. 1. Training Iteration of FedFusion. During the training procedure on clients, the local and global feature extractors are concatenated by a *feature fusion module*.

In this paper, we propose a new federated learning algorithm with feature fusion mechanism (FedFusion) to address the above problems. By introducing the feature fusion modules, we aggregate the features from both the local and global models after the feature extraction stage with little extra computation cost. These modules make the training process on each client more efficient and handle the Non-IID data distribution more pertinently, as each client will learn the most appropriate feature fusion module for itself.

In conclusion, our contributions are as follows:

- To the best of our knowledge, this is the first paper that introduces the feature fusion mechanism to the on-device training procedure in FL.



(a) Conv (b) Multi (c) Single

Fig. 2. Three types of feature fusion modules. The fusion operator is actually a mapping function, $F : \mathbb{R}^{2C\times H\times W} \rightarrow \mathbb{R}^{C\times H\times W}$

- The proposed feature fusion modules aggregate the features from the local and global models in an efficient and personalized way.
- Experiments on popular FL datasets show that the proposed FedFusion outperforms baselines in both accuracy and generalization ability while reducing the number of communication rounds by more than 60%.

2. RELATED WORK

Federated learning is proposed by McMahan et al. [3] to tackle the problem of decentralized training over massively distributed intelligent devices without access to the privacy-sensitive data directly.

Considering that communication cost remains the main constraint in FL, some research efforts have already been made. Konečný et al. [2] proposed structured and sketched updates in the context of client-to-server communication. Yao et al. [5] introduced extra constraints to the on-device training procedure, aiming to integrate more knowledge from other clients while fitting the local data. Caldas et al. [6] proposed federated dropout to train subsets on clients and extended the lossy compression [7] to server-to-client communication.

3. METHODS

In this section, we will first introduce the proposed feature fusion modules and then give our federated learning algorithm with feature fusion mechanism (FedFusion).

3.1. Feature Fusion Modules

The detailed architectures of feature fusion modules are illustrated in Fig. 2.

Concretely, an input image $\boldsymbol{x}$ is transformed into two feature spaces by the local feature extractor E_l and the global one E_g respectively, with the feature maps $E_l(x), E_g(x) \in \mathbb{R}^{C\times H\times W}$.

Then a fusion operator F embeds $E_l(x)$ and $E_g(x)$ into a fusion feature space, where $F(E_l(x), E_g(x)) \in \mathbb{R}^{C\times H\times W}$. In this paper, we introduce three types of fusion operator as follows.

Algorithm 1 Federated learning with feature fusion

```
Server:
1: initialize $G_0$
2: for each round r = 0, 1, 2, ... do
3:     $S_r \leftarrow$ random sample $m$ clients
4:     for each client $t \in S_r$ do in parallel
5:         $\mathcal{L}^t_{r+1} \leftarrow$ Client($G_r$)
6:     end for
7:     $G_{r+1} \leftarrow \frac{1}{n_{S_r}} \sum_{t\in S_r} n_t L^t_{r+1}$
8: end for
Client: Round $r$ on client $t$
1: $L^t_r = C \circ F \circ E_l \leftarrow G_r$        // $C$ is the classifier
2: for each batch $(x, y)$ do
3:     Compute $\mathcal{L}(C \circ F(E_l(x), E_g(x)), y)$
4:     Update $E_l, F, C$ by backpropagation
5: end for
6: return $L^t_{r+1}$ to server
```

Conv operator (F_{conv}) is implemented as 1×1 convolutions,

$$F_{conv}(E_l(x), E_g(x)) = \boldsymbol{W}_{conv}(E_g(x)||E_l(x)) \quad (1)$$

where $\boldsymbol{W}_{conv} \in \mathbb{R}^{2C\times C}$ is the learned weight matrix and $||$ denotes the operation that concatenates the feature maps along channel axis.

Multi operator (F_{multi}) introduces a learned weight *vector* $\boldsymbol{\lambda} \in \mathbb{R}^C$ and computes the weighted sum between the local and global feature maps,

$$F_{multi}(E_l(x), E_g(x)) = \boldsymbol{\lambda} E_g(x) + (\boldsymbol{1} - \boldsymbol{\lambda})E_l(x) \quad (2)$$

where the weighted vector $\boldsymbol{\lambda}$ is first broadcasted to the shape of $C \times H \times W$ and then multiplied by the feature maps element-wise, as illustrated in Fig. 2(b).

Single operator (F_{single}) uses a learned weight *scalar* λ and computes the weighted sum between the local and global feature maps,

$$F_{single}(E_l(x), E_g(x)) = \lambda E_g(x) + (1 - \lambda)E_l(x) \quad (3)$$

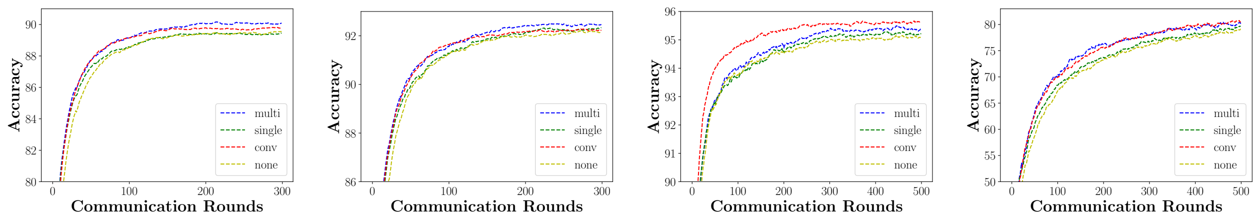
where the global and local feature maps are scaled by λ and $1 - \lambda$ respectively and then added together element-wise, as shown in Fig. 2(c).

3.2. Federated Learning with Feature Fusion Mechanism

A typical training iteration of the proposed FedFusion is shown in Fig. 1.

At the beginning of round i, we keep the feature extractor of the global model (E_g) instead of throwing it away as in FedAvg after initialization.

During training, E_g is frozen and an additional feature fusion module described in Sec. 3.1 is introduced. In practice,



(a) (b) (c) (d)

Fig. 3. The overall *accuracy* vs. *communication rounds* curves under different settings. (a) and (b): the artificial Non-IID partitions of CIFAR10. (c): user specific Non-IID partition, which is implemented by applying different permutations to MNIST. (d): IID partition of CIFAR10. *multi*, *single* and *conv* are FedFusion with corresponding fusion operators while *none* denotes FedAvg.

it's possible to record the global feature maps generated by E_g in one round forward inference. In other words, the additional feature fusion module brings limit extra computation cost.

After the on-device training procedure, the local model combined with the feature fusion module will be sent to the central server for model aggregation. For *multi* and *single* operators, we use an exponential moving average strategy to smooth the update.

The pseudo code of FedFusion is shown in Algorithm 1.

4. EXPERIMENTS

In this section, we will first present the experimental setup (Sec. 4.1) and then show the results under several different settings (Sec 4.2, 4.3 and 4.4).

4.1. Experimental Setup

Datasets

We use MNIST [8] and CIFAR10 [9] as basic datasets in our experiments. We further proposed three types of data partitions to benchmark FL algorithms, e.g., our FedFusion and the original FedAvg.

The first is *Artificial Non-IID Partition*, which is implemented by splitting an existing IID dataset to meet the FL scenario and commonly used in previous FL studies [1, 2, 3, 6, 10]. In this partition, a single client usually has a subset of the classes of the total data. For example, most clients have up to two digits of MNIST in [3].

The second is *User Specific Non-IID Partition*, where the data on different clients usually have similar classes but follows different distributions. This is commonly used in multi-task learning studies [11, 12, 13].

The last is *IID Partition*, which is a simple yet necessary partition to evaluate FL algorithms [3, 6].

Models

For MNIST digits recognition task, we use the the same model as FedAvg [3]: a CNN with two 5×5 convolution layers (the first with 32 channels while the second with 64, each followed by a ReLU activation and 2×2 max pooling), a fully connected layer with 512 units (with a ReLU activation and random dropout), and a final softmax output layer.

For CIFAR10 we use a CNN with two 5×5 convolution layers (both with 64 channels, each followed by a ReLU activation and 3×3 max pooling with stride size 2), two fully connected layers (the first with 384 units while the second with 192, each followed by a ReLU activation and random dropout) and a final softmax output layer.

Table 1. The convergence accuracy of FedFusion and FedAvg under different setups. (a-d) corresponds to those in Fig. 3.

	(a)	(b)	(c)	(d)
FedAvg	89.89	92.21	95.20	80.01
FedFusion+Single	89.77	92.32	95.25	80.85
FedFusion+Multi	**90.51**	**92.78**	95.43	**82.95**
FedFusion+Conv	90.11	92.53	**95.79**	82.15

4.2. Artificial Non-IID Partition

Under the artificial Non-IID scenarios, we use a learning rate of 3e-3 with an exponential decay factor 0.985 each round for all our FedFusion methods (with different fusion operators) and the compared FedAvg. The convergence behaviors of them are illustrated in Fig. 3 (a) and (b) while the final accuracies at convergence are shown in Table 1.

The curve representing FedFusion with *multi* operator is always above others, which means it achieves a higher accuracy at less communication cost. The accuracy of FedFusion with *conv* also raises faster at the beginning but fails to reach a better convergence point. FedFusion with *single* and FedAvg perform relatively worse.

Such results are obviously due to the *multi* fusion operator. As stated before, most clients have a subset of the total classes in artificial Non-IID scenarios. The *multi* operator allows the models on clients to select the feature maps that are helpful to their local data. In contrast, FedAvg does not offer

Table 2. Number of communication rounds to reach certain accuracy milestones. FedAvg is considered as the baseline, and reductions in communication rounds are listed.

	94%		95%	
	rounds	reduce	rounds	reduce
FedAvg	100	(ref)	256	(ref)
FedFusion+Single	87	13.0%	227	11.3%
FedFusion+Multi	78	22.0%	201	21.5%
FedFusion+Conv	**34**	**66.0%**	**92**	**64.1%**

the selection and the *single* operator does not provide enough room for adjustment.

4.3. User Specific Non-IID Partition

To simulate the *user specific Non-IID partition*, we apply different permutations to MNIST on each client, which is the so-called *Permuted MNIST* in several previous studies [14, 15]. We use a learning rate of 2e-3 with an exponential decay factor 0.99 each round for all the methods.

The number of communication rounds to reach certain accuracy milestones (94% and 95% here), as well as the reduction in communication rounds versus FedAvg, is shown in Table 2. The results indicate that FedFusion with *conv* leads in a large margin, which is different from that in *artificial Non-IID partition*. FedFusion with *conv* achieves the best performance while reducing the number of communication rounds by more than 60%. In *user specific Non-IID partition*, the data on clients have similar classes but follow different distributions. The *conv* operator has better ability to integrate the feature maps from both the local and global models, in other words, the knowledge from other clients and data distributions. It is worth noting that *user specific 'Non-IID partition* is closer to the realistic FL scenarios, thus the improvement makes more sense in this case.

Additionally, we study the generalization ability of the model that was usually ignored in previous FL research works. The local epochs to reach convergence for newly incoming clients are illustrated in Fig. 4. As we can see, when a new client joins an existing FL system, FedFusion with *conv* provides a better initialization than other algorithms and thus speeds up the process of convergence.

4.4. IID Partition

The *IID partition* is a simple yet necessary partition to evaluate FL algorithms. If one strategy cannot handle this partition, its effectiveness is questionable.

As shown in Fig. 3 (d), FedFusion with *multi* and *conv* achieve higher accuracy at less communication cost. In terms of the final convergence accuracy, FedFusion with *multi* and *conv* have an impressive improvement than other methods.

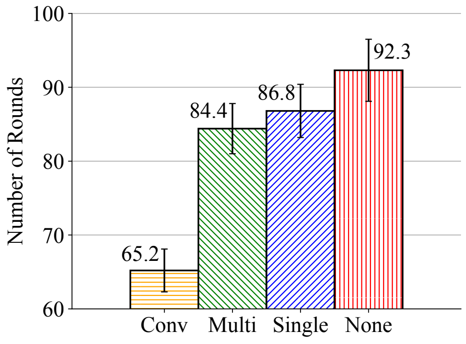


Fig. 4. Number of local epochs to reach convergence for newly incoming clients.

To make a brief conclusion on the feature fusion operators as follows: The *multi* operator offers flexible choices between the local and global feature maps and is more interpretable. Entries in the weight vector $\boldsymbol{\lambda}$ account for the proportions of the global feature maps in corresponding channels. When there were gaps in the classes of data, *multi* operators would learn to choose the most helpful feature maps. The *conv* operator is better at integrating the knowledge from the global and local models. If the data on clients had similar classes but followed different distributions, *conv* operator performs much better. Our experiments indicate that *single* operator has few improvements and should not be adopted in practice.

5. CONCLUSION

The heavy communication cost of federated learning is an emergency problem to solve. In this paper, we try to do some improvements from the perspective of reducing the communication rounds. We propose a new FL algorithm with feature fusion modules and evaluate it in popular FL setups. The experimental results show that the proposed method achieves a higher accuracy while reducing the communication rounds by more than 60%. Furthermore, we observe that FedFusion offers better generalization for newly incoming clients.

Future work may include extending our algorithm to more complicated models and scenarios, as well as combining the communication rounds reduction strategy with other kinds of methods, e.g., gradient estimation and compression.

6. ACKNOWLEDGEMENT

Thanks to the anonymous reviewers for their valuable feedback. This work is supported by the National Key R&D Program of China (2018YFB1003703), National Natural Science Foundation of China (61521002), as well as Beijing Key Lab of Networked Multimedia (Z161100005016051).

7. REFERENCES

[1] Jakub Konečnỳ, Brendan McMahan, and Daniel Ramage, "Federated optimization: Distributed optimization beyond the datacenter," *arXiv preprint arXiv:1511.03575*, 2015.

[2] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[3] 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*, 2017, pp. 1273–1282.

[4] Eunjeong Jeong, Seungeun Oh, Hyesung Kim, Jihong Park, Mehdi Bennis, and Seong-Lyun Kim, "Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data," *arXiv preprint arXiv:1811.11479*, 2018.

[5] Xin Yao, Chaofeng Huang, and Lifeng Sun, "Two-stream federated learning: Reduce the communication costs," in *Visual Communications and Image Processing (VCIP), 2018*. IEEE, 2018, pp. 1–4.

[6] Sebastian Caldas, Jakub Konečny, H Brendan McMahan, and Ameet Talwalkar, "Expanding the reach of federated learning by reducing client resource requirements," *arXiv preprint arXiv:1812.07210*, 2018.

[7] Ananda Theertha Suresh, Felix X Yu, Sanjiv Kumar, and H Brendan McMahan, "Distributed mean estimation with limited communication," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 3329–3337.

[8] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[9] Alex Krizhevsky and Geoffrey Hinton, "Learning multiple layers of features from tiny images," Tech. Rep., Citeseer, 2009.

[10] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[11] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar, "Federated multi-task learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 4424–4434.

[12] Fei Chen, Zhenhua Dong, Zhenguo Li, and Xiuqiang He, "Federated meta-learning for recommendation," *arXiv preprint arXiv:1802.07876*, 2018.

[13] Sebastian Caldas, Peter Wu, Tian Li, Jakub Konečnỳ, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar, "Leaf: A benchmark for federated settings," *arXiv preprint arXiv:1812.01097*, 2018.

[14] Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio, "An empirical investigation of catastrophic forgetting in gradient-based neural networks," *arXiv preprint arXiv:1312.6211*, 2013.

[15] Friedemann Zenke, Ben Poole, and Surya Ganguli, "Continual learning through synaptic intelligence," in *International Conference on Machine Learning*, 2017, pp. 3987–3995.