Federated Learning with Additional Mechanisms on Clients to Reduce Communication Costs

Xin Yao, Tianchi Huang, Chenglei Wu, Rui-xiao Zhang, Lifeng Sun
Department of Computer Science and Technology
Tsinghua University, Beijing, China
{yaox16,htc19,wucl18,zhangrx17}@mails.tsinghua.edu.cn
sunlf@tsinghua.edu.cn

Abstract

Federated learning (FL) 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* (FedAvg), suffers from heavy communication cost and inevitable performance drop, especially when the local data is distributed in a non-IID way. To alleviate this problem, we propose two potential solutions by introducing additional mechanisms to the on-device training.

The first (FedMMD) is adopting a two-stream model with the MMD (Maximum Mean Discrepancy) constraint instead of a single model in vanilla FedAvg to be trained on devices. Experiments show that the proposed method outperforms baselines, especially in non-IID FL settings, with a reduction of more than 20% in required communication rounds.

The second is FL with feature fusion (FedFusion). By aggregating the features from both the local and global models, we achieve 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 FL scenarios show that our FedFusion outperforms baselines in both accuracy and generalization ability while reducing the number of required communication rounds by more than 60%.

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 issuing 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) [7, 8, 13] enables on-device training directly over the distributed networks. The aim of federating learning is to train a model from data $\{X^1, ..., X^K\}$ generated by K distributed clients (each device is treated as a client). Each client, $t \in [K]$, generates data in a non-IID manner, which means the data distribution on client t, $X^t \sim P^t$, is not a uniform sample of the whole distribution. The leading algorithm in FL, i.e., *federated averaging* (FedAvg) [13], assume a synchronous update scheme that proceeds in rounds

---

This is a combination version of our papers in VCIP 2018 [18] and ICIP 2019 [19].

of communication. Considering a fixed number of K clients, each with a local dataset, a random fraction C of clients is selected to participate this round of update at the beginning of each round. The server sends the current *global model* to each of these chosen clients. Then each client computes a unique *local model* based on the global model and its local data, and reports it to the server. Finally, the server updates the global model by model averaging and begins the next round. By adding more computation on each client during the optimization iterations, federated learning algorithm reduces the required communication rounds compared with traditional SGD based methods. However, further studies [6, 18] 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.

To alleviate this problem, we propose two potential solutions by introducing additional mechanisms to the on-device training.

We first propose adopt a two-stream model, which is commonly used in transfer learning and domain adaptation [11, 15, 22], instead of a single model to be trained on devices in federated learning settings. Maximum Mean Discrepancy (MMD) constraint [5] is introduce into the on-device training iterations of our method, which forces the local model into learning more knowledge from the global one. Further experiments show the proposed methods (FedMMD) bring a reduction in required communication rounds without affecting the final test performance.

Then we further propose a new FL 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:

- We use a two-stream model with MMD (Maximum Mean Discrepancy) constraint instead of a single model in vanilla FedAvg to be trained on devices.

- We propose aggregating the features from both the local and global models during the on-device training.

- Experiments on popular FL settings show that the proposed methods outperform baselines in both accuracy and generalization ability while reducing the number of communication rounds by up to more than 60%.

2   Related Work

2.1   Federated Learning

Federated learning is proposed by McMahan et al. [13] 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. [8] proposed structured and sketched updates in the context of client-to-server communication. Yao et al. [18] 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. [1] proposed federated dropout to train subsets on clients and extended the lossy compression [16] to server-to-client communication.

2.2   Maximum Mean Discrepancy

As the name suggests, Maximum Mean Discrepancy (MMD) measures the distance between the means of two data distributions, and is widely used in domain adaptation problems [11][12][17], describing the difference between features generated from source and target domains. By minimizing the MMD loss, they force the two-stream model extract more generalized features, which is very similar to our purpose of learning better representations in the whole dataset. In this paper, we focus on the multiple kernel variant of MMD (MK-MMD) proposed by Gretton et al [5], which first maps the data distributions to a Reproducing Kernel Hilbert Space (RKHS).
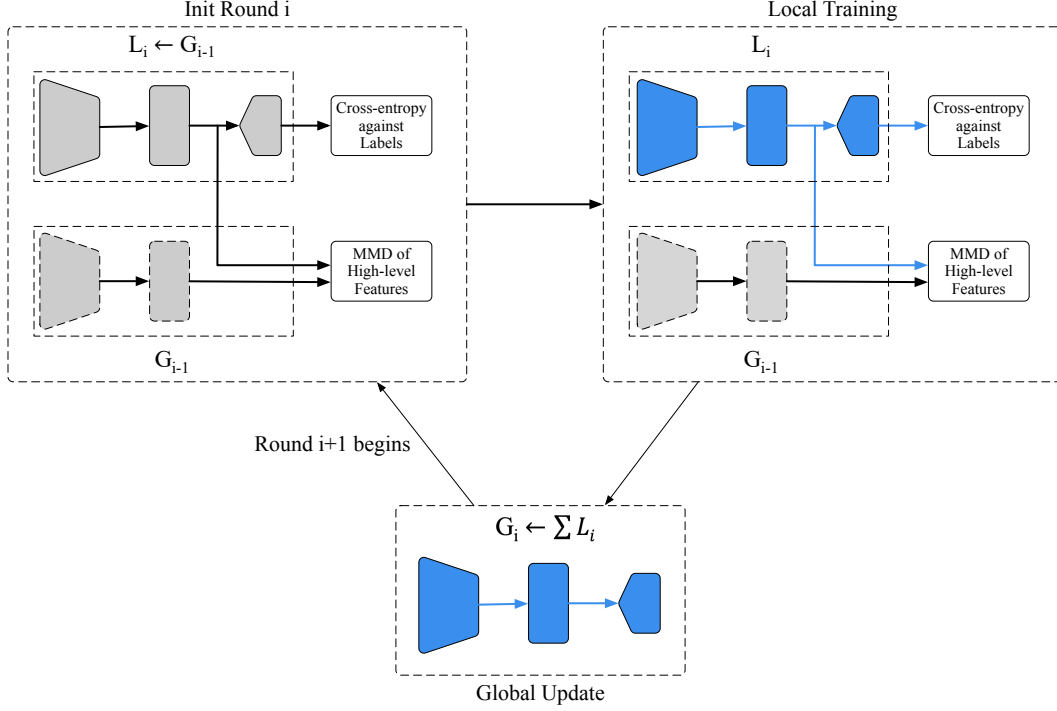
Figure 1: Two-stream model with MMD: the global model is fixed while the local model is trained through back propagation. (Better viewed in color)

Given two data distributions x and y, the square of MMD between them can be expressed as:

$$MMD^2(x, y) = \|E[\phi(x)] - E[\phi(y)]\|^2 \tag{1}$$

where $\phi(\cdot)$ denotes the mapping to RKHS. In practice, this mapping is unknown. Using the kernel trick to replace the inner product and we have:

$$MMD^2(x, y) = E[K(x, x)] + E[K(y, y)] - 2E[K(x, y)] \tag{2}$$

where $K(x, y) = \langle \phi(x), \phi(y) \rangle$ is the desired kernel function. In this paper, we use a standard radial basis function (RBF) kernel with multiple width.

3 Methods

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

3.1 Two Stream Federated Learning

Formally, let Θ^G and Θ^L be the model parameters, that is, the weights and biases, of all the layers in the global and local models respectively. Let $X^t = \{x_i^t\}_{i=1}^{n_t}$ and $Y^t = \{y_i^t\}_{i=1}^{n_t}$ denote the training data and the corresponding labels on client t, where n_t is the number of examples.

Figure 1 shows how our two-stream model works on the specific client t. As show in the figure, the global model is received from the server at the beginning of the current round and fixed in the following training process, while the local model is initialize with the parameters of the global model ($\Theta^L \leftarrow \Theta^G$) and then trained on the local data X^t and labels Y^t by minimizing a loss function in the

---

Algorithm 1 FedMMD

Server:
 1: initialize Θ_0^G
 2: **for** each round r = 1, 2, ... **do**
 3:     $S_r \leftarrow$ (random set of $C \cdot K$ clients)
 4:     **for** each client $t \in S_r$ **do**
 5:         $\Theta_{r+1}^t \leftarrow Client(\Theta_r^G)$
 6:     **end for**
 7:     $\Theta_{r+1}^G \leftarrow \sum_{t \in S_r} \Theta_{r+1}^t$
 8: **end for**

Client: Run on client t
 1: **for** each local epoch **do**
 2:     **for** each batch b **do**
 3:         Update Θ^L by minimizing $L(\Theta^L | \Theta^G, X^t, Y^t)$
 4:     **end for**
 5: **end for**
 6: return Θ^L to server

---

form:

$$L(\Theta^L | \Theta^G, X^t, Y^t) = L_{cls} + L_{MMD} \tag{3}$$

$$L_{cls} = \frac{1}{n^t} \sum_{i=1}^{n^t} J(\theta^L(x_i^t), y_i^t) \tag{4}$$

$$L_{MMD} = \lambda MMD^2(\theta^G(X^t), \theta^L(X^t)) \tag{5}$$

where $\theta^G(X^t)$ and $\theta^L(X^t)$ denote the outputs of the global model Θ^G and local model Θ^L with the corresponding input X^t respectively. $J(\theta(x), y)$ denotes a standard classification loss, such as cross-entropy loss function in our experiments. L_{MMD} denotes the MMD loss between the outputs of the global and local models computed by Equation (2). This term is weighted by coefficient λ.

Federated learning algorithm is indeed a cycle process of learning local representations, merging knowledge from different clients and then learning again. In other words, the global model contains more knowledge from multiple clients while the local model learns better representations of the local data. Compared with a single local model being trained in standard federated learning settings, we keep the global model as a reference instead of throwing it away after initializing. As discussed in Section 2.2, MMD is a measurement of the distance between the means of two data distributions. By minimizing the MMD loss term between the outputs of the global and local models, we force the local model learn more knowledge from other clients in addition to the representations of data on the current client, thus accelerating the convergence of the training process, in other words, reducing the communication rounds.

Our two-stream federated learning algorithm is described as Algorithm 1.

3.2 Feature Fusion Modules

The detailed architectures of feature fusion modules are illustrated in Figure 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.

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

$$F_{conv}(E_l(x), E_g(x)) = \boldsymbol{W}_{conv}(E_g(x) || E_l(x)) \tag{6}$$

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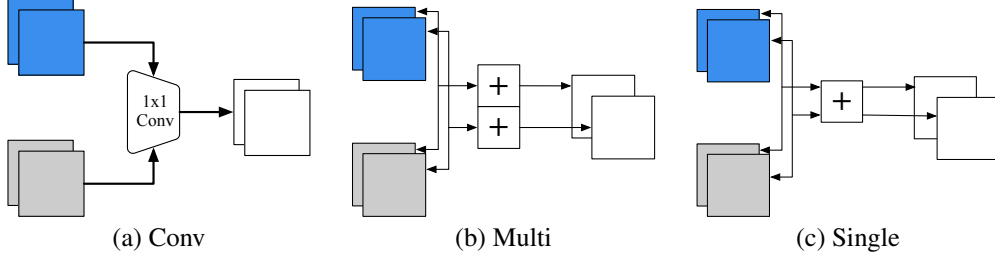
Figure 2: Three types of feature fusion modules. The fusion operator is actually a mapping function, $F : \mathbb{R}^{2C \times H \times W} \rightarrow \mathbb{R}^{C \times H \times W}$
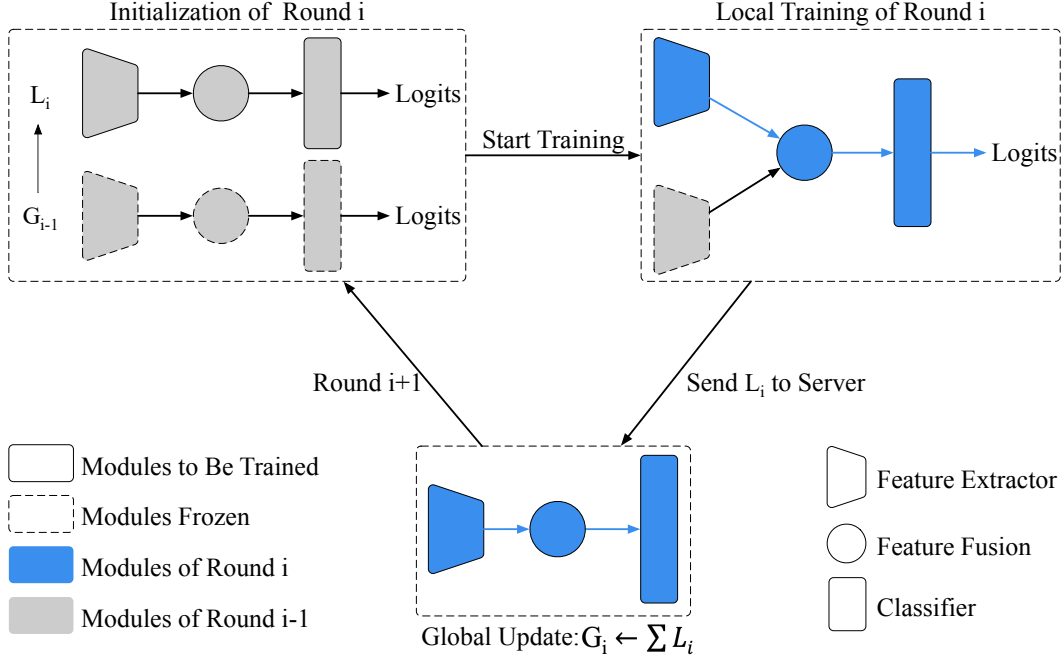

Figure 3: Training Iteration of FedFusion. During the training procedure on clients, the local and global feature extractors are concatenated by a *feature fusion module*.

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,

$$F_{multi}(E_l(x), E_g(x)) = \boldsymbol{\lambda} E_g(x) + (\mathbf{1} - \boldsymbol{\lambda}) E_l(x) \tag{7}$$

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 Figure 2(b).

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

$$F_{single}(E_l(x), E_g(x)) = \lambda E_g(x) + (1 - \lambda) E_l(x) \tag{8}$$

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

3.3 Federated Learning with Feature Fusion Mechanism

A typical training iteration of the proposed FedFusion is shown in Figure 3.

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

---

Algorithm 2 FedFusion

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:        $L_{r+1}^t \leftarrow$ **Client**(G_r)
 6:     **end for**
 7:     $G_{r+1} \leftarrow \frac{1}{n_{S_r}} \sum_{t \in S_r} n_t L_{r+1}^t$
 8: **end for**

Client: Round r on client t
 1: $L_r^t = C \circ F \circ E_l \leftarrow G_r$ // 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_{r+1}^t to server

---

feature fusion module described in Section 3.2 is introduced. In practice, 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 2.

4 Experiments

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

4.1 Experimental Setup

Datasets

We use MNIST [10] 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, 7, 8, 13, 21]. 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 [13].

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 [2, 3, 14].

The last is *IID Partition*, which is a simple yet necessary partition to evaluate FL algorithms [1, 13].

4.1.1 Models

For MNIST digits recognition task, we use the the same model as FedAvg [13]: 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
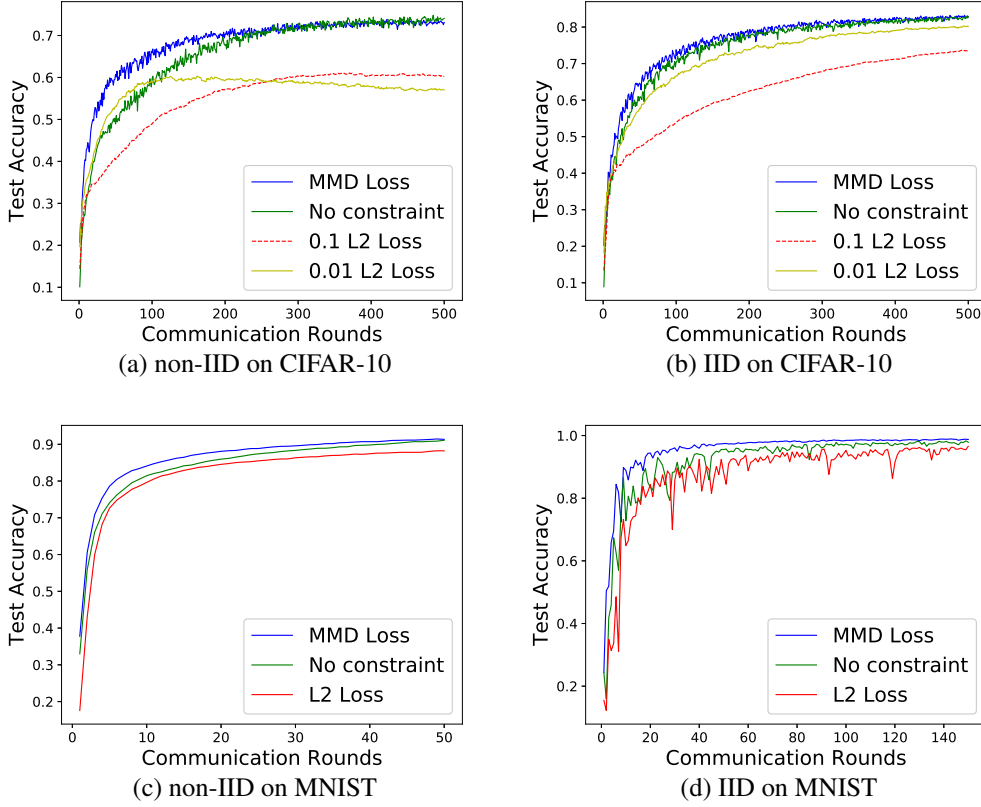
Figure 4: Test accuracy over communication rounds for FedMMD on CIFAR-10 (upper row) and MNIST (lower row) with non-IID (left column) and IID (right column) settings. (Better viewed in color)

with 384 units while the second with 192, each followed by a ReLU activation and random dropout) and a final softmax output layer.

4.2 FedMMD

4.2.1 CNN on CIFAR-10

For convenience but without loss of generality, we select a fixed group of hyper-parameters in federated learning algorithm, that is, two clients participating in the training process, with a local batch size $B = 128$ and local epochs $E = 2$. During the training, we use a SGD optimizer with the learning rate of 2×10^{-3}. The penalty parameter λ for L_{MMD} is set to 0.1 in this experiment. As for the penalty parameter for L2 norm, we have tried 0.1 and 0.01, and the results are shown in Figure 4 (a, b).

We explore both non-IID and IID data distributions and show that our two-stream model with MMD constraint outperforms the baseline methods, especially in non-IID data distribution. Figure 4 (a) shows the non-IID situation, where we split the 10 classes of images in CIFAR-10 dataset into 2 parts, each containing 5 classes without overlap, indicating the non-IID data distribution. Our method needs less communication rounds to get to convergence. More concretely, our method reaches the test accuracy of 0.72 by 260 rounds of communication while the standard method needs 326 rounds, achieving a reduction of 20.2%. It is worth noting that, MMD forces the local model learn more knowledge from the global model but does not introduce new information into the overall optimization system compared with the standard method, thus the final convergence results are the same, which is consistent as expected. Figure 4 (b) shows the IID situation. As we can see, the two-stream model performs similar to the standard federated algorithm. In this case, the data on each

7

Table 1: The convergence accuracy of FedFusion and FedAvg under different setups. (a-d) corresponds to those in Figure 5.

	(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

client is a uniform sample of the overall dataset, which means the local model is able to learn the complete representations, and as a result, the role of MMD constraint is weakened. Two-stream model constrained by L2 norm underperforms the other methods, indicating that selection of constraints does matter.

4.2.2 CNN on MNIST

Similar to the CIFAR-10 experiments, we first explore the binary separated non-IID distributions and the results are shown in Figure 4 (c). During the training, we use a SGD optimizer with the learning rate of 1×10^{-4}. The penalty parameter λ for L_{MMD} and L_2 are 0.1 and 0.001, respectively. We use a rather small λ for L_2 as we find that a larger one may lead to non-convergence situation. As we can see, our two-stream model shows faster convergence rate compared with other methods and does not lower the final test accuracy. The reduction of communication rounds is less than the result on CIFAR-10. We speculate that the variety of feature representations in MNIST is much less due to the black-and-white images and simple lines in them.

We further study a more complex non-IID data distribution described in [13], where we first sort the data in MNIST by digit label, divide it into 200 shards of size 300, and assign each of 100 clients 2 shards. This is a typical non-IID partition of the data, as most clients will only have examples of two digits. And we set $C = 0.1$, with local batch size $B = 10$ and local epochs $E = 2$. As shown in Figure 4 (d), the curve of test accuracy dithers due to the extremely pathological data distribution. Our two-stream model with MMD constraint achieves a test accuracy of 98% by 72 rounds of communication while the standard method needs 128 rounds, which means a reduction of 23.4%.

4.3 FedFusion

4.3.1 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 Figure 5 (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 the selection and the *single* operator does not provide enough room for adjustment.

4.3.2 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 [4, 20]. We use a learning rate of 2e-3 with an exponential decay factor 0.99 each round for all the methods.
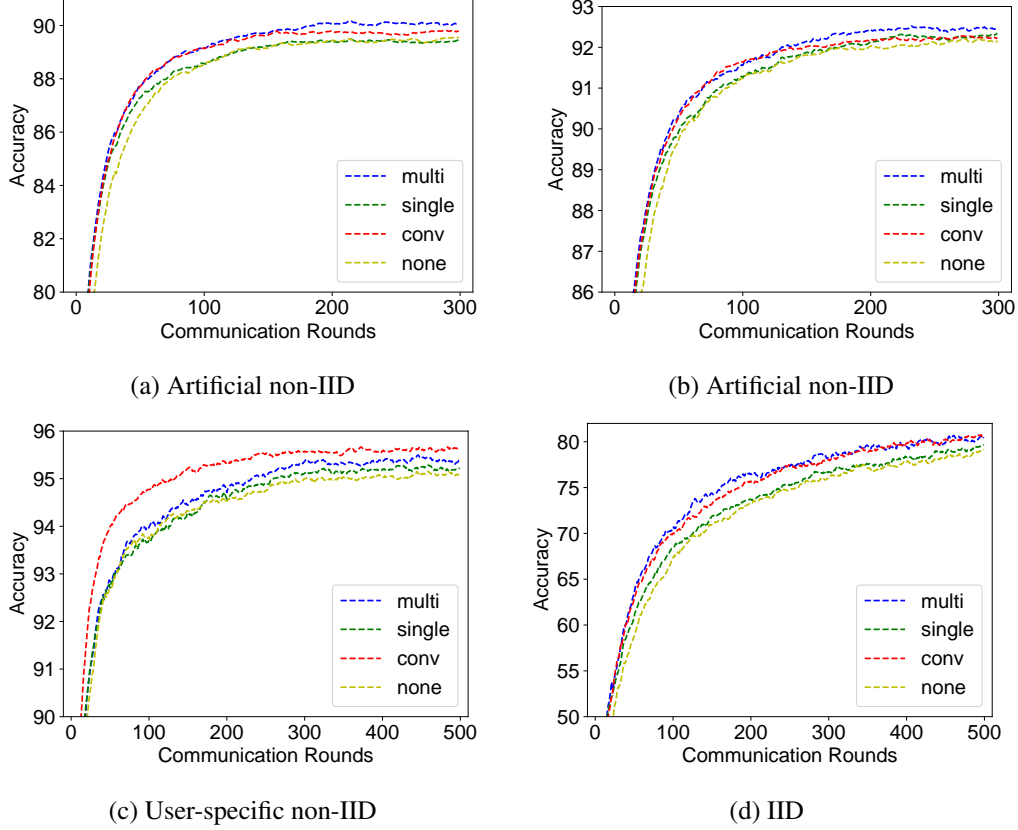
Figure 5: Test accuracy vs. communication rounds under different settings. (a) and (b): the artificial non-IID partitions of CIFAR10. (c): user specific non-IID partition, which is implemented by applying different permutations to MNIST. (d): IID partition of CIFAR10. *multi*, *single* and *conv* are FedFusion with corresponding fusion operators while *none* denotes FedAvg.

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 Figure 6. 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.

4.3.3 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 Figure 5 (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.

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

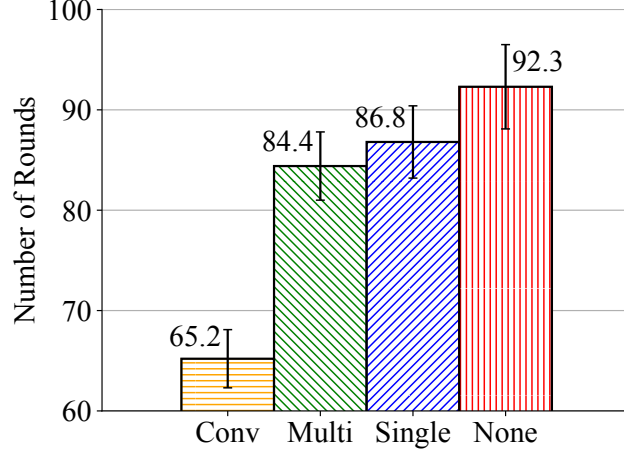|	94%		95%	
	rounds	reduce	rounds	reduce
FedAvg	100	(ref)	256	(ref)
FedFusion+Single	87	13.0%	227	11.3%
FedFusion+Multi	78	22.0%	201	21.5%
FedFusion+Conv	**34**	**66.0%**	**92**	**64.1%**

Figure 6: 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 λ 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 first replace the single model trained on clients in federated learning settings with a two-stream model including the global and local models. Our experiments show that introducing the MMD constraint into the optimization algorithm will bring a reduction in communication rounds, especially in non-IID FL settings. Further 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%. What is more, we observe that FedFusion offers better generalization for newly incoming clients.

6 Acknowledgement

References

[1] S. Caldas, J. Konečny, H. B. McMahan, and A. Talwalkar. Expanding the reach of federated learning by reducing client resource requirements. *arXiv preprint arXiv:1812.07210*, 2018.

[2] S. Caldas, P. Wu, T. Li, J. Konečnỳ, H. B. McMahan, V. Smith, and A. Talwalkar. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*, 2018.

[3] F. Chen, Z. Dong, Z. Li, and X. He. Federated meta-learning for recommendation. *arXiv preprint arXiv:1802.07876*, 2018.

[4] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.

[5] A. Gretton, D. Sejdinovic, H. Strathmann, S. Balakrishnan, M. Pontil, K. Fukumizu, and B. K. Sriperumbudur. Optimal kernel choice for large-scale two-sample tests. In *Advances in neural information processing systems*, pages 1205–1213, 2012.

[6] E. Jeong, S. Oh, H. Kim, J. Park, M. Bennis, and S.-L. Kim. Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data. *arXiv preprint arXiv:1811.11479*, 2018.

[7] J. Konečnỳ, B. McMahan, and D. Ramage. Federated optimization: Distributed optimization beyond the datacenter. *arXiv preprint arXiv:1511.03575*, 2015.

[8] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[9] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[10] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[11] M. Long, Y. Cao, J. Wang, and M. I. Jordan. Learning transferable features with deep adaptation networks. *arXiv preprint arXiv:1502.02791*, 2015.

[12] M. Long, J. Wang, G. Ding, J. Sun, and S. Y. Philip. Transfer feature learning with joint distribution adaptation. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 2200–2207. IEEE, 2013.

[13] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282, 2017.

[14] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar. Federated multi-task learning. In *Advances in Neural Information Processing Systems*, pages 4424–4434, 2017.

[15] B. Sun and K. Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *European Conference on Computer Vision*, pages 443–450. Springer, 2016.

[16] A. T. Suresh, F. X. Yu, S. Kumar, and H. B. McMahan. Distributed mean estimation with limited communication. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3329–3337. JMLR. org, 2017.

[17] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014.

[18] X. Yao, C. Huang, and L. Sun. Two-stream federated learning: Reduce the communication costs. In *Visual Communications and Image Processing (VCIP), 2018*, pages 1–4. IEEE, 2018.

[19] X. Yao, T. Huang, C. Wu, R. Zhang, and L. Sun. Towards faster and better federated learning: A feature fusion approach. In *IEEE International Conference on Image Processing*, 2019.

[20] F. Zenke, B. Poole, and S. Ganguli. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, pages 3987–3995, 2017.

[21] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

[22] J. Zhuo, S. Wang, W. Zhang, and Q. Huang. Deep unsupervised convolutional domain adaptation. In *Proceedings of the 2017 ACM on Multimedia Conference*, pages 261–269. ACM, 2017.