



EB-FedAvg: Personalized and Training Efficient Federated Learning with Early-Bird Tickets

Dongdong Li^(✉)

School of Computer Science and Engineering, South China University of Technology,
Guangzhou 510000, GD, China
201910106942@mail.scut.edu.cn

Abstract. Federated learning is a well-known way to improve privacy in distributed machine learning. Its major goal is to learn a global model that provides good performance to the broadest number of participants. Statistical heterogeneity (also known as non-IID) and training efficiency are two key unresolved concerns in the rapidly developing field of technology. In this paper, we propose Early-Bird FedAvg (EB-FedAvg), a customized federated learning architecture with personalization and training effects based on Early-Bird Tickets. By applying for the early-bird tickets, each client learns an early-bird ticket network (i.e., a sub-network of the base model), and only these early-bird ticket networks are communicated between the server and clients. Instead of learning a shared global model as in traditional federated learning, each client learns a personalized model with EB-FedAvg; communication costs can be greatly reduced due to the compact size of the early-bird ticket network. Experiments on these datasets show that EB-FedAvg outperforms existing systems in personalization, training, and communication cost.

Keywords: Personalization · Efficient federated learning systems · Data heterogeneity

1 Introduction

Federated learning (FL) is a well-liked distributed machine learning framework that enables several clients to train a common global model cooperatively without transmitting their local data [1]. The FL process is coordinated by a central server, and each participating client exchanges only the model parameters with the central server while maintaining local data privacy. By overcoming privacy issues, FL enables machine learning models to learn from decentralized data. FL has been used in numerous real-world situations when data is dispersed across clients and is too delicate to be collected in one location. For instance, FL has been shown to work well when it is used to predict the next word on a smartphone [2].

Participating clients want a shared global model that performs better than their models. Data distribution between clients is non-IID [1, 3]. Due to statistical unpredictability, it's difficult to design a worldwide model that works for all clients. Several research has used FL personalization techniques like meta-learning, multi-task learning, transfer learning, etc. to decrease statistical heterogeneity [4–10].

These solutions frequently entail two steps: 1) developing a global model together, and 2) adapting it for each customer using local data. Two-step customization increases costs. FL's computational cost and diverse devices cause the central server to wait while clients are trained, consuming a lot of energy. Inference speedup and model compression are important FL training advancements. The progressive pruning and training practice involves training a large model, pruning it, and then retraining it to increase performance (the process can be iterated several times). This is a conventional model compression strategy, but recent research links it to more effective training [11]. New research reveals that dense, randomly begun networks contain microscopic subnetworks that, when trained independently, can approach the test accuracy of original networks [12, 13].

Unpredictability in the communication channel between the central server and participating clients might cause transmission delays owing to bandwidth limits. Compressing data between the server and client solves the bottleneck. Sparsification, quantization, etc. [14, 15]. Are common methods. Few efforts have been made to solve both challenges at once. LG-FedAvg may be the only exception [16]. LG-FedAvg was developed with an unreasonable FL configuration, despite each client having enough training data (300 images per class for MNIST and 250 for CIFAR-10).

Our work: We construct EB-FedAvg utilizing Early-Bird Tickets, a bespoke FL framework for training and communication [17]. The Early-Bird Ticket phenomenon helps locate sparse subnetworks within a large base model (EBTNs). Given the same training, EBTN often outperform a non-sparse base model. Inspired by this fact, we suggest communicating only EBTN parameters between clients and servers in FL after collecting each client's EBTN during each communication round. After adding all of the clients' EBTN, the server displays the modified EBTN parameters to each client. Finally, each client will learn a tailored model, not a shared global model. The EBTN includes data-dependent features because it's built by trimming the underlying model using local client data. One client's EBTN may not overlap with others when non-IID data is included. After the server completes the aggregate, each EBTN's customization is preserved. Due to the lower EBTN, the required model parameters are also smaller. So, FL's communication efficacy can be boosted.

Our contribution can be summarized as follow:

1. We propose a revolutionary FL framework, namely, EB-FedAvg, that can achieve personalization, more effective training, and effective communication in both IID and non-IID settings;
2. We conduct experiments to compare EB-FedAvg with standalone, FedAvg, Per-FedAvg, and LG-FedAvg [1, 16, 18]. The results of the experiments show that EB-FedAvg is much better than the other methods in terms of personalization, cost of communication, and effectiveness of training.

2 Related Works

Personalization. To achieve personalization, the global model must be modified due to statistical heterogeneity (i.e., the distribution of non-IID data between clients). Personalization is accomplished in existing work by meta-learning, multi-task learning, transfer learning, etc. [4–10]. However, all of the current attempts to achieve personalization through two distinct phases come with additional overhead: 1) A federated global model is learned, and 2) the global model is tailored to each client based on local data.

The Winning Ticket Theory. According to the lottery ticket hypothesis, a fully trained dense network can be pruned to identify a small subnetwork called the winning ticket [12]. By training the isolated winning ticket with the same weight initialization as the dense network’s corresponding weights, the dense network can then be trained to have a test accuracy that is comparable to that of the isolated subnetwork. Finding winning tickets requires pricey (iterative) pruning and retraining, though. Morcos et al. investigate the transferability of winning tickets across several datasets [19].

Communication Efficiency. The main barrier for FL is communication because the links between clients and the server frequently function at low rates and can be expensive. Several studies try to lower FL’s communication expenses [14, 15]. By combining FedAvg with data compression methods like sparsification, quantization, sketching, etc., the main goal is to reduce the amount of data that is sent between the server and clients.

Efficient Inference and Training. Model compression has been thoroughly investigated for the inference that it is lighter weight. Popular methods include pruning, weight factorization, weight sharing, quantization, dynamic inference, and network architecture search [20–32]. On the other hand, it seems like there is significantly less research on effective training. A small number of studies focus on reducing the total amount of time spent training in situations where people are working in parallel and communicating well [35–37].

3 Design of EB-fedAvg

Early-Bird FedAvg (EB-FedAvg) is an end-to-end combination of Early-Bird Tickets and FedAvg. Figure 1 illustrates an overview of EB-FedAvg. By applying for the Early-Bird Tickets, each participating client discovers an Early-Bird Ticket Network (EBTN). In particular, the EBTN is trained by trimming the base model using the local data of each client. The base model will not be transmitted between the clients and the server, only the parameters of EBTNs. The server will only do the aggregate on the incoming EBTNs after that, and each client will receive the updated parameters for the corresponding EBTNs as a result. After the EBTNs’ parameters have been updated, the clients resume their training. Before we explain how to learn local EBTNs and run the global aggregate, we define the following notations that we use in our paper.

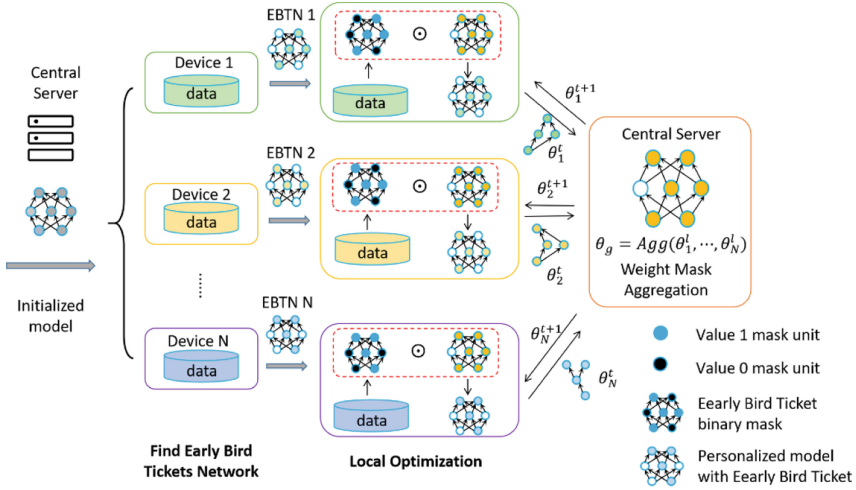


Fig. 1. The EB-FedAvg workflow diagram is displayed. It makes use of Early Bird Tickets to identify the Early Bird Network and, as a result, establish the network structure for Early Learning. Second, sparse weight masks are employed to keep the client personalized during model updates and to increase the effectiveness of data transmission between the client and the server.

Notations: We define $S \subset C$ as a group of clients chosen at random during each training cycle, with $C = \{C_1, \dots, C_N\}$ representing the N available clients, where C_i signifies the i -th client. S_t denotes the group of clients selected in the t -th round. K denotes the sampling ratio of each round. Let $\theta_i (i \neq g)$ represent the local model parameters on each client C_i , and let θ_g represent the parameters of the base model on the global server. To represent the model parameters of the Early-Bird Ticket Network (EBTN) found in round t , we additionally use the superscript t , θ_i^t . And creates a local early bird ticket binary mask $m_i^t \in \{0, 1\}^{|\theta_i^t|}$. Therefore, the parameters of the relevant weight mask for the client C_i are indicated by the symbol $\theta_i^t \odot m_i^t$. Given the data D_i held by C_i . we split D_i into the training data D_i^{Train} , and test data D_i^{Test} . Pruning probability is p , and the scaling factor for structured pruning is r . The FIFO queue Q has a length of l .

3.1 Training Algorithm

The main distinction between EB-FedAvg and FedAvg is that EBTNs are the only form of communication between clients and the FL server. In each communication round, the EBTNs alone are therefore the only ones used in the server’s aggregate. In Algorithm 1, the specifics of the EB-FedAvg training algorithm are described. In general, the training algorithm has the following steps:

Algorithm 1: Training Algorithm of EB-FedAvg

Data: (D_1, \dots, D_N) where D_i is the local data on C_i

Server Executes:

- 1 initialize the global model θ_g with θ_0
- 2 $S \leftarrow \{C_1, \dots, C_N\}$
- 3 **for** $C_i \in S$ **in parallel do**
- 4 download θ_g from Global Server
- 5 initialize the client mask m_k^0
- 6 **end for**
- 7 **for** each round $t=1,2,\dots$ **do**
- 8 $k \leftarrow \max(N \times K, 1)$
- 9 $S_t \leftarrow \{C_1, \dots, C_k\}$
- 10 **for** each client C_k **in parallel do**
- 11 $\theta_k^{t+1} \leftarrow \text{ClientUpdate}(C_k, \theta_g^t)$
- 12 **end for**
- 13 $\theta_g^{t+1} \leftarrow (\text{aggregate EBTNS}\{\theta_k^{t+1}\})$
- 14 **end for**
- 15 **ClientUpdate** (C_k, θ_g^t) :
- 16 $\theta_k^t = \theta_g^t \odot m_k^t$
- 17 $\mathcal{B} \leftarrow$ (split local data D_k^{Train} into batches)
- 18 **for** each local epoch, i from 1 to Epoch **do**
- 19 **for** batch $b \in \mathcal{B}$ **do**
- 20 $\theta_k^{t+1} \leftarrow \theta_k^t - \eta \nabla_{\theta_k^t} \ell(\theta_k^t, b)$
- 21 **end for**
- 22 Perform structured pruning based on r towards the target ratio p , and generate the mask m_k^t ;
- 23 Calculate the **mask distance** between the current and last subnetworks and add to Q ;
- 24 **if** $\text{Max}(Q) < \epsilon$ **then**
- 25 Updating the early bird ticket binary mask m_k^{t+1}
- 26 return $\theta_k^{t+1} \odot m_k^{t+1}$ to server
- 27 **end if**
- 28 **end for**
- 29 return $\theta_k^{t+1} \odot m_k^t$ to server

Step I: Send the global server initialization weights θ_g to the client C_i and initialize the client C_i 's mask m_k^0 .

Step II: The server randomly selects a group of clients S_t given the t -th communication round.

Step III: From the server, each $C_k \in S_t$ client downloads its matching EBTN θ_k^t , where $\theta_k^t = \theta_g^t \odot m_k^t$.

Step IV: Each client C_k start training the local model with θ_k^t . To create a mask m_k^t , structured pruning using the scaling factor r and the pruning probability p . And to store the successively generated subnetworks into a first-in-first-out (FIFO) queue Q with a length of $l = 5$, and calculate the mask m_k^t distances between them.

Step V: Exits when the greatest mask m_k^t distance in the FIFO falls below a predetermined criterion (default 0.1 with normalized distances of $[0,1]$).

3.2 Structured Pruning

Since it is hardware-friendly and best connects to our objective of effective training and performance enhancement, we use the same channel pruning as Liu et al. [21]. To use the scaling factor r in batch normalization (BN) layers as an indicator of channel importance, we follow Liu et al. We generate a mask for each filter separately, and filter channels are pruned under the pruning threshold. To make the pruning threshold easy to use, a percentile of all scaling factors, such as $p\%$ of channels, is used to figure it out.

3.3 Early Bird Ticket Masked Weights

The main distinction between EB-FedAvg and existing FL methods is that only the learned weight masks are sent to the central server by each device, which learns an EBTN that only belongs to itself. Figure 2 illustrates how structured pruning can be used to obtain each device’s respective EBTN. We can obtain the pruned network mask in response, which we refer to as the Early Bird Ticket Binary Mask (EBTB Mask). To send the mask weights to the central server, we can then use the learned EBTB Mask to perform a real-time structured pruning of the EBTN.

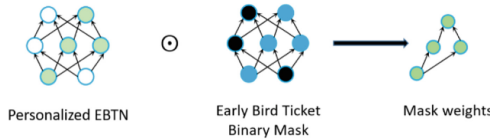


Fig. 2. The process of mask weight. Using the EBTN network, which can be acquired after training or updating, we can produce the corresponding early bird ticket binary masks. A mask calculation between the early bird ticket binary mask and the EBTN network can be used to determine the weight mask.

3.4 Aggregate Heterogeneous Weight Masks

The majority of FL methods aggregate data using the FedAvg aggregation strategy, which involves averaging. Instead of updating the full weights in EB-FedAvg, aggregation is done on the weight masks. Additionally, not all elements are overlapped due to the heterogeneity of the weight masks across devices. Because of this, we can’t just use an averaging strategy to do aggregation in EB-FedAvg.

Our objective in developing the aggregation strategy is to maximize the retention of personalized data contained in the heterogeneous weight masks. We propose a weighted mask aggregation scheme that accomplishes this goal by independently aggregating each element of the weight mask. As Fig. 3 shows, only elements that appear in two or more weight masks are averaged by the central server. The central server then uses the aggregated values to update these elements in the corresponding weight masks. The central server ignores elements that are not shared in the weight masks and do not aggregate them. Finally, the device will receive the updated weight mask.

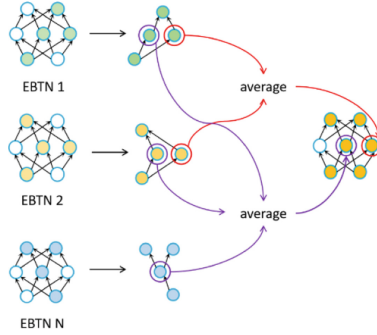


Fig. 3. Weight mask aggregation process. To the server, each device sends its unique weight mask. On the server side, overlapping nodes are aggregated rather than using a strategy that distributes weights evenly as FedAvg does.

3.5 Generate Personalized Model

The weight mask is updated once the weight mask aggregation operation is completed, as shown in Fig. 4. After the weight masks have finished training, the EBTN mask of each device is applied to the updated weight masks one at a time to create a unique model. The EBTN mask makes it more likely that devices will be able to share information while also giving a sparse, personalized model and resolving statistical heterogeneity.

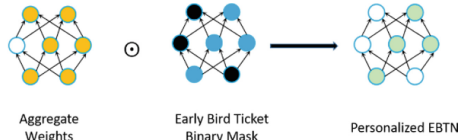


Fig. 4. Personalization via EBTN mask. The client does not update the model immediately after receiving it from the server, unlike FedAvg, in order to preserve the model’s personalized nature. The client will choose the received weights using the early bird ticket binary mask to maintain personalization while ensuring information sharing.

4 Evaluation

4.1 Datasets, and Models

In our investigations, we employ the MNIST and CIFAR-10 datasets [38, 40]. We sample the non-IID dataset using the Dirichlet distribution and allocate it to each client with an alpha of 0.2. All of the test sets for the training dataset labels for each client are used to create the evaluation data. We utilized LeNet-5 and AlexNet as our architectures for MNIST and CIFAR-10. In the LeNet-5 and AlexNet designs, we also include a batch-normalization layer after each convolutional layer.

4.2 Hyper-Parameter Setting

We built up 100 clients with local batch sizes of 32, and 50 for local epochs, and an SGD optimizer with a 0.1 learning rate and 0.9 momentum for all experiments. Between the server and the client, there are 50 communication rounds. For structured pruning methods, the threshold for mask distance is 0.1 as well. Additionally, we used a pruning rate of 0.3 and a scale sparse factor of 10^{-4} .

4.3 Compared Methods

Baselines. To comprehensively evaluate the performance of EB-FedAvg, we compare EB-FedAvg against four baselines:

Standalone: each device trains a model independently using only local data without collaborating with other devices [39]. Be aware that using the Standalone technique won't incur any communication costs.

FedAvg is the most classic FL method, and it is employed in commercial products. Devices talk to the central server to send updated local parameters and download the global model so that local training can happen all the time.

Per-FedAvg adds MAML, a prominent meta-learning approach, with FedAvg for customization [18, 41].

LG-FedAvg is a cutting-edge FL method that lets you customize it and improves the efficiency of communication while decreasing the efficiency of computing [16].

4.4 Evaluation Metrics

To evaluate the performance of EB-FedAvg during the training process, we use the following evaluation metrics:

- (1) **Inference Accuracy:** We assess the inference accuracy of the test data for each device and report the overall average accuracy for evaluations.
- (2) **Communication Cost:** The total number of parameters the model uploads and downloads during training serves as our proxy for the communication cost, which is a significant bottleneck in federal learning.
- (3) **Computation Cost:** We quantify the computation time spent on devices for 50 training rounds.

4.5 Training Performance

Inference Accuracy vs. Computation Cost: we compare EB-FedAvg with the baselines in terms of the accuracy-computation tradeoff. Table 1 shows that EB-FedAvg can improve the accuracy of inferences by a lot while reducing the cost of computations by a lot.

First, compared to LG-FedAvg, EB-FedAvg can improve inference accuracy and training computation cost simultaneously. In particular, EB-FedAvg improves inference accuracy by $1.17\times$, $1.78\times$, $1.11\times$, $1.67\times$ on LeNet-5-MNIST, LeNet-5-CIFAR-10, AlexNet-MNIST, and AlexNet-CIFAR-10 in IID, respectively. Inference accuracy

improves by $2.96\times$, $2.23\times$, $1.79\times$, and $1.96\times$ on LeNet-5-MNIST, LeNet-5-CIFAR-10, AlexNet-MNIST, and AlexNet-CIFAR-10 in non-IID, respectively.

Second, EB-FedAvg can dramatically reduce computation costs compared to Per-FedAvg, which is specifically designed for personalization. In particular, EB-FedAvg reduces the computation costs of LeNet-5-MNIST, LeNet-5-CIFAR-10, AlexNet-MNIST, and AlexNet-CIFAR-10 in IID by $1.79\times$, $1.53\times$, $1.46\times$, and $1.36\times$, respectively. In non-IID, EB-FedAvg reduces computation costs by $1.43\times$ and $1.53\times$ on LeNet-5-CIFAR-10 and AlexNet-CIFAR-10, respectively.

Table 1. Comparison between EB-FedAvg and baselines in inference accuracy-computation cost space.

Models	Algorithm	IID		non-IID	
		Acc.(%)	Computation cost. (s)	Acc. (%)	Computation cost. (s)
LeNet-5 MNIST	EB-FedAvg	96.31	673	89.56	780
	FedAvg	96.1	683	89.32	775
	LG- FedAvg	82.31	1281	30.28	703
	Per- FedAvg	94.79	1204	83.87	695
	Standalone	92.14	225	50.14	283
LeNet-5 CIFAR-10	EB-FedAvg	46.67	794	29.31	839
	FedAvg	46.59	792	27.93	766
	LG- FedAvg	26.2	1409	13.06	1410
	Per- FedAvg	36.2	1214	28.2	1200
	Standalone	38.01	353	24.52	372
AlexNet MNIST	EB-FedAvg	95.09	1435	88.22	1369
	FedAvg	94.48	1436	85.57	1305
	LG- FedAvg	85.79	2143	49.11	1408
	Per- FedAvg	91.69	2097	83.62	1346
	Standalone	90.29	516	49.89	418
AlexNet CIFAR-10	EB-FedAvg	49.75	1292	30.09	1377
	FedAvg	48.42	1312	31.1	1360
	LG- FedAvg	29.73	2007	15.38	2447
	Per- FedAvg	40.77	1762	18.29	2100
	Standalone	36.2	514	24.07	788

Second, EB-FedAvg can dramatically reduce computation costs compared to Per-FedAvg, which is specifically designed for personalization. In particular, EB-FedAvg reduces the computation costs of LeNet-5-MNIST, LeNet-5-CIFAR-10, AlexNet-MNIST, and AlexNet-CIFAR-10 in IID by $1.79\times$, $1.53\times$, $1.46\times$, and $1.36\times$, respectively. In non-IID, EB-FedAvg reduces computation costs by $1.43\times$ and $1.53\times$ on LeNet-5-CIFAR-10 and AlexNet-CIFAR-10, respectively.

Third, it's not surprising that EB-FedAvg does a lot better than FedAvg in terms of how well it makes inferences and how much it costs to compute. FedAvg is a general FL method that isn't optimized for computation or personalization.

Even though Standalone doesn't have any communication costs, EB-FedAvg does a better job than Standalone because it uses all local data instead of only a few training samples on each device.

Low Communication Cost: Fig. 5 illustrates the comparison of communication costs between EB-FedAvg and the baselines.

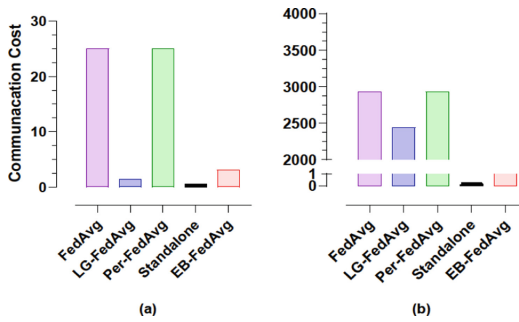


Fig. 5. In terms of communication costs, EB-FedAvg is compared to baselines. LeNet-5 and AlexNet architectures for sending are shown in (a) and (b), respectively. Data sent is measured in megabytes (MB).

As Fig. 5 shows, EB-FedAvg is more communication-efficient in all applications when compared to baselines because of the structured sparsity. Specifically, EB-FedAvg can save $8\times$, $8.06\times$, and per-FedAvg on LeNet communication costs, respectively. EB-FedAvg can save $27\times$, $27.86\times$, and per-FedAvg on AlexNet communication costs, respectively. EB-FedAvg can save $0.47\times$ and $23.18\times$ in communication costs on LeNet-5-LG-FedAvg and AlexNet-Per-FedAvg, respectively.

The number of Participating Devices: Table 2 shows the BE-FedAvg, which we use to figure out how adding more devices affects how well each communication round works.

We experiment on MNIST and CIFAR10 and change the quantities of participating devices by 5, 20, and 40. As Table 2 illustrates, with more devices participating in each communication round, the inference accuracy marginally improves. For instance, increasing the number of participating devices from 5 to 40 on IID and non-IID enhances the inference accuracy on MNIST-LeNet-5 by 1% and 1.02%, respectively. When the number of devices in IID and non-IID goes from 5 to 40, the inference accuracy on CIFAR-10-LeNet-5 goes up by 1.03% and 1.11%, respectively.

Table 2. The impact of the number of participating devices on EB-FedAvg performance.

Models	Number of devices	IID Acc. (%)	non-IID Acc. (%)
MNIST-LeNet-5	5	96.31	89.56
	20	96.28	90.69
	40	96.49	91.74
CIFAR-10-LeNet-5	5	46.67	29.31
	20	48.78	31.36
	40	47.93	32.57

Data Imbalance Ratio: The amount and type of data on the device have a big effect on how well the FL method works. In practice, there are several bad situations when the data amount is constrained. In addition to having limited data, data on a device frequently displays an imbalance between different classifications. It is challenging for FL strategies to train customized models that perform as well across classes as they do within them. To figure out how the amount of data and the degree of different data types affect the way EB-FedAvg works. We conducted experiments on the MNIST and CIFAR-10 datasets to compare the performance of both datasets on LeNet and AlexNet. The Dirichlet distribution’s alpha value, whose greater value roughly equates to a more uniform distribution of data types and amounts among clients, allows us to control the data imbalance rate. The alpha values selected are 0.0001, 0.001, 0.01, and 0.1. As Fig. 6 illustrates, for a fixed number of participating training clients, the accuracy of an inference decreases slightly for smaller alpha values. With the participation of 20 clients,

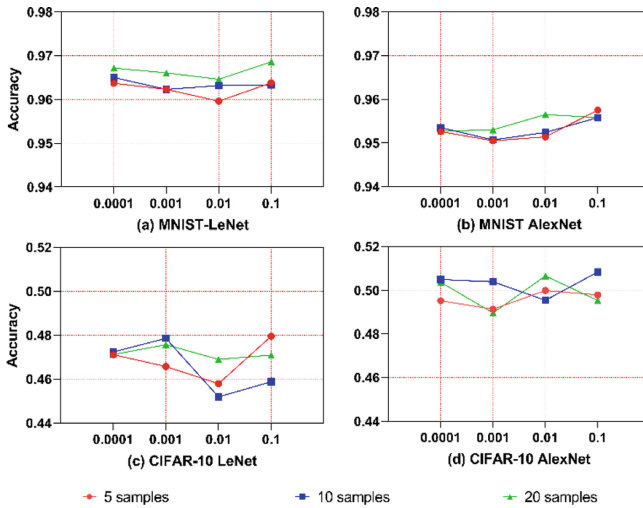


Fig. 6. The impact of the data imbalance rate on EB-FedAvg performance. Three factors are combined in this: data heterogeneity, device heterogeneity, and model heterogeneity. Data heterogeneity refers to data with various balance rates; device heterogeneity refers to the various training devices, and model heterogeneity refers to the utilization of various two models.

the value in Fig. 6(a) dropped from 96.86% to 96.72% when the alpha dropped from 0.1 to 0.0001. In several additional comparisons, the situation is similar. In addition, the increase in the number of clients participating in the training when setting the same alpha value can reduce the impact of the data imbalance degree, in addition to improving the effectiveness of the training. In Fig. 6(a), for example, the accuracy varies between 2% and 4%.

5 Conclusion

We created EB-FedAvg, a customized, effective training and communication FL framework that is motivated by Early-Bird Tickets. The technique removes the generic model parameters from clients' models while maintaining the customized ones by iteratively pruning the neural network channels. Our results show that obtaining the winning ticket at a very early stage, i.e., EB, can achieve the same or better performance than standard training and other personalized architectures. The EBTN obtained through EB-FedAvg not only ensures personalized training but also greatly reduces the consumption during the communication process in the transmission of FL. We assume there are still a lot of promising issues that need to be solved. Testing low-precision EB Train methods on larger models and datasets is an immediate future task. We are also interested in finding out if EB Train could be connected to any further less expensive training methods.

References

1. McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Artificial Intelligence and Statistics, pp. 1273–1282 (2017)
2. Hard, A., Rao, K., Mathews, R., Ramaswamy, S., Beaufays, F., Augenstein, S., Ramage, D.: Federated learning for mobile keyboard prediction. arXiv preprint [arXiv:1811.03604](https://arxiv.org/abs/1811.03604). (2018)
3. Li, T., Sahu, A.K., Talwalkar, A., Smith, V.: Federated learning: Challenges, methods, and future directions. *IEEE Signal Process. Mag.* **37**(3), 50–60 (2020)
4. Jiang, Y., Konečný, J., Rush, K., Kannan, S.: Improving federated learning personalization via model agnostic meta learning. arXiv preprint [arXiv:1909.12488](https://arxiv.org/abs/1909.12488). (2019)
5. Khodak, M., Balcan, M.F.F., Talwalkar, A.S.: Adaptive gradient-based meta-learning methods. In: Advances in Neural Information Processing Systems, 32 (2019)
6. Chen, F., Luo, M., Dong, Z., Li, Z., He, X.: Federated meta-learning with fast convergence and efficient communication. arXiv preprint [arXiv:1802.07876](https://arxiv.org/abs/1802.07876). (2018)
7. Smith, V., Chiang, C.K., Sanjabi, M., Talwalkar, A.S.: Federated multi-task learning. In: Advances in Neural Information Processing Systems, 30 (2017)
8. Zantedeschi, V., Bellet, A., Tommasi, M.: Fully decentralized joint learning of personalized models and collaboration graphs. In: International Conference on Artificial Intelligence and Statistics, pp. 864–874 (2020)
9. Wang, K., Mathews, R., Kiddon, C., Eichner, H., Beaufays, F., Ramage, D.: Federated evaluation of on-device personalization. arXiv preprint [arXiv:1910.10252](https://arxiv.org/abs/1910.10252). (2019)
10. Mansour, Y., Mohri, M., Ro, J., Suresh, A.T.: Three approaches for personalization with applications to federated learning. arXiv preprint [arXiv:2002.10619](https://arxiv.org/abs/2002.10619). (2020)
11. Han, S., Mao, H., Dally, W.J.: Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint [arXiv:1510.00149](https://arxiv.org/abs/1510.00149). (2015)

12. Frankle, J., Carbin, M.: The lottery ticket hypothesis: Finding sparse, trainable neural networks. arXiv preprint [arXiv:1803.03635](https://arxiv.org/abs/1803.03635). (2018)
13. Liu, Z., Sun, M., Zhou, T., Huang, G., Darrell, T.: Rethinking the value of network pruning. arXiv preprint [arXiv:1810.05270](https://arxiv.org/abs/1810.05270). (2018)
14. Konečný, J., McMahan, H.B., Yu, F.X., Richtárik, P., Suresh, A.T., Bacon, D.: Federated learning: Strategies for improving communication efficiency. arXiv preprint [arXiv:1610.05492](https://arxiv.org/abs/1610.05492). (2016)
15. Alistarh, D., Grubic, D., Li, J., Tomioka, R., Vojnovic, M.: QSGD: Communication-efficient SGD via gradient quantization and encoding. In: *Advances in Neural Information Processing Systems*, 30 (2017)
16. Liang, P.P., et al.: Think locally, act globally: Federated learning with local and global representations. arXiv preprint [arXiv:2001.01523](https://arxiv.org/abs/2001.01523). (2020)
17. You, H., et al.: Drawing early-bird tickets: Towards more efficient training of deep networks. arXiv preprint [arXiv:1909.11957](https://arxiv.org/abs/1909.11957). (2019)
18. Fallah, A., Mokhtari, A., Ozdaglar, A.: Personalized federated learning: A meta-learning approach. arXiv preprint [arXiv:2002.07948](https://arxiv.org/abs/2002.07948). (2020)
19. Morcos, A., Yu, H., Paganini, M., Tian, Y.: One ticket to win them all: generalizing lottery ticket initializations across datasets and optimizers. In: *Advances in Neural Information Processing Systems*, 32 (2019)
20. Li, H., Kadav, A., Durdanovic, I., Samet, H., Graf, H.P.: Pruning filters for efficient convnets. arXiv preprint [arXiv:1608.08710](https://arxiv.org/abs/1608.08710). (2016)
21. Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., Zhang, C.: Learning efficient convolutional networks through network slimming. In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2736–2744 (2017)
22. He, Y., Kang, G., Dong, X., Fu, Y., Yang, Y.: Soft filter pruning for accelerating deep convolutional neural networks. arXiv preprint [arXiv:1808.06866](https://arxiv.org/abs/1808.06866). (2018)
23. Wen, W., Wu, C., Wang, Y., Chen, Y., Li, H.: Learning structured sparsity in deep neural networks. In: *Advances in Neural Information Processing Systems*, 29 (2016)
24. Luo, J. H., Wu, J., Lin, W.: Thinet: A filter level pruning method for deep neural network compression. In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5058–5066 (2017)
25. Liu, S., Lin, Y., Zhou, Z., Nan, K., Liu, H., Du, J.: On-demand deep model compression for mobile devices: A usage-driven model selection framework. In: *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*, pp. 389–400. (2018)
26. Denton, E.L., Zaremba, W., Bruna, J., LeCun, Y., Fergus, R.: Exploiting linear structure within convolutional networks for efficient evaluation. In: *Advances in Neural Information Processing Systems*, 27 (2014)
27. Wu, J., Wang, Y., Wu, Z., Wang, Z., Veeraraghavan, A., Lin, Y.: Deep k-means: Re-training and parameter sharing with harder cluster assignments for compressing deep convolutions. In: *International Conference on Machine Learning*, pp. 5363–5372. (2018)
28. Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., Bengio, Y.: Quantized neural networks: training neural networks with low precision weights and activations. *J. Mach. Learn. Res.* **18**(1), 6869–6898 (2017)
29. Wang, Y., Nguyen, T., Zhao, Y., Wang, Z., Lin, Y., Baraniuk, R.: Energynet: Energy-efficient dynamic inference (2018)
30. Wang, Y., et al.: Dual dynamic inference: Enabling more efficient, adaptive, and controllable deep inference. *IEEE J. Selected Topics Signal Process.* **14**(4), 623–633 (2020)
31. Shen, J., Wang, Y., Xu, P., Fu, Y., Wang, Z., Lin, Y.: Fractional skipping: Towards finer-grained dynamic cnn inference. In: *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34(4), pp. 5700–5708 (2020)

32. Zoph, B., Le, Q.V.: Neural architecture search with reinforcement learning. arXiv preprint [arXiv:1611.01578](https://arxiv.org/abs/1611.01578). (2016)
33. Wang, Y., Xu, C., Xu, C., Xu, C., Tao, D.: Learning versatile filters for efficient convolutional neural networks. In: Advances in Neural Information Processing Systems, 31 (2018)
34. Wang, Y., Xu, C., Xu, C., Tao, D.: Packing convolutional neural networks in the frequency domain. *IEEE Trans. Pattern Anal. Mach. Intell.* **41**(10), 2495–2510 (2018)
35. Goyal, P., et al.: Accurate, large minibatch sgd: Training imagenet in 1 hour. arXiv preprint [arXiv:1706.02677](https://arxiv.org/abs/1706.02677). (2017)
36. Cho, M., Finkler, U., Kumar, S., Kung, D., Saxena, V., Sreedhar, D.: Powerai ddl. arXiv preprint [arXiv:1708.02188](https://arxiv.org/abs/1708.02188). (2017)
37. You, Y., Zhang, Z., Hsieh, C.J., Demmel, J., Keutzer, K.: Imagenet training in minutes. In: Proceedings of the 47th International Conference on Parallel Processing, pp. 1–10. (2018)
38. Deng, L.: The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Process. Mag.* **29**(6), 141–142 (2012)
39. Ang, L., Jingwei, S., Xiao, Z., Mi, Z., Hai, L., Yiran, C.: Fedmask: Joint computation and communication-efficient personalized federated learning via heterogeneous masking. In: Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems (2021)
40. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images (2009)
41. Finn, C., Abbeel, P., Levine, S.: Model-agnostic meta-learning for fast adaptation of deep networks. In: International Conference on Machine Learning, pp. 1126–1135. (2017)