Once-for-All Federated Learning: Learning From and Deploying to Heterogeneous Clients

Kamala Varma
Amazon
Cambridge, MA, USA
kvarma@amazon.com

Enmao Diao
Amazon
Cambridge, MA, USA
emdiao@amazon.com

Tanya G. Roosta
Amazon
Sunnyvale, CA, USA
troosta@amazon.com

Jie Ding
Amazon
Minneapolis, MN, USA
jiedi@amazon.com

Tao Zhang
Amazon
Cambridge, MA, USA
taozhng@amazon.com

ABSTRACT

Federated learning (FL) enables multiple client devices to train a single machine learning model collaboratively. As FL often involves various smart devices, it is important to adapt the FL pipeline to accommodate device resource constraints. This work addresses the problem of training and storing memory-intensive deep neural network architectures on resource-constrained devices. Existing solutions often involve computationally expensive methods. We propose Once-for-All Federated Learning (OFA-FL) to overcome this limitation by learning a model that concurrently optimizes sub-networks of various sizes. Clients can therefore receive the sub-network best suited for their device resources without extra computation. Our experiments show that each component of OFA-FL contributes to well-performing FL-produced sub-networks while maintaining a global network design that supports the efficient deployment of device resource-specific sub-networks.

CCS CONCEPTS

• **Computing methodologies** → *Ensemble methods*; *Neural networks*; • **Computer systems organization** → **Client-server architectures**;

KEYWORDS

federated learning, Internet of Things, heterogeneity

ACM Reference Format:
Kamala Varma, Enmao Diao, Tanya G. Roosta, Jie Ding, and Tao Zhang. 2023. Once-for-All Federated Learning: Learning From and Deploying to Heterogeneous Clients. In *KDD FL4Data-Mining '23, August 7, 2023, Long Beach, CA, USA.*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
KDD FL4Data-Mining '23, August 7, 2023, Long Beach, CA, USA
© 2023 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Federated learning (FL) is a distributed learning paradigm that enhances privacy by not requiring training data to leave client devices [9, 13, 19]. In FL, a central server keeps a global model whose parameters are updated using an aggregate of locally-trained client model parameters [18]. Federated Averaging (FedAvg) [19] is a sample size-weighted averaging algorithm that is commonly used in practice for aggregating local parameters and it will be used in our approach. After each global model update, the server sends the new parameters to the clients [19]. However, many client devices cannot accommodate them due to limited storage and computational resources. In addition, the global model is sometimes too big to be deployed on memory-constrained devices that wish to receive the final learned model [1].

To address this problem, we introduce Once-for-All Federated Learning (OFA-FL) (Figure 1), which adapts the concept of Once-for-All (OFA) networks from the centralized setup in which they were initially introduced [3] into an FL setting. OFA networks concurrently optimize sub-networks contained within the global architecture. OFA networks use a progressive shrinking algorithm to jointly optimize sub-networks during training, which involves an efficient, generalized pruning method for realizing sub-architectures [8]. We translate this training approach to an FL system by employing a progressive Lottery Ticket Pruning (LTP) strategy and by formulating a local training process that jointly optimizes sub-networks [7, 16]. Our contributions include the following: 1) We present a new method of globally applying LTP to an FL system to produce a series of parameter masks, which can be used to instantaneously extract sub-networks for clients with different capacities. 2) We adapt the loss function for centralized OFA networks to the local training process of FL to jointly optimize sub-networks within a global network. 3) We provide numerical studies to show the promising performance of the proposed approach.

Related work. The prior work most relevant to our approach is the concept of the Once-for-All (OFA) network, which solves the problem of achieving efficient inference across devices with diverse resource constraints [3]. Through decoupling the training process and the process of specializing the full network into sub-networks, OFA networks avoid the costly approach of separately re-training specialized networks after the architectures are established. We

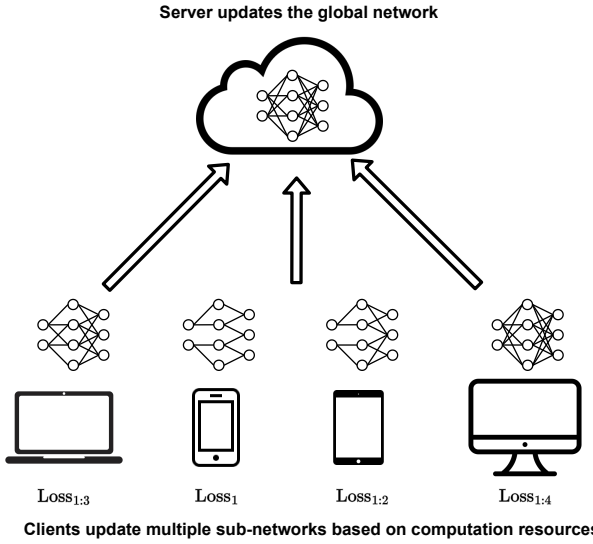


Figure 1: Illustration of Once-for-All Federated Learning.

utilize two key components of the OFA progressive shrinking algorithm: 1) the joint optimization approach that allows clients to train sub-networks of different sizes locally, 2) a generalized progressive pruning strategy based on LTP [7] that progressively looks for sparse sub-networks to train more effectively. A variety of FL methods have been proposed that in some way involve compressed sub-architectures being utilized by clients [2, 4, 6, 10–12, 16, 17, 20]. Their main shortcomings that OFA-FL is designed to overcome include imposing extra model-specialization costs on the clients, either skipping or using expensive methods for specialization, not catering to client-specific constraints, not concurrently optimizing all sub-networks, and pruning in a one-shot (as opposed to progressive) or an inflexible manner.

2 METHOD

In this section, we walk through the full OFA-FL training pipeline, whose pseudocode is provided in Algorithm 1, which involves helper functions that are detailed in Algorithms 2, 3, and 4.

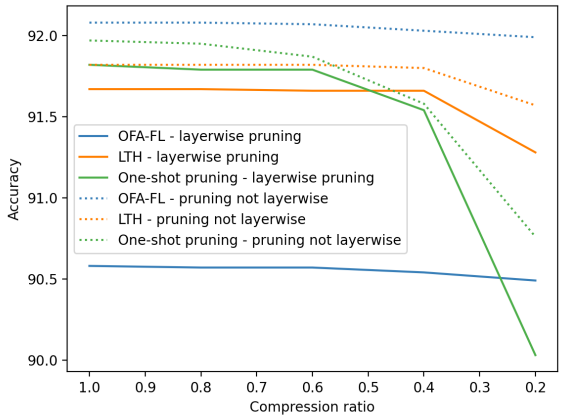
Setup and Initialization. We assume the following two hyperparameters are predetermined. First, a static list of compression ratios, $\mathcal{R} = [1.0, r_{s-2}, r_{s-3}, \ldots, r_0]$, where each r_j is one of s ratios of total global parameters associated with a sub-network that the FL system will support. Second, a list, $C = [c_0, c_1, \ldots c_i, \ldots, c_n]$, where each c_i corresponds to the maximum ratio, chosen from $\mathcal{R}$, of global model parameters that client i (of n total clients) can accommodate locally. The following values are initialized at the server (line 1 of Algorithm 1). $\mathcal{M}_g$ represents the global OFA model that the FL system will learn, where θ_g denotes the global parameters of the model, whose initial values are stored in θ_{initial}. $\mathcal{M}$ and m are both dictionaries of key-value pairs where each key is a compression ratio that has been pruned for. The value for m is the 0-1 mask, m_r, associated with the ratio, r, and the value for $\mathcal{M}$ is the associated masked model, $f(x; \theta_r)$. Each mask is a list of tensors, with each tensor corresponding to a layer of parameters from θ_g. For an individual layer, k, parameters are specified by θ_k, mask parameters are specified by m_k, and each parameter within θ_k can be indexed by l (i.e. $\theta_k[l]$).

Algorithm 1: Full Pipeline

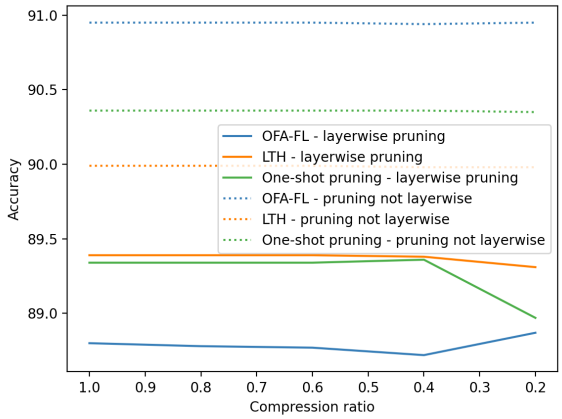
This is executed by the server in an FL system with n clients. The global OFA model will support s sub-networks of different sizes. The server is provided with a list, $\mathcal{R} = [1.0, r_{s-2}, r_{s-3}, \ldots, r_0]$, where each r_j corresponds to a ratio of total global parameters that the FL system will prune sub-networks for, and a list, $C = [c_0, c_1, \ldots, c_n]$, where each c_i corresponds to the maximum ratio (from $\mathcal{R}$) of global model parameters that client i can accommodate.

1 $\theta_g \leftarrow$ random initialization, $\theta_{\text{initial}} \leftarrow \theta_g$, $\mathcal{M}_g \leftarrow f(x; \theta_g)$, $\mathcal{M} \leftarrow \{\}$, $m \leftarrow \{\}$
2 **for** *each* r *in* $\mathcal{R}$ **do**
3 $m_r \leftarrow$ copy of θ_g with all values set to 1
4 $m[r] = m_r$
5 $\mathcal{R}_{\text{pruned}} \leftarrow [1.0]$
6 $n_{\text{pruned}} \leftarrow 0$
7 **for** *each global epoch* $i = 0, 1, \ldots, \epsilon - 1$ **do**
8 **if** $n_{pruned} < s$ **then**
9 **if** *ratio* $\mathcal{R}_{pruned}[n_{pruned}]$ *model has converged* **then**
10 increment n_{pruned}
11 append $\mathcal{R}[n_{\text{pruned}}]$ to $\mathcal{R}_{\text{pruned}}$
12 $r_{\text{relative}} \leftarrow 1 - \frac{\mathcal{R}_{\text{pruned}}[n_{\text{pruned}}] - \mathcal{R}_{\text{pruned}}[n_{\text{pruned}}-1]}{\mathcal{R}_{\text{pruned}}[n_{\text{pruned}}-1]}$
13 $m[\mathcal{R}[n_{\text{pruned}}]] \leftarrow$ $Prune(\theta_{\text{initial}}, m[\mathcal{R}[n_{\text{pruned}} - 1]] * \theta_g, r_{\text{relative}})$
14 $\theta_g \leftarrow Reinitialize(\theta_{\text{initial}}, \theta_g, m[\mathcal{R}[n_{\text{pruned}}]])$
15 $\mathcal{M} \leftarrow ApplyMasks(\theta_g, \mathcal{R}_{\text{pruned}}, m)$
16 $\mathcal{M}_{\text{local}} \leftarrow []$
17 **for** *each client index* $i = 0, \ldots, n$ **do**
18 **if** $C[i]$ *in* $\mathcal{R}_{pruned}$ **then**
19 append $InitLocal(C[i], m, M[C[i]], \mathcal{R}_{\text{pruned}})$ to $\mathcal{M}_{\text{local}}$
20 $\Theta \leftarrow []$
21 **for** *each local model index* $i = 0, \ldots, length(\mathcal{M}_{local})$ **do**
22 append $LocalUpdate(i, \mathcal{M}_{\text{local}}[i][0], \mathcal{M}_{\text{local}}[i][1])$ to Θ
23 $\theta_g \leftarrow$ Aggregate(Θ)
24 $\mathcal{M}_g \leftarrow f(x; \theta_g)$

Progressive Lottery Ticket Pruning. To start each of ϵ total global epochs, pruning might occur. The progressive shrinking algorithm used in centralized OFA networks ensures that smaller sub-network parameters will not interfere with larger sub-network parameters and gives the compressed networks a head start on training. We incorporate these components of progressive shrinking into our OFA-FL method through progressive LTP. In the 'Prune' function (invoked in line 13 of Algorithm 1 and implemented in Algorithm 2),



(a) Accuracies from the global epoch with maximized average accuracy across all ratios.



(b) Accuracies achieved after 75 global epochs of training.

Figure 2: The accuracy on the MNIST classification task of sub-networks of different compression ratios produced by either OFA-FL, LTH pruning, or one-shot pruning, either layerwise pruning (pruning executed separately for each network layer) or non-layerwise pruning (pruning simultaneously executed across the entire network), and an MLP architecture.

Algorithm 2: Lottery Ticket Pruning
LTP executed by the server.

```
def Prune(θ_initial, θ, r):
    α ← 100 * (1 − r) percentile value of nonzero
      parameters in θ
    m ← []
    for each layer θ_k in θ do
        m_k ← copy(θ_k)
        for each parameter index l in layer θ_k do
            if θ_k[l] ≥ α then
                m_k[l] ← 1
            else
                m_k[l] ← 0
            append m_k to m
    return m
```

Algorithm 3: Additional Server Methods
Additional helper methods executed by the server.

```
def InitLocal(ratio, m, M_i, R_pruned):
    m_i ← []
    for r in R_pruned do
        if r < ratio then
            append m[r] to m_i
    return (M_i, m_i)
def Reinitialize(θ_initial, θ_g, mask):
    for each layer index k in θ_g do
        for each parameter index l in layer θ_g[k] do
            if mask[k][l] is 1 then
                θ_g[k][l] ← θ_initial[l]
    return θ_g
def ApplyMasks(θ_g, R_pruned, m, M):
    for each r in R_pruned do
        θ_r ← m[r] * θ_g
        M[r] ← f(x; θ_r)
    return M
def Aggregate(Θ):
    θ ← (Σ Θ) / |Θ|
    return θ
```

the server progressively prunes by starting with the full network and initially pruning it only to the next smallest sub-network size, i.e., ratio $\mathcal{R}[1]$. For convenience, the server keeps track of ratios that have been pruned for in a list, $\mathcal{R}_{\text{pruned}}$. For an arbitrary round of pruning, round n_{pruned}, the ratio $\mathcal{R}[n_{\text{pruned}} - 1]$ sub-network is pruned to result in a mask for the $\mathcal{R}[n_{\text{pruned}}]$ network. A relative ratio, r_{relative}, will be computed between the two networks' ratios. It is used to compute a threshold, α, and parameters with magnitudes less than α will be pruned from the $\mathcal{R}[n_{\text{pruned}} - 1]$ network to achieve the $\mathcal{R}[n_{\text{pruned}}]$ network. Each round of pruning can occur either after we determine that the current smallest sub-network being supported has been sufficiently trained or after some predetermined number of epochs. We consider the network sufficiently trained if it has converged, meaning either the evaluation loss from the current round has increased from the previous round or it has only improved by some small predefined threshold. The parameters of the global network that have not been pruned

Algorithm 4: Local Once-for-All Training

Executed by each participating client. Each client i will each have a local dataset, D_i, split into batches d, and a list of masks, m_i, corresponding to all sub-networks smaller than the client's designated sub-networks. Clients train for ϵ_{local} local epochs and use learning rate η.

```
def LocalUpdate(i, f(x;θ_i), m_i):
    for ε_local epochs do
        for each d in D_i do
            L_d = [ℓ(d;θ_i)]
            for each m_{i,j} in m_i do
                append ℓ(d; m_{i,j} * θ_i) to L_d
            ℓ_avg ← average(L_d)
            θ_i ← θ_i − η∇ℓ_avg
    return θ_i
```

in the current round will be restored to the values that they were initialized with before the start of the training. This will prompt the clients to fine-tune this new smallest sub-network by training it from scratch yet incorporating only the parameters that have been proven important.

Local Once-for-All Training. We adapt the joint optimization from centralized OFA networks to the local training step in OFA-FL (the 'LocalUpdate' function invoked in line 22 of Algorithm 1 and implemented in Algorithm 4). Clients participate in training once the global OFA network supports their designated sub-network size. The server provides clients with tuples that contain their designated sub-network, $\mathcal{M}_i$, and a list, m_i, of all masks corresponding to all ratios smaller than the client's designated ratio. The server keeps all tuples in a list, $\mathcal{M}_{\text{local}}$. The local OFA training process executes for ϵ_{local} epochs per global epoch. Client i will forward-pass their local training dataset, D_i, through their designated sub-network, $f(x;\theta_i)$. For each mini-batch, d, they will compute a loss value with the output of the forward pass using some loss function, $\ell(\cdot)$, and use this loss to initialize a list of losses, $\mathcal{L}_d$. Then, the client will apply each of the masks in m_i to its designated sub-network. The same mini-batch will be used with each of these masked sub-networks to compute new loss values that are appended to $\mathcal{L}_d$. This is why, in Figure 1, the client that has the least computational resources and therefore holds the smallest sub-network is only considering a single loss value. The clients associated with the larger sub-networks consider loss values associated with their own sub-network size in addition to the losses of all smaller sub-network sizes. The gradient of the average of $\mathcal{L}_d$ is used to update the parameter values in θ_i. This updated list of parameters is returned to the server for aggregation.

Aggregation. For aggregation, each parameter contained in the full global model architecture's parameter matrix, θ_g, is updated to be the sample size-weighted average of all corresponding values taken from each local update in the list of local parameters, Θ, that contains the parameter.

3 EXPERIMENTS

Experiment setup. We evaluate the performance of our algorithm through ablation studies. Our approach is unique in that it supports multiple sub-networks simultaneously within a single global network, which is not addressed in existing FL methods. To demonstrate the effectiveness of this key component of our method, we compare the performance of OFA-FL to other state-of-the-art FL algorithms that only involve learning from compressed models. In our studies, we simulate an FL system with 10 client devices. Two clients can accommodate a maximum number of model parameters corresponding to each of the five ratios of total global parameters: 1.0, 0.8, 0.6, 0.4, and 0.2. We use either a convolutional neural network (CNN) with the same architecture as in [4] or a simple multi-layer perception (MLP) architecture to classify MNIST handwritten digits [15]. We use a ResNet50 architecture [21] to classify CIFAR-10 images [14]. We use one local epoch, a 0.0005 learning rate decay, and a 0.9 momentum coefficient.

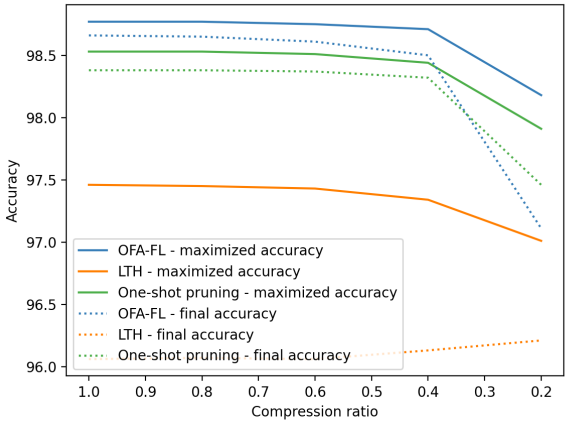


Figure 3: Accuracies from the final epoch and the epoch with maximized average accuracy across all ratios resulting from the use of OFA-FL, LTH pruning, or one-shot pruning with a CNN model and MNIST data.

Experiment results. For MNIST classification, we use a mini-batch size of 10 and an initial learning rate of 0.01. Figure 2 includes plots of the MLP results, and Figure 3 plots the CNN results. They collectively demonstrate the fact that our OFA-FL method generally results in the highest average sub-network accuracy and the lowest variance across the accuracies of different sub-networks. This variance-minimizing effect is important since it shows that the global network is being optimized across all sub-network sizes. Instead of only prioritizing the optimization of the largest sub-network or some subset of the larger sub-networks, which would likely cause the performance of the smaller sub-networks to be severely compromised, OFA-FL is prioritizing optimization across all sub-network sizes simultaneously. This ensures that the smaller sub-networks can still perform comparably well to the larger sub-networks, as opposed to having a high variance among the larger

Table 1: Classifying CIFAR-10 images, waiting for 200 epochs to start pruning, and then waiting for the number of epochs in the pruning frequency column for each subsequent round of pruning. The accuracy reported corresponds to five compression ratios (1.0, 0.8, 0.6, 0.4, 0.2), including final values and values from the epoch with the maximum average accuracy across ratios.

Pruning frequency	Method	Maximized accuracy per ratio					Variance across the five max accuracies	Final accuracy per ratio					Variance across the five final accuracies
		1.0	0.8	0.6	0.4	0.2		1.0	0.8	0.6	0.4	0.2	
50	OFA-FL	34.67	34.68	34.67	34.66	34.65	.0001	34.67	34.68	34.67	34.66	34.65	.0001
	LTP	58.95	58.9	58.79	53.93	45.34	27.89	43.49	43.48	43.45	43.18	41.8	.4227
	One-shot	80.78	80.78	80.72	79.32	66.95	29.26	39.53	39.53	39.53	38.19	32.96	6.489
100	OFA-FL	49.83	49.81	49.8	44.09	37.52	23.79	43.70	43.70	43.70	43.70	43.61	.0012
	LTP	53.72	53.72	53.63	46.04	38.87	35.44	40.27	40.25	40.26	40.11	38.14	.6973
	One-shot	80.78	80.78	80.72	79.32	66.95	29.26	36.4	36.4	36.39	34.66	29.78	6.568

Table 2: Classifying CIFAR-10 images. The accuracy reported corresponds to five compression ratios (1.0, 0.9, 0.8, 0.7, 0.6), including final values and values from the epoch with the maximum average accuracy across ratios.

Method	Maximized accuracy per ratio					Variance across the five max accuracies	Final accuracy per ratio					Variance across the five final accuracies
	1.0	0.9	0.8	0.7	0.6		1.0	0.9	0.8	0.7	0.6	
OFA-FL	76.99	76.99	76.99	76.98	76.96	.0001	76.99	76.99	76.99	76.98	76.96	.0001
LTP	79.72	79.7	79.7	79.65	79.58	.0026	79.17	79.17	79.18	79.19	79.13	.0004

sub-network performance and the smaller sub-network performance. Figure 2 additionally compares the result of using layer-wise pruning against the use of pruning that is executed across the entire network. Here, it is evident that the non-layer-wise pruning that we propose for OFA-FL, which resembles the flexibility of the centralized OFA network progressive shrinking algorithm, is superior to the more conventional layer-wise pruning implementation.

For CIFAR-10 classification, we use a mini-batch size of 128 and an initial learning rate of 0.0004. We initially wait 200 epochs to start the pruning. In Table 1, we compare results where subsequent pruning rounds occur after either every 50 or 100 epochs. Although waiting for the full model to converge before the initial pruning is crucial for ensuring that smaller sub-network parameters will not interfere with larger sub-network parameters, these results show that the frequency with which subsequent pruning occurs is not as important. We also find that the accuracy of the system peaks when only the full network has been trained, which is demonstrated in the plot of each sub-network's accuracy throughout training with OFA-FL (Figure 4). This is right after it first converges and before any pruning occurs. Then, with each round of pruning, the sub-network accuracies drop as a new sub-network joins the system, then the accuracies increase again until the next round of pruning, but they increase to lower accuracies for each of these cycles. This explains why one-shot pruning can achieve the highest maximum average accuracy. We only consider epochs where all sub-network sizes are included in the system when finding the maximum average accuracy, and for one-shot pruning, this includes

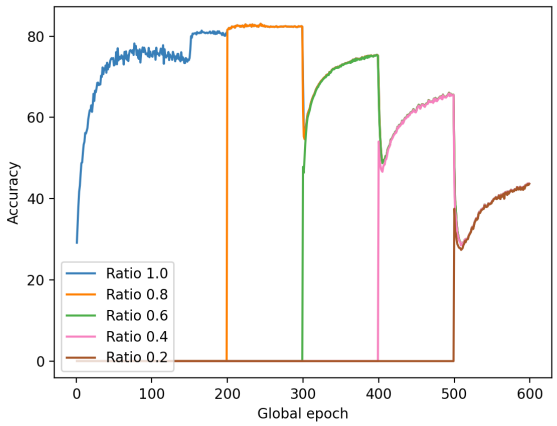


Figure 4: Accuracies across 600 epochs of global training for each sub-network size on the CIFAR-10 classification task using our OFA-FL method. Initial pruning occurred after 200 epochs and subsequent pruning occurred every 100 epochs. Note that in this plot, a sub-network's compression ratio's accuracy is considered 0 until that sub-network size has been pruned for by the server. The different lines are difficult to distinguish between because they are mostly overlapping.

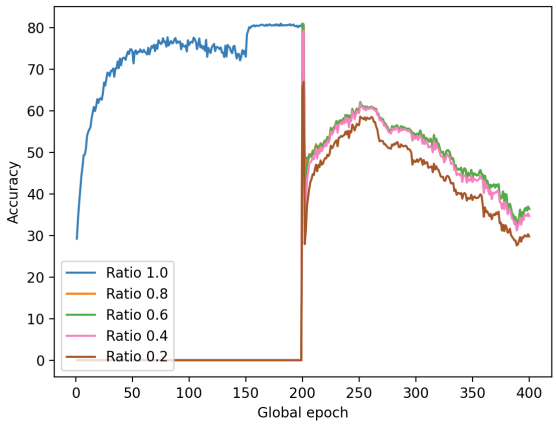


Figure 5: Accuracies across 400 epochs of global training for each sub-network size on the CIFAR-10 classification task using our one-shot pruning. Pruning occurred after 200 epochs. Note that in this plot, a sub-network's compression ratio's accuracy is considered 0 until that sub-network size has been pruned for by the server.

the epochs following just one round of pruning, whereas with OFA-FL, all sub-networks are not included in the system until 4 rounds of pruning have occurred and each one causes the system to converge at increasingly low accuracies. An example of this pattern for the one-shot pruning method can be seen in Figure 5. The low final accuracy across all methods in this set of experiments suggests that the results are not reliable, so we ran additional experiments that use a set of compression ratios, 1.0, 0.9, 0.8, 0.7, and 0.6, that do not shrink the overall network as drastically. These results are reported in Table 2 and again show that OFA-FL leads to minimized variance across sub-network accuracies, which demonstrates the efficacy of its joint optimization approach.

4 CONCLUSION

We presented a new FL approach to learning a global model where clients have device resource constraints that may not be able to accommodate the full global architecture. Our method translates the centralized concepts of OFA networks to an FL setting through a server-side application of LTP and client-side local OFA training. We have shown promising results of the proposed method through numerical studies. A future problem is to study more sophisticated pruning techniques, e.g., those based on the PQ Index [5].

REFERENCES

[1] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, H Brendan McMahan, et al. 2019. Towards federated learning at scale: System design.

[2] Nader Bouacida, Jiahui Hou, Hui Zang, and Xin Liu. 2020. Adaptive Federated Dropout: Improving Communication Efficiency and Generalization for Federated Learning. https://doi.org/10.48550/ARXIV.2011.04050

[3] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. 2019. Once-for-All: Train One Network and Specialize it for Efficient Deployment. https://doi.org/10.48550/ARXIV.1908.09791

[4] Enmao Diao, Jie Ding, and Vahid Tarokh. 2021. HeteroFL: Computation and Communication Efficient Federated Learning for Heterogeneous Clients. In *International Conference on Learning Representations (ICLR)*.

[5] Enmao Diao, Ganghua Wang, Jiawei Zhan, Yuhong Yang, Jie Ding, and Vahid Tarokh. 2023. Pruning Deep Neural Networks from a Sparsity Perspective. In *International Conference on Learning Representations (ICLR)*.

[6] Alireza Fallah, Aryan Mokhtari, and Asuman E. Ozdaglar. 2020. Personalized Federated Learning: A Meta-Learning Approach. arXiv:2002.07948 https://arxiv.org/abs/2002.07948

[7] Jonathan Frankle and Michael Carbin. 2018. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. https://doi.org/10.48550/ARXIV.1803.03635

[8] Song Han, Huizi Mao, and William J Dally. 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding.

[9] Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. 2018. Federated learning for mobile keyboard prediction.

[10] Chaoyang He, Murali Annavaram, and Salman Avestimehr. 2020. Towards Non-I.I.D. and Invisible Data with FedNAS: Federated Deep Learning via Neural Architecture Search. https://doi.org/10.48550/ARXIV.2004.08546

[11] Samuel Horvath, Stefanos Laskaridis, Mario Almeida, Ilias Leontiadis, Stylianos I. Venieris, and Nicholas D. Lane. 2021. FjORD: Fair and Accurate Federated Learning under heterogeneous targets with Ordered Dropout. https://doi.org/10.48550/ARXIV.2102.13451

[12] Yuang Jiang, Shiqiang Wang, Victor Valls, Bong Jun Ko, Wei-Han Lee, Kin K. Leung, and Leandros Tassiulas. 2019. Model Pruning Enables Efficient Federated Learning on Edge Devices. https://doi.org/10.48550/ARXIV.1909.12326

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

[14] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. 2009. CIFAR-10 (Canadian Institute for Advanced Research). http://www.cs.toronto.edu/~kriz/cifar.html

[15] Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. , 2278–2324 pages.

[16] Ang Li, Jingwei Sun, Binghui Wang, Lin Duan, Sicheng Li, Yiran Chen, and Hai Li. 2020. LotteryFL: Personalized and Communication-Efficient Federated Learning with Lottery Ticket Hypothesis on Non-IID Datasets. https://doi.org/10.48550/ARXIV.2008.03371

[17] Ang Li, Jingwei Sun, Xiao Zeng, Mi Zhang, Hai Li, and Yiran Chen. 2021. FedMask: Joint Computation and Communication-Efficient Personalized Federated Learning via Heterogeneous Masking. In *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems* (Coimbra, Portugal) *(SenSys '21)*. Association for Computing Machinery, New York, NY, USA, 42–55. https://doi.org/10.1145/3485730.3485929

[18] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. 2020. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine* 37, 3 (2020), 50–60.

[19] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2016. Communication-Efficient Learning of Deep Networks from Decentralized Data. arXiv:1602.05629 [cs.LG]

[20] Anish K. Vallapuram, Pengyuan Zhou, Young D. Kwon, Lik Hang Lee, Hengwei Xu, and Pan Hui. 2022. HideNseek: Federated Lottery Ticket via Server-side Pruning and Sign Supermask. https://doi.org/10.48550/ARXIV.2206.04385

[21] Sergey Zagoruyko and Nikos Komodakis. 2016. Wide Residual Networks. https://doi.org/10.48550/ARXIV.1605.07146