# Architecture Personalization in Resource-constrained Federated Learning

**Mi Luo**[1], **Fei Chen**[2], **Zhenguo Li**[2], **Jiashi Feng**[1]
[1]National University of Singapore [2]Huawei Noah's Ark Lab
{romyluo7, jshfeng}@gmail.com, {chen.f, li.zhenguo}@huawei.com

## Abstract

Federated learning aims to collaboratively train a global model across a set of clients without data sharing among them. In earlier studies, a global model architecture, either predefined by experts or searched automatically, is applied to all the clients. However, this convention is impractical for two reasons: 1) The clients may have heterogeneous resource constraints and only be able to handle models with particular configurations, imposing high requirements on the model's versatility; 2) Data in the real-world federated system are highly non-IID, which means a model architecture optimized for all clients may not achieve optimal performance on personalized data on individual clients. In this work, we address the above two issues by proposing a novel framework that automatically discovers personalized model architectures tailored for clients' specific resource constraints and data, called Architecture Personalization Federated Learning (`APFL`). `APFL` first trains a sizable global architecture and slims it adaptively to meet computational budgets on edge devices. Then, `APFL` offers a communication-efficient federated partial aggregation (FedPA) algorithm to allow mutual learning among clients with diverse local architectures, which largely boosts the overall performance. Extensive empirical evaluations on three federated datasets clearly demonstrate that `APFL` provides affordable and personalized architectures for individual clients, costing fewer communication bytes and achieving higher accuracy compared with manually defined architectures under the same resource budgets.

## 1 Introduction

Recently, some research works have been devoted to Federated Neural Architecture Search (Federated NAS) [1–4]. They aim to search neural architectures in the federated setting where data are not independent and identically distributed (non-IID) among multiple clients (such as mobile phones or online systems of different organizations with constrained data access) and cannot be uploaded to the central server due to privacy concern. Federated NAS finds model architectures automatically, saving much time and human effort compared with the hand-design manner. Thus, it is attracting increasing attention.

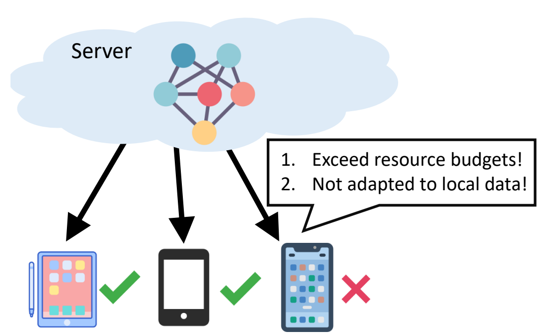


Figure 1: Don't apply the same architecture for all clients.

Existing Federated NAS methods strive to search the optimal *global* model architecture in a decentralized way based on popular NAS algorithms [5–7], which is then applied to all the clients after being fully trained. Though effective, two major issues arise when deploying the resultant model architecture on clients. As shown in Figure 1, firstly, each client may have different hardware configurations, posing resource constraints (such as latency, energy and memory footprint) for the architecture to run properly, thus the global architecture maybe not affordable. Secondly, as suggested by [8–10], the architecture that achieves the best performance on particular data distribution may not generalize well to another. In federated learning, the data distributed on clients are non-IID, thus it's implausible that a single architecture optimized globally performs best for all the clients.

To address these issues, this work makes the first effort to explore the problem of discovering personalized architectures that are tailored to specific resource budgets and heterogeneous data distributions of a diverse set of clients in federated learning. This problem is challenging due to privacy and regulatory reasons. Personal data cannot leave the client, thus the architecture search process should take place locally. Most NAS methods are time-consuming and computationally expensive [11, 12]. Though many approaches have accelerated searching or reduce the required computation overhead [5, 13, 14], the inherent drawback of the large search space still remains. So they are not suitable for deployment on resource-constrained edge devices. Further, to boost the local task performance while affording the budget, optimal accuracy-efficiency trade-offs should be achieved.

To battle these problems, we propose a new federated NAS algorithm called *Federated Channel Search (FedCS)*. The main idea is to maintain a powerful global architecture (called "Super-network") with strong generalization ability on the server, being tailorable for different clients and then pruning it at the channel-level to obtain efficient and customized local model architectures. Note that local pruning requires no further back-propagation on the local data and can be executed on the fly. Compared with previous federated NAS methods, *FedCS* is more suitable for the resource-constrained federated setting. It largely compresses the scale of search space by only searching for channel configurations. Further, it is computationally-cheap since the searching cost mainly lies in the training process of super-network which could be largely shrunk.

Typically, architectures searched by federated NAS approaches are trained from scratch using FedAvg [15] to boost their performance. However, the parameter aggregation scheme of FedAvg requires each client to share the same architecture and doesn't fit for the setting where each client has diverse architectures. Thus we propose a novel *Federated Partial Aggregation (FedPA)* algorithm to achieve continuous improvements on the performance of sub-networks. In *FedPA*, all the clients collaboratively update the weights of the super-network maintained on the server in a periodic manner, then inherit the corresponding part of weights from the super-network. Since only partial weights of the super-network are transmitted, in our experiments, *FedPA* saves much communication cost to attain the same accuracy with the state-of-the-art FedAvg. Note that *FedPA* is not a supplementary algorithm for *FedCS*, but can be applicable to the general setting where all clients own different architectures sampled from the same search space.

As shown in Figure 2, the primary contribution of this work is a novel *Architecture Personalization Federated Learning (APFL)* framework which includes the above two sequential steps. It addresses a federated learning problem with great practical significance but not much prior literature: each client has its own resource constraint and heterogeneous data and needs to be served with personalized model architecture. *APFL* advances the progress of federated learning in two aspects. 1) First, it provides a computationally-efficient *FedCS* algorithm which conducts neural architecture search for channel number on mobile devices while preserves user privacy. Empirical results show that, with *FedCS*, we are able to acquire personalized architectures which make good trade-offs between accuracy and resource-efficiency. Moreover, personalized architectures searched by *FedCS* cost less communication cost compared with human-designed architectures. 2) Second, *APFL* gives *FedPA*, an off-the-shelf solution to the federated optimization problem under the setting where clients have diverse architectures. Extensive experiments show that *FedPA* achieves much more accuracy gains compared with FedAvg.

## 2 Related Work

This work is related to Personalization in Federated Learning, Neural Architecture Search, and Federated Neural Architecture Search. Please refer to the Appendix for more discussions.

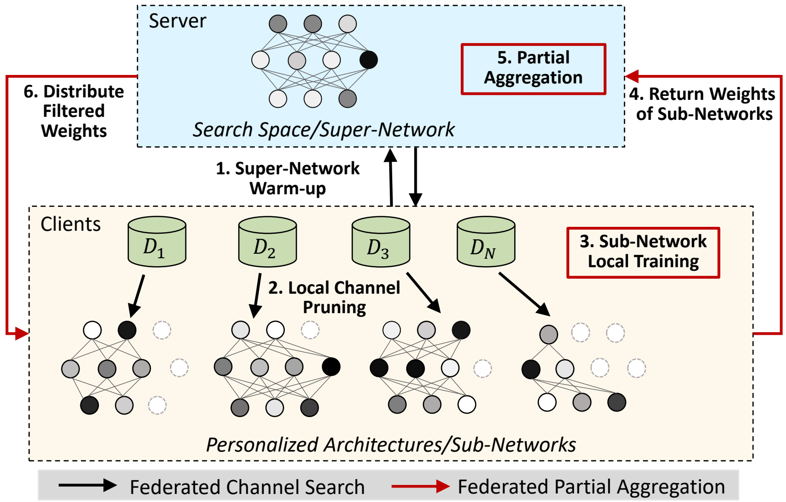


Figure 2: An overview of *APFL* framework. It divides the federated learning process into two subsequent phases: 1) Federated Channel Search (marked with black arrow), which searches hardware-friendly architectures for the clients equipped with heterogeneous resources. 2) Federated Partial Aggregation (marked with red arrow), which aims to make the searched architectures ready for real-time inference.

# 3 Problem Setup

We consider image classification task on a central server and $K$ clients. Each client $u_k$ stores a local private dataset $D^k$, with $n_k$ data samples. Each local data sample in $D^k$ is drawn from a local distribution $P_k$ different from the ones of other clients. Their formed global training data $D = \bigcup_k D^k$ with $n = \sum_k n_k$ samples are non-IID.

More specifically, we address a *resource-constrained federated learning* setting. That is, each client $u_k$ has its own computation resource budget $B_k$. Two resource constraints are considered, *i.e.*, FLOPs and the model size (the number of parameters) which the client can afford.

Formally, we aim to search for the optimal and personalized architecture parameter $\theta_k^*$, along with the optimized model weight parameter $w_k^*$ for each client $u_k$, whose resource cost $C(\theta_k)$ does not exceed the local budget $B_k$. The objective can be defined as

$$\begin{aligned} \min_{W,\Theta} \; & F(W,\Theta) \triangleq \sum_{k=1}^{K} p_k \mathcal{L}(w_k, \theta_k; D^k) \\ \text{s.t.} \; & C(\theta_k) \leq B_k, \quad \forall k = 1, 2, ..., K, \end{aligned} \tag{1}$$

$W = [w_1, w_2, ..., w_K]$ denote their corresponding model weight parameters, and $p_k = n_k/n$ is the client-wise weighting factor when participating in aggregation. The function $\mathcal{L}$ is the local objective function which measures the classification loss over the local dataset $D^k = \{z_1, \ldots, z_{n_k}\}$:

$$\mathcal{L}(w_k, \theta_k; D^k) \triangleq \frac{1}{n_k} \sum_{i=1}^{n_k} \ell(w_k, \theta_k; z_i), \tag{2}$$

where $\ell(\cdot, \cdot; \cdot)$ is the cross entropy loss function.

**Search Space.** Our adopted search space is MobileNet [16–18], which has been proven to be stable and efficient for mobile setting. To further shrink the search space into an acceptable range for resource-constrained federated learning, we set the number of the channels as the only changeable option of the network. Thus, $\theta$ is a vector containing the channel configuration of a network. Note that another popular search space DARTS [5] used in [1, 2] is less hardware-friendly and has lower scalability to architecture personalization, thus we don't adopt it.

# 4 Architecture Personalization Federated Learning (APFL)

**Algorithm 1: Federated Channel Search.** $T$ is the number of communication rounds; $E$ stands for the number of local training epochs; $w_m^{(t)}$ represents the network weights at client $u_m$ in round $t$; $B_k$ stands for the computation budget at client $u_k$; $C_k$ is the computation cost of the current network adopted at client $u_k$, $R \in (0, 1)$ is the shrink ratio of channels.

# **Super-Network Warm-up**:
**for** $t = 0$ *to* $T - 1$ **do**
  Server samples a set of $M$ high capacity clients, indexed by $I^{(t)}$, and broadcasts $w^{(t)}$ to the sampled clients;
  **for** *each client* $u_m$ *with* $m \in I^{(t)}$ **do**
    $w_m^{(t+1)} \leftarrow$ TrainingUSNet$(w^{(t)})$
    Send $w_m^{(t+1)}$ back to the server;
  **end**
  Server aggregates the received gradients as:
  $w^{(t+1)} \leftarrow \sum_{m \in I^{(t)}} \frac{n_m}{n^{(t)}} w_m^{(t+1)}$
**end**
# **Local Channel Pruning**:
Server broadcasts current super-network weights $w$ to all the clients;
**for** *each client* $u_k$ **do**
  **while** $C(\theta_k) > B_k$ **do**
    **for** *number of channels* $\theta_k^j$ *for each layer* $j$ **do**
      $\theta_k^j \leftarrow \lceil \theta_k^j (1 - R) \rceil$
      $A^j \leftarrow AccDrop(\theta_k)$
      $\theta_k^j \leftarrow \lfloor \theta_k^j / (1 - R) \rfloor$
    **end**
    $j^* \leftarrow \arg\min_j A^j$,
    $\theta_k^{j^*} \leftarrow \lceil \theta_k^{j^*} (1 - R) \rceil$
  **end**
**end**

**Challenges.** The above optimization problem is difficult since both the optimal values of $W$ and $\Theta$ should be obtained. In this work, we solve it by fixing one and optimizing the other. The problem can be disassembled into two stages: 1) Architecture search, which finds optimal architectures $\Theta$. 2) Personlized Architecture Training, which keeps $\Theta$ unchanged and determines the corresponding $W$. The first sub-question is tricky due to privacy concerns in federated learning. All computation must be completed on resource constrained edge devices, imposing high requirements on the efficiency of the search strategy and the scale of the search space. The second sub-question is also arduous because the heterogeneity of model architectures of clients makes the classic federated optimization algorithm FedAvg fail. More specifically, The underlying assumption of FedAvg is that all clients share the same architecture, while this is not met in our problem.

**General Framework.** As shown in figure 2, *APFL* consists of two fundamental components: *Federated Channel Search (FedCS)* and *Federated Partial Aggregation (FedPA)*, which are responsible for Architecture Search and Personlized Architecture Training respectively.

*FedCS* aims to search for the optimal channel configurations which satisfy the resource budgets of individual clients. The general idea is inspired by network pruning [19–22], to reduce a heavy network to a lightweight one by removing redundant weights or neurons. Network pruning is a natural choice for architecture personalization in federated learning, because the same trained super-network can be flexibly applied for local pruning on a large amount of clients. Common pruning criteria includes the magnitudes of weights [23], scaling parameters of batch normalization [19], etc. However, these criteria are not directly applicable to *APFL*. The main reason is that the over-parameterized super-network should be trained in a federated way, thus the numerical values of the weights only reflect the corresponding importance on the global training data $D$, rather than the importance on the personalized local data $D^k$. To make the pruned architecture more customized, we propose to ground the pruning criteria on the local task performance. Specifically, we use the instant inference accuracies of the sub-networks contained in the super-network to determine whether a channel should be pruned. Note that we prune at the level of the channel rather than the weight, avoiding unstructured sparse matrix operations which require special hardware support to be effective [17]. To conclude, in *FedCS*, a global tailorable architecture is firstly trained in a distributed way, then it is pruned through client-wise additional learning on the local data to fit the local resource constraints. This approach can fully exploit the generalization properties of the global model as well as the learned customized information from the local distribution.

*FedPA* aims to solve the federated optimization problem where the client's architecture are incongruous sub-networks cut out from the super-network. It retains the benefit of collaborative training by enabling the local customized models to learn from each other. The intuition is to only update the corresponding part of weights in the super-network when a sub-network makes contributions to it.

### 4.1 Federated Channel Search (FedCS)

*FedCS* starts with training a sizable global super-network with FedAvg. However, plain training does not optimize the sub-networks contained in the global model. Therefore, the super-network is not a good estimator to rank the relative accuracies of the sub-networks contained in it. So the sub-networks cannot be directly drawn from the super-networks to acquire its instant inference accuracy which will be served as the criteria for local pruning. In view of the above considerations, we resort to slimmable networks [24, 25, 8] which can be executed at arbitrary widths as the global architecture. In slimmable networks, US-Net [25] is particularly suitable because it adopts the sandwich rule and inplace distillation so that sub-networks with different channel configurations can be optimized simultaneously.

**Super-Network Warm-up.** Different from most network pruning methods, *FedCS* does not fully train the super-network to achieve its utmost performance. The reason is that we only care about the relative accuracy of the sub-networks, so roughly training for 30% of the full rounds will be enough, according to our experiments. This practice benefits federated learning by saving much communication and computation cost. As shown in Algorithm. 1, *FedCS* first initializes the weight of super-network $w^0$ randomly. In the $t$-th communication round, the server selects a random subset from the clients which are capable of running the super-network properly, indexed by $I^{(t)} \subseteq \{1, \ldots, K\}$, and distributes the current $w^{(t)}$ to them. Then each participating client $u_m$ performs local training based on the received parameters $w^{(t)}$, and sends the updated parameters $w_m^{(t)}$ back to the central server. Specifically, the typical training paradigm of US-Net is adopted in the local learning procedure, details are covered in the Appendix. After receiving the architecture parameters and network weights from all the participating clients, the central server then takes a weighted average of them to update the global model. The weighting factor is $\frac{n_m}{n^{(t)}}$, where $n_m$ is the number of local samples of the client $u_m$, and $n^{(t)} = \sum_{m \in I^{(t)}} n_m$ is the total number of data samples used in this communication round.

**Local Channel Pruning.** Once the super-network is obtained, unnecessary channels are pruned from it layer-wisely. We adopt greedy pruning strategy [8, 26] to achieve an inherent trade-off between the accuracy and the resource-efficiency of the personalized architecture. Local pruning requires no further training of the super-network. The sub-networks can be directly sampled from the super-network to acquire the instant inference accuracy as the pruning criterion. Algorithm. 1 shows the detailed local channel pruning procedure. The central server first broadcasts the weights $w$ of the super-network obtained from the warm-up stage to all clients. After receiving $w$ from the server, each client $u_k$ fine-tunes the channel number of the received architecture based on its local resource budget $B_k$. Concretely, we predefine a shrink ratio $R \in (0, 1)$. In each iteration, each layer is shrunk by $R$ tentatively, then the accuracy of the pruned network is computed on the local training dataset. After finishing all the layers, the layer $j$ with the minimum accuracy drop $A^j$ will be truly shrunk. Then the computation cost $C(\theta_k)$ of the pruned network will be checked. If it is lower than the local resource budget $B_k$, the pruning process would be terminated.

### 4.2 Federated Partial Aggregation (FedPA)

After attaining personalized architectures by *FedCS*, one can choose to fully train them on the local datasets. However, to make the best use of the generalization capacity of the super-network and also the global information encoded in it, the local model can inherit the corresponding part of parameters from the super-network. This is inspired by the lottery ticket hypothesis [27] that sub-networks can benefit from the initial weights of the original super-network. However, as suggested by [28, 29], simply training on the local data restrains individual clients from learning beneficial knowledge from each other and sacrifices the core advantage of federated learning: win-win cooperations among clients. After an extensive survey on federated optimization algorithms, we find that most off-the-shelf methods, such as FedAvg[15], FedMA [30], FedProx [31], are grounded on the assumption that all clients share the same architecture, thus not fit for our problem. In view of this, we propose a novel federated partial aggregation algorithm *FedPA* which solves federated optimization problem with heterogeneous edge architectures. Full details of FedPA can be found in the Appendix.

A crucial trick of FedPA is always to keep batch normalization [32] statistics locally, for three reasons. First, running mean and variance of feature representations are privacy-sensitive, thus should never leave the clients. Second, retaining batch normalization statistics locally is naturally suited to eschew aggregation disorder brought by heterogeneous channel configuration. Additionally, for non-IID data, the local running means and variances may vary across clients considerably, thus average them to accumulate global BN statistics may not help advance local task performance.

# 5 Experiment

## 5.1 Experiment Setup

We consider image classification task and adopt three datasets from the popular FedML benchmark [33], including CIFAR-10 [34], CIFAR-100 [34] and CINIC-10 [35]. Note that CINIC-10 is constructed from ImageNet [36] and CIFAR-10, whose samples are very similar but not drawn from identical distributions. Therefore, it naturally introduces distribution shifts which is suited to the heterogeneous nature of federated learning. We are interested in two data partition strategies : IID partition and NIID partition. The detailed partition strategy, statistics and visualizations of the datasets are summarized in the Appendix. For the local resource constraints $B_k$, we divide the clients into 3 groups with 3 possible resource configurations: high budget, medium budget and low budget. To check the performance of *APFL* under the scenarios with different overall computation capacities, we consider two settings: 1) HIGH CAPACITY, where 50% of clients are equipped with high computation budgets, 30% and 20% of clients have medium and low computation budgets. 2) LOW CAPACITY, where only 20% of clients are equipped with high computation budgets, 30% and 50% of clients have medium and low budgets. Other implementation details are covered in the Appendix.

## 5.2 Accuracy Improvement of APFL

The classic federated optimization algorithm *FedAvg* requires all clients to share the same architecture. To make it comparable in our setting, we set the global architecture of *FedAvg* as MobileNetV2-0.5x which is acceptable to all the clients. However, this baseline is weak considering the overall computation capacity is still low. So we propose a stronger baseline that allows each client to own heterogeneous local architectures. We term it as *federated learning with uniform channels (FedUniform)*. Concretely, *FedUniform* modifies the global width multiplier of the super-network adaptively for each client till its resource constraints are fulfilled. According to [16, 17], It's effective for trading off between resource efficiency and accuracy. In *FedUniform*, the local architectures adopted by clients with high, medium and, low resource budgets are MobileNetV2 with 1x, 0.75x and, 0.5x width multipliers respectively. The local architectures obtained by *FedCS* and *FedUniform* are both trained with *FedPA* from scratch. Note that for a fair comparison, sub-networks of *FedCS* didn't inherit weights from the super-network which was trained in the warm-up stage. However, we also find that using the parameters of super-networks for initialization further improves the performance of *FedCS*. After training, the local testing tasks can be performed either by downloading the global super-network from the server for inference or directly running with the local architectures. We report accuracies of both evaluation methods and abbreviate them as "Glo" and "Loc".

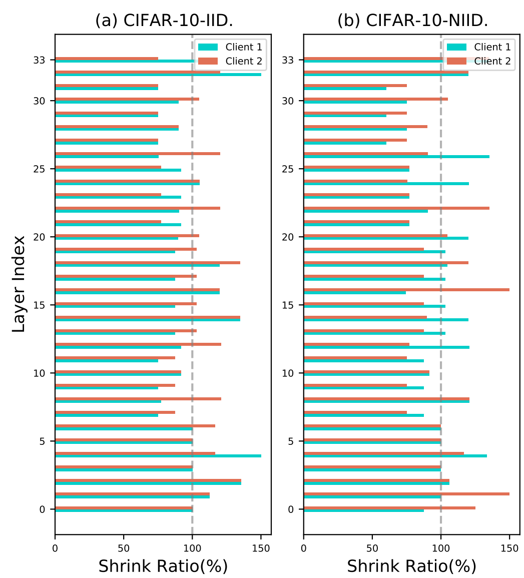


Figure 3: Comparing architectures searched on different clients.

We summarize all the results in Table 1. For all the datasets, it can be observed that *FedCS* achieves accuracy improvements over *FedAvg*, with the accuracy gain up to 15.17% on CINIC-10. We then

Table 1: Accuracy@1 (%) on CIFAR-10, CIFAR-100 and CINIC-10.

| | Methods | CIFAR-10 | | | | CIFAR-100 | | | | CINIC-10 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | IID | | NIID | | IID | | NIID | | IID | | NIID | |
| | | Glo. | Loc. | Glo. | Loc. | Glo. | Loc. | Glo. | Loc. | Glo. | Loc. | Glo. | Loc. |
| HIGH CAPACITY | FedAvg | 92.10 | | 89.49 | | 66.36 | | 64.6 | | 77.63 | | 60.21 | |
| | FedUniform | 92.58 | 92.55 | 92.47 | 92.52 | 68.1 | 67.80 | 65.83 | 65.46 | 79.21 | 79.19 | 71.11 | 73.57 |
| | FedCS (Ours) | **92.74** | **92.75** | **93.08** | **93.29** | **68.39** | **68.46** | **66.01** | **65.60** | **79.33** | **79.31** | **74.51** | **75.38** |
| LOW CAPACITY | FedAvg | 92.10 | | 89.49 | | 66.36 | | 64.6 | | 77.63 | | 60.21 | |
| | FedUniform | 91.66 | 91.68 | 92.38 | 92.45 | 67.61 | 67.51 | 65.69 | 65.39 | 78.76 | 78.78 | 71.04 | 72.22 |
| | FedCS (Ours) | **92.63** | **92.64** | **92.82** | **93.05** | **67.83** | **68.15** | **65.86** | **65.44** | **79.01** | **79.02** | **73.04** | **74.42** |

compare *FedCS* with *FedUniform* and find that *FedCS* performs consistently better than *FedUniform*. We also notice that when increasing the proportion of clients with lower resource budgets (from HIGH CAPACITY to LOW CAPACITY), the accuracies of *FedCS* and *FedUniform* both decrease, but *FedCS* still performs better than *FedUniform*. It well demonstrates that the architectures searched by *FedCS* are more customized to the local data distribution. Another surprising discovery is that the performance of applying personalized architectures for local inference is not necessarily worse than that of applying the full super-network. On CINIC-10, it even provides a 2.46% accuracy gain, which further validates the advantage of architecture personalization. To conclude, all results confirm that *FedCS* is able to find personalized architectures which not only meet the local resource constraints but also boost the performance of local tasks.

### 5.3 Visualization of Personalized Architectures

To analyze the characteristics of the personalized architectures searched by *APFL*, we now provide additional visualization results. First, we compare the local architectures of the clients who belong to the same resource group. On the IID and NIID versions of CIFAR-10, we plot the shrink ratios of the pruned architectures (compared to the original MobileNetV2). As shown in Figure 3, the architectures searched on the NIID version are visually more heterogeneous. We also compute the L2 distance between the two architecture parameters $\theta$ and find that the distance between architectures searched on the IID version is indeed smaller. This suggests that the personalized architectures obtained by *APFL* are effectively adapted to local distributions.

### 5.4 System Overhead

We now present another benefit brought by *FedCS*. As shown in Figure 5, we compare the average FLOPs and number of parameters of local architectures searched by *FedCS* with that of *FedUniform*. It can be seen that with nearly the same FLOPs, *FedCS* provides more lightweight architectures which consume less local storage space than *FedUniform*. As shown Figure 4, training architectures searched by *FedCS* obviously costs fewer communication bytes to achieve the same accuracy as that of *FedUniform*. This is particularly inspiring, as communication cost is the main bottleneck of federated learning.

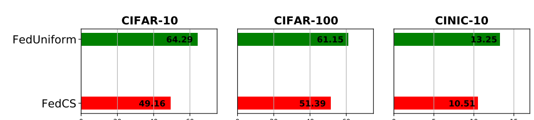

Figure 4: Communication Cost (GBytes).

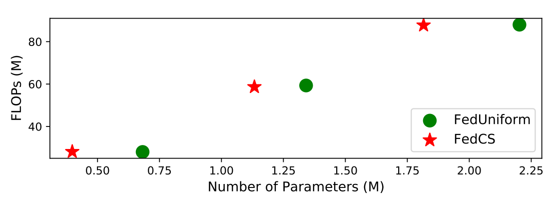

Figure 5: FLOPs and number of parameters.

## 6 Conclusion

In this paper, we initialize the concept of architecture personalization and propose a novel federated architecture learning framework named *APFL* which is able to discover resource-efficient personalized model architectures for individual clients. Extensive empirical results well demonstrate the effectiveness and high efficiency of *APFL*. As a starting point, we hope the proposed *APFL* framework could contribute to simulating research on architecture personalization in federated learning. We leave to future work many open problems, such as convergence analysis of *FedPA* and more effective federated partial aggregation algorithms.

## References


[1] He, C., M. Annavaram, S. Avestimehr. Fednas: Federated deep learning via neural architecture search. 2020.

[2] Singh, I., H. Zhou, K. Yang, et al. Differentially-private federated neural architecture search. *arXiv preprint arXiv:2006.10559*, 2020.

[3] Zhu, H., Y. Jin. Real-time federated evolutionary neural architecture search. *arXiv preprint arXiv:2003.02793*, 2020.

[4] Xu, M., Y. Zhao, K. Bian, et al. Neural architecture search over decentralized data. *arXiv preprint arXiv:2002.06352*, 2020.

[5] Liu, H., K. Simonyan, Y. Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

[6] He, C., H. Ye, L. Shen, et al. Milenas: Efficient neural architecture search via mixed-level reformulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11993–12002. 2020.

[7] Yang, T.-J., A. Howard, B. Chen, et al. Netadapt: Platform-aware neural network adaptation for mobile applications. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 285–300. 2018.

[8] Yu, J., T. Huang. Autoslim: Towards one-shot architecture search for channel numbers. *arXiv preprint arXiv:1903.11728*, 2019.

[9] Chin, T.-W., R. Ding, C. Zhang, et al. Towards efficient model compression via learned global ranking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1518–1528. 2020.

[10] Luo, M., F. Chen, P. Cheng, et al. Metaselector: meta-learning for recommendation with user-level adaptive model selection. In *Proceedings of The Web Conference 2020*, pages 2507–2513. 2020.

[11] Zoph, B., Q. V. Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

[12] Real, E., A. Aggarwal, Y. Huang, et al. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, vol. 33, pages 4780–4789. 2019.

[13] Xu, Y., L. Xie, X. Zhang, et al. Pc-darts: Partial channel connections for memory-efficient architecture search. *arXiv preprint arXiv:1907.05737*, 2019.

[14] Dong, X., Y. Yang. Searching for a robust neural architecture in four gpu hours. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1761–1770. 2019.

[15] McMahan, B., E. Moore, D. Ramage, et al. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *AISTATS*, pages 1273–1282. 2017.

[16] Howard, A. G., M. Zhu, B. Chen, et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[17] Sandler, M., A. Howard, M. Zhu, et al. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520. 2018.

[18] Howard, A., M. Sandler, G. Chu, et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1314–1324. 2019.

[19] Liu, Z., J. Li, Z. Shen, et al. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2736–2744. 2017.

[20] He, Y., X. Zhang, J. Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1389–1397. 2017.

[21] Huang, Q., K. Zhou, S. You, et al. Learning to prune filters in convolutional neural networks. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 709–718. IEEE, 2018.

[22] Lee, N., T. Ajanthan, P. H. Torr. Snip: Single-shot network pruning based on connection sensitivity. *arXiv preprint arXiv:1810.02340*, 2018.

[23] Han, S., J. Pool, J. Tran, et al. Learning both weights and connections for efficient neural networks. *arXiv preprint arXiv:1506.02626*, 2015.

[24] Yu, J., L. Yang, N. Xu, et al. Slimmable neural networks. *arXiv preprint arXiv:1812.08928*, 2018.

[25] Yu, J., T. S. Huang. Universally slimmable networks and improved training techniques. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1803–1811. 2019.

[26] Ye, M., C. Gong, L. Nie, et al. Good subnetworks provably exist: Pruning via greedy forward selection. In *International Conference on Machine Learning*, pages 10820–10830. PMLR, 2020.

[27] Frankle, J., M. Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

[28] Liang, P. P., T. Liu, Z. Liu, et al. Think locally, act globally: Federated learning with local and global representations. *ArXiv*, abs/2001.01523, 2019.

[29] Hanzely, F., P. Richtárik. Federated learning of a mixture of global and local models. *arXiv preprint arXiv:2002.05516*, 2020.

[30] Wang, H., M. Yurochkin, Y. Sun, et al. Federated learning with matched averaging. In *International Conference on Learning Representations*. 2020.

[31] Li, X., K. Huang, W. Yang, et al. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189*, 2019.

[32] Ioffe, S., C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[33] He, C., S. Li, J. So, et al. Fedml: A research library and benchmark for federated machine learning. *arXiv preprint arXiv:2007.13518*, 2020.

[34] Krizhevsky, A., G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[35] Darlow, L. N., E. J. Crowley, A. Antoniou, et al. Cinic-10 is not imagenet or cifar-10. *arXiv preprint arXiv:1810.03505*, 2018.

[36] Russakovsky, O., J. Deng, H. Su, et al. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[37] Fallah, A., A. Mokhtari, A. Ozdaglar. Personalized federated learning: A meta-learning approach. *arXiv preprint arXiv:2002.07948*, 2020.

[38] Chen, F., M. Luo, Z. Dong, et al. Federated meta-learning with fast convergence and efficient communication. *arXiv preprint arXiv:1802.07876*, 2018.

[39] Jiang, Y., J. Konečnỳ, K. Rush, et al. Improving federated learning personalization via model agnostic meta learning. *arXiv preprint arXiv:1909.12488*, 2019.

[40] Smith, V., C.-K. Chiang, M. Sanjabi, et al. Federated multi-task learning. In *Advances in Neural Information Processing Systems*, pages 4424–4434. 2017.

[41] Khodak, M., M.-F. F. Balcan, A. S. Talwalkar. Adaptive gradient-based meta-learning methods. In *Advances in Neural Information Processing Systems*, pages 5917–5928. 2019.

[42] Bui, D., K. Malik, J. Goetz, et al. Federated user representation learning. *arXiv preprint arXiv:1909.12535*, 2019.

[43] Liang, P. P., T. Liu, L. Ziyin, et al. Think locally, act globally: Federated learning with local and global representations. *arXiv preprint arXiv:2001.01523*, 2020.

[44] Cai, H., T. Chen, W. Zhang, et al. Efficient architecture search by network transformation. *arXiv preprint arXiv:1707.04873*, 2017.

[45] Zoph, B., V. Vasudevan, J. Shlens, et al. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710. 2018.

[46] Cai, H., L. Zhu, S. Han. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332*, 2018.

[47] Tan, M., B. Chen, R. Pang, et al. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2820–2828. 2019.

[48] Wu, B., X. Dai, P. Zhang, et al. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10734–10742. 2019.

[49] Jin, X., J. Wang, J. Slocum, et al. Rc-darts: Resource constrained differentiable architecture search. *arXiv preprint arXiv:1912.12814*, 2019.

[50] Cai, H., C. Gan, T. Wang, et al. Once-for-all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791*, 2019.

[51] Guo, Z., X. Zhang, H. Mu, et al. Single path one-shot neural architecture search with uniform sampling. *arXiv preprint arXiv:1904.00420*, 2019.

[52] Yurochkin, M., M. Agarwal, S. Ghosh, et al. Bayesian nonparametric federated learning of neural networks. In *International Conference on Machine Learning*, pages 7252–7261. PMLR, 2019.