PerFedRLNAS: One-for-All Personalized Federated Neural Architecture Search

Dixi Yao*, **Baochun Li**

University of Toronto
dixi.yao@mail.utoronto.ca, bli@ece.toronto.edu

Abstract

Personalized federated learning is a new paradigm to address heterogeneous problems (e.g. issues with non-i.i.d. data) in federated learning. However, existing personalized federated learning methods lack standards for how personalized and shared parts of the models are designed. Sometimes, manual design can even lead to worse performance than non-personalization. As a result, we propose a new algorithm for personalized federated neural architecture search, called PerFedRLNAS, to automatically personalize the architectures and weights of models on each client. With such an algorithm, we can solve the issues of low efficiency as well as failure to adapt to new search spaces in previous federated neural architecture search work. We further show that with automatically assigning different client architectures can solve heterogeneity of data distribution, efficiency and memory in federated learning. In our experiments, we empirically show that our framework shows much better performance with respect to personalized accuracy and overall time compared to state-of-the-art methods. Furthermore, PerFedRLNAS has a good generalization ability to new clients, and is easy to be deployed in practice.

Introduction

Federated learning (McMahan et al. 2017) (FL) has emerged as an important paradigm in distributed machine learning, where a large number of clients participate in a collaborative training session while keeping their data private. In federated learning, clients may have different computational speed and different data distributions, and such heterogeneity may significantly affect the performance of federated learning (Caldas et al. 2018), especially with the conventional federated averaging (FedAvg) algorithm (Oh, Kim, and Yun 2022). As a result, *personalized federated learning* (Tan et al. 2023) (PFL) emerged as a new paradigm to address such heterogeneity and achieve better performance. The objective of personalized federated learning is to train a personalized model for each client (Collins et al. 2021; Fallah, Mokhtari, and Ozdaglar 2020).

Though many solutions have been proposed for personalized federated learning, each of them has its own definition of the parts of the model to be personalized. Such a lack of conventional wisdom and standards may confuse users when they choose their solutions. Under different settings of federated learning, users need to choose different manually defined personalization policy to reach optimal results. For instance, Figure 1 includes several typical methods of personalized federated learning, and summarizes how they defined shared and personalized parts of the models. Different methods have different manually set definitions of which parts we should share or personalize. Such manual design led to varying performance across these methods, and even with an abundance of existing work, failed to answer two questions adequately: (1) Why personalizing models can solve such a heterogeneous problem in federated learning? (2) Since there are so many personalization policies, is there a standard and automated way of deciding personalized and shared parts of the models on clients to achieve optimal performance?

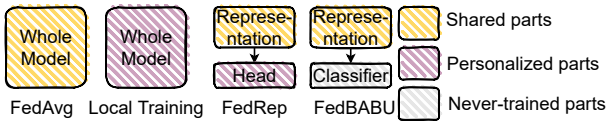

Figure 1: Summary of shared and personalized parts of a model in four typical methods in personalized FL.

Apart from conventional personalized federated learning methods, new methods leveraging automated machine learning (AutoML) are proposed to seek for automatic ways to achieve better performance. Federated Neural Architecture Search (FedNAS) (He, Annavaram, and Avestimehr 2020) was proposed to directly conduct neural architecture search (NAS) in the federated learning setting, with which each client can obtain a model of a different structure along with different weights. However, previous work either search for only a global neural architecture shared among all or manually defined clusters of the clients instead of personalized architecture for each client (Yao et al. 2021; Garg, Saha, and Dutta 2021; Laskaridis, Fernandez-Marques, and Dudziak 2022) or incrementally carry out a neural architecture search on each client (He, Annavaram, and Avestimehr 2020; Mushtaq et al. 2021). Such simple replacements of fixed models in conventional personalized federated learning with an NAS supernet still cannot provide an insight on personalizing models without manual settings.

*Dixi Yao is the corresponding author.

In this paper, towards setting a new standard for deciding personalized and shared parts of the models in an automated fashion, we propose a new framework called *personalized federated neural architecture search via reinforcement learning* (PerFedRLNAS). PerFedRLNAS works as a general framework, and can partially personalize models automatically without manual settings for personalized and shared parts. In our framework, each client has a model of a customized architecture with customized weights. Which parts of their models need sharing, which parts need personalizing, and which parts should be shared among which clients will be automatically decided with the optimal policy. In other words, the personalized and shared parts of the models are decided directly by the objective. Back to the first question, we show that having partly personalized architectures and models on each client can help solve data and system heterogeneity.

Apart from the failure of answering those two core questions, another existing limitation of current personalized federated learning and federated neural architecture search solutions is that they lack flexibility adapting to various deep neural network models, which prevents them from enjoying better performance. However, the structures of neural networks evolve very quickly, and the new framework we proposed can be implemented with arbitrary NAS supernet models. If a new promising NAS supernet model is proposed later, PerFedRLNAS can adapt it into the federated setting.

In addition, there is another limitation that prevents previous federated NAS methods from being applied in personalized federated learning effectively: *low efficiency*. We keep the architecture search phase entirely on the server, and only a single model — instead of the entire supernet — is transmitted between the server and the clients. With our approach, clients only need to perform local training over a normal scaled neural network model, and no extra communication and local computation overhead will be incurred. The overall efficiency is comparable to FedAvg.

Finally, we empirically show that PerFedRLNAS outperforms FedAvg, local training and existing state-of-the-art personalized and heterogeneous federated learning methods. PerFedRLNAS can achieve much better performance in terms of accuracy and efficiency among different objectives, different heterogeneous settings and different backbone models and search space. We also verify that PerFedRLNAS generalizes well to new clients.

Related Work

Personalized Federated Learning

Recently, a variety of works has studied personalized federated learning, for example, adopting partial personalization policy (Pillutla et al. 2022; Collins et al. 2021; Oh, Kim, and Yun 2022). FedTP (Li et al. 2023b) customized attentions in vision image transformers (ViT) since previous methods do not work as well for ViTs as for convolutional neural networks. Ditto (Li et al. 2021) additively mixed the local and global models. Other methods leveraged meta learning (Fallah, Mokhtari, and Ozdaglar 2020), KNN algorithm (Marfoq et al. 2022), Gaussian processes (Achituve et al. 2021), or hyper-network (Shamsian et al. 2021). HeteroFL (Diao, Ding, and Tarokh 2021) aggregated models of different channel widths to solve system heterogeneity. Some researchers (Qu et al. 2022) investigated this problem through changing model architecture. FedDyn (Acar et al. 2021) proposed dynamic regularization method to solve device heterogeneity. However, existing methods still cannot automatically decide how the personalized parts and shared parts of the models are customized directly based on the objectives.

Neural Architecture Search

Supernet-based one-stage neural architecture search (NAS) (He, Zhao, and Chu 2021; Elsken, Metzen, and Hutter 2019) enables efficient architecture optimization and can achieve state-of-the-art results. A supernet is an over-parameterized neural network. After the supernet is well trained, we can sample various single models with different architectures without further retraining. For example, NASViT (Gong et al. 2022) is one of the existing state-of-the-art work, searching for ViTs. MobileNetV3 (Howard et al. 2019) is a family of convolutional neural networks (CNN) obtained by neural architecture search.

Federated Neural Architecture Search

Federated neural architecture search conducts NAS in the federated learning setting, so that each client can obtain models with different architectures and weights. Direct FedNAS (Garg, Saha, and Dutta 2021) and FedNAS (He, Annavaram, and Avestimehr 2020) directly treated the supernet as the global model in FedAvg. FedorAS (Laskaridis, Fernandez-Marques, and Dudziak 2022) needed three phases: supernet training, searching and the fine-tuning. These methods still search for a global model and failed to do personalization. SPIDER (Mushtaq et al. 2021) added a customized architecture search phase during the client local computing while the overhead is very large for transmitting the whole supernet and conducting NAS on the local clients. FedRLNAS (Yao et al. 2021) sent only the subspace to each client, but it still only searched for a global model and had three phases: warm-up, searching and training which is also inefficient. Zhu and Jin (Zhu and Jin 2021) proposed to leverage evolution methods to do federated NAS. However, evolution methods are quite inefficient to apply into real world use cases. We propose a new method which can overcome these drawbacks and has the flexibility to be implemented on arbitrary neural architecture search supernets.

Problem Formulation

In personalized federated learning, we have K clients in total, and each client has its own local objective. We denote them as $f_i, \forall i \in [K]$. In personalized federated learning, customized weight w_i does not need to be and is usually not the same as global weights. It is specific to each client. So, the problem is

$$\min_{w_i} \sum_{i \in [K]} f_i(w_i) \tag{1}$$

where we are trying to make performance on each local dataset good. If $w_i = w$, only a global model is used. If $f_i(w_i) = \mathbb{E}_{(x_i,y_i)\sim D_i}[l(w(x_i), y_i)], \forall i \in [K]$ where w is a

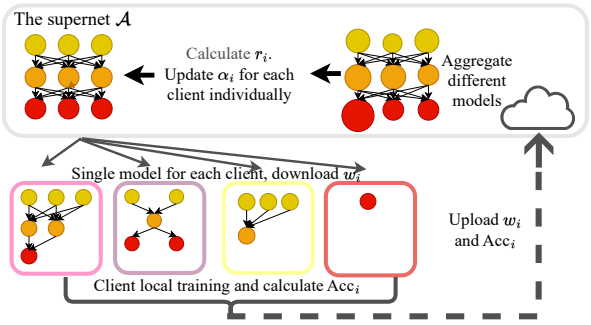

Figure 2: The overview flowchart of the PerFedRLNAS

Input Resolution	128, 192, 224, 256		
Block	Channel Width	Block Numbers	Kernel Size
Convolution	16, 24	1	3, 5, 7
Transformer	16,24, 32	1, 2, 3, 4, 5	-

Table 1: An example of search space of the ViT based NAS structure, where we have search dimensions over input resolution, convolutions blocks and transformer blocks.

shared global model and l is the loss function, this is consistent with the standard formulation of FedAvg. We use federated neural architecture search to automatically decide the shared weights and customize the local weights of clients. The formulation is

$$\min_{\alpha_i, \mathcal{A}} \sum_{i \in [K]} f_i(w_i), \tag{2}$$

$$s.t.\ w_i = S(\alpha_i, \mathcal{A}) \tag{3}$$

where $\mathcal{A}$ is the supernet, α_i are structure parameter indicating the structure of each local model, S is a sampling function: for example, passing α_i through a softmax function and pick one operation on each edge of the supernet.

The difference between different personalized federated learning methods is how the constraints (Equation (3)) are designed. Such constraints do not have to involve w_i. In conventional personalized federated learning methods, there are no supernet and architecture parameters. Equation (3) is predefined by a manually set policy. For example, in FedRep (Collins et al. 2021), w_i is directly composed of head h_i and representation ϕ_i. The constraint can also be a regularization form over loss function (FedProx (Li et al. 2020)).

Personalized Federated Neural Architecture Search via Reinforcement Learning

In previous work on federated NAS, methods were specific to a particular NAS search space. However, as new models and new search spaces are continuously proposed, the federated NAS framework should have the ability to adapt to new NAS supernets. In this section, we will give out the routine building up PerFedRLNAS framework with an arbitrarily given NAS supernet and the solution to searching for the best model architecture on each client.

Build up Personalized Federated NAS Framework

We use $\mathcal{A}$ to denote the supernet, or we call it search space, which is placed on the server. A supernet can be of any structure and each has many different search dimensions with various candidate operations. Each search dimension does not need to have the same number of candidate operations. We use an example search space shown in the Table 1 to better illustrate our method.

In a supernet, there are D search dimensions. In this example, a convolution block has search dimensions of width, block numbers and kernel sizes; a transformer block has search dimensions of width and block numbers; and the input resolution has search dimensions of width. As a result, there are 6 search dimensions. For each search dimension, we assign the probability of each choice: $p_1^d, \cdots, p_{k_d}^d$, where d means the index of each dimension and k_d is the number of choices. In this example, when $d = 4$, it means the kernel size dimension of Conv. We have three choices: 3, 5, 7. So, we have $k_4 = 3$. We can know that p_1^4 is the probability of choosing 3, p_2^4 is the probability of choosing 5 and p_3^4 is the probability of choosing 7. We have $\sum_{j \in [k_d]} p_j^d = 1, \forall d \in [D]$. Next, we assign an architecture parameter $\alpha_{i,j}^d \in R$ for each probability $p_{i,j}^d$, where i is the client index. We can get the probability through a softmax.

$$p_{i,j}^d = \frac{e^{\alpha_{i,j}^d}}{\sum_{j \in [k_d]} e^{\alpha_{i,j}^d}}, \forall i \in [K], d \in [D] \tag{4}$$

For each client, we can assign such architecture parameters. We use α_i to represent the model structure on each client. With each α_i^d, we can sample and generate an one-hot vector v_i^d. For example, $v^2 = [1\ 0]^\top$ means in current communication round, on the dimension of width of Conv., we choose the option of 16. We can sample the model w_i for client i with all $v_i^d, d \in [D]$ of client i, which is the sample policy in $w_i = S(\alpha_i, \mathcal{A})$. So, we can represent arbitrary supernet models in such a formulation.

To aggregate the single models into the supernet during aggregation in each round, we follow the conventional methods of aggregation in previous federated neural architecture search methods. We directly expand the single model to the same dimension as the supernet with filling in the zero and conducting aggregation. When we conduct the averaging operation, we divide the model weight with the counts of non-zero occurrences regarding each parameter.

Solve PerFedNAS via Reinforcement Learning

In the next step, we view our problem as a Markov Decision Process (MDP) and illustrate the solution to the architecture search via reinforcement learning. We use on-policy online RL here. The advantage of adapting RL methods into FL is that each client can be solved with an individual agent evaluating the action. Each client observes the reward individually.

Markov Decision Process. We view each communication round as one step of MDP and follow the conventional steps in reinforcement learning to define states and actions:

State: The state include two parts: the supernet $\mathcal{A}$ and K clients parameters α_i. Though these are the states of clients, they are stored and processed only on the server. The initial value of α_i are all set to 0 which represents that all operations have the same probabilities to be selected.

Action: Each client will have a virtual agent placed on the server responsible for searching architectures for each client. In each communication round, this agent will send a personalized model server sampled from the supernet according to the α_i to the client. Each client will then do the local training on their local dataset and return the trained model, along with test accuracy to the server. We use $a_i, i \in [K]$ to denote action of each agent.

Policy: We can use the architecture parameter we design to represent our policy. $\pi_{\alpha_i}(\mathcal{A}, a_i^t, r_i^t | \mathcal{A}, a_i^{t-1}, r_i^{t-1})$ is the policy at t communication round for each virtual agent. The state is $(\mathcal{A}, \alpha_i^{t-1})$ and takes the action a_i^{t-1} to turn state into $(\mathcal{A}, \alpha_i^t)$, according to reward r_i.

In the final state, the α_i on each client will indicate the best architecture for each client searched, the model weights on clients can be directly deployed without any further training.

Search through policy gradient. So, the key part is to optimize the policy parameters so as to find the best personalized model for each client. In each communication round, for each participated client, it will have a reward and we leverage this reward r_i to update the policy net, which is parameterized by α_i. We maximize the expected reward of the sub-model sampled from the supernet by ascending gradients:

$$\begin{aligned} \nabla J_{\alpha_i}(\alpha_i) &= \nabla_{\alpha_i} \mathbb{E}_{v_i \sim \alpha_i}[r_i] \\ &= \sum_{d \in [D]} \nabla_{\alpha_i} p_i^d r(\mathcal{A}(e = o^d)) \\ &\approx r_i \nabla_{\alpha_i} \log(p_i(v_i)) \end{aligned} \tag{5}$$

The meaning of $e = o^d$ means on the search dimension d, the chosen option is o^d. We can further calculate Equation (5)

$$\nabla_{\alpha_i^d} \log(p_i^d(v_i^d)) = \delta - p_i^d, \forall d \in [D] \tag{6}$$

where we assume the n-th entry of v_i^d is 1 and for $m \in [k_d]$:

$$\delta_m = \begin{cases} 0, m \neq n \\ 1, m = n \end{cases} \tag{7}$$

With such a formulation, we can then optimize the policy network which is one layer of softmax. This makes the server barely have extra overhead besides aggregation.

Design of reward functions. To accommodate arbitrary levels of heterogeneity in federated learning, we can design reward functions based on our objective. Here, we set our objective of solving statistical and computing speed heterogeneity with designing a reward function considering accuracy and efficiency: $r_i(w_i) = \mathrm{Acc}_i - \mathrm{RT}_i$ to add in the tradeoff between round time (RT) of each client and local inference accuracy. Since each client has a reward in each communication to reflect their performance individually, each reward will not have extreme steep changes. The round time is the time spent from sending out the model until receiving the clients' updated model parameters. However, directly using the value of accuracy and round time can cause the training to be unstable. As a result, in each round, we just need to update the accuracy of the participated clients this round in the history information and calculate average accuracy among clients. And we can still select a part of the clients to participate in each round as usual FL. Clients only need to send accuracy besides model parameters, which approximately adds no communication overhead. The round time is in real time. And $\min(\mathrm{RT})$ means the shortest round time over participated clients in each communication round. To address stability concerns, probably caused by too large variance between different clients' training time, we use λ_{time} to regulate accuracy and round time to the same scale.

$$r_i(w_i) = \mathrm{Acc}_i - \overline{\mathrm{Acc}} - \lambda_{\mathrm{time}}(\mathrm{RT}_i - \min(\mathrm{RT})) \tag{8}$$

Algorithm 1: PerFedRLNAS

Input: K clients, with objective functions $f_i, \forall i \in [K]$.
Initialize supernet $\mathcal{A}_0$, α_i.
for each round $r \leftarrow 1$ to R **do**
  On Server:
  Sample N clients: $\mathcal{N} \subseteq \{1, \ldots, K\}$.
  Each client weight $w_i \leftarrow \mathrm{Sample}(\alpha_i, \mathcal{A})$ and send.
  On Client:
  for on client $i \in \mathcal{N}$ parallel **do**
    Conduct Local Update with objective function f_i.
    Upload w_i and test accuracy to the server.
  end for
  On Server:
  Receive w_i and check the round time of client i and calculate score r_i, $i \in \mathcal{N}$.
  Aggregate w_i to update $\mathcal{A}$.
  for Each client $i \in \mathcal{N}$ **do**
    Use r_i as the reward function of each policy π_{α_i} and update α_i with policy gradient.
  end for
end for

PerFedRLNAS Algorithm

With the framework of PerFedRLNAS and the reinforcement learning solution, we can initiate personalized federated learning as shown in the Algorithm 1.The whole process is depicted in Figure 2, the server will sample the client models and distributed them to the clients. As standing on the view of clients, their duties are exactly the same as those in FedAvg, As a result, our algorithm adds no extra overhead over clients. While on the server, it conducts the detailed updates of architecture parameters through virtual RL agents for each client along with aggregation. Since the policy net is a shallow net with one layer of softmax, the overhead on the server is negligible. In each round, the server receives w_i and aggregates w_i to the supernet.

	ViT				MobileNetV3			
	CIFAR10 $D_\alpha = 0.3$		CIFAR100 $D_\alpha = 0.1$		CIFAR10 $D_\alpha = 0.3$		CIFAR100 $D_\alpha = 0.1$	
Methods	$\overline{\text{Acc}}\pm$Std	Latency	$\overline{\text{Acc}}\pm$Std	Latency	$\overline{\text{Acc}}\pm$Std	Latency	$\overline{\text{Acc}}\pm$Std	Latency
FedAvg	72.22 ± 6.70	34.95	52.70 ± 4.24	34.30	74.44 ± 6.11	19.12	47.19 ± 4.24	13.76
Local Training	60.26 ± 8.27	–	37.66 ± 5.28	–	57.60 ± 13.60	–	31.98 ± 7.08	–
FedRep	71.37 ± 6.44	35.40	48.37 ± 4.34	34.98	74.75 ± 5.97	19.42	48.75 ± 4.07	18.08
FedBABU	72.32 ± 6.28	37.11	54.35 ± 4.38	34.41	75.00 ± 5.92	17.81	48.46 ± 4.07	19.41
FedTP	80.27 ± 8.39	6.05	48.27 ± 5.62	6.74	–	–	–	–
PerFedRLNAS	**85.02 ± 4.11**	**28.14**	**65.08 ± 3.97**	**26.73**	**82.02±4.85**	**9.96**	**63.85±4.02**	**9.68**

Table 2: Comparison of accuracy and efficiency over 100 clients under cross-device scenario (participation rate 5%) using ViT and CNN structures. Higher average accuracy, lower std, and shorter elapsed wall clock time (hours) indicate better performance.

Experiments

We empirically compare PerFedRLNAS with the state-of-the-art personalized federated learning methods and previous federated neural architecture works to see how well our method solves the heterogeneous problems.

Dataset, tasks, and models. We study on image classification tasks with CIFAR10 and CIFAR100 (Krizhevsky, Hinton et al. 2009). For measuring flexibility of different methods and having a comprehensive evaluation, we employ ViT (NASViT (Gong et al. 2022) as search space, LeViT (Graham et al. 2021) as fixed model) and CNN (MobileNetV3 (Howard et al. 2019) and DARTS (Liu, Simonyan, and Yang 2019) as search space, MobileNetV3-Large as fixed model). NASViT is the supernet for one-shot and block-wise search. MobileNetV3 is the supernet for block-wise and evolutionary-based search, while DARTS is the supernet for path-wise and differentiable search.

Non-i.i.d. datasets and implementation. We use Dirichlet distribution with parameter $D_\alpha < 1$ to generate non-i.i.d. datasets. Smaller D_α indicates stronger non-i.i.d. assumptions. All experiments use the same non-i.i.d. datasets. The training samples and test samples are equally partitioned over all the clients. We mainly consider the cases of cross-device scenario, where some clients participate in the federated learning process. All experiments are performed on the federated learning framework Plato (Li et al. 2023a).

Personalization Performance

To fairly compare the results and simulate real federated learning settings, during each round, 5 out of 100 clients are selected. In each communication round, each client does the local training for 5 epochs. We evaluate performance in terms of accuracy and efficiency. For accuracy, we report average test accuracy and standard deviation over local datasets on these $K = 100$ clients. For efficiency, we directly measure the latency for the whole system to reach the reported accuracy including time spent on the client, transmission and the server. We set the upload and download data transmit rate to 100Mbps. Random seeds are fixed during all experiments. The variance between different runs of experiments is much smaller than the variance between local accuracy of different clients (e.g. **0.5** comparing to **4.11** when using PerFedRLNAS on NASViT search space, 5 times run with different seeds). All methods are trained to convergence.

Comparing with conventional methods. For conventional personalized FL, we choose FedRep (Collins et al. 2021), FedBABU (Oh, Kim, and Yun 2022) and FedTP (Li et al. 2023b) because they are representative work of personalizing part of the models on clients. More importantly, these methods show better performance than previous methods such as FedProx (Li et al. 2020), Ditto (Li et al. 2021), and SCAFFOLD (Karimireddy et al. 2020). To set up a basic benchmark, we also compare PerFedRLNAS with FedAvg (with 10 epochs post-fine tuning on the local dataset after training) and only training locally. In the method of FedTP, we directly use the structure self-designed in the paper. Since our methods have flexibility to different search space, we compare both ViT and CNN structured models and show numerical results in Table 2.

Sharing is necessary for personalization. From the results, only local training performs the worst and has a large variance across all the clients. Even the FedAvg can provide much improvement over local training. On the other hand, if we adopt a reasonable personalization policy, performance can greatly be improved, empirically showing that we need personalization for solving heterogeneity. Our method reaches a good tradeoff between the shared parts and personalized parts, achieving good performance.

PerFedRLNAS shows better performance in terms of accuracy and efficiency. The conventional state-of-the-art methods bring little improvement over FedAvg when using state-of-the-art manually tuned models. PerFedRLNAS has more improvements in terms of both accuracy and efficiency when data is more non-i.i.d. distributed. FedTP has great improvement, while it suffers from large variance. Apart from that, it also lacks the flexibility to apply itself to arbitrary transformer models, for example, LeViT, whose attention maps have different sizes over different layers. Although FedTP has smaller customized models, our method needs only 3.29 hours on CIFAR10 and 1.13 hours on CIFAR100 to reach the same accuracy. Despite that our searched models have similar architectures with manually tuned models, we

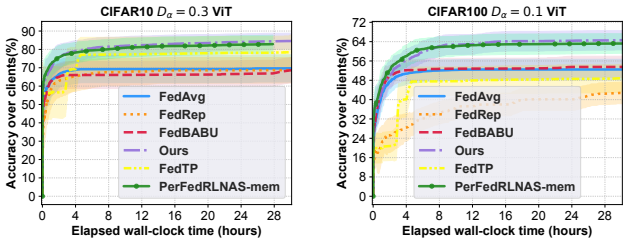

Figure 3: The average inference accuracy and the standard deviation among K local clients on search space NASViT.

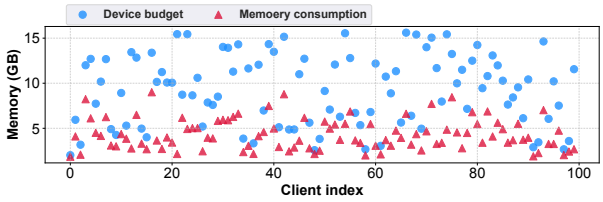

Figure 4: The memory budget and consumption on each device when we use reward functions in Equation (9).

need less communication overhead and training time.

Comparing with federated NAS methods. We compare our method with previous federated NAS work in Table 3. We omit the method DirectFedNAS as it is a similar method to FedNAS except for changing a supernet. For multi stage methods like FedNAS (He, Annavaram, and Avestimehr 2020), FedRLNAS (Yao et al. 2021), FedorAS (Laskaridis, Fernandez-Marques, and Dudziak 2022), the results are generated after all their stages are applied instead of only supernet training stage. For FedorAS, it partitions clients into four clusters and trains a model for each cluster in the final stage. So, we report the accuracy of the personalized model for each cluster. For fair comparison, all methods are implemented over DARTS search space. Since DARTS is a CNN structured search space, we can see the results obtained by federated NAS methods are better than those obtained by conventional personalized federated learning methods with CNN model MobileNetV3. The reason is that the federated NAS can seek in a larger space to find the parts that need personalization.

PerFedRLNAS shows better performance than other federated NAS baselines. In addition, PerFedRLNAS has much less transmission overhead than methods like FedNAS, SPIDER as they need to transmit the whole search space across the clients and the server.

PerFedRLNAS is a one-stage algorithm that naturally combines warm-up, post fine-tuning, and retraining. At the beginning stage of PerFedRLNAS, all architectures have the close probabilities to be sampled, which is the same as the warm-up stage. At the end stage closing to reaching convergence, as architectures are mostly fixed, the personalized parts of the models are only updated by particular clusters of clients. This process has similar functionalities as the post fine-tuning phase. As a result, PerFedRLNAS does not need warm-up or post fine-tune, but is competitive to multi-stage methods such as FedNAS, FedRLNAS, and FedorAS.

Why Can Personalizing Models Solve Heterogeneous Problem in Federated Learning?

Back to the first question, it is intuitive that different architectures can solve system heterogeneity as more powerful devices can get larger models and vice versa. Regarding statistical heterogeneity, if all clients share the same distribution, we only need one global model. However, when different clients have different distributions, the optimized architecture on each client should be different. On the other hand, each architecture they still have similarities with each other. Clients with closer data distributions will have more similar architectures. So these clients will have more parts of the model to be shared. In this way, clients can form clusters of any scale automatically without a manual definition of client clustering or revealing datasets privacy to the server. Without loss of generality, we assume we have 2 clients participating in federated learning, each holding a local dataset: X_1 and X_2 respectively. The corresponding distribution probabilities are $P_1(y|x)$ and $P_2(y|x)$ respectively for a sample x from the sample space $\mathcal{X}$ and $y \in \mathcal{Y}$. Their expected optimal architecture sampling parameters are p_1 and p_2.

Theorem 1. *For any two clients picked from federated learning setting. If $P_1 = P_2$, then $\mathbb{E}[p_1] = \mathbb{E}[p_2]$. When Kullback–Leibler divergence $D_{KL}(P_1 \| P_2)$ gets much larger, $\langle \mathbb{E}[p_1], \mathbb{E}[p_2] \rangle$ correspondingly gets smaller.*

This theorem answers the first question at the very beginning. From this theorem, it shows that applying both sharing and proper personalization, applying both model weights and architectures of personalization can benefit PFL most.

Awareness of Resource Budgets

Besides resolving the data heterogeneity and efficiency heterogeneity, we can also consider a case that different clients may have various memory budgets. Hence, we design another reward function to realize resource (memory) budget awareness to assign different architectures to clients depending on their memory limitations. We design such a reward function considering memory utilization.

$$r(w_i) = \mathrm{Acc}_i - \overline{\mathrm{Acc}} - \lambda_{\mathrm{time}}(\mathrm{RT}_i - \min(\mathrm{RT})) - \phi_i \quad (9)$$

where ϕ_i is the memory utilization of each client device. If the device cannot fit the currently assigned model into the physical (GPU) memory, its ϕ_i will be 1 and the accuracy will be 0. We set the RT_i in this case to the maximal round time of all clients participated in current round.

Here, we repeat the experiment of adopting NASViT supernet and train over the CIFAR10 $D_\alpha = 0.3$ dataset and CIFAR100 $D_\alpha = 0.1$ dataset. NASViT covers a wide range of model flops from 159.22 M Flops to 1293.98 M Flops. We simulate the memory budgets of each client with a uniform sample from 2 GB to 16GB. In Figure 3, we show the training curve of inference accuracy comparing to the use case of no memory budget in the reward function. The final test accuracy on CIFAR10 is $82.87 \pm 4.87\%$ and the training time is

	CIFAR10 $D_\alpha = 0.3$		CIFAR100 $D_\alpha = 0.1$	
Methods	$\overline{\text{Acc}}\pm$Std	Latency	$\overline{\text{Acc}}\pm$Std	Latency
FedNAS	79.88 ± 5.27	22.33	55.42 ± 4.32	22.22
FedRLNAS	79.97 ± 5.81	5.92	60.85 ± 4.43	5.80
FedorAS	$[74.32 \pm 5.87, 75.39 \pm 6.11$ $74.91 \pm 6.63, 75.52 \pm 5.92]$	6.27	$[48.23 \pm 5.18, 49.93 \pm 4.52$ $50.96 \pm 4.91, 55.41 \pm 4.35]$	7.67
SPIDER	71.14 ± 6.74	13.67	47.90 ± 5.17	14.50
PerFedRLNAS	$\mathbf{80.80 \pm 5.48}$	**5.76**	$\mathbf{62.72 \pm 4.17}$	**3.72**

Table 3: Comparison of accuracy and efficiency (in hours) over 100 clients with under cross-device scenario (participation rate 5%) between different federated neural architecture search methods, on DARTS search space.

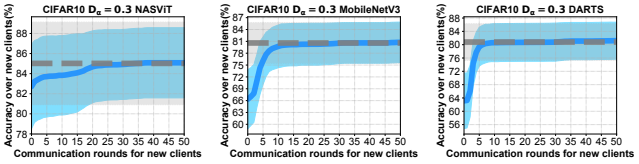

Figure 5: The average validation accuracy and the standard deviation among new 100 local clients. Gray patterns are accuracies achieved by PerFedRLNAS in previous training.

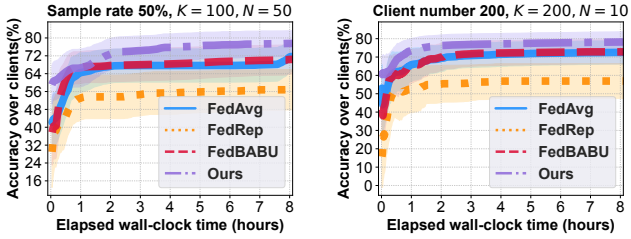

Figure 6: The average inference accuracy and the standard deviation among K local clients with different federated settings on CIFAR10 $D_\alpha = 0.3$ and search space MobileNetV3.

27.84 hours and the results on CIFAR100 are $63.24 \pm 4.07\%$ and 26.56 hours. We also show the memory budgets of each clients and the consumed memory of searched architectures on each client in Figure 4. We can see that each device can receive models of different architectures according to their memory budget. With this example, we show that PerFedRLNAS is flexible to achieve different types of heterogeneity through setting different reward functions.

Generalization Ability

After supernet training, new clients, uninvolved in federated training, solely test received models. We select 100 clients with non-i.i.d datasets, unused in federated training. Figure 5 displays the accuracy of these 100 new clients after R rounds of architecture search only. The supernet weights remain unchanged; clients neither train models nor upload weights during this process.

With high accuracy achieved and good generalization ability, PerFedRLNAS is easy to use during model deployment in inference. Even if some clients did not participate in federated training, they only need a few rounds of testing to get a better-personalized network. Though conventional methods only need one round of testing, their performance is worse. On the other hand, as we can see in Figure 5, at the start point, a randomly sampled subnet shows better performance than that by FedAvg.

Ablation Study of Different Federated Settings

Here, we study robustness in different federated settings. We change the number of participants in each round to 50 while controlling for other variables. Next we change the total client K to 200, but use the same sample rate (10%) to control the variable. From the convergence along elapsed wall-clock time shown in the Figure 6, we can see that PerFedRLNAS still outperforms the baselines by taking less time to achieve the same accuracy and converging to higher accuracy in the end. We also studied the effect of moving time items in the reward function. Over the setting CIFAR10 $\mathcal{D}_\alpha = 0.3$, using MobileNetV3, we set λ_{time} to 0. The accuracy can be improved to $83.33\% \pm 4.97$ while the latency will be 10.65 hours. The accuracy is improved but efficiency is decreased when we do not consider efficiency into the objective.

Conclusion

In this paper, we propose a new federated neural architecture search framework PerFedRLNAS. It can automatically search for customized architectures and weights on each client directly through optimization over various heterogeneity objectives including non-i.i.d awareness, efficiency awareness, and memory budget awareness. The framework gives the solution to designing personalized parts of models in personalized federated learning automatically. Finally, we show PerFedRLNAS outperforms other state-of-the-art personalized federated learning methods over non-i.i.d datasets in terms of local accuracy and efficiency. Furthermore, it generalizes well to new clients, which is an excellent property for us to deploy clients in inference easily. PerFedRLNAS is also robust in different federated settings. Our source code is released in https://github.com/TL-System/plato/tree/main/examples/model_search/pfedrlnas.

References

Acar, D. A. E.; Zhao, Y.; Matas, R.; Mattina, M.; Whatmough, P.; and Saligrama, V. 2021. Federated Learning Based on Dynamic Regularization. In *Proc. International Conference on Learning Representations (ICLR)*.

Achituve, I.; Shamsian, A.; Navon, A.; Chechik, G.; and Fetaya, E. 2021. Personalized Federated Learning With Gaussian Processes. In *Proc. Advances in Neural Information Processing Systems 34 (NeurIPS)*, volume 34, 8392–8406. Curran Associates, Inc.

Caldas, S.; Duddu, S. M. K.; Wu, P.; Li, T.; Konečnỳ, J.; McMahan, H. B.; Smith, V.; and Talwalkar, A. 2018. LEAF: A Benchmark for Federated Settings. arXiv:1812.01097.

Collins, L.; Hassani, H.; Mokhtari, A.; and Shakkottai, S. 2021. Exploiting Shared Representations for Personalized Federated Learning. In *Proc. the 38th International Conference on Machine Learning (ICML)*, volume 139, 2089–2099. PMLR.

Diao, E.; Ding, J.; and Tarokh, V. 2021. HeteroFL: Computation and Communication Efficient Federated Learning for Heterogeneous Clients. In *Proc. International Conference on Learning Representations (ICLR)*.

Elsken, T.; Metzen, J. H.; and Hutter, F. 2019. Neural Architecture Search: A Survey. *The Journal of Machine Learning Research (JMLR)*, 20(1): 1997–2017.

Fallah, A.; Mokhtari, A.; and Ozdaglar, A. 2020. Personalized Federated Learning with Theoretical Guarantees: A Model-Agnostic Meta-Learning Approach. In *Proc. Advances in Neural Information Processing Systems 33 (NeurIPS)*, volume 33, 3557–3568. Curran Associates, Inc.

Garg, A.; Saha, A. K.; and Dutta, D. 2021. Direct Federated Neural Architecture Search. In *Proc. ICLR 2021 Workshop on Distributed and Private Machine Learning (DPML)*.

Gong, C.; Wang, D.; Li, M.; Chen, X.; Yan, Z.; Tian, Y.; qiang liu; and Chandra, V. 2022. NASViT: Neural Architecture Search for Efficient Vision Transformers with Gradient Conflict aware Supernet Training. In *Proc. International Conference on Learning Representations (ICLR)*.

Graham, B.; El-Nouby, A.; Touvron, H.; Stock, P.; Joulin, A.; Jégou, H.; and Douze, M. 2021. LeViT: A Vision Transformer in ConvNet's Clothing for Faster Inference. In *Proc. the IEEE/CVF International Conference on Computer Vision (ICCV)*, 12259–12269.

He, C.; Annavaram, M.; and Avestimehr, S. 2020. FedNAS: Federated Deep Learning via Neural Architecture Search. In *Proc. CVPR 2020 Workshop on Neural Architecture Search and Beyond for Representation Learning*.

He, X.; Zhao, K.; and Chu, X. 2021. AutoML: A Survey of the State-of-the-art. *Knowledge-Based Systems (KBS)*, 212: 106622.

Howard, A.; Sandler, M.; Chu, G.; Chen, L.-C.; Chen, B.; Tan, M.; Wang, W.; Zhu, Y.; Pang, R.; Vasudevan, V.; Le, Q. V.; and Adam, H. 2019. Searching for MobileNetV3. In *Proc. the IEEE/CVF International Conference on Computer Vision (ICCV)*, 1314–1324.

Karimireddy, S. P.; Kale, S.; Mohri, M.; Reddi, S.; Stich, S.; and Suresh, A. T. 2020. SCAFFOLD: Stochastic Controlled Averaging for Federated Learning. In *Proc. the 37th International Conference on Machine Learning (ICML)*, volume 119, 5132–5143. PMLR.

Krizhevsky, A.; Hinton, G.; et al. 2009. *Learning Multiple Layers of Features from Tiny Images*. Master's thesis, University of Toronto.

Laskaridis, S.; Fernandez-Marques, J.; and Dudziak, Ł. 2022. Cross-device Federated Architecture Search. In *Proc. Workshop on Federated Learning: Recent Advances and New Challenges (in Conjunction with NeurIPS 2022)*.

Li, B.; Su, N.; Ying, C.; and Wang, F. 2023a. Plato: An Open-Source Research Framework for Production Federated Learning. In *Proc. the ACM Turing Award Celebration Conference-China (TURC)*, 1–2.

Li, H.; Cai, Z.; Wang, J.; Tang, J.; Ding, W.; Lin, C.-T.; and Shi, Y. 2023b. FedTP: Federated Learning by Transformer Personalization. *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, 1–15.

Li, T.; Hu, S.; Beirami, A.; and Smith, V. 2021. Ditto: Fair and Robust Federated Learning Through Personalization. In *Proc. the 38th International Conference on Machine Learning (ICML)*, volume 139, 6357–6368. PMLR.

Li, T.; Sahu, A. K.; Zaheer, M.; Sanjabi, M.; Talwalkar, A.; and Smith, V. 2020. Federated Optimization in Heterogeneous Networks. In *Proc. Machine learning and systems (MLSys)*, volume 2, 429–450.

Liu, H.; Simonyan, K.; and Yang, Y. 2019. DARTS: Differentiable Architecture Search. In *Proc. International Conference on Learning Representations (ICLR)*.

Marfoq, O.; Neglia, G.; Vidal, R.; and Kameni, L. 2022. Personalized Federated Learning through Local Memorization. In *Proc. he 39th International Conference on Machine Learning (ICML)*, volume 162, 15070–15092. PMLR.

McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; and Arcas, B. A. y. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proc. the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 54, 1273–1282. PMLR.

Mushtaq, E.; He, C.; Ding, J.; and Avestimehr, S. 2021. SPIDER: Searching Personalized Neural Architecture for Federated Learning. arXiv:2112.13939.

Oh, J.; Kim, S.; and Yun, S.-Y. 2022. FedBABU: Toward Enhanced Representation for Federated Image Classification. In *Proc. International Conference on Learning Representations (ICLR)*.

Pillutla, K.; Malik, K.; Mohamed, A.-R.; Rabbat, M.; Sanjabi, M.; and Xiao, L. 2022. Federated Learning with Partial Model Personalization. In *Proc. the 39th International Conference on Machine Learning (ICML)*, volume 162, 17716–17758. PMLR.

Qu, L.; Zhou, Y.; Liang, P. P.; Xia, Y.; Wang, F.; Adeli, E.; Fei-Fei, L.; and Rubin, D. 2022. Rethinking Architecture Design for Tackling Data Heterogeneity in Federated Learning. In *Proc. the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 10061–10071.

Shamsian, A.; Navon, A.; Fetaya, E.; and Chechik, G. 2021. Personalized Federated Learning using Hypernetworks. In *Proc. the 38th International Conference on Machine Learning (ICML)*, volume 139, 9489–9502. PMLR.

Tan, A. Z.; Yu, H.; Cui, L.; and Yang, Q. 2023. Towards Personalized Federated Learning. *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, 34(12): 9587–9603.

Yao, D.; Wang, L.; Xu, J.; Xiang, L.; Shao, S.; Chen, Y.; and Tong, Y. 2021. Federated Model Search via Reinforcement Learning. In *Proc. the IEEE International Conference on Distributed Computing Systems (ICDCS)*, 830–840.

Zhu, H.; and Jin, Y. 2021. Real-time Federated Evolutionary Neural Architecture Search. *IEEE transactions on evolutionary computation (TEC)*, 26(2): 364–378.