Solving Heterogeneity via Personalized Federated Learning

Dixi Yao
dixi.yao@mail.utoronto.ca
University of Toronto

***Abstract*—Federated learning has emerged as an important paradigm in modern distributed machine learning. However, different from conventional distributed learning, the clients in federated learning are placed in a wild environment where clients do not have consensus over data, systems, privacy and others. What's more, the heterogeneity problems in federated learning can greatly affect performance. Personalized federated learning is a method to customize the models on each client and try to find efficient and effective ways to let there clients share particular knowledge and personalize, so as to achieve the best performance over local data.**

In this paper, we go through several representative personalized federated learning methods, particularly over three types: global model, local customized model and AutoML based methods. We fairly compare these methods on board with consideration of effectiveness, feasibility and ubiquitousness. We also do some experiments to show that some methods are not as effective as they were claimed in the original papers. Then we delve deeper into the federated neural architecture search based methods and find out it is a promising direction to solve the heterogeneity problems despite of several drawbacks. On the basis of that, we give out several directions for future work about how to improve current federated NAS based methods and make them promising. We also give out discussion about future work on designing better personalized federated learning methods.

I. Introduction

Federated learning [1] has emerged as an important paradigm in modern large-scale machine learning. Unlike in traditional centralized learning where models are trained using large datasets stored in a central server, in federated learning, the training data remains distributed over a large number of clients, and these clients could be any devices such as mobile phones, IoT services or edge servers. In each communication round, a subset of clients are selected to conduct local training and send updates on models to the server. The server then aggregate their updates and train a global model [2].

However, since these clients are local clients and they can be any device and the communication between

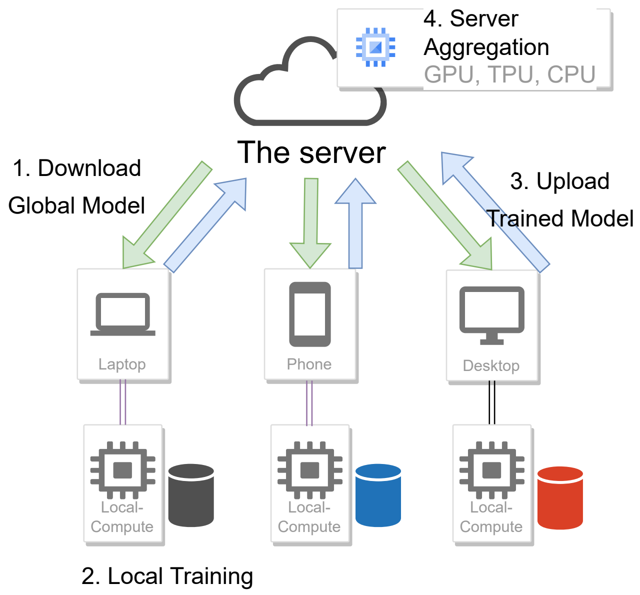

Fig. 1. The flow chart of typical federated averaging in federated learning

clients and server is also different with the conventional distributed learning in central server: unreliable and unstable network connection. As a result, a key challenge in federated learning is dealing with heterogeneity [3]. One of the typical concerns of heterogeneity is statistical heterogeneity, where the most famous problem is the none independent and identical distribution problem. In a federated setting with statistical heterogeneity, different clients can hold their own datasets and these datasets do not follow the rule of independent and identically distributed data. Though, it has been studied by large ranges of researchers, it has not been well-solved. Non-i.i.d problem makes federated learning much harder to converge to optimization point [4] and can greatly disturb the convergence rate.

In this paper, we will go through a wide range of heterogeneity concerns in federated learning including: statistical, system, privacy, model and task heterogeneity. Table I summarizes the cause and the detailed presentation forms of these heterogeneity concerns in federated

TABLE I
SUMMARIZATION OF FIVE TYPICAL POSSIBLE HETEROGENEITY IN FEDERATED LEARNING.

Type of Heterogeneity	Causes and forms
Statistical	Local datasets are of non-i.i.d, the gradients on each client diverge a lot.
System	Round time on each client diverges greatly because of various computation abilities and communication contexts.
Privacy	Different clients use different policy to protect privacy, for example, different gaussian noise in differential privacy.
Model	Different clients use different model structures.
Task	Different clients have different local objectives to optimize.

learning.

In federated learning, we narrow our scope to horizontal federated learning and the conventional update way in such setting is doing federated averaging [1]. As shown in figure 1, in each communication round the server delivers a global model to each client, then the client do the local training over local datasets. After that, each client sends the trained model back to the server and the server will aggregate these models to generate a new global model. So, we can call this form of aggregation the weight sharing scheme in federated averaging. Then, naturally, we will be raised with a question:

Is the Weight Sharing Scheme in Federated Averaging reasonable when we meet the heterogeneity?

Before answering this question, let's see the solution to heterogeneity, which is called **personalized federated learning** where the models on the clients and the server are no longer a same global model but they can have their personalized parts. The start point of using personalized learning to solve heterogeneity in federated learning comes from the idea of applying multi-task learning into federated learning [5]. It is useful to solve statistical challenges and system-aware optimization after putting multi-task learning into a distributed manner.

Many methods have been purposed via personalized federated learning. We can divide them into three categories: global model, customized local model, and AutoML methods. In the global model method, the server is the keypoint to the solution where usually, it will have specially designed aggregation or optimization methods, to fit a universe model suited for all clients. Normally, after federated learning is done, clients can choose to finetune customized models on their own datasets further. In the customized local model method, each client owns a model with customized weights or even a customized model structure. In each communicate round, part of the model on a client is only trained by itself and part of the model is used for aggregation on the server. In these two methods, researchers stepped forward to design and add special regularization forms to the formulations to

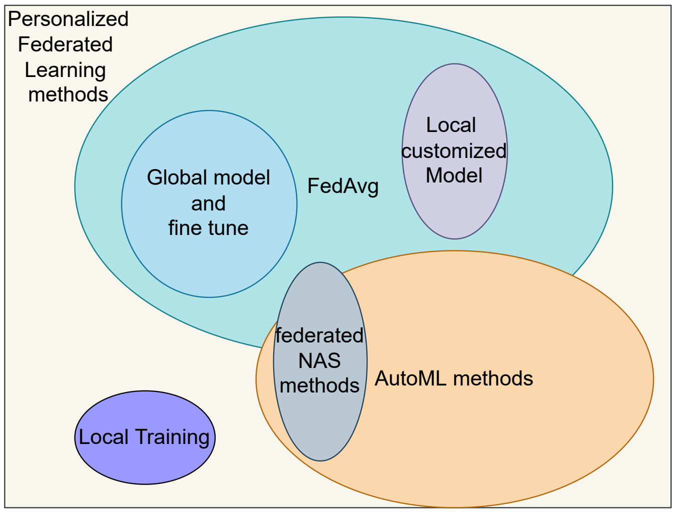

Fig. 2. The inclusion relationship of different types of personalized federated learning methods.

achieve a better solution.

The third method is more novel where they leverage Auto Machine Learning (AutoML) methods to let clients and servers automatically find the suitable models and weights for different heterogeneous settings. In this method, the information or knowledge is no longer shared simply in the process of weight aggregating. Clients can share more high-level knowledge to better customize their local models. One solution under this method is using knowledge distillation, where each client can have a completely different structure of models while they share their local dataset knowledge through logits, the same as manners of knowledge distillation. Another more interesting solution is using Federated Neural Architecture Search to allow users to share more diverse knowledge through the supernet. Implicit knowledge is shared between clients; for example, different clients will select a different neural architecture as their local model and this can implicitly reflect the customized information of each client. We summarize the inclusion relationship in figure 2.

II. Problem Formulation

In personalized Federated Learning, we have K clients in total, and each client has its local objective, where we denote it as $F_i, \forall i \in [K]$. We use w_i to denote the model weights of each client's local model. In personalized federated learning w_i do not need to and usually not to be the same as global weights but customized weights, specific for each client. The problem is

$$\min_{w_i} F_i(w_i), \forall i \in [K] \tag{1}$$

This is a general formulation for personalized federated learning where we are trying to make performance on each local dataset good.

III. Examining Methods of Personalized Federated Learning

In this section, we will have a through examination of these three approaches to solving heterogeneity and have a fair comparison of their pros and cons. We will compare these methods in mainly three considerations: whether they are effective, feasible and ubiquitous. Regarding effectiveness, the solution should have a clear theoretical convergence guarantee and can empirically solve the heterogeneity problem.

Regarding being feasible, we will consider two sides. First, the solution should have a proper cost when implemented in the federated setting. And the assumptions proposed in the papers should be reasonable in real worlds. For example, the assumption that the server is an oracle having all clients information is invalid. Regarding of being ubiquitous, we will evaluate whether the method only fits for the problem setting in the papers and the solution is aimed for a particular kind of heterogeneity or can solve different kinds of heterogeneity at the same time. On the other hand, the method should be flexible and easy to extend. For example, for method FedAvg, we can use any model, no matter if it is a convolution neural network, recurrent neural network, graph network or transformer. If the method can only be applicable for a small class of models or only one or two specific models, it is not flexible. So, it is not feasible enough.

For evaluation of ubiquitous, we will check if each method is applicable to the following five kinds of heterogeneity: statistical, system, privacy, model and task heterogeneity. Statistical heterogeneity means the classical problem of solving clients with non-i.i.d dataset. System heterogeneity means that each client can have different local update step [6], different local training time and communication time due to heterogeneous network, resources and system settings. Privacy heterogeneity means different clients have different privacy budgets; for example, they can add different Gaussian noise according to different ϵ requirement according to differential privacy [7]. Model heterogeneity is straightforward: each client can hold a different structure of local models. Task heterogeneity means that clients can solve different tasks such as vertical federated learning or transfer federated learning [8]. Another example of task heterogeneity is we can consider a toy federated learning setting where we have only three tasks: client one needs to classify categories 1,2,3,4,5 in CIFAR10 [9], client two needs to classify categories 6,7,8,9,10 in CIFAR10 and client three needs to classify categories 1,3,5,7,9, in CIFAR10.

A. Global Model and Fine Tune

The simplest way for personalized federated learning is FedAvg [4] (baseline of baselines) and actually, it is proven that FedAvg can still converge on non-i.i.d data: a convergence rate of $\mathcal{O}(\frac{1}{T})$ for strongly convex and smooth problems, where T is the number of SGDs. However, such a method produced poor performance over evaluation metrics such as accuracy. As a result, a matter of course is to fine tune global models on clients for several steps on the local dataset. However, this can lead to them having poor generalization ability. This method is simple and easy to implement, and also ubiquitous. However, unfortunately, it is not effective enough.

PFedMe [10] leveraged the idea in conventional non-personalized federated learning to add a regularization form, so as to train a global model which can be adapted to different client. Then, they use Moreau Envelope Algorithm to solve their formulation with regularization. Qu et al. [11] rethink the problem in another angle. They argue that simply changing the model from a convolution network into a vision image transformer (ViT) can greatly help tackle the data heterogeneity problem in federated learning.

B. Local Customized Model

Normally, global model methods still need to fine-tune the trained global model on their own datasets to fully realize personalization. As a result, a more elegant method is to encapsulate such a process into the federated learning process. The general idea is to let the model on clients have some parts of weights only locally trained, which we call them personalized parts and have some parts of weights to participate in the aggregation in federated averaging, which we call them shared parts.

FedRep [12] define that several last layers of the model as classifier and the rest of the models are representation. Then, in each local training round, each client

TABLE II
SUMMARIZATION OF SHARED AND PERSONALIZED PARTS OF A MODEL IN TYPICAL METHODS FOR IN PERSONALIZED FL.

Method	Shared Parts	Personalized Parts	Shared Policy	Shared Dimension
FedAvg [1]	Whole	None	-	Whole
Local Training	None	Whole	-	Whole
PFedMe [10]	Whole	None	-	Whole
Qu et al. [11]	Whole	None	Use ViT models	Whole
FedRep [12]	Representation	Classifier	Manual Setting	Layer
FedBabu [13]	Representation	None	Manual Setting	Layer
FedTP [14]	Except Attention Maps	Attention Maps	Manual Setting	Layer
GCN-based [15]	Whole	Whole	GCN decided	Weights
APFL [16]	$\bar{w}$	$w_{loc,i}$	α_i decided	Weights
L2GD [17]	$\bar{w}$	w_i	λ decided	Weights
Ditto [18]	$\bar{w}$	w_i	λ decided	Weights
FedHN [19]	Hypernet	Whole	Hypernet decided	None FedAvg scheme
FedPerAvg [20]	Meta parameters	Whole	MAML decided	None FedAvg scheme
PerFedCKT [21]	Distilled knowledge	Whole	KD decided	None FedAvg scheme
FML [22]	Proxy global model	Whole	KD decided	None FedAvg scheme
FedNAS [23]	Global supernet	Model structure	DARTS decided	NAS
Direct FedNAS [24]	Global supernet	Model structure	DSNAS decided	NAS
SPIRDER [25]	Server's supernet	Model structure and part of the weights	Progressive NAS decided	NAS
FedRLNAS [26]	Server's supernet	Model structure and part of the weights	RL decided	NAS
FedorAS [27]	Whole	None	FedorAS search space decided	NAS

will have a few more local steps to update the classifier and only share the representation for global aggregation. While on the other hand, FedBaBU [13] argues that, classifier should not never be updated and we just need to share representation in global aggregation can better tackle the data heterogeneity problems. APFL [16] views the shared policy in another dimension and defines the personalized model on each client i as

$$w_i = \alpha_i w_{loc,i} + (1 - \alpha_i)\bar{w} \tag{2}$$

where $h_{loc,i}$ is the local personalized weights of the model and $\bar{h}$ is the shared global model weights. And α_i is a parameter for each client. Each client will train its own personalized weights and also do training to contribute to the shared global model weights. FedTP [14] delves deeper particularly into the ViT and transformer models, they only personalized the attention map of transformer models on each client and the rest parts remain the shared parts. Chen et al. proposed an interesting GCN-method [15] to solve such problem. They use a graph convolution network (GCN) to aggregate each local model uploaded by each client, along with a graph network containing the relationship between these clients. Then, they use that GCN to generate personalized weights for each client.

To further improve the generalization ability of models and add constraints over the distances between model weights. Many methods combine the technique of regularization. L2GD [17] introduces the personalization of weights on the clients through a L2 penalty parameter and the new formulation of the problem is

$$\min_{w_i} F_i(w_i) + \frac{\lambda}{2K} \sum_{j \in [K]} \|w_i - \bar{w}\|^2, \forall i \in [K] \tag{3}$$

where $\bar{w}$ is the shared global weights. Ditto [18] adopted the similar formulation as L2GD but provide a better proof of convergence and generalization ability.

C. AutoML

Local Customized Models seems to be promising enough; however they still cannot resolve the heterogeneity problems well enough. This naturally arose the question: Is the Weight Sharing Scheme the most suitable for heterogeneous federated learning? The answer is non-trivial and actually weight sharing neglects some important but implicit information to share between clients, which can possibly be critical to personalized federated learning. As a result, a series of personalized federated learning methods based on AutoML are proposed to dig out implicit information.

Instead of doing aggregation on the server, FedHN [19] uses a hypernet on the server which inputs are embedding vectors representing each client

TABLE III
EVALUATION TYPICAL METHODS IN PERSONALIZED FEDERATED LEARNING CONSIDERING EFFECTIVENESS, FEASIBILITY AND UBIQUITOUSNESS.

Method	Theoretically effective	Empirically effective	Cost feasible	Extension feasible	Ubiquitous (Statistical, System, Privacy, Model, Task)
FedAvg [1]	✓	✓	✓	✓	(×, ×, ×, ×, ×)
Local Training	✓	✓	✓	✓	(×, ×, ×, ×, ×)
PFedMe [10]	✓	✓	✓	✓	(✓, ×, ×, ×, ×)
Qu et al. [11]	✓	✓	✓	✓	(✓, ×, ×, ×, ×)
FedRep [12]	✓	✓	✓	✓	(✓, ×, ×, ×, ×)
FedBabu [13]	×	✓	✓	✓	(✓, ×, ×, ×, ×)
FedTP [14]	×	✓	✓	×	(✓, ×, ×, ×, ×)
GCN-based [15]	×	✓	×	×	(✓, ×, ×, ×, ×)
APFL [16]	✓	✓	✓	✓	(✓, ×, ×, ×, ×)
L2GD [17]	✓	✓	✓	✓	(✓, ×, ×, ×, ×)
Ditto [18]	✓	✓	✓	✓	(✓, ×, ×, ×, ×)
FedHN [19]	×	✓	✓	×	(✓, ✓, ✓, ✓, ✓)
FedPerAvg [20]	✓	✓	✓	✓	(✓, ×, ✓, ×, ×)
PerFedCKT [21]	✓	✓	×	✓	(✓, ✓, ✓, ✓, ×)
FML [22]	×	✓	✓	✓	(✓, ✓, ✓, ✓, ✓)
FedNAS [23]	×	✓	×	✓	(✓, ✓, ✓, ✓, ✓)
Direct FedNAS [24]	×	✓	×	×	(✓, ✓, ✓, ✓, ✓)
SPIRDER [25]	×	✓	×	×	(✓, ✓, ✓, ✓, ✓)
FedRLNAS [26]	×	✓	×	×	(✓, ✓, ✓, ✓, ✓)
FedorAS [27]	×	✓	×	×	(✓, ✓, ✓, ✓, ✓)

and the hypernet will directly output the personalized weights for each client. However such a hypernet cannot be easily trained. FedPerAvg [20] innovatively introduces a model-Agnostic meta-learning (MAML) method to train and learn meta parameters. Then each client uses these meta parameters to generate the personalized model weights. PerFedCKT [21] leverages another way for weight sharing. Clients no longer update the trained model weighs. Instead they update the logits to the server and the server downloads the loss to clients. The clients use these losses to train personalized models with methods of knowledge distillation (KD). However the drawback is that to do knowledge distillation, the server and the clients should both hold the same public datasets. FML [22] also leverages knowledge distillation methods. But different from previous methods, they assign a personalized model on each client and there is also another shared global model for aggregation. The knowledge distillation is conducted locally, between each personalized model and the shared global model on each client. In this way, no further public datasets are needed and each client can hold models of different structures and weights, which is a better method comparing to PerFedCKT.

Besides, Meta Learning and Knowledge Distillation, there exists a new kind of AutoML, personalized federated learning, what we call it FedNAS, which allows the clients to decide the structure of their local models and the parts of weights to be shared automatically. We will discuss this detailedly in the next section.

IV. New Aggregating Scheme: personalized federated NAS (PerFedNAS)

A. Neural Architecture Search

Neural Architecture Search (NAS) is a powerful tool for automating efficient neural architecture design. It often targets at searching for the best model in a search space under given efficiency-related constraints. A particular promising NAS scheme is one-shot supernet-based NAS, where the search space is a very big super net that can directly output targets. The candidate networks are a subset of the supernet, and once the supernet is fully trained, we can directly sample a subnet as a trained model instead of retraining that model in a two-stage NAS. It has been a normal method to use supernet-based NAS in computer vision [28, 29, 30, 31, 32]. Here, we introduce some popular supernets in previous NAS works.

NASVIT [28] is a one-stage one-shot supernet based neural architecture search where a huge over-parameterized supernet is firstly well-trained and then

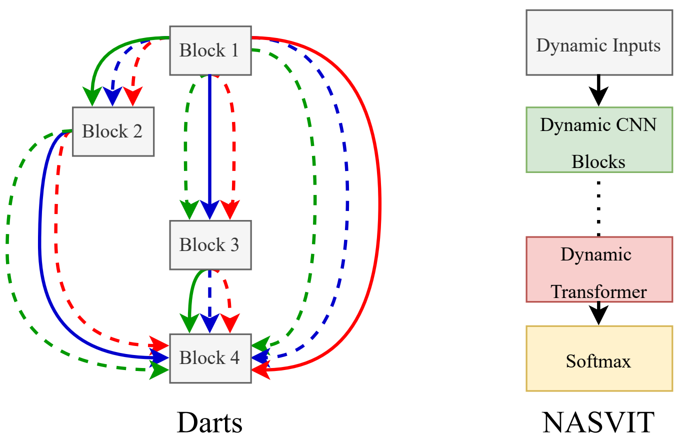

Fig. 3. The searching space of DARTS and NASVIT

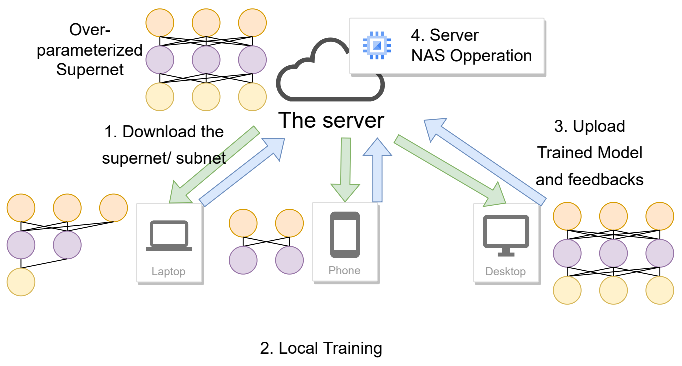
Over-parameterized Supernet
4. Server NAS Opperation
The server
1. Download the supernet/ subnet
3. Upload Trained Model and feedbacks
Laptop
Phone
Desktop
2. Local Training

Fig. 4. The framework of general personalized federated neural architecture search

they use a random forest model to search for the best model structure. DARTS [29] is another supernet based neural architecture search. It uses a method of differential gradients method to solve NAS problem where the architecture parameters can be directly optimized through the backward chain rule from the loss. Figure 3 shows the search space of these two popular supernets and we can see different neural architecture search methods have different definition on the search space and the NAS related works are also fast evolving. There also exist a lot of other NAS work and they are not limited to computer vision area, also including natural language processing, graph areas and these wide areas of NAS all have good applications.

B. Federated Neural Architecture Search and Personalized federated neural architecture search

Federated neural architecture search, though is not initially designed for solving personalized federated learning, is powerful in solving such a problem. In federated neural architecture, which parts of the model should be personalized, which parts of the model should be shared and for the shared parts, they are shared among which clients can be automatically searched in the framework without any manual interference.

Figure 4 shows the framework of general personalized federated neural architecture search. The server will hold a very huge over-parameterized supernet and in each communication round it will send the supernet or the subnet depending on the detailed algorithm. After local training, the clients will report the trained weights along with some feedback. Then the server will use them to do specific algorithm.

FedNAS [23] is the first work to apply neural architecture search into federated learning setting. They directly replace the traditional model with the DARTS supernet and send the whole huge supernet between the clients and the server. Direct FedNAS [24] is similar to the original FedNAS, but they change the search space into a more efficient search space DSNAS [33]. SPIRDER [25], also assigns the whole NAS supernet to each client and each client will first do NAS locally over local datasets and the server use a progressive NAS method to aggregate these supernets instead of federated averaging. FedRLNAS [26] is the first work that the server only sends the subnet to the clients and the clients do the local training and updating with subnets, the same as in FedAvg. The FedRLNAS pipeline has three stages: warmup, searching and training. FedorAS [27] improves the FedRLNAS with only having one stage in the pipeline. After the federated neural architecture searching is done, we can obtain a global model. They do not need a warmup and re-training phase.

Existing federated NAS work provides a good direction to solve heterogeneity in federated learning. On the other hand, they still exist several drawbacks which make them still cannot work as feasibly as the methods introduced in the last section.

Pros:

1) First, federated neural architecture search has a natural advantage to assign model of any structures to each client. As a result, clients can have the maximum flexibility to hold different models so as to solve different heterogeneity objectives.
2) Second, federated neural architecture search leverage an automatic way to find how to do the weight sharing. This kind of weighted sharing policy can help such methods achieve the best performance.
3) Third, federated neural architecture search adopt another scheme of aggregation on the server instead of averaging aggregation. The new aggregation methods can jump out of the box and provide

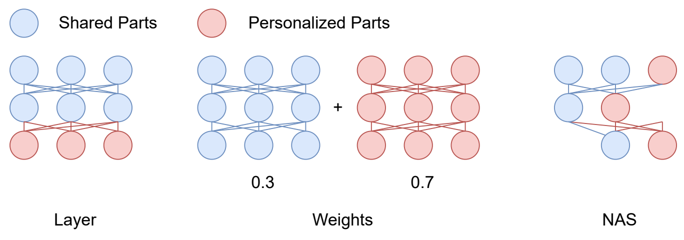

Fig. 5. Illustration of three kinds of weight sharing dimension.

more space for future researchers to find new promising algorithms.

Cons: However, existing FedNAS papers have three problems.

1) First, FedNAS [23], Direct FedNAS [24], FedorAS [27], and FedRLNAS [26] only searched a global model for all clients and relied on further local fine-tune. This scenario makes them cannot fully take advantage of federated NAS and loses the meaning of doing that. Though indeed through NAS, they can find a better model architecture and train better weights, which can improve the global accuracy, they just replicate the advantage of NAS in centralized NAS onto federated learning. But not take the advantage of federated NAS into federated learning.
2) Second, other work such as SPIRDER [25], FedNAS [23] directly conducted neural architecture search on clients. In federated learning settings, the devices' resources can be constrained and neural architecture search needs large computation resources. It is infeasible in a real system to conduct a heavy NAS workload on clients.
3) Third, which is also the most constrained point of current federated NAS method, is they are lack of flexibility. In each work of federated NAS, they define a specific NAS search space and their methods can be only applicable over such search space. If we want to use another search space, for example replace the DARTS search space with NASVIT, FedRLNAS may not work. However, as we have mentioned that the NAS search space evolves very fast, if the current federated NAS methods cannot adapt to new NAS search space, the problems it can solve will be very limited. As a result, a new federated NAS which allows such flexibility should be proposed.

V. Summarization

The key to solve the heterogeneity problems in federated learning is how to design a good **personalized weight sharing policy**. Table II summarizes the how each existing method design the weight sharing policy, and how each method define the shared parts of the models and the personalized parts of the models on each client. Different methods use different sharing dimensions. As shown in figure 5, layer dimension means some layers of the model is used as the shared parts and other layers are used as the personalized part. While for weights dimension, we both have a shared model and a personalized model and use some factors to add them with weights like 0.3 and 0.7 here. For NAS sharing dimensions and those methods not using the FedAvg scheme, the sharing can happen at arbitrary dimensions. Also, different methods use a different policy to assign the shared and personalized parts of the model. As we can see, some methods use manual setting while other methods use some parameters to optimize and decide such things. However, a natural question arises:

Can we use some automatic way to directly assign the shared policy based on our objective?

Among these methods, a promising answer is the method of AutoML, where we automatically search for the shared policy. As a result, if we want to solve the heterogeneity problems in personalized federated learning, we should design a method, which can provide the clients as much flexibility as possible to design their search policy and optimize directly by the clients and the server, instead of using any manual settings.

Back to our three considerations for evaluating methods, we now compare these methods of effectiveness, feasibility and ubiquitousness. Table III summarizes the comparison of all personalized federated learning methods mentioned in this paper. The common problem with conventional personalized federated learning methods is their lack of ubiquitousness. Actually, though in their papers, they argue that they can tackle the statistical system. I also tried some of these methods.

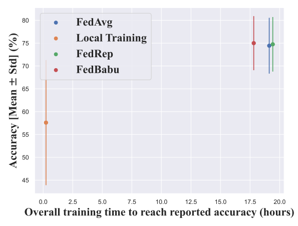

Fig. 6. Comparison of different personalized federated learning methods over Cifar10 dataset with non-i.i.d. parameter 0.3.

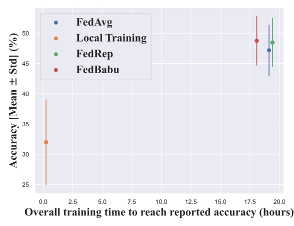

Fig. 7. Comparison of different personalized federated learning methods over Cifar100 dataset with non-i.i.d. parameter 0.1.

Experimental evaluation of typical personalized federated learning methods. Here we use the dataset

CIFAR10 [34]. We control the statistical heterogeneticity of CIFAR10 by randomly partitioning samples with the same label among clients according to Dirichlet distribution with parameter $\alpha = 0.3$. And we also use the dataset CIFAR100 [34] with parameter $\alpha = 0.1$. In the implementation, we use 100 clients and during each communication round 5 clients are sampled. We conducted 5 epochs of local training over the local datasets. As CIFAR10 and CIFAR100 have 60000 samples, we sample 600 samples on each client, and 500 of them are used for training and the rest are used for validation to report the accuracy. We use the MobileNetV3-Large [35] model as the backbone network, which is one of the state-of-the-art model in federated learning settings. The training hyper-parameters are the same as those reported in the MobileNetV3 paper except the batch size is 32. Due to the time limitation, I only ran the experiments each once but with a fixed seed. I used the open source federated learning framework `plato`[1][2] to do the simulation experiments. `Plato` a new software framework to facilitate scalable federated learning research. I report the mean accuracies and standard deviation over these 100 clients and also the overall wall clock training time. As shown in figure 6 and figure 7, the improvements over FedAvg are minor. While on the other hand, most of them are easy to extend with different model structures. Regarding AutoML methods except FedPerAvg, they are well ubiquitous but are lack of theoretical analysis. Regarding federated neural architecture search based methods, we will later discuss these drawbacks detailedly and how to solve them.

VI. Discussion and Future Work

From analysis over previous methods in personalized federated learning, we can see there exist various drawbacks in different methods. Among these methods, though federated neural architecture search based exist drawbacks, they still provide a promising direction for designing next generation personalized federated learning algorithms to solve heterogeneity problems. As a result, I would like to provide several insights about how to improve the existing federated neural architecture based methods and solve existing problems.

Algorithm 1 is the improved PerFedNAS algorithm. We still hold a huge supernet on the server. While during each communication round, the server should only assign a small sampled subnet instead of the complete supernet the the clients. The client then can use the small subnet to do local training. In this form the communication

[1]https://platodocs.netlify.app/

[2]https://github.com/TL-System/plato

Algorithm 1 Improved PerFedNAS

```
Input: $N$ clients, with objective functions $F_i, \forall i \in [N]$.
Initialize supernet $\mathcal{A}_0$.
for each round $r = 1 \cdots R$ do
    On Server:
    sample clients $N \subseteq \{1 \cdots K\}$
    each client weight $w_i \leftarrow$ Sampled subnet from $\mathcal{A}$) and send.
    On Client:
    for on client $i \in N$ parallel do
        Conduct Local Update with objective function $F_i$.
    end for
    On Server:
    Receive $w_i$ and calculate score $r_i$, $i \in N$.
    Aggregate $w_i$ to update $\mathcal{A}$.
    Use $r_i$ to do some NAS related operations.
end for
```

and computation overhead for the clients are just the same as those in the FedAvg. Then the clients report the feedback and the client weights back to the server. The server should design a particular algorithm to compute a score r_i for each client and design a particular policy for each client, so as to let each client hold different models instead of still using a global model. In this way, we can solve the problem of only searching a global model and directly conducting NAS on the clients.

Such an algorithm only solves the problem of cost infeasibility. As a result, regarding the future work. We still have two points that need to be improved.

- Provide a theoretical analysis about convergence guarantee. The lack of convergence analysis is a common drawback for all existing federated neural architecture search work. As a result, providing a theoretical analysis would be a promising direction for future work.
- Make the framework extension feasible. As we have discussed, NAS are fast evolving. Hence, how to design an algorithm that can be applicable to wide ranged or arbitrary NAS search space is an important future direction.

Besides, these work, there also exist another interesting problem for federated neural architecture search. For future work, we can design a special neural architecture search space or supernet in particular for federated learning. Different from centralized NAS search space, there exist specific challenges in federated learning, for example, the heterogeneity which is the main challenge we have discussed in this paper. It would be an interesting and promising direction to design a special search space which is more privacy preserving or more efficient or maybe can yield higher performance.

Although in this paper, we put more emphasis on the federated neural architecture search based methods than other AutoML methods such as methods using meta-learning, hypernet and knowledge distillation. It is also an interesting problem that if we can leverage these useful AutoML tools to solve the problem. For example, how to design a promising hypernet to generate model weights so as to make it extension feasible. Also, for example, FML is a promising method leveraging knowledge distillation and how to provide a theoretical analysis in the future work is waiting to be answered. We can also explore the infinite possibilities of these AutoML methods.

Another problem we have discussed is that different methods have their own definitions in heterogenous settings. And as shown in our tiny experiments, some methods which are claimed useful in their papers actually do not perform so well in simulation tests and can provide minor improvements over baselines. As a result, a promising future work would be to create a fair benchmark and federated learning settings particularly for personalized federated learning and we have a fair benchmark to compare these methods numerically on board.

VII. Conclusion

In this paper, we study the problem of how to solve general heterogeneity problems in federated settings. We investigate through three types of personalized federated learning methods to solve such problem, which are global model, customized local model and AutoML-based methods respectively. We fairly compare these methods considering their effectiveness, feasibility and ubiquitousness. We also show through the experiments that conventional personalized federated learning are not so effective as they claimed as. Among these three methods, AutoML-based methods are the most promising one. In particular, we find that federated neural architecture search methods are one future direction to solve heterogeneity despite existing limitations. Based on that, we propose several possible directions for the future study of personalized federated learning and how to solve heterogeneity problems in federated learning.

VIII. Acknowledgement

I would like to thank Prof. Baochun Li because he delivers perfect lectures. I would also like to thank teaching assistant Chen Ying for active and valuable feedback on midterm evaluation and critique papers. By writing this survey, I have learned a lot about how to do research, brain storm, paper reading, academic paper writing and any other things we did to write this paper.

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-efficient learning of deep networks from decentralized data,” in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[2] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, “Advances and open problems in federated learning,” *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[3] Y. Mansour, M. Mohri, J. Ro, and A. T. Suresh, “Three approaches for personalization with applications to federated learning,” *CoRR*, vol. abs/2002.10619, 2020.

[4] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, “On the convergence of fedavg on non-iid data,” in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=HJxNAnVtDS

[5] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, “Federated multi-task learning,” *Advances in neural information processing systems*, vol. 30, 2017.

[6] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor, “Tackling the objective inconsistency problem in heterogeneous federated optimization,” *Advances in neural information processing systems*, vol. 33, pp. 7611–7623, 2020.

[7] Z. Liu, S. Hu, Z. S. Wu, and V. Smith, “On privacy and personalization in cross-silo federated learning,” *arXiv preprint arXiv:2206.07902*, 2022.

[8] Q. Yang, Y. Liu, T. Chen, and Y. Tong, “Federated machine learning: Concept and applications,” *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[9] A. Krizhevsky, G. Hinton *et al.*, “Learning multiple layers of features from tiny images,” 2009.

[10] C. T Dinh, N. Tran, and J. Nguyen, “Personalized federated learning with moreau envelopes,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 21 394–21 405, 2020.

[11] L. Qu, Y. Zhou, P. P. Liang, Y. Xia, F. Wang, E. Adeli, L. Fei-Fei, and D. Rubin, “Rethinking architecture design for tackling data heterogeneity in federated learning,” in *Proceedings of the*

IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 10 061–10 071.

[12] L. Collins, H. Hassani, A. Mokhtari, and S. Shakkottai, "Exploiting shared representations for personalized federated learning," in *International Conference on Machine Learning*. PMLR, 2021, pp. 2089–2099.

[13] J. Oh, S. Kim, and S.-Y. Yun, "Fedbabu: Toward enhanced representation for federated image classification," in *International Conference on Learning Representations*, 2021.

[14] H. Li, Z. Cai, J. Wang, J. Tang, W. Ding, C.-T. Lin, and Y. Shi, "Fedtp: Federated learning by transformer personalization," *arXiv preprint arXiv:2211.01572*, 2022.

[15] F. Chen, G. Long, Z. Wu, T. Zhou, and J. Jiang, "Personalized federated learning with graph," *arXiv preprint arXiv:2203.00829*, 2022.

[16] yuyang deng, M. M. Kamani, and M. Mahdavi, "Adaptive personalized federated learning," 2021. [Online]. Available: https://openreview.net/forum?id=g0a-XYjpQ7r

[17] F. Hanzely and P. Richtarik, "Federated learning of a mixture of global and local models," 2021. [Online]. Available: https://openreview.net/forum?id=nLYMajjctMh

[18] T. Li, S. Hu, A. Beirami, and V. Smith, "Ditto: Fair and robust federated learning through personalization," in *International Conference on Machine Learning*. PMLR, 2021, pp. 6357–6368.

[19] A. Shamsian, A. Navon, E. Fetaya, and G. Chechik, "Personalized federated learning using hypernetworks," in *International Conference on Machine Learning*. PMLR, 2021, pp. 9489–9502.

[20] A. Fallah, A. Mokhtari, and A. Ozdaglar, "Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach," *Advances in Neural Information Processing Systems*, vol. 33, pp. 3557–3568, 2020.

[21] Y. J. Cho, J. Wang, T. Chiruvolu, and G. Joshi, "Personalized federated learning for heterogeneous clients with clustered knowledge transfer," *arXiv preprint arXiv:2109.08119*, 2021.

[22] R. Yang, J. Tian, and Y. Zhang, "Regularized mutual learning for personalized federated learning," in *Asian Conference on Machine Learning*. PMLR, 2021, pp. 1521–1536.

[23] C. He, E. Mushtaq, J. Ding, and S. Avestimehr, "Fednas: Federated deep learning via neural architecture search," 2021.

[24] A. Garg, A. K. Saha, and D. Dutta, "Direct federated neural architecture search," *arXiv preprint arXiv:2010.06223*, 2020.

[25] E. Mushtaq, C. He, J. Ding, and S. Avestimehr, "Spider: Searching personalized neural architecture for federated learning," *arXiv preprint arXiv:2112.13939*, 2021.

[26] D. Yao, L. Wang, J. Xu, L. Xiang, S. Shao, Y. Chen, and Y. Tong, "Federated model search via reinforcement learning," in *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2021, pp. 830–840.

[27] L. Dudziak, S. Laskaridis, and J. Fernandez-Marques, "Fedoras: Federated architecture search under system heterogeneity," *arXiv preprint arXiv:2206.11239*, 2022.

[28] C. Gong, D. Wang, M. Li, X. Chen, Z. Yan, Y. Tian, qiang liu, and V. Chandra, "NASVit: Neural architecture search for efficient vision transformers with gradient conflict aware supernet training," in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=Qaw16njk6L

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

[30] D. Wang, C. Gong, M. Li, Q. Liu, and V. Chandra, "Alphanet: improved training of supernets with alpha-divergence," in *International Conference on Machine Learning*. PMLR, 2021, pp. 10 760–10 771.

[31] J. Yu, P. Jin, H. Liu, G. Bender, P.-J. Kindermans, M. Tan, T. Huang, X. Song, R. Pang, and Q. Le, "Bignas: Scaling up neural architecture search with big single-stage models," in *European Conference on Computer Vision*. Springer, 2020, pp. 702–717.

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

[33] S. Hu, S. Xie, H. Zheng, C. Liu, J. Shi, X. Liu, and D. Lin, "Dsnas: Direct neural architecture search without parameter retraining," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 12 084–12 092.

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

[35] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan *et al.*, "Searching for mobilenetv3," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 1314–1324.