FAQS: Communication-efficient Federated DNN Architecture and Quantization Co-Search for personalized Hardware-aware Preferences

Hongjiang Chen 1, Yang Wang 1, Leibo Liu 1, Shaojun Wei 1, Shouyi Yin 1
1 The School of Integrated Circuits, Tsinghua University, Beijing 100084, China.

Abstract—Due to user privacy and regulatory restrictions, federate learning (FL) is proposed as a distributed learning framework for training deep neural networks (DNN) on decentralized data clients. Recent advancements in FL have applied Neural Architecture Search (NAS) to replace the predefined one-size-fit-all DNN model, which is not optimal for all tasks of various data distributions, with searchable DNN architectures. However, previous methods suffer from expensive communication cost raised by frequent large model parameters transmission between the server and clients. Such difficulty is further amplified when combining NAS algorithms, which commonly require prohibitive computation and enormous model storage. Towards this end, we propose FAQS, an efficient personalized FL-NAS-Quantization framework to reduce the communication cost with three features: weight-sharing super kernels, bit-sharing quantization and masked transmission. FAQS has an affordable search time and demands very limited size of transmitted messages at each round. By setting different personalized pareto function loss on local clients, FAQS can yield heterogeneous hardware-aware models for various user preferences. Experimental results show that FAQS achieves average reduction of 1.58x in communication bandwidth per round compared with normal FL framework and 4.51x compared with FL+NAS framework.

Index Terms—Federate Learning, Neural Architecture Search, Quantization, Communication-efficiency

I. INTRODUCTION

Federated Learning (FL) is a promising distributed training approach for deep neural networks (DNN) when processing decentralized data due to user privacy and regulatory restrictions [1]. It allows multiple data client to collaboratively train a global predefined model by exchanging intermediate parameters (e.g. the weights and biases) while keeping the raw local data samples private. As such, FL has been extensively applied in various challenging machine learning tasks such as data mining, computer vision and natural language processing. Despite its widespread superiority, one major challenge involved in FL is data heterogeneity, which means that the data distributions across clients are not identically or independently (non-IID) in nature. As a result, it is difficult to manually design optimization schemes for pre-defined model architectures.

In order to relax this constraint, recent research have combined Neural Architecture Search (NAS) algorithm to automatically generate high performance models instead of exploiting pre-defined architectures [2]-[5]. As showed in Fig. 1, the common NAS-based FL framework [2] can be divided into 4 steps. First, a NAS agent is utilized as a client searcher to generate the network weights $w_r$ and the neural architecture parameters $\alpha_r$ with several epochs of local training at round $r$. Second, every client node uploads $w_r$ and $\alpha_r$ to the center server. Third, the server then aggregates all parameters to obtain a global version of $w_{r+1}$ and $\alpha_{r+1}$. At last, the clients pull the updated parameters for the following round of searching. However, there are two challenges in front of the current framework. First, the NAS algorithms conducted on local clients require prohibitive computation and enormous model storage, which makes the parameters transmission extremely expensive because attempting many rounds of training on edge devices results in a remarkably huge communication cost between the server and clients. Second, it still searches for a unified global model, which may not perform optimally on all local clients.

Based on the previous observations, we propose FAQS, an efficient personalized FL-NAS-Quantization Co-Search framework to reduce the communication cost in three dimensions. First, we adopt the single-path based NAS algorithm [12] as our searching agent featuring weight-sharing super kernel, which slims the redundant supernet to the similar size of a normal network. Second, the network parameters is further compressed with mixed-precision bit-sharing quantization, which is jointly conducted with neural architecture search to save extra search time expense. Third, only a part of the local network is transmitted using masked parameters when uploading or pulling to save considerable size of communication. All these factors contribute to a communication-efficient manner of generating personalized local models. In addition, FAQS can satisfy heterogeneous hardware-aware preferences...
by setting various personalized pareto functions on different local clients.

Our contributions of this work can be concluded as follows:

- We propose a communication-efficient personalized FL-NAS-Quantization framework to reduce the transmission cost between the server and clients. This is the first work (to the best of our knowledge) to jointly optimize NAS and Quantization policy in a FL framework.
- We diagnose and lift three techniques featuring weight-sharing super kernel, bit-sharing quantization and masked transmission to achieve average reduction of 1.58x in communication bandwidth per round compared with normal FL framework and 4.51x compared with FL+NAS framework.
- We demonstrate that FAQS can yield heterogeneous hardware-aware models for various user preferences by setting different personalized pareto functions on accuracy, latency and model size.

II. MOTIVATION

A. Differential Neural Architecture Search for FL

Great research interests have been raised in Neural Architecture Search for automatically generating high-performance DNNs without human bias. Current NAS algorithms is typically based on three methods: Reinforcement Learning (RL) [6]-[10], Evolutionary Algorithm (EA) [16], [17] and Gradient Differential supernet (GD) [11]-[15]. Both RL and EA require an enormous searching time to find the best operations due to ineffective search direction proposals. While GD-based methods dramatically decreases the search time by formulating a differentiable search space, which has obtained momentum in recent research of FL to search for efficient DNNs. Direct Federated NAS [3] utilizes DSNAS [21] with Fed averaging algorithm in search of a unified model. FedNAS [2] combines MileNAS [13] with Fed averaging algorithm to search for a global model. [22] explores the concept of differential privacy with DARTS [11] to analyze the trade-off between accuracy and privacy of a global model. However, all these models are faced with two challenges. First, prior research is based on a differentiable multi-path NAS algorithm which searches over a supernet that encompasses all candidate architecture paths. Although it reduces the total search epochs to acceptable quantity, the number of candidate paths grows exponentially w.r.t. the number of hyper-parameter types which makes the whole supernet overweighted. As a result, every client needs to transmit tremendous size of the local model to the center server on each round and thus leads to huge communication cost. Second, existing methods converge on a single unified model’s architecture and weight parameters. As a result, there can only be one predictive outcome which may not perform optimally on all local clients’ tasks.

B. Joint optimization on quantization and NAS

In order to achieve higher energy efficiency with limited resource budget, DNNs must be carefully designed in two steps: the architecture design and quantization policy choice. However, taking the two steps separately is time-consuming and leads to a sub-optimal final deployment [20]. Recent research have attempted joint optimization for both quantization and NAS. [20] utilizes a RL agent based on a multi-objective evolutionary search algorithm to search the models under the balance between model size and performance accuracy. [19] formulates the co-search problem by fusing DNN search variables and hardware implementation (quantization and other implementation) variables into one solution space, and maximize both algorithm accuracy and hardware implementation quality. [18] proposes a novel co-exploration framework which parameterizes the layer-wise quantization and searches these parameters jointly with the hyperparameters of the architecture. However, prior methods are lacking in searching efficiency. [18] and [20] simply wrap an outer loop of the original NAS framework to handle the quantization selection and [19] encompasses quantization policy as another multi-path dimension which increases both the time of search and the model size.

III. FAQS FRAMEWORK

In this section, we introduce our entire framework FAQS, an efficient personalized FL-NAS-Quantization Co-Search framework to reduce the communication cost between the server and clients. In the rest part of this section, we first introduce the problem definition including FAQS problem formulation and the communication cost. Next we lift and explain three key techniques used in our framework to reduce communication cost in different dimensions. Last, we give an overall explanation on FAQS algorithm.

A. Problem Definition

FAQS problem formulation: Mathematically, FAQS can be represented by the following optimization problem. In FAQS, the objective is to find a personalized model for each device $i$ that performs well on the local data distribution (we assume each client has the same number of samples for simplicity):

$$
\min_{\theta_1,..,\theta_k} G(\theta_1,..,\theta_k) = \frac{1}{k} \sum_{i=1}^{k} L_i(\theta_i, x_i, y_i)
$$

$$
L_i(\theta_i, x_i, y_i) = \alpha_i CE(\theta_i, x_i, y_i) + \beta_i Lat(\theta_i) + \gamma_i MS(\theta_i)
$$

$$
\alpha_i + \beta_i + \gamma_i = 1
$$

Where $k$ is the number of clients and $\theta_i$ is the trainable parameters including weights $w$ and architecture thresholds $t$. $(x_i, y_i)$ represents local data of each client. $L_i$ is the personalized loss function of client $i$, with soft pareto coefficient [19] $(\alpha_i, \beta_i, \gamma_i)$ on accuracy (converged by cross-entropy loss), latency and model size. We utilize the differentiable $Lat$ and $MS$ loss used in [27], which regards the weighted sum of latency or model size on each type of block, as final $Lat$ and $MS$ result for end-to-end training.
**Communication Cost:** The total communication cost (bits) transmitted between the server and clients through all rounds in FAQs can be formulated as:

\[
Comm = 2 \sum_{r=1}^{nRd} \sum_{i=1}^{nClient} \sum_{j=1}^{nLayer_i} \gamma_{i,r} N^j_{i,r}(w, t) \cdot Q^j_{i,r}(w, t)
\]

(2)

Where \(nRd\) represents the total rounds of FL, \(nClient\) is the number of clients involved in the collaborative training and \(nLayer_i\) is the \(i_{th}\) client’s total layer number. \(w\) and \(t\) are learnable weights and architecture parameters respectively. FAQs generates specific architectures based on thresholds \(t\) instead of the weighted path-level coefficient \(\alpha\) used in multi-path [2]. \(\gamma_{i,r}\) represents the proportion of the entire network parameters which are not masked during message transmission. \(N^j_{i,r}(w, t)\) is the total number of parameters including weights (or gradients) and architecture parameters while \(Q^j_{i,r}(w, t)\) represents the number of bits on layer \(j\) of client \(i\) in round \(r\). Since each round catains parameters pulling (from the server to clients) and uploading (from clients to the server), the communication cost is times by 2 for the final result. For a better understanding on previous parameters, we compare FAQs with FedAvg [26] and FedNAS+EDD [2], [19] (our implement of EDD with FedNAS) on each variable. As showed in Table I, we focus on a single muti-path MobileNetV2 block [23] which is the basic searchable unit in [2] and [19]. By adapting single-path weight-sharing super kernel and bit-sharing quantization, both the number of total parameters and the average quantization bits are reduced in FAQs. \(\gamma_{i,r}\) represents the percentage of the parameters that are actually transmitted. FedAvg and FedNAS-EDD both transmit the entire network for a full training, while FAQs masks part of the network inspired by [25]. In other words, FAQs only transmit the crucial parts that affect the gradient aggregation hence \(\gamma_{i,r}\) can be decreased.

| Method               | \(\gamma_{i,r}\) | \(N^i_{i,r}(w, t)\) | \(Q^i_{i,r}(w, t)\) | \(Comm^i_{i,r}\) (KB) | Personalized |
|----------------------|------------------|----------------------|----------------------|------------------------|--------------|
| FedAvg(MBV2)         | 1                | 9k                   | 32 (float)           | 72                     | no           |
| FedNAS+EDD           | 1                | 25k                  | 28 (quant)           | 175                    | yes          |
| FAQs (ours)          | 0.51             | 9k                   | 16 (quant)           | 18.4                   | yes          |

**B. Technique Details On Communication Reduce**

As showed in Fig. 2, the framework of FAQs adpats three techniques to reduce diverse parameters in Eq. 2 for a communication-efficient manner of generating personalized local models including weight-sharing super kernel (top middle), bit-sharing quantization (top right) and masked transmission (left).

**Weight-sharing super kernel:** In this dimension, we reduce the total number of parameters \(N^i_{i,r}(w, t)\) in Eq. 2 Compared with the famous differential multi-path NAS [11] (showed in Fig. 2 bottom middle), which constructs and directly trains an over-parameterized network named SuperNet containing all possible candidate paths. The key idea of single-path NAS is relaxing architecture decisions over a **weight-sharing over-parameterized kernel** called super kernel [12]. A super kernel is the max size of the potential kernel. For MobilenetV2, it is a \(5 \times 5\) kernel with the expansion ratio of 6. As showed in Fig. 2 (top middle), we divide the whole kernel in several parts. For searching kernel size, choosing \(3 \times 3\) or \(5 \times 5\) is equivalent to whether we choose the subset of \(w_{55}^{33}\) or not. And for searching expand ratio and skip opertaion, it...
is determined by if we choose to preserve the entire length, half length or zero of the kernel.

\[ w_k = w_{53} + \sigma(id(w_{55}\backslash33, t_k,5)) \ast w_{55\backslash33} \]  
\[ \hat{w} = \sigma(id(w_{k,3}, t_e,3)) \ast (w_{k,6\backslash3} + \sigma(id(w_{k,3}, t_e,6)) \ast w_{k,6\backslash3}) \]  

(3) 

(4)

Where \( w_{53} \) and \( w_{55\backslash33} \) represent weights of the 3*3 part and 5*5, 3*3 part of the original super kernel. \( w_k \) is the kernel shape selection output. While \( w_{k,3} \) and \( w_{k,6\backslash3} \) represent the first and the second half of the expanded super kernel. \( t_k,5, t_e,3 \) and \( t_e,6 \) are three thresholds fed on three \( \sigma(\cdot) \) sigmoid functions to decide whether we choose the subset of \( w_{55\backslash33} \), the skip operation and the expand ratio of 3. The original problem is a hard-decision problem which can be softened by using these sigmoid functions. The \( id(\cdot) \) function is used to calculate the indicator value as input of the sigmoid. We simply use the group Lasso term of the corresponding weight values and a threshold to make our decisions. For example, towards the following equation:

\[ id(w_{55\backslash33}, t_k,5) = \| w_{55\backslash33} \|^2 - t_{k,5} \]

We can deduce from the the \( id(\cdot) \) result that if \( id(\cdot) \) is much bigger than zero, then most variables in the \( w_{55\backslash33} \) are likely to have important values hence \( \sigma(\cdot) \) is almost equal to 1. As a result \( w_k \) yield a complete \( w_{55\backslash33} \) kernel size according to Eq. 2. By using this single-path NAS based on weight-sharing super kernels, the original redundant super network in multi-path NAS can be slimmed to the similar size of a normal network, making the entire search agent light and efficient. The relationship between the block type and \( id(\cdot) \) result is enumerated in Table II.

| \( id(w_{k,3}, t_{3,3}) \) | \( id(w_{k,3}, t_{e,6}) \) | \( id(w_{55\backslash33}, t_{k,5}) \) | block(size, ratio) |
|-----------------|-----------------|-----------------|-----------------|
| +               | +               | +               | 5 * 5, 6        |
| +               | +               | 3 * 3, 6        |
| +               | -               | +               | 5 * 5, 3        |
| +               | -               | -               | 3 * 3, 3        |
| -               | *               | *               | skip            |

Bit-sharing quantization: In this dimension, we focus on decreasing the average number of bits on all quantized variables \( Q_{ij}(w, t) \) in Eq. 2 when transmitting messages between the server and the clients. Quantization is a model compression method when deploying DNNs. We follow the widely used quantization aware training method [24], which uniformly round floating point values to the nearest integer point values based on the maximum and the minimum representative values. We formulate the quantization process as follows:

\[ w^* = \text{Norm}^{-1}(R(\text{Norm}(w), b)) = Q(w, b) \]  

(5)

Where \( (w, w^*) \) is the weight vector of the real value and the approximate quantization value, and \( b \) is the bit number. \( \text{Norm}(w) = \frac{w - w_{\text{min}}}{w_{\text{max}} - w_{\text{min}}} \) is a linear scaling function which normalizes the values of the vectors into [0,1] and \( \text{Norm}^{-1} \) is the inverse function. In particular, \( Q(\cdot) \) is the quantization function which takes in the normalized real value \( w \) and the bit number \( b \) and outputs the closest quantization value of the corresponding weight element. Different from the differencial multi-path quantization policy in [19] that allocates independent bits for diverse quantization policy storage (Fig. 2 bottom right), FAQS stores all quantization policies in shared space. For example, for a \{4, 8, 16\} bit search space, [19] takes \( 28(4 + 8 + 16) \) bits to store each potential quantization choice independently while FAQS only needs 16 fixed bits (b=16). As a result, 42% of model size can be saved on communications in FL. As showed in Fig. 2 (top right), instead of selecting quantization policy out of independent multi-path quantization policy, the basic idea of bit-sharing is similar to weight-sharing super kernels. For each layer, the quantization policy is decided by whether we choose to preserve the 5~8 bits and the 9~16 bits in a fixed 16-bit space or not.

\[ w_q = w_{i~4-8} + \sigma(id(w_{5-8}, t_{q,5-8}) \ast (w_{5-8} - \sigma(id(w_{9-16}, t_{q,9-16}) \ast w_{9-16})) = f(w', t_q) = f(Q(w, 16), t_q) = F_Q(w, t_q) \]

Where \( w_{i~4-8} \) is the \( i \)th to \( j \)th bits of the approximate quantization value in Eq. 5 (left to right: high bit to low bit). Similarly, \( t_{q,5-8} \) and \( t_{q,9-16} \) are two thresholds as inputs for respective \( \sigma(\cdot) \) functions to determine whether we preserve the 5~8 bits and the 9~16 bits. Similar to the former weight-sharing super kernels, towards the following \( id(\cdot) \) function:

\[ id(w_{5-8}, t_{q,5-8}) = \| w_{5-8} \|^2 - t_{q,5-8} \]

If \( \| w_{5-8} \|^2 \) is much bigger than the threshold \( t_{q,5-8} \), most weights’ 5 ~ 8 bits are non-zeros. As a result, an 8-bit(at least) quantization policy is needed for the fine-grained weight distribution. In contrast, when most 5 ~ 8 bits are zeros, we can approximately regard the original 16-bit weight as a 4-bit (1 ~ 4 bits) manner. The relationship between quantization policy and \( id(\cdot) \) result is enumerated in Table III.

| \( id(w_{5-8}, t_{q,5-8}) \) | \( id(w_{9-16}, t_{q,9-16}) \) | Quantization(bit) |
|-----------------|-----------------|-----------------|
| +               | +               | 16              |
| +               | -               | 8               |
| -               | *               | 4               |

Masked transmission: In this part, we seek the opportunity to reduce the valid proportion \( \gamma_{i,j} \) of network parameters transmission in Eq. 2. Inspired from FedMask [25] which masks unimportant channels and only aggregates the overlapped weights, the basic idea behind masked transmission is similar: we only transmit the important parts of kernels generated by local search agents. For example, as showed in
Algorithm 1 FAQS

**Input:** local searching epochs $E$; learning rounds $T$; clients number $K$; local training set $D^{train}_i$ and personalized pareto coefficients $(\alpha_i, \beta_i, \gamma_i)$ for each client $i$.

**Output:** Local model $P_i$ with architecture $A_i$, quantization $Q_i$ and Weight $W_i$

**Main Function:**

$\theta_0^{i} \leftarrow \text{Initialize Parameter}$

For $t = 1, 2, T$ Do

For $i = 1, 2, K$ in parallel Do

$\theta_t^{i} \leftarrow \text{Local Search}(\theta_t^{i-1}, D^{train}_i, \alpha_i, \beta_i, \gamma_i)$

$(\theta_t^{i}, ..., \theta_t^{K}) \leftarrow \text{Aggregation & Masked Pull}(\theta_t^{i}, ..., \theta_t^{K})$

For $i = 1, 2, K$ in parallel Do

$P_i \leftarrow \text{Local Finetune}(\theta_t^{i}, D^{train}_i)$

Local Search$(\theta, D^{train}, \alpha, \beta, \gamma)$:

$w, t \leftarrow \theta$

For $e = 1, 2, E$ Do

For batch$(x, y) \in D^{train}$ Do

$w_q \leftarrow \text{Bit Sharing Quantitize}(w, t)$ (Eq. 6, 7)

$\bar{w} \leftarrow \text{Weight Sharing Kernel}(w_q, t)$ (Eq. 4, 5)

$w \leftarrow w - \eta_w \nabla_w L(batch(x, y), \alpha, \beta, \gamma, \bar{w})$

$t \leftarrow t - \eta_t \nabla_t L(batch(x, y), \alpha, \beta, \gamma, \bar{w})$

$\tilde{w} \leftarrow \text{Masked Sample}(w)$ (Table II)

$\theta \leftarrow \tilde{w}, t$

Masked Upload $\tilde{\theta}$ to server

Local Finetune$(\theta, D^{train})$:

$w, t \leftarrow \theta$

$\tilde{W} \leftarrow \text{Masked Sample}(w)$ (Table II)

$A \leftarrow \text{Architecture Select}(w, t)$ (Table II)

$Q \leftarrow \text{Quantization Select}(w, t)$ (Table III)

$W \leftarrow \text{Train}(A, Q, \tilde{W}, D^{train})$

$P \leftarrow (A, Q, W)$

Return $P$

Fig. 2 (left): towards a particular layer in round $r$, masked transmission means that we do not pull or upload the entire MBV2 block with regressed values calculated by Eq. 3 and Eq. 4. Instead, discrete architectures are selected according to the mapping relationship in Table II. In this case, client$_i$, client$_j$ and client$_k$ select (3 * 3, 3), (3 * 3, 6) and (5 * 5, 6) MBV2 blocks respectively. Where $(k, k, e)$ represents the MBV2 block shape with kernel size $k$ and expand ratio $e$. Then FAQS only transmits the part of blocks with respective kernel shape and masks the others unimportant values. In this way, large amount of kernel weights can be left out and the valid proportion $\gamma_{r, e}$ of network parameters can be reduced to small numbers. When it comes to weights aggregation, only corresponding overlapped shape of kernels are calculated for aggregation. For example, the (3 * 3, 3) part is shared among all these clients while the (3 * 3, 6, 3) part accounts for client$_j$ and client$_k$. And the (5 * 5, 3 * 3, 6) part only accounts for client$_k$. As a result, (3 * 3, 3) is aggregated for all clients and the (3 * 3, 6, 3) part is pulled for client$_j$ and client$_k$, while (5 * 5, 3 * 3, 6) is monopolized by client$_k$. Since it is not required to transmit the complete block for all clients on each round, $\gamma_{r, e}$ is greatly reduced for a communication-efficient FL framework.

C. Overall Algorithm

As showed in Algorithm 1, FAQS is generally a two-step algorithm. First, a two-depth nested loop is adapted to generate personalized local models with a communication-efficient manner. Second, after the final epoch of local search and server aggregation, each local model is discretely selected for finetuning to yield the final model $P_i$ with architecture $A_i$, quantization $Q_i$ and Weight $W_i$. To be more specific, FAQS firstly initialize each client $i$ with trainable parameters $\theta_0^i$ including weights $w_0^i$ and architecture thresholds $r_0^i$. For each round, every client conducts local joint search on neural architecture and quantization. The process of Local_search function is simple. The original kernel weights is firstly quantized in a fixed 16-bit format using Eq. 5 and Eq. 6. Then the output quantized weights is transformed into a regressed combination of different parts of kernels using Eq. 3 and Eq. 4 (denoted as $\tilde{w}$). To this end, we regard $\tilde{w}$ as the direct form of kernel weights where output $= \text{Conv}(\tilde{w}, x)$. The personalized batch loss is a function parameterized by $\tilde{w}$ and pareto coefficients $(\alpha_i, \beta_i, \gamma_i)$, where $\tilde{w}$ is parameterized by $w$ and $t$. As a result, we can directly conduct mini-batch gradient descent on $w$ and $t$. Afterwards, we discretely select the masked weight $\tilde{w}$ of each layer according to Table II and transmit them to the server for aggregation. When finishing the final round of local search, FAQS generates the architecture and quantization selection based on the current $w$ and $t$ according to Table II and Table III. The finetune checkpoint starts on the masked sample weight $\tilde{W}$ and the final local model $P_i$ is generated with architecture $A_i$, quantization $Q_i$ and Weight $W_i$.

IV. Evaluations

Experiments are conducted on the standard classification benchmark to evaluate our method. We implement FAQS for distributed computing with nine nodes each equipped with a GPU (NVIDIA Tesla V100). We set up our experiment in a cross-silo setting for simplicity with one node representing the server and eight nodes representing the clients.

A. Experimental Setup

**Dataset preparation:** We select CIFAR10 dataset on image classification task for a fair comparison, which is commonly used as a benchmark in FL and NAS [2], [19]. We generate non-IID data across clients by exploiting LDA distribution [2] with parameter $(\alpha=0.2)$.

**Search space:** For neural architecture search space, our framework builds upon hierarchical MBV2-like search space used in [23], [12], [14], [19]. The main goal is to decide the type of mobile inverted bottleneck convolution. To be more specific, MBConv layer is decided by the kernel size of the depthwise convolution $k$ and by the expansion ratio $e$. In particular, we consider layers with kernel sizes (3, 5) and...
B. Training and Communication Efficiency

To reduce the complexity of the problem, we only quantize a k*k depthwise convolution, and a linear 1*1 convolution. The backbone has 4 blocks each block consists of 4 groups of a point-wise 1*1 convolution, and a k*k depthwise convolution, and a linear 1*1 convolution. To reduce the complexity of the problem, we only quantize weights and assume each layer within a group share the same quantization policy. The choices of quantization are limited within (4, 8, 16) bits.

C. Personalized Hardware-aware Preferences

In order to demonstrate the ability of generating heterogeneous models for personalized hardware-aware preferences, we use 8 groups of pareto loss with various \((\alpha, \beta, \gamma)\) for each node. By running through 150 rounds of FAQs, we record the accuracy curves and the scatterings on three clients' latency and model size. Client A with (0.8,0.1,0.1) prefers accuracy much more than latency and model size, while client B with (0.4,0.5,0.1) and C with (0.4,0.1,0.5) are lat-prior and ms-prior respectively. As showed in Fig. 4 (left), the model searched in node A outperforms B and C in accuracy. While both B and C yield high-performance models very near to corresponding Pareto boundaries showed in Fig. 4 (right). The searched architectures and quantization policies are detailed in Table VI along with their performance and hardware-aware parameters. We give a brief analysis on the searched architectures and quantization policies. As client A prefers accuracy much more than latency and model size, it tends to search for deep and wide architectures with long bit(16-b) quantizations. Latency is the main optimizing component for Client B. As a result, it prefers wide but shallow architectures as the GPU device is good at performing parallel computations.
Preserving large kernels can make up for the accuracy loss introduced by shallow architectures. For the ms-prior client $C$, many layers adapt narrow and shallow block types to reduce model size. We have similar findings in [14] that both lat-prior and ms-prior models have quite shallow layers but long quantization bit length in early stage, because features of early stages are relatively small and need higher resolution. Besides, both of them prefer larger MBConv and longer quantization bit in downsamplied layers to prevent severe accuracy loss.

V. CONCLUSION

In this paper, we have presented FAQS, the first joint neural architecture and quantization optimization framework in federate learning. FAQS reduces the communication cost by utilizing weight-sharing super kernels, bit-sharing quantization and masked transmission. Moreover, FAQS can yield heterogeneous hardware-aware models for various user preferences by setting different personalized pareto functions on accuracy, latency and model size. Our evaluation shows that FAQS can achieve average reduction of 1.58x in communication bandwidth per round compared with normal FL framework and 4.51x compared with FL+NAS framework.

REFERENCES

[1] McMahan, Brendan, et al. "Communication-efficient learning of deep networks from decentralized data." Artificial intelligence and statistics. PMLR, 2017.
[2] Chaoyang He, Murali Annavaram, and Salman Avestimehr. "Fednas: Federated deep learning via neural architecture search." arXiv e-prints, pages arXiv–2004, 2020. 2
[3] Garg, Anubhav, Amit Kumar Saha, and Debo Dutta. "Direct federated neural architecture search." arXiv preprint arXiv:2010.06223 (2020).
[4] Zhu, Hangyu, and Yaochu Jin. "Real-time federated evolutionary neural architecture search." IEEE Transactions on Evolutionary Computation 26.2 (2021): 364-378.
[5] Singh, Ishika, et al. "Differentially-private federated neural architecture search." arXiv preprint arXiv:2006.10559 (2020).
[6] Zoph, Barret, and Quoc V. Le. "Neural architecture search with reinforcement learning." arXiv preprint arXiv:1611.01578 (2016).
[7] Zoph, Barret, et al. "Learning transferable architectures for scalable image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.
[8] Baker, Bowen, et al. "Designing neural network architectures using reinforcement learning." arXiv preprint arXiv:1611.02167 (2016).
[9] Zhong, Zhao, et al. "Blockqnn: Efficient block-wise neural network architecture generation." IEEE transactions on pattern analysis and machine intelligence 43.7 (2020): 2314-2328.
[10] Pham, Hieu, et al. "Efficient neural architecture search via parameters sharing." International conference on machine learning. PMLR, 2018.
[11] Liu, Hanxiao, Karen Simonyan, and Yiming Yang. "Darts: Differentiable architecture search." arXiv preprint arXiv:1806.09055 (2018).
[12] Stoumboulis, Dimitrios, et al. "Single-path nas: Designing hardware-efficient convnets in less than 4 hours." Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Springer, Cham, 2019.
[13] He, Chaoyang, et al. "Milenas: Efficient neural architecture search via mixed-level reformulation." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020.
[14] Cai, Han, Ligeng Zhu, and Song Han. "Proxylessnas: Direct neural architecture search on target task and hardware." arXiv preprint arXiv:1812.00332 (2018).
[15] Dong, Xuan yi, and Yi Yang. "Searching for a robust neural architecture in four gpu hours." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019.
[16] Xie, Lingxi, and Alan Yuille. "Genetic cnn." Proceedings of the IEEE international conference on computer vision. 2017.