FedorAS: Federated Architecture Search under system heterogeneity

Anonymous authors
Paper under double-blind review

Abstract

Federated learning (FL) has gained considerable attention due to its ability to learn on decentralised data while preserving client privacy. However, it also poses additional challenges related to the heterogeneity of the participating devices, both in terms of their computational capabilities and contributed data. Meanwhile, Neural Architecture Search (NAS) has been successfully used with centralised datasets, producing state-of-the-art results in constrained or unconstrained settings. Such centralised datasets may not be always available for training, though. Most recent work at the intersection of NAS and FL attempts to alleviate this issue in a cross-silo federated setting, which assumes homogeneous compute environments with datacenter-grade hardware. In this paper we explore the question of whether we can design architectures of different footprints in a cross-device federated setting, where the device landscape, availability and scale are very different. To this end, we design our system, `FedorAS`, to discover and train promising architectures in a resource-aware manner when dealing with devices of varying capabilities holding non-IID distributed data. We present empirical evidence of its effectiveness across different settings, spanning across three different modalities (vision, speech, text), and showcase its superior performance compared to state-of-the-art federated solutions, while maintaining resource efficiency.

1 Introduction

As smart devices become omnipresent where we live, work and socialise, the ML-powered services that these provide grow in sophistication. With the recent advances in SoCs' capabilities (Ignatov et al., 2019; Almeida et al., 2021) and motivated by privacy concerns (Truong et al., 2021) over the custody of data, Federated Learning (FL) (McMahan et al., 2017) has emerged as a way of training on-device on user data without them ever directly leaving the device premises. However, FL training has largely been focused on the weights of a static global model architecture, assumed to be runnable by every participating client (Kairouz et al., 2019) in its vanilla form. Not only may this not be the case, but it can also lead to subpar performance of the overall training process in the presence of stragglers or biases in the case of consistently dropping certain low-powered devices. On the opposite end, more capable devices might not fully take advantage of their data if the deployed model is of reduced capacity to ensure all devices can participate (Li et al., 2020b). These aspects have been the subject of research in recent years Horvath et al. (2021); Diao et al. (2020); Jiang et al. (2022) which, through different methods, address the challenge of system heterogeneity in FL.

Parallel to these trends, Neural Architecture Search (NAS) has become the *de facto* mechanism to automate the design of DNNs that can meet the requirements (e.g. latency, model size) for these to run on resource-constrained devices. The success of NAS can be partly attributed to the fact that these frameworks are commonly run in datacenters, where high-performing hardware and/or large curated datasets (Krizhevsky, 2009; Deng et al., 2009; Cordts et al., 2015; John S. Garofolo et al., 1983; Panayotov et al., 2015) are available. However, this also poses two major limitations on current NAS approaches: i) *privacy*, i.e. these methods are often not designed to work in situations when user's data must remain on-device; and, consequently, ii) *tail data non-discoverability*, i.e. they might never be exposed to infrequent or time/user-specific data that exist in the wild but not necessarily in centralized datasets. On top of these, the whole cost is born by the provider

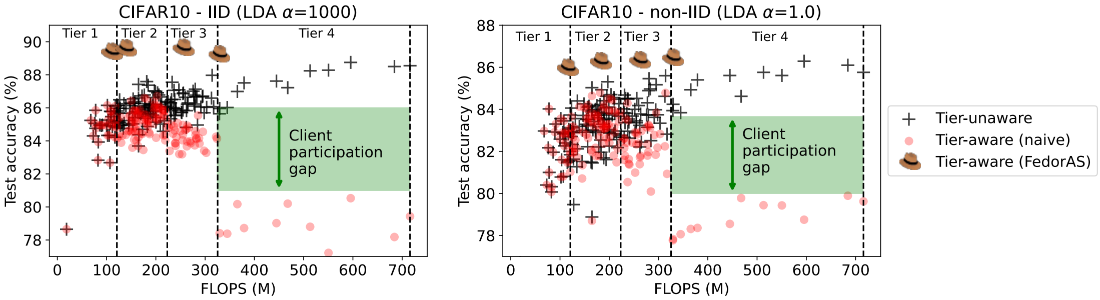

Figure 1: For two LDA settings, 160 architectures are randomly sampled from a ResNet-like search space and trained on a CIFAR-10 FL setup with 100 clients, with 10 clients participating on each round for a total of 500 rounds. Clients are uniformly assigned to a tier, resulting in 25 clients per tier. Given sufficient data and ignoring tier limits, model performance tends to improve as its footprint (FLOPS) increases (black crosses). However, when models are restricted to only train on clients that support them (tier-aware), the lack of data severely restricts the performance of more capable models (red dots). `FedorAS` successfully overcomes the challenges of tier-aware FL and outperforms existing system heterogeneous baselines, as shown later in Fig. 3.

and separate on-device modelling/profiling needs to be done in the case of hardware-aware NAS (Dudziak et al., 2020; Tan et al., 2019; Lee et al., 2021), which has mainly focused on inference performance hitherto. Performing NAS in a federated setting brings about several challenges, including communication, memory and computation cost. Prior work (He et al., 2020a; Mushtaq et al., 2021) has mainly focused in *cross-silo* deployments which assumes full client participation and system costs are a secondary issue. These assumptions are highly impractical for cross-device settings.

Since devices in the wild exhibit different compute capabilities, may support different operators and can hold non-IID distributed data, this results in *system* and *data heterogeneity*. In the context of NAS, system heterogeneity has a particularly significant effect, as we might no longer be able to guarantee that any model from the search space can be efficiently trained on all devices. This inability can be attributed to insufficient compute power, non-implemented DNN operators for accelerated compute, limited network bandwidth or unavailability of the client at hand. Consequently, some of the models might be deemed worse than others, not because of their worse ability to generalise, but because they might not be exposed to the same subsets of data as others. As shown in Fig. 1, models of different footprint trained across clients of varying capabilities under constrained (*tier-aware*) and full (*tier-unaware*) participation exhibit different levels of performance.

Motivated by the aforementioned participation phenomena and limitations of the existing NAS methods, we introduce `FedorAS`[1], a framework that performs NAS over *heterogeneous devices* holding *heterogeneous data* in a resource-aware and federated manner. To the best of our knowledge, `FedorAS` is the first system to perform *cross-device* federated NAS, optimising for both training overhead and inference deployment. To accomplish this, we design a supernet comprising operations covering a wide spectrum of compute complexities and memory footprints. This supernet acts both as *search space* and a *weight-sharing backbone* (Sec. 3.1 ⓿). Upon federation, it is only *partially* and *stochastically* shared to clients, respecting their bandwidth (Sec. 3.1 ❶) and computational capabilities (Sec. 3.1 ❷). Clients leverage resource-aware one-shot path sampling (Guo et al., 2020) that we re-formulate for lightweight on-device NAS. In this way, networks in a given search space are not only deployed in a resource-aware manner, but also trained as such, by tuning the downstream *communication* (i.e. the subspace explored by each client) and *computation* (i.e. FLOPs of sampled paths) to meet the device's training budget. This training budget can can be *static* – based on the device's capabilities – or *dynamic* – based on the device's current workload. Once federated training of the supernetwork has completed, usable pretrained networks can be extracted even before performing fine-tuning or personalising per device (Sec. 3.3), thus minimising the number of extra on-device training rounds to achieve competitive performance. In summary, in this work we make the following contributions:

- We adapt and extend a popular single-path one-shot NAS to enable resource-aware federated supernet training – towards this end we introduce and study the (mutual) effects of: *i)* subspace sampling to improve communication cost (Sec. 3.1 ❶), *ii)* a $\mathcal{O}(1)$ path-sampling mechanism that supports dynamic changes

---

[1] Anonymised source code: `https://anonymous.4open.science/r/FedorAS-TMLR-2023`

in a client's constraints (Sec. 3.1 ❷), *iii)* frequency-aware OPerator Aggregation (OPA) method (Sec. 3.1 ❸) to correctly weight updates when training entails working with stochastic architectures.
- We further establish and evaluate an alternative, federated way of ranking models from the global supernet, during search, which removes the requirement for having a server-residing validation set (Sec. 4.7).
- We perform extensive empirical study of the benefits of supernet weight-sharing in highly heterogeneous FL deployments – our results show clear gains from supernet-based initialisation (Sec. 4.3).
- Through extensive evaluation across various datasets, tasks and modalities, on different device distributions we demonstrate `FedorAS`' performance vs. state-of-the-art FL techniques. Indicatively, we achieve over +6pp (percentage points) accuracy on SpeechCommands with 33× fewer FLOPs, and +26pp for a highly non-IID CIFAR-10 setup vs the previous federated NAS best alternative.

2 Related Work

This section introduces background and related work relevant to `FedorAS`. For an overview of a typical FL pipeline and an introduction to NAS, please see Appendix D.1. and D.2 respectively. Comparison between `FedorAS` and selected FL techniques is additionally summarised in Table 1.

Federated Learning. Traditionally, works have focused on tackling the statistical data heterogeneity (Smith et al., 2017; Li & Wang, 2019; Hsieh et al., 2020; Fallah et al., 2020; Li et al., 2020c) or minimising the upstream communication cost (Li et al., 2021a; Konečný et al., 2016; Wang et al., 2018; Han et al., 2020; Amiri et al., 2020), as the primary bottleneck in the federated training process. However, it has become apparent that computational disparity between participating nodes becomes an equally important barrier for contributing knowledge in FL. Cross-device FL performs the bulk of the compute on a highly heterogeneous (Kairouz et al., 2019) set of devices in terms of their compute capabilities, availability and data distribution. In such scenarios, a trade-off between model capacity and client participation arises: larger architectures might result in more accurate models which may only be trained on a fraction of the available devices; on the other hand, smaller footprint networks could target more devices – and thus more data – for training, but these might be of inferior quality (see gap in Fig. 1).

System heterogeneous FL. To tackle the problem of stragglers and limited participation, there have been various approaches in the literature, leveraging structured (PruneFL (Jiang et al., 2022), HeteroFL (Diao et al., 2020)), unstructured (Adaptive Federated Dropout (Bouacida et al., 2021), LotteryFL (Li et al., 2021b)) or importance-based pruning (FjORD (Horvath et al., 2021)), quantisation (AQFL (Abdelmoniem & Canini, 2021)), low-rank factorisation (FedHM (Yao et al., 2021a)), sparsity-inducing training (ZeroFL (Qiu et al., 2021)) or distillation (GKT (He et al., 2020b)). However, most of these techniques focus on dynamically altering the model in a single dimension (e.g. model width or precision) and may require extra training overhead, multiple DNN copies or specialised hardware. Federated NAS, on the other hand, is a more

Table 1: Comparison of heterogeneous FL techniques. "System Het." – the ability of altering each client's workload; quantitative methods only scale amount of work (FLOPs) while qualitative can also change a network's structure or operations. "Dynamic" – the ability of changing each client's capabilities within a single round. "Model Het." – the property of having more than one usable model at the end; "scaling" is a weaker class of heterogeneity where different models are scaled-down versions of a larger one, while methods with a tick allow for topological diversity. "# models" – how many different models a method considers. "?" means a property is achievable but either not considered in the original work or with very limited evaluation.

Method	FL setup	Objective	System Het.	Dynamic	Model Het.	# models	Knowledge sharing
FjORD Horvath et al. (2021)	cross-device	global	quantitative	✓	scaling	# channels	randomized scaling
HeteroFL Diao et al. (2020)	cross-device	local[1]	quantitative	✓?	scaling	# tiers	static scaling
ZeroFL Qiu et al. (2022)	cross-device	global	quantitative	✓	✗	-	-
FedNAS He et al. (2020a)	cross-silo	local[1]	✗	-	✓	NAS	supernet
SPIDER Mushtaq et al. (2021)	cross-silo	both	✗	-	✓	NAS	supernet
SuperFed Khare et al. (2023)	cross-device	global	✗[2]	✗	✓	NAS	supernet
FedSup Kim & Yun (2022)	cross-device	both	✗[2]	-	✓	NAS?[3]	supernet
E-FedSup Kim & Yun (2022)	cross-device	both	qualitative	✗	✓	NAS?[3]	supernet
HAFL Litany et al. (2022)	cross-silo	local[4]	qualitative	✗	✓?	# tiers	hypernetwork
pFedHN Shamsian et al. (2021)	cross-device	local[4]	qualitative?	✗	✓?	# tiers	hypernetwork
`FedorAS`	cross-device	global	qualitative	✓	✓	NAS	supernet

[1] global performance is weak so we consider it a secondary objective. [2] only considers system heterogeneity at deployment.

[3] evaluation focuses on 3 arbitrarily selected models; architecture search is said to be "beyond the scope of the current study".

[4] strictly local – the possibility of obtaining a single, high-quality global model is unclear.

general technique that offers additional degrees of architectural freedom and, through our technique, a more efficient and expressive knowledge sharing mechanism.

Federated NAS. The concept of performing NAS in a federated setting has been considered before (He et al., 2020a; Mushtaq et al., 2021; Yao et al., 2021b; Zhang et al., 2022; Litany et al., 2022). Specifically, one of the first works in the area was FedNAS (He et al., 2020a), which adopts a DARTS-based approach and aims to find a globally good model for all clients, to be personalised at a later stage. This approach requires the whole supernet to be transmitted and kept in memory for architectural updates, which leads to excessive requirements (Table 3) that make it largely inapplicable for cross-device setups with clients of limited capabilities. To mitigate this requirement, (Yao et al., 2021b) proposes an RL-based approach for cross-silo FL-based NAS. Despite the intention, it still incurs significant overheads due to RL-based model sampling convergence and single model training per client. A somewhat different approach is adopted by HAFL (Litany et al., 2022) and pFedHN Shamsian et al. (2021), which leverage graph hypernetworks as a means to generate outputs of a model. While interesting, performance and scalability are not on par with current state-of-the-art. Closer to our method is FedSup (Kim & Yun, 2022), but their implementation suggests that clients of different capabilities do not participate simultaneously. In addition, their distillation component requires sequential training between teacher and student models, thus harming local training latency. SuperFed (Khare et al., 2023) is another system performing Federated NAS, but only focuses on inference cost, therefore assuming all clients can run all models of the supernet. Moreover, their "sandwitch" rule requires the maximal network to be always sampled per round, thus incurring additional costs. On the front of personalisation, FedPNAS (Hoang & Kingsford, 2022) searches for architectures with a base component shared across clients and a personalised component unique per client. This work, however, is only aimed at IID vision tasks and involves a meta-step for personalisation, which increases training overheads significantly. At the other extreme for cross-silo personalised FL, SPIDER (Mushtaq et al., 2021) aims at finding personalised architectures per client. It requires the whole space on-device and federation is accomplished through a second static on-device network. These overheads make porting SPIDER to the cross-device setting non-trivial.

`FedorAS` brings Federated NAS to the *cross-device* setting and is designed with resource-awareness and system heterogeneity in mind. This way, communication and computation cost is kept within the defined limits, respecting the runtime on constrained devices. Crucially, our system does not assume full participation of clients and is able to efficiently exchange knowledge through the *supernet weight sharing*, as demonstrated, and generalise across different modalities and granularity of tasks. By combining NAS and cross-device FL, it becomes possible to discover and train neural network architectures that are optimised for the distributed data available on a diverse range of devices, while preserving privacy. Of course, Federated NAS can be combined with techniques from the system heterogeneous FL literature to achieve further performance gains.

3 The FedorAS Framework

`FedorAS` is a resource-aware Federated NAS framework that combines learning from clients across all tiers and yielding models tailored to each tier that benefit from this collective knowledge.

Workflow. `FedorAS`' workflow consists of three stages, outlined in Fig. 2): *i) supernet training*, *ii) model search and validation* and *iii) model fine-tuning*. First, we train the supernet in a resource-aware and federated manner (**Stage 1**, Sec. 3.1). We then search for models from the supernet with the goal of finding the best architecture per tier (**Stage 2**, Sec. 3.2). Models are effectively sampled, validated on a global validation set and ranked per tier. These architectures and their associated weights act as initialisation to the next phase, where each model is fine-tuned in a per-tier manner (**Stage 3**. Sec. 3.3). The end goal of our system is to have the best possible model per each cluster of devices.

Design rationale. Designing a search space that can be trained under computational constraints and stochastic exposure to heterogeneous data is no trivial task. We build our system around the concept of a supernet to facilitate weight-sharing between architectures of various footprints. Operations in the supernet are samplable paths (i.e. models) of different footprint. As such, while normally large models would not be directly trained on data of low-tier clients, our design allows for indirectly sharing this knowledge through the association of the same operation to different paths. To ensure efficient training of a supernet, we base our

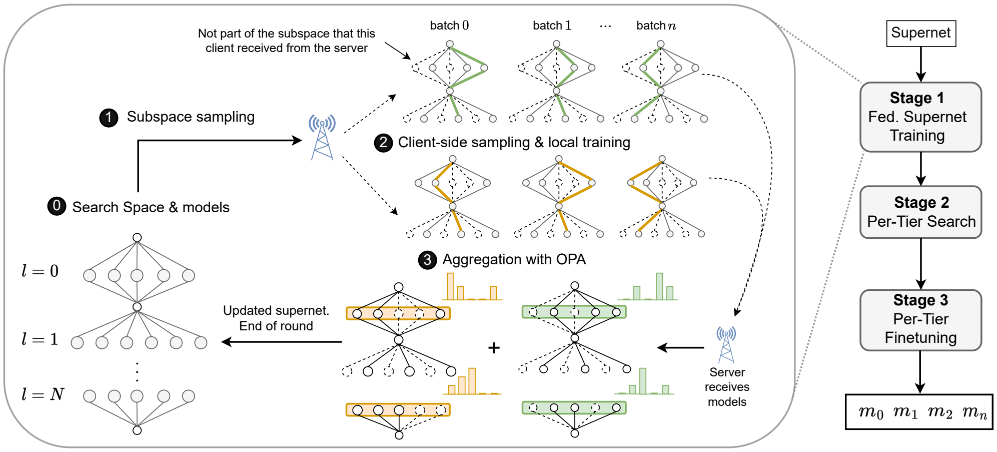
Not part of the subspace that this client received from the server
batch 0
batch 1
···
batch n
❶ Subspace sampling
❷ Client-side sampling & local training
⓿ Search Space & models
❸ Aggregation with OPA
$l = 0$
$l = 1$
$l = N$
Updated supernet. End of round
+
Server receives models
Supernet
Stage 1 Fed. Supernet Training
Stage 2 Per-Tier Search
Stage 3 Per-Tier Finetuning
m_0 m_1 m_2 m_n

Figure 2: Training process workflow with `FedorAS`.

approach on SPOS (Guo et al., 2020) and adapt it (see Eq. 1-5) since its training procedure is lightweight, with marginal overhead in terms of memory consumption and required FLOPs. Last, we cluster devices into "tiers" based on their computational capabilities and search for an architecture for each tier as a balance between having one model to fit all needs (He et al., 2020a) and a different architecture per client (Mushtaq et al., 2021).

3.1 Federated SuperNet Training

⓿ **Search space & models.** First, we define the search space in terms of a range of operators options per layer in the network that are valid choices to form a network. This search space resides as a whole on the server and is only partially communicated to participating clients of a training round to keep communication overheads minimal. Specific models (i.e. paths) consist of selections of one operator option per layer, sampled in a single-path-one-shot manner on-device per local iteration. For each layer subject to search, l, we denote the set of candidate operations as $\mathbb{O}_l$.

❶ **Subspace sampling.** It is obvious that communicating the whole space of operators along with the associated weights to each device becomes quickly intractable, especially bearing in mind that communication is usually a primary bottleneck in Federated Learning (Kairouz et al., 2019; Li et al., 2020b). To this direction, `FedorAS` adopts a uniform parameter size budget, $B_{\Phi_{\text{comm}}}$, and samples[2] the search space for operators until this limit is hit (Eq. 1). Setting the limit to half the size of a typical network deployed for a task[3] worked well in our experiments and in fact accelerated convergence (Sec. 4.5).

$$\sum_{l=1}^{L}\sum_{o\in\mathbb{O}_l}\mathbb{1}_{\hat{\mathbb{O}}_l}(o)\Phi_{\text{comm}}(o) < B_{\Phi_{\text{comm}}}, \quad \hat{\mathbb{O}}_l \subseteq \mathbb{O}_l \wedge \hat{\mathbb{O}}_l \neq \varnothing \tag{1}$$

where L is the number of searchable layers in the supernet, $\mathbb{O}_l$ the candidate and $\hat{\mathbb{O}}_l$ the selected operations for layer l, $\mathbb{1}_X$ an indicator function of a set X and Φ_{comm} a relevant size measure.

In terms of sampling strategies, we experimented with uniform operator sampling and found it to work sufficiently well, providing uniform coverage over the search space. It is worth noting that different, but possibly overlapping, subspaces can be selected for each client in a round.

❷ **Client-side sampling & local training.** Participating clients receive the subspace sampled on the server, $\{\hat{\mathbb{O}}_l\}_{l=1}^{L}$, from which they sample a single operator on every layer. This constitutes a path along the supernet (p_L) representing a single model. For every batch, clients sample paths that do not surpass the assigned training budget $B_{\Phi_{\text{train}}}^{\text{Tier}}$. Throughout this work, we consider $\Phi_{\text{train}}(\cdot)$ to be a cost function that counts the FLOPs of a given operator, and the budget is the maximum total FLOPs per forward pass[4]. Since

---

[2] Non-parametric operations are always sent and layers without such options are prioritised to guarantee a valid network.

[3] Typical means here a commonly used baseline network for the task at hand in the literature.

[4] For qualitative constraints, we can easily exclude unsupported operations (x) by defining $\Phi_{\text{train}}(x) = \infty$.

our setting assumes clients to form computational tiers, practically, we also assume clients from a single tier to share the same budget. However, note that path sampling is an autonomous process happening independently on each client. As such, in general, clients retain the ability to adjust their sampling dynamically, in both qualitative and quantitative manner. Ultimately, irrespective of each client's constraints, our goal is to sample valid paths uniformly, to ensure systematic coverage of the entire (sub) search space:

$$p_L = \bigcup_{l=1}^{L} o_l \sim \mathcal{U}\{\hat{\mathbb{O}}_l\} \;\text{ s.t. }\; \sum_{l=1}^{L} \Phi_{\text{train}}(o_l) < B^{\text{Tier}}_{\Phi_{\text{train}}} \tag{2}$$

However, realising Eq. 2 efficiently is not a trivial task. Originally, Guo et al. (2020) considered a naive approach of repeatedly sampling a path until it fits the given budget, which results in non-negligible overhead if the probability of finding a model under the threshold is low. Were we to employ such a method, the most restricted devices, for which the set of eligible models is the smallest, would be the ones burdened with the largest overhead. Therefore, we propose a greedy approximation in which operations are selected sequentially. Specifically, in order to obtain a path $p_L = \{o_i\}_{i=1}^{L}$ we sample operations layer-by-layer, according to a random permutation σ, in such a way that the i-th operation is chosen from the candidates for layer $\sigma(i)$ whose total overhead would not violate the constraint:

$$o_i \sim \mathcal{U}\{\, o : o \in \hat{\mathbb{O}}_{\sigma(i)} \wedge \sum_{j=1}^{i} \Phi_{\text{train}}(o_j) + \Phi_{\text{train}}(o) < B^{\text{Tier}}_{\Phi_{\text{train}}} \,\} \tag{3}$$

We ensure that Eq. 3 obtains a valid architecture without resampling by having the "identity" function (i.e. no-op) in their candidate operations and prioritising the selection of those layers which do not.

After having sampled the path, a model is instantiated and a single optimization step using a single batch of data is performed. The number of samples passing through each operator are kept and communicated back to the server, along with the updates, for properly weighting updated parameters upon aggregation, as we will see next. Overall, if we denote the distribution of valid paths the i^{th} client can sample as P_i and its local dataset as D_i, we can define the local training objective to be:

$$\omega_i = \arg\min_{\omega} \mathbb{E}_{p\sim P_i} \mathbb{E}_{x\sim D_i} \mathcal{L}(f_\omega(x,p)), \tag{4}$$

where input p can be modeled as a sparse matrix $\{0,1\}^{L\times|O|}$ of architectural parameters controlling path selection in a supernet f_ω (see D.2).

❸ **Aggregation with OPA.** An operator gets stochastically exposed to clients data. This stems from *subspace sampling* and client-side *path sampling*. As such, naively aggregating updates in an equi-weighed manner (Plain-Avg) or total samples residing on a client (FedAvg (McMahan et al., 2017)) would not reflect the training experience during a round. For this reason, we propose OPA, OPerator Aggregation, an aggregation method that weights updates based on the relative experience of an operator across all clients that have updated that operator. Concretely, our method is a generalisation of FedAvg where normalisation is performed independently for each layer, rather than collectively for full models. In order to enable that, we keep track of how many examples were used to update each searchable operation o_l, independently on each client, and later use this information to weight updates. Formally:

$$\omega_g{}^{(t+1)}(o_l) = \begin{cases} \sum_{i=1}^{k} \frac{|D^{(t)}_{i,o_l}|}{\sum_{j=0}^{k} |D^{(t)}_{j,o_l}|} \omega_i^{(t)}(o_l) & \text{if } |C^{(t)}_{o_l}| > 1 \\ \omega_g^{(t)}(o_l) & \text{otherwise} \end{cases} \tag{5}$$

where $\omega_g^{(t)}(\cdot)$ are global weights, $\omega_i^{(t)}(\cdot)$ are local weights of client i at global step t, $|D^{(t)}_{i,o_l}|$ is the number of samples having backpropagated through an operator o_l for client i in round t, and $C^{(t)}_{o_l}$ is the set of clients i s.t. $|D^{(t)}_{i,o_l}| > 0$. Updates to $\omega_g^{(t)}$ happen only if $|C^{(t)}_{o_l}| > 1$ in order to protect the privacy of single clients. Finally, if an operation is always selected, then Eq. 5 recovers FedAvg, which means we can effectively use OPA throughout the model and not only for searchable layers. Analogous to other FL approaches, the high-level global objective of our supernet training is similar to the local objective shown in Eq. 4, but with both

Table 2: Datasets for evaluating `FedorAS`. We partition CIFAR-10/100 following the Latent Dirichlet Allocation (LDA) partitioning (Hsu et al., 2019), with each client receiving approximately equisized training sets. For CIFAR-10, we consider $\alpha \in \{1000, 1.0, 0.1\}$ configurations, while for CIFAR-100, we adopt $\alpha = 0.1$ (Reddi et al., 2021). Other datasets come naturally partitioned.

Dataset	Search Space	Number Clients	Number Examples	Target Task
CIFAR10	CNN-L	100	$50K$	Image classification
CIFAR100	CNN-L	500	$50K$	Image classification
Speech Comm.	CNN-S	$2,112$	$105.8K$	Keyword Spotting
Shakespeare	RNN	715	$38K$	Next char. prediction

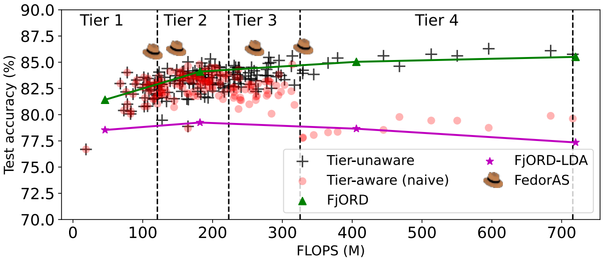

Figure 3: `FedorAS` outperforms other approaches (details in Appendix E.7). CIFAR-10 (non-IID, α=1.0). FjORD is represented as a line as it can switch between operating points on-the-fly via Ordered Dropout.

expectations being taken over global alternatives of distributions P_i and D_i. Consequently, although both our model (supernet) and its input space (pairs of (x, p)) might seem more complex compared to standard FL approaches, they follow the same fundamental principles regarding training dynamics.

3.2 Model Search & Validation

After training the supernet, a search phase is implemented to discover the best trained architecture per device tier. Models are sampled with NSGA-II (Deb et al., 2002) and evaluated on a global validation set. The rationale behind selecting this algorithm for our search is the fact that it is multi-objective and allows us for efficient space exploration with the goal of optimising for model size in a specific tier and accuracy. Other search algorithms can be used in place of NSGA-II, based on the objective at hand. Search can stop after a specified number of steps or when an accuracy threshold is met. At the end of this stage, we have a model architecture per tier, already encompassing knowledge across devices of different tiers, which serves as initialisation for further training. Formally, model selection for the i^{th} tier can be defined as:

$$p_i^\star = \arg\min_{p \in \mathbb{P}} \mathbb{E}_{x \sim V} \mathcal{L}(f_{\omega_g^\star}(x, p)) \quad \text{s.t.} \quad \Phi_{\text{train}}(p) \leq B^{(i)}_{\Phi_{\text{train}}}, \tag{6}$$

where $\mathbb{P}$ is the set of all paths, V is a validation dataset at hand, $B^{(i)}_{\Phi_{\text{train}}}$ is the budget for tier i and $\omega_g^\star$ are converged global weights obtained from supernet training outlined above. Specifically for V, we evaluate a centralised approach, as well as a federated alternative (Sec. 4.7).

3.3 Fine-tuning Phase

Subsequently, we move to train each of the previously produced models in a federated manner across all eligible clients. This means that architectures falling in a specific tier, in terms of footprint, can be trained across clients belonging to that tier and above. In the context of Eq. 4 and the related global objective, fine-tuning can be seen as performing analogous optimisation on a subset of the input space. This is due to the input p begin now fixed to one of the selected $p_i^\star$ and x being drawn from a – not necessarily strict – subset of the global distribution D. The main difference lies in the consequences of fixing p, *i.e.*, the set of eligible clients can be drastically limited, resulting in exposure to a smaller subset of D. Considering our supernet training solves a generalised version of the fine-tuning problem with better guarantees about exposure to the training data, we fathom it can provide us with a much stronger initialisation compared to the typical random one. In particular, we anticipate to see significant benefits in cases when otherwise the set of eligible clients for a path $p_i^\star$ would be very small. Here, we use conventional FedAvg and partial client participation per round with the goal of training a single network per tier.

4 Evaluation

This section provides a thorough evaluation of `FedorAS` across different tasks to shows its performance and generality. First, we compare `FedorAS` to existing approaches in the context of federated NAS in the *cross-device* and *cross silo* setting. Next, we draw from the broader FL literature and showcase our technique's performance gains compared to *homogeneous* and *heterogeneous* federated solutions (Sec. 4.2). It is worth noting that many of these techniques (e.g. Reddi et al. (2021); Horvath et al. (2021); Lai et al. (2021)) remain

Table 3: Cross-device federated NAS on CIFAR-10.

Dataset	Method	Mem. Peak (MB)	Perf. (%)
CIFAR10$_{\alpha=1}$	FedNAS	3837	**90.02**
	FedNAS (adj. batch size)	**1919**	85.45
	`FedorAS`	1996	86.46$\pm$0.32
CIFAR10$_{\alpha=0.1}$	FedNAS	3837	65.28
	FedNAS (adj. batch size)	**1919**	54.84
	`FedorAS`	1996	**81.53**$\pm$0.29

Table 4: Cross-silo federated NAS on CIFAR-10.

Dataset	Method	Validation acc.	Test acc.	#clients
CIFAR10$_{\alpha=0.5}$	FedNAS personalised*	90.4$\pm$2.4	-	20
	FedNAS global	-	81.2	16
	`FedorAS`$_{\text{cross-silo}}$	**97.2**$\pm$0.4	**90.6**$\pm$0.2	20
CIFAR10$_{\alpha=0.2}$	SPIDER	-	**92.00**$\pm$2.0	8
	`FedorAS`$_{\text{cross-silo}}$	-	90.82	8

*FedNAS reports validation acc for this setting.

orthogonal to our system and can be combined for additional performance gains. Nevertheless, our gains can be traced back to the benefits of supernet *weight sharing*; we subsequently quantify its contribution by comparing it against randomly initialised networks trained on eligible clients in a federated way (Sec. 4.3) without weight sharing. Last, we showcase the convergence behaviour of `FedorAS` (Sec. 4.6), its impact on client fairness 4.4, the contribution of specific components of our system through an *ablation study* (Sec. 4.5) and the behaviour of *alternative search methods* (Sec. 4.7).

4.1 Experimental Setup

Datasets & baselines. Datasets are summarised in Tab. 2. More information in Appendix E.

Search space. The adopted search spaces are specific to the dataset and task at hand, both in terms of size and type of operators. In general, we assume our highest tier of model sizes to coincide with a typical network footprint used for that task in the FL literature (e.g. ResNet-18 for CIFAR-10). Nevertheless, there may be operators in the space that cannot be sampled in some tiers due to their footprint. `FedorAS` sets the communication budget $B_{\Phi_{\text{comm}}}$ to be half the size of the supernet. The full set of available operators per task is provided in the Appendix E.3.

Clusters definition. In our experiments, we cluster clients based on the amount of operations they can sustain per round of training (#FLOPs), for simplicity. Other resources (e.g. #parameters, energy consumption) can be used in place or in conjunction with FLOPs. More details in Appendix E.4.

4.2 Performance Evaluation

Federated NAS. We start the evaluation by comparing our system with existing works in the federated NAS domain. Specifically, we find two systems that perform federated NAS, namely FedNAS (He et al., 2020a) and SPIDER (Mushtaq et al., 2021) and compare in the *cross-device* and *cross-silo* settings. In the former setting, we adapt FedNAS to support partial participation and compare their technique to `FedorAS` under the same *cross-device* settings in CIFAR-10. Results are shown in Tab. 3, with `FedorAS` performing 1.04% better than FedNAS on CIFAR-10$_{\alpha=1}$ and 48.7% on CIFAR-10$_{\alpha=0.1}$, for the same training memory footprint. Further details can be found in Appendix E.6. At the same time, while running in *cross-silo* setting is not the main focus of `FedorAS`, we have adapted our experimental setting to match that of FedNAS and SPIDER. Results are shown in Tab. 4 with `FedorAS` performing +11.6% and -1.3% than the baselines, respectively, on the test set of their selected settings.

Federated Learning baselines. Next, we compare the performance of `FedorAS` with different federated baselines, including *homogeneous* (Reddi et al., 2021) and system *heterogeneous* frameworks (Horvath et al., 2021; Lai et al., 2021; Qiu et al., 2022). In the former setting, we compare with results from (Reddi et al., 2021) on CIFAR100$_{\alpha=0.1}$. `FedorAS` performs 1 pp[5] better than the FedAvg baseline, at 45.84%, but at a fraction of the cost[6]. Simultaneously, retraining the discovered model from scratch using random initialisation under the same training setting as the baseline results in 11.43 pp higher accuracy than the best FedAdam (63.94% vs 52.50%), showcasing the quality of `FedorAS`-produced bespoke architectures.

At this point, we should clarify that NAS has never constituted a direct replacement for techniques stemming from Efficient ML, be it in centralised or federated settings. Effectively, they control different free variables in the architecture of a DNN and explore different optimisation spaces, which can ultimately be combined

---

[5] We use points and percentage points (p and pp, resp.) for absolute performance difference and % for relative difference.

[6] 1.11 vs 0.16 GFLOPS, 11.4M vs 1.62M parameters, 4000 vs 850 global rounds (750 for supernet + 100 for fine-tuning)

Table 5: Comparison with heterogeneous federated baselines. `FedorAS` performs better across datasets.

Dataset	Method	MFLOPs	Params (M)	Performance
CIFAR10$_{\alpha=1000}$	ZeroFL$_{s=90\%}$ (Qiu et al., 2022)	557‡	11.17	82.82±0.64
	FjORD$_{\text{LDA}}$ (Horvath et al., 2021)	[**35**, **139**, 313, 556]	[**0.70**, **2.79**, 6.28, 11.16]	[78.19±1.20, 78.63±1.31, 78.25±1.06, 77.19±0.85]
	`FedorAS`$_{\text{per tier}}$	[111, 140, **256**, **329**]	[2.96, 2.93, **3.35**, **4.32**]	[**89.40±0.19, 89.60±0.15, 89.64±0.22, 89.24±0.29**]
CIFAR10$_{\alpha=1}$	ZeroFL$_{s=90\%}$ (Qiu et al., 2022)	557‡	11.17	81.04±0.28
	FjORD$_{\text{LDA}}$ (Horvath et al., 2021)	[**35**, **139**, 313, 556]	[**0.70**, **2.79**, 6.28, 11.16]	[78.54±0.12, 79.25±0.51, 78.66±0.29, 77.35±0.44]
	`FedorAS`$_{\text{per tier}}$	[116, 183, **262**, **330**]	[2.59, 2.90, **3.55**, **4.31**]	[**85.99±0.13, 86.30±0.41, 86.34±0.19, 86.46±0.32**]
CIFAR10$_{\alpha=0.1}$ (Acc. (%) ↑ is better)	FjORD$_{\text{LDA}}$ (Horvath et al., 2021)	[**35**, **139**, 313, 556]	[**0.70**, **2.79**, 6.28, 11.16]	[61.43±0.39, 60.81±1.42, 59.72±5.19, 57.44±3.53]
	`FedorAS`$_{\text{per tier}}$	[117, 159, **238**, **345**]	[2.17, 3.13, **2.49**, **2.61**]	[**81.01±0.46, 81.53±0.29, 80.64±0.66, 80.85±0.28**]
Shakespeare (Perplexity ↓ is better)	FjORD (Horvath et al., 2021)	[**1, 3, 7, 11, 17**]	[**0.01, 0.04, 0.08, 0.14, 0.21**]	[4.44±0.07, 3.91±0.10, 3.87±0.13, 3.87±0.13, 3.87±0.13]
	`FedorAS`$_{\text{per tier}}$	[7, 12, 15, 21, 24]	[0.09, 0.15, 0.18, 0.26, 0.30]	[**3.43±0.01, 3.39±0.04, 3.38±0.03, 3.40±0.01, 3.42±0.01**]
SpeechCommands (35 classes) (Accuracy (%) ↑ is better)	Oort (Lai et al., 2021)†	2382	21.29	62.20
	PyramidFL (Li et al., 2022)†	2382	21.29	63.84
	`FedorAS`$^{\star}_{\text{best}}$	**10**	**0.63**	**70.10**

† (Lai et al., 2021; Li et al., 2022) perform client selection based on system heterogeneity and are provided for context. FLOPs computed assuming the common (Zhang et al., 2018) 40×51 MFCC features input.
‡ (Qiu et al., 2022) speeds-up training w/ highly-sparse convs, attainable only with specialised h/w. ★ result obtained from the best model of setup in Appendix G.

Table 6: Models discovered by `FedorAS` benefit from weight sharing across tiers. Models resulted from the search stage in `FedorAS` and subsequently FL-finetuned are compared to models using the same architecture but trained end-to-end in an FL manner (i.e. randomly initialised, *rand-init*) on eligible clients.

Dataset	Clients	Setting	Partitioning	Initialisation	Classes	Tier 1	Tier 2	Tier 3	Tier 4
CIFAR-10 (Acc. (%) ↑ is better)	100	Standard	IID$_{\alpha=1000}$	Supernet	10	**89.40±0.19**	**89.60±0.15**	**89.64±0.22**	**89.24±0.29**
				rand-init		89.05±0.17	87.84±0.38	86.18±0.38	81.27±0.81
			non-IID$_{\alpha=1.0}$	Supernet	10	85.99±0.13	**86.30±0.41**	**86.34±0.19**	**86.46±0.32**
				rand-init		**87.12±0.44**	86.29±0.86	85.10±0.44	80.10±1.92
			non-IID$_{\alpha=0.1}$	Supernet	10	**81.01±0.46**	**81.53±0.29**	**80.64±0.66**	**80.85±0.28**
				rand-init		70.61±2.16	70.30±1.90	68.29±0.49	64.87±1.48
CIFAR-100 (Acc. (%) ↑ is better)	500	Standard	non-IID$_{\alpha=0.1}$	Supernet	100	**45.25±0.13**	**45.84±0.18**	**45.42±0.39**	**45.07±0.71**
				rand-init		36.30±0.96	39.26±1.21	39.06±1.32	36.77±1.32
Speech Commands (Acc. (%) ↑ is better)	2112	Standard	*given*	Supernet	12	80.19±1.78	**80.47±1.69**	**81.0±1.58**	**80.56±0.40**
				rand-init		**81.92±1.32**	79.94±0.84	78.57±1.42	79.36±1.67

Dataset	Clients	Setting	Partitioning	Initialisation	Classes	Tier 1	Tier 2	Tier 3	Tier 4	Tier 5
Shakespeare (Perplexity ↓ is better)	715	Standard	*given*	Supernet	90	**3.43±0.01**	**3.39±0.04**	**3.38±0.03**	**3.40±0.01**	**3.42±0.01**
				rand-init		3.44±0.03	3.50±0.02	3.47±0.08	3.52±0.07	3.60±0.04

at deployment. Thus, the goal of the following comparison is to put `FedorAS` in the context of the literature tackling similar issues in *cross-device* FL, rather than replace them.

Nevertheless, with respect to heterogeneous baselines (ZeroFL (Qiu et al., 2022), FjORD(Horvath et al., 2021)), we see that `FedorAS` consistently leads to models with higher accuracy across tiers that are in the respective clients' computational budget (Tab. 5). At the same time, we depict how `FedorAS` performs compared to FjORD and randomly selected architectures trained naively in a resource-aware manner in Fig. 3. One can clearly see that the degrees of architectural freedom that our solution offers leads to significantly better accuracy per resource tier. Notably, we perform 15.20% and 12.58% better on average than FjORD on CIFAR-10 and Shakespeare, respectively, while still respecting client eligibility.

Model accuracy may not seem to scale proportionally to their size. We attribute this to the limited participation eligibility of clients, an innate trait of heterogeneous systems landscape. Normally, this can cause performance gaps due to limited exposure to federated data (Fig. 1). `FedorAS` is able to bridge this gap (+0.03 points (p) avg), by means of weight sharing and OPA, +1.72 p more effectively than FjORD (avg Tier 4 vs Tier 1 gap across datasets).

Additional results on the communication cost and convergence of `FedorAS` as well as alternative client allocation (Lai et al., 2021) to clusters are provided in Appendix I, H.2 and J.3, respectively.

4.3 Supernet Weight-sharing with FedorAS

Here, we evaluate the impact of weight sharing through `FedorAS`' supernet. We compare the performance of `FedorAS`' models to the same architectures models, but trained end-to-end in a federated manner, across

Table 7: Quantification of fairness with per client accuracy statistics, comparing end-to-end with `FedorAS`'s performance. We report on the mean, standard deviation per tier and across tiers and minimum performance (min accuracy or max perplexity) across datasets with per-tier client test sets. Lower standard deviation is better.

Dataset	Mode	# Total Clients	Perf. (across tiers)	Perf. (per tier)	Worst Perf.
CIFAR-10 (Acc. (%) ↑ is better)	Fedoras	100	**81.36±8.58**	[**82.43±7.87, 81.41±9.34, 81.78±8.49, 79.80±8.42**]	**47.00**
	rand-init		65.70 ±13.11	[67.43±12.29, 66.10±14.05, 66.14±13.02, 63.12±12.76]	11.00
CIFAR-100 (Acc. (%) ↑ is better)	Fedoras	500	**45.61**±13.15	[**44.46**±13.81, **45.51**±12.28, **45.19**±13.15, **47.29**±13.15]	**5.00**
	rand-init		31.08±**12.12**	[30.72±**13.16**, 30.65±**11.27**, 30.43±**11.25**, 32.52±**12.59**]	0.00
Shakespeare (Perplexity ↓ is better)	Fedoras	715	**2.93±1.01**	[**2.93±0.97**, **2.94±0.99**, **2.88±1.03**, **2.98±1.04**, **2.94±1.01**]	8.43
	rand-init		3.07±1.10	[3.07±1.05, 3.08±1.10, 3.02±1.13, 3.11±1.13, 3.07±1.10]	**8.36**

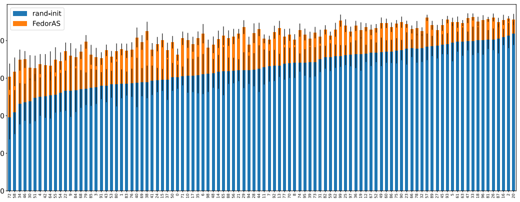

Figure 4: Accuracy per client of `FedorAS` vs. end-to-end (rand-init) FL-trained models (same architecture) on CIFAR-10. Accuracy is quantified on each client's dataset from the model associated with that device's tier and error bars represent the standard deviation of accuracy between different runs. Across runs, the allocation of data to clients does not change. Ordered by ascending accuracy.

all four datasets. With this comparison, we aim to showcase that `FedorAS` not only comes up with better architectures, but it also effectively transfers knowledge between tiers through its supernet structure and OPA aggregation scheme, without the need for full client participation. To accomplish that, we first train with `FedorAS` across the three stages described in Sec. 3 (*supernet-init*). Subsequently, we take the output architectures from our system, randomly initialise them and train end-to-end in a federated manner, where each model trains across all eligible clients (*rand-init*). Results are presented in Tab. 6.

Indeed, models benefit from being initialised from a supernet across cases; this means that weight-sharing mitigates both the limited model capacity of the lower-tier and the limited data exposure of large models. The accuracy improvement is further magnified as non-IID-ness increases, leading up to +15 pp over *rand-init*. Results on different tasks of the same dataset presented in Appendix G.

4.4 Fairness Evaluation

Up to this point, we have only reported global test accuracy, which is aggregated across the clients participating during the test. To provide further insights into the effects our system has on each client individually, here we present a more fain-grained evaluation. In particular, we quantify fairness (how much different clients benefit from weight sharing) by the means of variance and worst-case statistical measures of client performance on their respective test set.

Results are shown in Tab. 7 for the different datasets offering per-client test data splits for all clients. It can be seen that `FedorAS` consistently leads to better performance compared to end-to-end FL trained networks of the same architectures, where only eligible clients can train models. With respect to the standard deviation of per client performance, we witness `FedorAS` offering lower variance, except for the case of CIFAR-100. We consider this to be a consequence of our significantly higher accuracy. Similar behaviour is also witnessed when we measure variance per tier of devices. Last, we also showcase the worst-case result of a clients performance in the last column of the table.

An in-depth view of how each client behaves is depicted in Fig. 4 for CIFAR-10, where we show performance of Fedoras vs. end-to-end FL-trained models per client.

4.5 Ablation Study

Next, we measure the contribution of specific components of `FedorAS` to the quality and performance of these models. To this direction, we firstly compare the performance of our system's aggregation compared to naive averaging. Subsequently, we measure the impact of sending smaller subspaces to clients for supernet training. We provide indicative results on CIFAR-$10_{\alpha=0.1}$.

OPA vs. naive averaging. OPA compared to FedAvg leads to increased accuracy by +1.2, +0.9, +1.01 and +1.78 pp for tiers 1-4, respectively. More details in Appendix H.2.

Subspace sampling size. `FedorAS` samples the supernet before communicating it to a client. For CIFAR-10 ($\alpha = 1.0$) and when $B_{\Phi_{\text{comm}}}$ is set to 22.3M parameters (i.e. allowing sending whole supernet), `FedorAS` yields 85.48%, 86.73%, 86.50% average accuracies across tiers for full, 1/2 and 1/4 of the size of the search space, respectively. The overhead of communication reduced with `FedorAS`and the convergence to the same

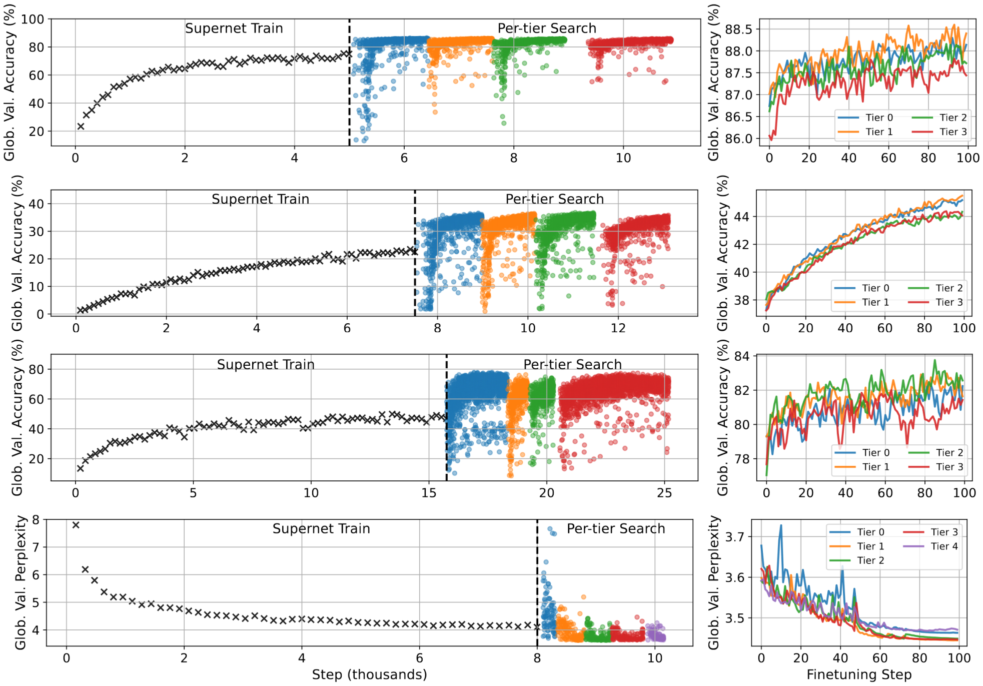

Figure 5: Convergence of `FedorAS` for (from top to bottom): CIFAR-10, CIFAR-100, Speechcommands and Shakesear. For each stage, displayed values follow the relevant optimisation objective discussed in Sec. 3.1, 3.2 and 3.3, respectively from left to right. E.g., notably, supernet training shows performance of the supernet averaged across both data and paths (c.f. Eq. 4). Please note that the notion of a step changes between stages. For Supernet training, one step is equivalent to performing local training on a single client. For search, one step means fully evaluating a single path. Finally, for fine-tuning, one step is one global round. Be mindful of the different ranges of the Y-axes.

level of accuracy is faster. We hypothesise that this is due a better balance in the exploration vs. exploitation trade-off. Details in H.2.

4.6 Convergence Behaviour

In Sec. 3 we argue that our supernet training is analogous to the typical FL training but performed on the extended input space. Here we attempt to empiricaly verify this conjecture. In Fig. 5 we show the behaviour of `FedorAS` across all stages for the datasets considered in this work. In particular, we show representative results from one of the seeds in Table 6 using the default hyperparameters (as per Tables 9 and 10 of the Appendix) and with all datasets using as communication budget ($B_{\Phi_{\text{comm}}}$) half of their respective supernet size. We can see that: 1) across all tasks, our supernet *converges stably to its own local minima*; 2) its absolute performance (avg. across data and paths) is considerably lower than what can be obtained with a single model, which is expected as the joint distribution of data and paths is much harder to optimise compared to just inputs; 3) at the same time, for all tasks and tiers, we can find lots of paths with strong performance in the trained supernet, although there are also those that perform exceptionally bad; 4) in all cases, fine-tuning the weights from *the supernet does not collapse any point* – even though performance might fluctuate, it is always significantly higher than the performance of an average path, the supernet or a randomly initialised network and, with the only exception being Shakespeare Tier 0, which is always higher than the starting point – this is important as it supports our conjecture that fine-tuning, being done on the subset of all data, is not in conflict with supernet training and learned features can be easily transferred.

4.7 Evaluation of the Search Phase

To assess the quality of models during search (Stage 2), so far we have needed a proxy dataset to evaluate different paths and rank them. We consider this as a set of examples that each application provider has centrally to measure the quality of their deployed models in production settings.

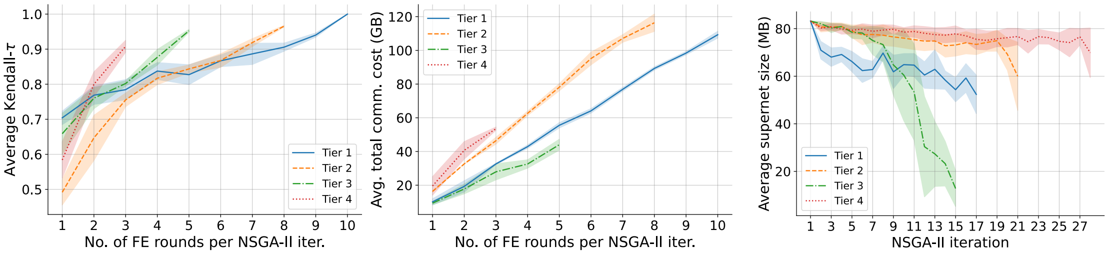

Figure 6: Ranking quality & cost of federated evaluation (FE) of models for CIFAR-10$_{\alpha=0.1}$ during federated search. Each time a new population of models is evaluated, a minimal supernet encompassing selected models is sent to a sample of clients: **left)** ranking correlation between scores produced by FE & centralised evaluation, as a function of FE rounds (↑ rounds = ↑ clients); **middle)** total communication of sending all necessary supernets to all clients to run a full search; **right)** changes in supernet size as NSGA-II progresses.

Validation set size. The size and representativeness of the centralised dataset might affect the quality of the search. To gauge the sensitivity of the end models to the underlying distribution of the validation set, we sample down the validation set, as a held-out portion of the clients datasets, to 20% and 50% of the original size. We find no noticeable impact on final model quality. More in App. K.1.

Federated search. There may be also cases where no such dataset can be centralised. To this direction, we test whether our search can operate under a federated setting with partial participation in order to faithfully rank the quality of models stemming from the supernet. In this setting, we have implemented a *federated version* of NSGA-II. Instead of candidate models being evaluated on the same validation set, they are stochastically evaluated on sampled federated client datasets (Paulik et al., 2021). We hypothesise it is possible to maintain faithful ranking of models compared to centralised evaluation if enough clients are leveraged to evaluate models, at the cost of increased communication cost. Furthermore, we expect the overall cost of achieving robust evaluation to increase as non-IID-ness and #clients increase, and the instantaneous cost of sending models to decrease over time, as NSGA-II converges to well-performing models (i.e. decreased diversity of models). Fig. 6 shows results for CIFAR10$_{\alpha=0.1}$, with extra results and details presented in Appendix K.2. Our experiments support the aforementioned conjectures. Noticeably, even under highly non-IID settings, we can attain faithful FE at a reasonable cost (4 rounds to Kendall-τ>0.8 for the results presented in Fig. 6).

Correlation with final accuracy. We observe that the best models after Stage 2 are rather unlikely to be the best ones after Stage 3 (Kendall-τ 0.2-0.3). However, they tend to achieve decent results, suggesting they are a safe choice if we want to keep fine-tuning to the minimum. See App. K.3.

5 Conclusion

In this work, we have presented `FedorAS`, a system that performs resource-aware federated NAS in the cross-device setting. Not only does `FedorAS` achieve a significantly lower overhead compared to previous federated NAS solutions, but also reaches state-of-the-art accuracy compared to heterogeneous FL techniques from the literature. This is largely enabled via our proposed supernet-training method that enables effective knowledge sharing among clients of different dynamics.

References

Mohamed S Abdelfattah, Abhinav Mehrotra, Łukasz Dudziak, and Nicholas Donald Lane. Zero-cost proxies for lightweight NAS. In *International Conference on Learning Representations (ICLR)*, 2021. URL `https://openreview.net/forum?id=0cmMMy8J5q`.

Ahmed M Abdelmoniem and Marco Canini. Towards mitigating device heterogeneity in federated learning via adaptive model quantization. In *Proceedings of the 1st Workshop on Machine Learning and Systems*, EuroMLSys '21, pp. 96–103, New York, NY, USA, April 2021. Association for Computing Machinery.

Mario Almeida, Stefanos Laskaridis, Abhinav Mehrotra, Lukasz Dudziak, Ilias Leontiadis, and Nicholas D. Lane. Smart at what cost? characterising mobile deep neural networks in the wild. In *Proceedings* *of the 21st ACM Internet Measurement Conference*, IMC '21, pp. 658–672, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450391290. doi: 10.1145/3487552.3487863. URL `https://doi.org/10.1145/3487552.3487863`.

Mohammad Mohammadi Amiri, Deniz Gunduz, Sanjeev R Kulkarni, and H Vincent Poor. Federated Learning with Quantized Global Model Updates. *arXiv preprint arXiv:2006.10672*, 2020.

James Henry Bell, Kallista A Bonawitz, Adrià Gascón, Tancrède Lepoint, and Mariana Raykova. Secure Single-Server aggregation with (Poly)Logarithmic overhead. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1253–1269. Association for Computing Machinery, New York, NY, USA, October 2020.

Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc V. Le. Understanding and simplifying one-shot architecture search. In *International Conference on Machine Learning (ICML)*, 2018.

Axel Berg, Mark O'Connor, and Miguel Tairum Cruz. Keyword Transformer: A Self-Attention Model for Keyword Spotting. In *Proc. Interspeech 2021*, pp. 4249–4253, 2021. doi: 10.21437/Interspeech.2021-1286.

Daniel J. Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Titouan Parcollet, and Nicholas D. Lane. Flower: A friendly federated learning research framework, 2020.

Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for Privacy-Preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pp. 1175–1191, New York, NY, USA, October 2017. Association for Computing Machinery.

Keith Bonawitz et al. Towards Federated Learning at Scale: System Design. In *Proceedings of Machine Learning and Systems (MLSys)*, 2019.

Nader Bouacida, Jiahui Hou, Hui Zang, and Xin Liu. Adaptive federated dropout: Improving communication efficiency and generalization for federated learning. In *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 1–6, 2021. doi: 10.1109/INFOCOMWKSHPS51825.2021.9484526.

James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. Quasi-recurrent neural networks. In *International Conference on Learning Representations (ICLR)*, 2017. URL `https://openreview.net/pdf?id=H1zJ-v5xl`.

H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. *CoRR*, October 2017.

Andrew Brock, Theo Lim, James M. Ritchie, and Nick Weston. SMASH: One-shot model architecture search through hypernetworks. In *International Conference on Learning Representations (ICLR)*, 2018. URL `https://openreview.net/pdf?id=rydeCEhs-`.

Han Cai, Ligeng Zhu, and Song Han. ProxylessNAS: Direct neural architecture search on target task and hardware. In *International Conference on Learning Representations (ICLR)*, 2019. URL `https://arxiv.org/pdf/1812.00332.pdf`.

Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečnỳ, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*, 2018.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches, 2014.

D. Coimbra de Andrade, S. Leo, M. Loesener Da Silva Viana, and C. Bernkopf. A neural attention model for speech command recognition. *ArXiv e-prints*, August 2018.

Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Scharwächter, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset. In *CVPR Workshop on The Future of Datasets in Vision*, 2015.

S. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(4): 357–366, 1980. doi: 10.1109/TASSP.1980.1163420.

K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002. doi: 10.1109/4235.996017.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

Enmao Diao, Jie Ding, and Vahid Tarokh. Heterofl: Computation and communication efficient federated learning for heterogeneous clients. In *International Conference on Learning Representations*, 2020.

Xuanyi Dong and Yi Yang. Searching for a robust neural architecture in four gpu hours. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1761–1770, 2019.

Łukasz Dudziak, Thomas Chau, Mohamed Abdelfattah, Royson Lee, Hyeji Kim, and Nicholas Lane. Brp-nas: Prediction-based nas using gcns. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 10480–10490. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/file/768e78024aa8fdb9b8fe87be86f64745-Paper.pdf`.

Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized Federated Learning with Theoretical Guarantees: A Model-Agnostic Meta-Learning Approach. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. In *European Conference on Computer Vision*, pp. 544–560. Springer, 2020.

Pengchao Han, Shiqiang Wang, and Kin K Leung. Adaptive Gradient Sparsification for Efficient Federated Learning: An Online Learning Approach. In *IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2020.

C He, M Annavaram, and S Avestimehr. Fednas: Federated deep learning via neural architecture search. *arXiv e-prints*, 2020a.

Chaoyang He, Murali Annavaram, and Salman Avestimehr. Group knowledge transfer: Federated learning of large cnns at the edge. *Advances in Neural Information Processing Systems*, 33:14068–14080, 2020b.

Minh Hoang and Carl Kingsford. Personalized neural architecture search for federated learning, 2022. URL `https://openreview.net/forum?id=WcZUevpX3H3`.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, nov 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL `https://doi.org/10.1162/neco.1997.9.8.1735`.

Samuel Horvath, Stefanos Laskaridis, Mario Almeida, Ilias Leontiadis, Stylianos I Venieris, and Nicholas D Lane. Fjord: Fair and accurate federated learning under heterogeneous targets with ordered dropout. *arXiv preprint arXiv:2102.13451*, 2021.

Kevin Hsieh, Amar Phanishayee, Onur Mutlu, and Phillip Gibbons. The Non-IID Data Quagmire of Decentralized Machine Learning. In *International Conference on Machine Learning (ICML)*, 2020.

Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*, 2019.

Andrey Ignatov, Radu Timofte, Andrei Kulik, Seungsoo Yang, Ke Wang, Felix Baum, Max Wu, Lirong Xu, and Luc Van Gool. Ai benchmark: All about deep learning on smartphones in 2019. In *International Conference on Computer Vision (ICCV) Workshops*, 2019.

Yuang Jiang, Shiqiang Wang, Víctor Valls, Bong Jun Ko, Wei-Han Lee, Kin K. Leung, and Leandros Tassiulas. Model pruning enables efficient federated learning on edge devices. *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–13, 2022. doi: 10.1109/TNNLS.2022.3166101.

William M. Fisher John S. Garofolo, Lori F. Lamel et al. Timit acoustic-phonetic continuous speech corpus. In *Linguistic Data Consortium, Philadelphia*, 1983. doi: https://doi.org/10.35111/17gk-bn40.

Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

Jiawen Kang, Zehui Xiong, Dusit Niyato, Shengli Xie, and Junshan Zhang. Incentive mechanism for reliable federated learning: A joint optimization approach to combining reputation and contract theory. *IEEE Internet of Things Journal*, 6(6):10700–10714, 2019. doi: 10.1109/JIOT.2019.2940820.

Alind Khare, Animesh Agrawal, Myungjin Lee, and Alexey Tumanov. Superfed: Weight shared federated learning. *arXiv preprint arXiv:2301.10879*, 2023.

Taehyeon Kim and Se-Young Yun. Supernet training for federated image classification under system heterogeneity, 2022.

Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Peter Richtarik, Ananda Theertha Suresh, and Dave Bacon. Federated Learning: Strategies for Improving Communication Efficiency. In *NeurIPS Workshop on Private Multi-Party Machine Learning*, 2016.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Fan Lai, Xiangfeng Zhu, Harsha V. Madhyastha, and Mosharaf Chowdhury. Oort: Efficient federated learning via guided participant selection. In *15th USENIX Symposium on Operating Systems Design and Implementation (OSDI 21)*, pp. 19–35. USENIX Association, July 2021. ISBN 978-1-939133-22-9. URL `https://www.usenix.org/conference/osdi21/presentation/lai`.

Hayeon Lee, Sewoong Lee, Song Chong, and Sung Ju Hwang. Help: Hardware-adaptive efficient latency prediction for nas via meta-learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Ang Li, Jingwei Sun, Pengcheng Li, Yu Pu, Hai Li, and Yiran Chen. Hermes: an efficient federated learning framework for heterogeneous mobile clients. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, MobiCom '21, pp. 420–437, New York, NY, USA, October 2021a. Association for Computing Machinery.

Ang Li, Jingwei Sun, Binghui Wang, Lin Duan, Sicheng Li, Yiran Chen, and Hai Li. Lotteryfl: Empower edge intelligence with personalized and communication-efficient federated learning. In *2021 IEEE/ACM Symposium on Edge Computing (SEC)*, pp. 68–79, 2021b. doi: 10.1145/3453142.3492909.

Changlin Li, Jiefeng Peng, Liuchun Yuan, Guangrun Wang, Xiaodan Liang, Liang Lin, and Xiaojun Chang. Blockwisely supervised neural architecture search with knowledge distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020a.

Chenning Li, Xiao Zeng, Mi Zhang, and Zhichao Cao. Pyramidfl: A fine-grained client selection framework for efficient federated learning. In *Proceedings of the 28th Annual International Conference on Mobile Computing and Networking*, MobiCom '22. Association for Computing Machinery, 2022.

Daliang Li and Junpu Wang. FedMD: Heterogenous Federated Learning via Model Distillation. In *NeurIPS 2019 Workshop on Federated Learning for Data Privacy and Confidentiality*, 2019.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127*, 2018.

Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated Learning: Challenges, Methods, and Future Directions. *IEEE Signal Processing Magazine*, 2020b.

Tian Li, Maziar Sanjabi, Ahmad Beirami, and Virginia Smith. Fair resource allocation in federated learning. In *International Conference on Learning Representations*, 2020c. URL `https://openreview.net/forum?id=ByexElSYDr`.

Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. Ditto: Fair and robust federated learning through personalization. In *International Conference on Machine Learning*, pp. 6357–6368. PMLR, 2021c.

Or Litany, Haggai Maron, David Acuna, Jan Kautz, Gal Chechik, and Sanja Fidler. Federated learning with heterogeneous architectures using graph hypernetworks, 2022. URL `https://openreview.net/forum?id=7x_47XJULn`.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *International Conference on Learning Representations (ICLR)*, 2019.

Renqian Luo, Fei Tian, Tao Qin, Enhong Chen, and Tie-Yan Liu. Neural architecture optimization. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pp. 7827–7838, Red Hook, NY, USA, 2018. Curran Associates Inc.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pp. 1273–1282. PMLR, 2017.

Bert Moons, Parham Noorzad, Andrii Skliar, Giovanni Mariani, Dushyant Mehta, Chris Lott, and Tijmen Blankevoort. Distilling optimal neural networks: Rapid search in diverse spaces. *arXiv:2012.08859*, 2020.

Erum Mushtaq, Chaoyang He, Jie Ding, and Salman Avestimehr. SPIDER: Searching personalized neural architecture for federated learning. *CoRR*, December 2021.

Xuefei Ning, Changcheng Tang, Wenshuo Li, Zixuan Zhou, Shuang Liang, Huazhong Yang, and Yu Wang. Evaluating efficient performance estimators of neural architectures. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pp. 5206–5210. IEEE, 2015.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 8026–8037, 2019.

Matthias Paulik, Matt Seigel, Henry Mason, Dominic Telaar, Joris Kluivers, Rogier van Dalen, Chi Wai Lau, Luke Carlson, Filip Granqvist, Chris Vandevelde, et al. Federated evaluation and tuning for on-device personalization: System design & applications. *arXiv preprint arXiv:2102.08503*, 2021.

Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameters sharing. In *International Conference on Machine Learning (ICML)*, pp. 4095–4104, 2018.

Xinchi Qiu, Titouan Parcollet, Javier Fernandez-Marques, Pedro Porto Buarque de Gusmao, Daniel J Beutel, Taner Topal, Akhil Mathur, and Nicholas D Lane. A first look into the carbon footprint of federated learning. *arXiv preprint arXiv:2102.07627*, 2021.

Xinchi Qiu, Javier Fernandez-Marques, Pedro PB Gusmao, Yan Gao, Titouan Parcollet, and Nicholas Donald Lane. ZeroFL: Efficient on-device training for federated learning with local sparsity. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=2sDQwC_hmnM`.

Mirco Ravanelli, Philemon Brakel, Maurizio Omologo, and Yoshua Bengio. Light gated recurrent units for speech recognition. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2(2):92–102, apr 2018. doi: 10.1109/tetci.2017.2762739. URL `https://doi.org/10.1109%2Ftetci.2017.2762739`.

Mirco Ravanelli, Titouan Parcollet, Peter Plantinga, Aku Rouhe, Samuele Cornell, Loren Lugosch, Cem Subakan, Nauman Dawalatabad, Abdelwahab Heba, Jianyuan Zhong, Ju-Chieh Chou, Sung-Lin Yeh, Szu-Wei Fu, Chien-Feng Liao, Elena Rastorgueva, François Grondin, William Aris, Hwidong Na, Yan Gao, Renato De Mori, and Yoshua Bengio. SpeechBrain: A general-purpose speech toolkit, 2021. arXiv:2106.04624.

Sashank J. Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and Hugh Brendan McMahan. Adaptive federated optimization. In *International Conference on Learning Representations*, 2021.

Aviv Shamsian, Aviv Navon, Ethan Fetaya, and Gal Chechik. Personalized federated learning using hypernetworks. In *International Conference on Machine Learning*, pp. 9489–9502. PMLR, 2021.

Han Shi, Renjie Pi, Hang Xu, Zhenguo Li, James Kwok, and Tong Zhang. Bridging the gap between sample-based and one-shot neural architecture search with bonas. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1808–1819. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/file/13d4635deccc230c944e4ff6e03404b5-Paper.pdf`.

Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. Federated Multi-Task Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V. Le. MnasNet: Platform-aware neural architecture search for mobile. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Nguyen Truong, Kai Sun, Siyao Wang, Florian Guitton, and YiKe Guo. Privacy preservation in federated learning: An insightful survey from the gdpr perspective. *Computers & Security*, pp. 102402, 2021.

Roman Vygon and Nikolay Mikhaylovskiy. Learning efficient representations for keyword spotting with triplet loss. In *Speech and Computer*, pp. 773–785. Springer International Publishing, 2021. doi: 10.1007/978-3-030-87802-3_69. URL `https://doi.org/10.1007%2F978-3-030-87802-3_69`.

Hongyi Wang, Scott Sievert, Shengchao Liu, Zachary Charles, Dimitris Papailiopoulos, and Stephen Wright. ATOMO: Communication-Efficient Learning via Atomic Sparsification. *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Pete Warden. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209*, 2018.

Colin White, Arber Zela, Binxin Ru, Yang Liu, and Frank Hutter. How powerful are performance predictors in neural architecture search? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Dezhong Yao, Wanning Pan, Yao Wan, Hai Jin, and Lichao Sun. Fedhm: Efficient federated learning for heterogeneous models via low-rank factorization. *arXiv preprint arXiv:2111.14655*, 2021a.

Dixi Yao, Lingdong Wang, Jiayu Xu, Liyao Xiang, Shuo Shao, Yingqi Chen, and Yanjun Tong. Federated model search via reinforcement learning. In *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*, pp. 830–840, July 2021b.

Kaicheng Yu, Christian Sciuto, Martin Jaggi, and Mathieu Salzmann Claudiu Musat. Evaluating the search phase of neural architecture search. In *International Conference on Learning Representations (ICLR)*, 2020.

Arber Zela, Julien Siems, and Frank Hutter. Nas-bench-1shot1: Benchmarking and dissecting one-shot neural architecture search. In *International Conference on Learning Representations (ICLR)*, 2020.

Chunhui Zhang, Xiaoming Yuan, Qianyun Zhang, Guangxu Zhu, Lei Cheng, and Ning Zhang. Towards tailored models on private AIoT devices: Federated direct neural architecture search. *Corr*, February 2022.

Yuge Zhang, Zejun Lin, Junyang Jiang, Quanlu Zhang, Yujing Wang, Hui Xue, Chen Zhang, and Yaming Yang. Deeper insights into weight sharing in neural architecture search, 2020.

Yundong Zhang, Naveen Suda, Liangzhen Lai, and Vikas Chandra. Hello edge: Keyword spotting on microcontrollers, 2018.

Dongzhan Zhou, Xinchi Zhou, Wenwei Zhang, Chen Change Loy, Shuai Yi, Xuesen Zhang, and Wanli Ouyang. Econas: Finding proxies for economical neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.