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 Machine Learning (ML) powered services that these provide grow in sophistication. With the recent advances in System-On-Chip (SoC) 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 and separate on-device modelling/profiling needs to be done in the case of hardware-aware NAS (Dudziak

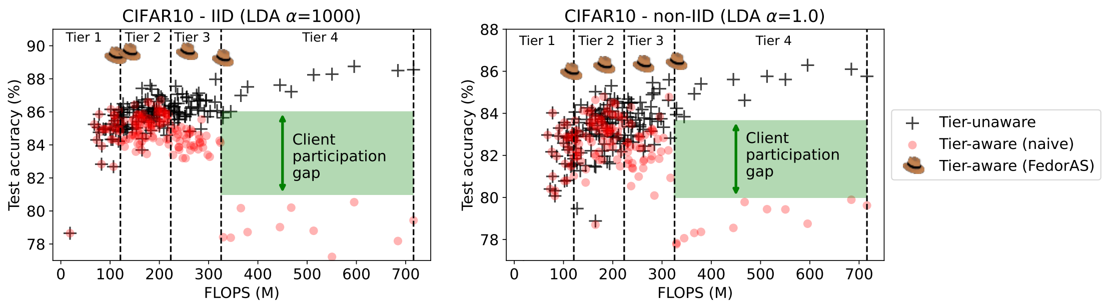


Figure 1: For two LDA settings, 160 architectures are randomly sampled from a ResNet-like search space and trained on a CIFAR-10 FL setup with 100 clients, with 10 clients participating on each round for a total of 500 rounds. Clients are uniformly assigned to a tier, resulting in 25 clients per tier. Tier-awareness refers to whether tier limits are respected or not. Given sufficient data and ignoring tier limits (tier-unaware), 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.

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 exhibit different levels of performance under constrained (*tier-aware*) and unconstrained (*tier-unaware*) participation. The former setting respects device computational capabilities, while the latter does not, which effectively means that larger networks are exposed to more data.

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

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

communication cost (Sec. 3.1 ❶), *ii)* a $\mathcal{O}(1)$ path-sampling mechanism that supports dynamic changes 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), FedRolex (Alam et al., 2022)), 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

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
FedRolex (Alam et al., 2022)	cross-device	global	quantitative	✓	scaling	# tiers	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
FedPM (Isik et al., 2023)	cross-device	global	✗	✗	✓?	-	sampled randomised network
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.

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

Clustered FL. There have been various works (Ghosh et al., 2020; Ruan & Joe-Wong, 2022) that cluster clients together in order to form cohorts and train more efficiently. `FedorAS` assumes that device tiers reflect the device capabilities and are orthogonal to their underlying data distributions, which still remain non-IID. As such, such techniques remain largely complimentary to our contribution.

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. Similar is the case with (Isik et al., 2023), which trains binary masks generated by Bernoulli sampling over frozen randomized networks. While conceptually interesting and with upstream bandwidth savings, performance is heavily impacted. 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

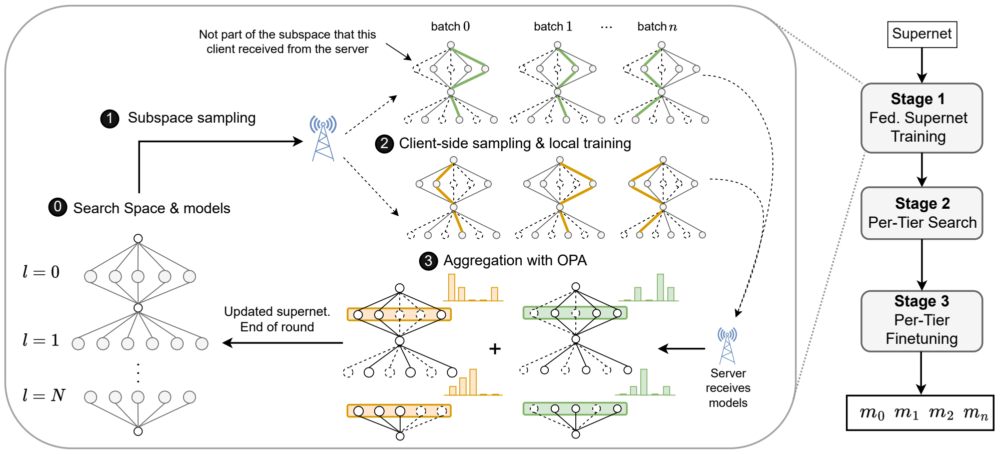

Not part of the subspace that this client received from the server
batch 0
batch 1
batch n
1 Subspace sampling
2 Client-side sampling & local training
0 Search Space & models
$l = 0$
$l = 1$
$l = N$
3 Aggregation with OPA
Updated supernet. End of round
Server receives models
Supernet
Stage 1 Fed. Supernet 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`.

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 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

⓪ 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). In practice, it is up to the model developer to decide the value of $B_{\Phi_{\text{comm}}}$, according to their needs and the characteristic of their deployment environment — `FedorAS` does not assume it to be any particular value. In our experimental setting, we decided to set it to the size of a typical network that can be found in the literature, to facilitate comparison. This happens to be roughly half of a supernet used for each task. We investigate the impact of different budgets in Sec. 4.5 and G.2. Notably, we observe that sending fewer parameters can in fact accelerate convergence.

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

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

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 — which prunes operators from a supernet with equal probability until its size fits in the budget — 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 $B^{\text{Tier}}_{\Phi_{\text{train}}}$. 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[3]. Since 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

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

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:

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

where $\omega_g^{(t)}(\cdot)$ are global weights, $\omega_i^{(t)}(\cdot)$ are local weights of client i at global step t, $|D_{i,o_l}^{(t)}|$ is the number of samples having backpropagated through an operator o_l for client i in round t, and $C_{o_l}^{(t)}$ is the set of clients i s.t. $|D_{i,o_l}^{(t)}| > 0$. Updates to $\omega_g^{(t)}$ happen only if $|C_{o_l}^{(t)}| > 1$ 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 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:

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

where $\mathbb{P}$ is the set of all paths, V is a validation dataset at hand, $B_{\Phi_{\text{train}}}^{(i)}$ 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

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

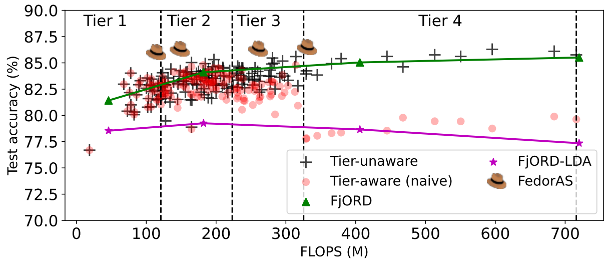


Figure 3: `FedorAS` outperforms other approaches (details in Appendix E.7). CIFAR-10 (non-IID, α=1.0). FjORD is represented as a line as it can switch between operating points on-the-fly via Ordered Dropout.

noting that many of these techniques (e.g. Reddi et al. (2021); Horvath et al. (2021); Lai et al. (2021)) remain 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[4] better than the FedAvg baseline, at 45.84%, but at a fraction of the cost[5]. Simultaneously, retraining the discovered model from scratch using random initialisation

[4]We use points and percentage points (p and pp, resp.) for absolute performance difference and % for relative difference.
[5]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 3: Cross-device federated NAS on CIFAR-10.

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

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

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

*FedNAS reports validation acc for this setting.

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

Dataset	Method	MFLOPs	Params (M)	Performance
$\text{CIFAR10}_{\alpha=1000}$	$\text{ZeroFL}_{s=90\%}$ (Qiu et al., 2022)	557‡	11.17	82.82±0.64
	$\text{FjORD}_{\text{LDA}}$ (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]
	$\texttt{FedorAS}_{\text{per tier}}$	[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**]
$\text{CIFAR10}_{\alpha=1}$	$\text{ZeroFL}_{s=90\%}$ (Qiu et al., 2022)	557‡	11.17	81.04±0.28
	$\text{FjORD}_{\text{LDA}}$ (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]
	$\texttt{FedorAS}_{\text{per tier}}$	[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**]
$\text{CIFAR10}_{\alpha=0.1}$ (Acc. (%) ↑ is better)	$\text{FjORD}_{\text{LDA}}$ (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]
	$\texttt{FedorAS}_{\text{per tier}}$	[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]
	$\texttt{FedorAS}_{\text{per tier}}$	[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
	$\texttt{FedorAS}^{\star}_{\text{best}}$	**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 H.

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 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. Additional results on the per-tier computation and communication cost, along with the convergence behaviour of `FedorAS` are provided in Appendix E.4, I, G.2, respectively. Moreover, we provide evaluations on alternative client allocation to clusters in Appendix J.3.

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 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

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	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
(Acc. (%) ↑ is better)			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

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
	FjORD		61.37±15.58	[56.34±15.37, 60.96±12.69, 62.92±14.04, 65.24±20.22]	25.00
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**

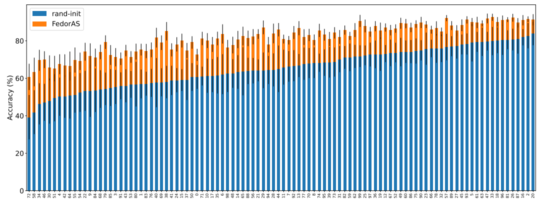


Figure 4: Accuracy per client of FedorAS vs. end-to-end (rand-init) FL-trained models (same architecture) on CIFAR-10. Accuracy is quantified on each client's dataset from the model associated with that device's tier and error bars represent the standard deviation of accuracy between different runs.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

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 H.

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

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

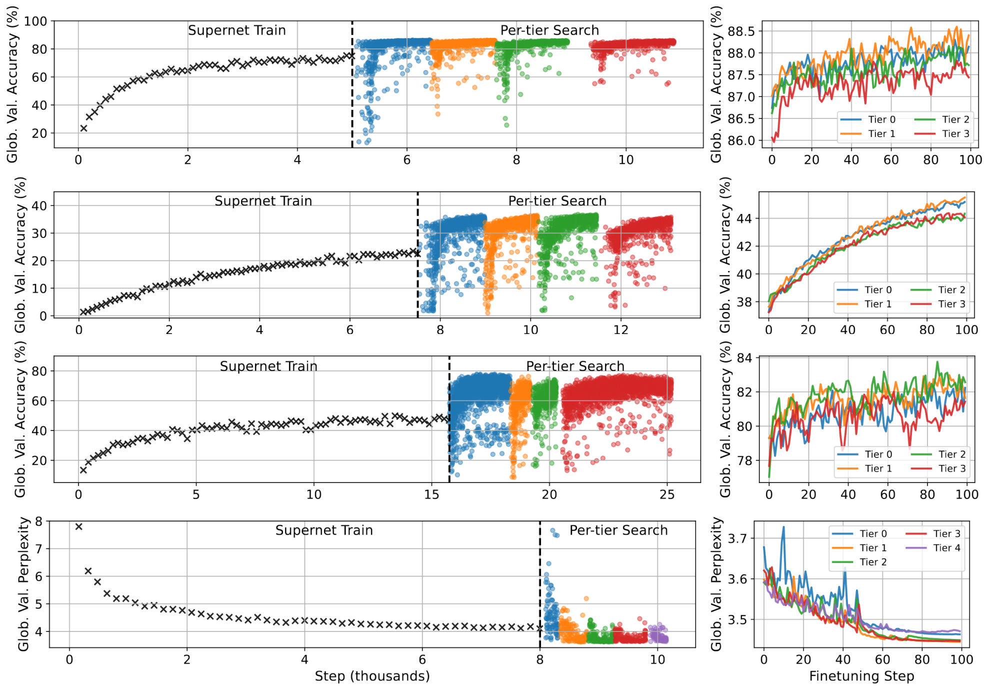


Figure 5: Convergence of `FedorAS` for (from top to bottom): CIFAR-10, CIFAR-100, Speechcommands and Shakesear. For each stage, displayed values follow the relevant optimisation objective discussed in Sec. 3.1, 3.2 and 3.3, respectively from left to right. E.g., notably, supernet training shows performance 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.

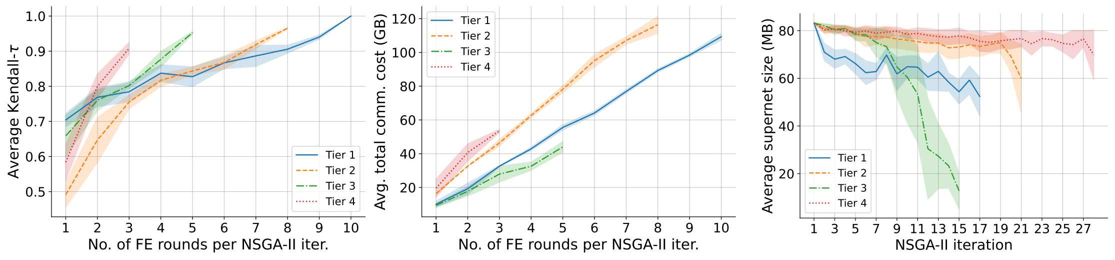


Figure 6: Ranking quality & cost of federated evaluation (FE) of models for CIFAR-10$_{\alpha=0.1}$ during federated search. Each time a new population of models is evaluated, a minimal supernet encompassing selected models is sent to a sample of clients: **left)** ranking correlation between scores 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.

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.

Samiul Alam, Luyang Liu, Ming Yan, and Mi Zhang. Fedrolex: Model-heterogeneous federated learning with rolling sub-model extraction. *Advances in Neural Information Processing Systems*, 35:29677–29690, 2022.

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.

Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. An efficient framework for clustered federated learning. *Advances in Neural Information Processing Systems*, 33:19586–19597, 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.

Berivan Isik, Francesco Pase, Deniz Gunduz, Tsachy Weissman, and Zorzi Michele. Sparse random networks for communication-efficient federated learning. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=k1FHgri5y3-`.

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.

Yichen Ruan and Carlee Joe-Wong. Fedsoft: Soft clustered federated learning with proximal local updating. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 8124–8131, 2022.

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.

Supplementary Material

Table of Contents

A Introduction

This Appendix extends the content of the main paper by providing support material to the study presented in Sec. 3, 4 and, additional insights about `FedorAS`. Concretely, the Appendix is divided into three main blocks of extra content:

- **Impact and Limitations.** We concisely present the broad impact and limitations of our work as well as future research directions in Sec. B, C.
- **Experimental Setup.** Sec. E provides details on the libraries used to build `FedorAS`, the datasets considered for experiments as well as the hyperparameters used to obtain the results presented in Section 4. Crucially, we provide a detailed description of the search spaces utilised in for each dataset domain, how tiers are defined and, a concise description of each baseline method included in this work.
- **Additional Experiments.** Sec. F.1 through 4.4 study different aspects of `FedorAS` such as: *i)* learning multiple tasks using the supernet in Sec. H; or *ii)* the convergence behaviour in Sec. G.2; *iii)* the fairness aspect of `FedorAS`; or under new scenarios altogether, such as: *iv)* an alternative procedure to assign clients to tiers in Section J.3; or *iv)* alternative search methods for Stage 2 in Sec. K.

Overall, the following content substantially extends what is already presented in the main text.

B Broader Impact

Our system, `FedorAS`, performs federated and resource-aware NAS in the cross-device setting. Despite the benefits illustrated compared to centralised NAS and cross-silo FL solutions, running Neural Architecture Search is still a resource-demanding process, in terms of compute, memory and network bandwidth. While `FedorAS`'s target devices can be of significantly lower TDP (i.e. smartphones and IoT devices vs. server-grade GPUs) – with consequences on the overall energy consumption of training (Qiu et al., 2021) – they are resources not directly owned by the operator. As such, special care needs to be taken with respect to how many device resources are leveraged at any point in, so as not hinder the usability of the device or invoke monetary costs to the user (Kang et al., 2019).

C Limitations & Future Work

Despite the challenges addressed by `FedorAS`, our prototype has certain limitations. First and foremost, we have opted to cluster devices (i.e. in tiers) based on their FLOPS. While it is perfectly normal to divide clusters based on other criteria (e.g. memory, latency, energy) or in a multi-objective manner, we have kept it simple. Moreover, one can opt for biased sampling[6] of i) clients participating in a round, ii) the subspace they get communicated and iii) the paths sampled from that subspace, we opted for uniform sampling for all of the above for simplicity, uniform coverage of the search space and fairness in participation. We leave the exploration of such strategies as future work, .

Efficiency-wise, while `FedorAS` can achieve superior performance to the selected baselines, it is evident that we need to perform additional rounds for supernet training and fine-tuning compared to single model training (Horvath et al., 2021; Qiu et al., 2022). This involves pretraining a supernet, which – sampling-aside – affects the communication cost of distributed training proportionally to the size of the search space. Our current search spaces reflected the networks commonly deployed in the selected tasks. As a future research direction, we would like to delve into designing spaces tailored for federated deployment across tasks along with more sophisticated sampling and low-precision methods to minimise the communication cost.

Our method also has not stepped into the area of federated personalisation. Therefore, we have steered away of comparing against such solutions from the literature. Nevertheless, as follow-up work, one could explore how to further tune the architecture and weights of models in a more granular manner, per client device.

[6]based on the data or layer importance.

Last but not least, we have considered privacy-enhancing techniques, such as Differential Privacy (Brendan McMahan et al., 2017) or Secure Aggregation (Bonawitz et al., 2017; Bell et al., 2020), as orthogonal to our scheme. Combining Federated NAS with such strategies can expose interesting trade-offs of exploration-exploitation-privacy budgets that could be explored in the future.

D Extended Background

This section briefly introduces Federated Learning and NAS for a general audience.

D.1 Federated Learning Overview

A typical FL pipeline is comprised of three distinct stages: given a *global* model initialised on a central server, $\omega_g^{(t=0)}$, *i)* the server randomly samples k clients out of the available K ($k \ll K$ for *cross-device*; (Bonawitz et al., 2019) $k = K$ for *cross-silo* setting) and sends them the current state of the global model; *ii)* those k clients perform training on-device using their own data partition, D_i, for a number of epochs and send the updated models, $\omega_i^{(t)}$, back to the server after local training is completed; finally, *iii)* the server aggregates these models and a new *global* model, $\omega_g^{(t+1)}$, is obtained. This aggregation can be implemented in different ways (McMahan et al., 2017; Reddi et al., 2021; Li et al., 2018). For example, in FedAvg (McMahan et al., 2017) each update is weighted by the relative amount of data on each client: $\omega_g^{(t+1)} = \sum_{i=0}^{k} \frac{|D_i|}{\sum_{j=0}^{k} |D_j|} \omega_i^{(t)}$. Stages *ii)* and *iii)* repeat until convergence. The quality of the *global* model ω_g can be assessed: on the *global* test set; by evaluating the fit of the ω_g to each participating client's data (D_i) and derive fairness metrics (Li et al., 2020c); or, by evaluating the adaptability of the ω_g to each client's data or new data these might generate over time, this is commonly known as *personalised FL* (Fallah et al., 2020; Li et al., 2021c).

D.2 Neural Architecture Search Overview

Neural Architecture Search. NAS is usually defined as a bi-level optimisation problem:

$$a^\star = \underset{a \in \mathbb{A}}{\arg\min}\, \mathcal{L}(\omega_a^*(D_t), D_v), \quad \text{where } \omega_a^*(D_t) = \underset{\omega_a}{\arg\min}\, \mathcal{L}(\omega_a, D_t) \tag{7}$$

where $\mathbb{A}$ is a finite (discrete) set of architectures to search from (a *search space*), $\mathcal{L}$ is a loss function, ω_a are weights of a model with architecture a, and $D_{\{v,t\}}$ are validation and training datasets, respectively. The main challenge of NAS comes directly from the fact that in order to assess quality of different architectures (*outer optimisation*), we have to obtain their optimal weights which entitles conducting full training (*inner optimisation*).

There exist multiple approaches to speed up NAS (Pham et al., 2018; Liu et al., 2019; Dong & Yang, 2019; Cai et al., 2019; Dudziak et al., 2020; Zhou et al., 2020; Abdelfattah et al., 2021; Luo et al., 2018; Shi et al., 2020; Guo et al., 2020; Moons et al., 2020). More relevant to our work are those utilising the concept of a supernet (Brock et al., 2018; Bender et al., 2018); where a single model that encapsulates all possible architectures from the search space is created and trained. Specifically, a supernet is constructed by defining an operator that incorporates all candidate operations (the set of which we denote by $\mathbb{O}_l$) for each searchable layer l. A common choice is to define it as a weighted sum of candidates' individual outputs $y_l = \sum_{o \in \mathbb{O}_l} \alpha_l^{(o)} * o(y_{l-1})$, where factors $\{\alpha_l^{(o)}\}_{o \in \mathbb{O}_l}$ of each layer l can be defined differently for different methods (e.g, continuous parameters (Liu et al., 2019; Cai et al., 2019; Dong & Yang, 2019) or random one-hot vectors (Guo et al., 2020)). Importantly for us, methods that use sparse weighting factors can avoid executing operations associated with zero weights, saving both memory and compute (Cai et al., 2019; Guo et al., 2020).

After a supernet has been constructed and trained, searching for $a^\star$ is usually performed by either investigating architectural parameters (Liu et al., 2019; Dong & Yang, 2019; Cai et al., 2019), or using zero-th order optimisation methods to directly solve the outer loop of Eq. 7 while approximating ω_a^* with weights taken from the supernet (thus avoiding the costly inner optimisation) (Li et al., 2020a; Guo et al., 2020). The final model can then be retrained in isolation using either random initialisation or weights from the supernet.

E Detailed Experimental Setup

E.1 Implementation

`FedorAS` was implemented on top of the Flower (v0.18) (Beutel et al., 2020) framework and PyTorch (v1.11.0) (Paszke et al., 2019). We run all our experiments on a private cloud cluster in a simulated manner, across four iterations each and report averages and standard deviations.

E.2 Datasets

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 $\alpha = 0.1$ as in (Reddi et al., 2021). The remaining datasets come naturally partitioned.

CIFAR-10/100. The CIFAR-10 datasets contains 10k and 50k 32x32 RGB images in its test and training sets respectively comprising ten classes. The goal is to classify these images correctly. Similarly, CIFAR-100 follows an identical partitioning but, this time, across 100 classes (*fine* lebels) or 20 superclasses (*coarse* labels). Both CIFAR datasets have a uniform coverage across their classes.

SpeechCommands. The Speech Commands datasets (Warden, 2018) is comprised of 105,829, 16KHz 1-second long audio clips of a spoken word (e.g. "yes", "up", "stop") and the task is to classify these in a 12 or 35 classes setting. The datasets comes pre-partitioned into 35 classes and in order to obtain the 12-classes version, the standard approach (Berg et al., 2021; Coimbra de Andrade et al., 2018; Vygon & Mikhaylovskiy, 2021) is to keep 10 classes of interest (i.e. "yes", "no", "up", "down", "left", "right", "on", "off", "stop", "go"), place the remaining 25 under the "unknown" class and, introduce a new class "silence" where no spoken word appear is the audio clip. In this work we consider SpeechCommandsV2, the most recent version of this dataset. The dataset spans three disjoint set of speakers: 2112 form the training set, 256 for validation and, 250 for testing. In `FedorAS`, the supernet training phase makes uses of the 2112 clients in the training partition only. The results are obtained by measuring the performance of the discovered models on the data of the 250 clients comprising the test set. Data is not uniformly distributed and some clients have more data than others. This dataset is processed by first extracting MFCC (Davis & Mermelstein, 1980) features from each audio clip (Zhang et al., 2018; Berg et al., 2021). Across our experiments we extract 40 MFCC features from a MelSpectrogram where each audio signal is first sub-sampled down to 8KHz and then sampled using 40ms wide time windows strided 20ms appart. This results in 1-second audio clip being transformed into a 1×40×51 input that can be passed to a CNN.

Shakespeare. This dataset is built (Caldas et al., 2018; McMahan et al., 2017) from *The Complete Works of William Shakespeare* and partitioned in such a way data that from each role in the play is assigned to a client. This results in a total of 1,129 partitions where the average number of samples per device is 3.7K and the standard deviation 6.2K samples. This makes Shakespeare a relatively imbalanced dataset. The task is to correctly predict the next character in a dialog given the previously seen characters in the sentence. The vocabulary considered has 86 English characters as well as four special tokens: start and end of sentence, padding and out-of-vocabulary tokens.

E.3 Search Spaces

We assume that the largest model in the search space is the maximal size that any of the clients can handle. The minimum cost is set to the fixed cost of the network (i.e. cost of non-searchable components). Given this range, and number of clusters C, we define the first and last clusters at $prct_l, prct_r$ percentiles of a randomly sampled set of models from the search space and linearly scale the $C - 2$ clusters in-between. All supernet search spaces are depicted in Fig. 7 and described below.

CIFAR-10 and CIFAR-100. We use a ResNet-like search space similar to the one used in, for example, Cai et al. (Cai et al., 2019). Specifically, our model is a feedforward model with a fixed (*i.e.* non-searchable) stem layer followed by a sequence of 3 searchable blocks, each followed by a fixed reduction block. A standard

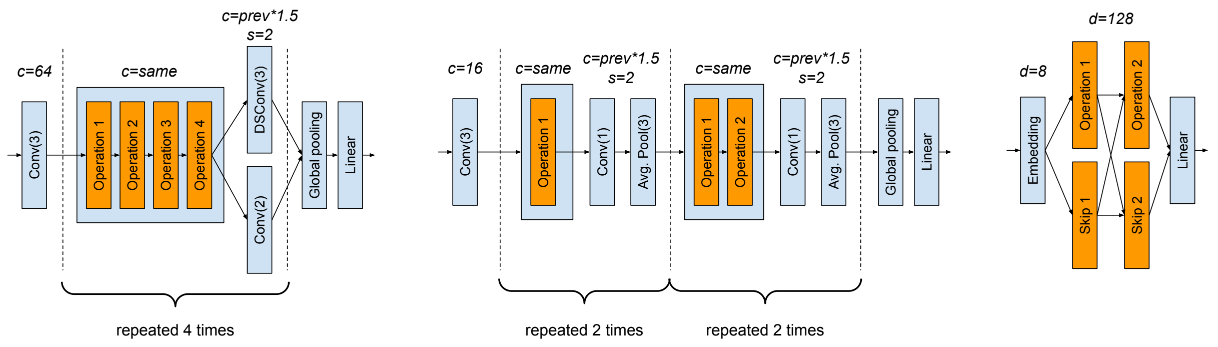


Figure 7: Summary of models used in our experiments: **left)** CIFAR-10 and CIFAR-100, **middle)** Speech Commands, **right)** Shakespeare. Blocks highlighted in blue are fixed, orange blocks represent searchable layers. c – output channels, s – stride, d – output feature dimension, DSConv – depthwise separable convolution. Whenever a layer has more than one input they are added. All convolutions are followed by BN and ReLU. For convolution and pooling operations, numbers in parentheses represent window sizes.

block consists of 4 searchable layers organized again in a simple feedforward manner. Operations within a standard block preserve shape of their inputs, but are allowed to change dimensions of intermediate results. On the other hand, the goal of reduction blocks is to reduce spatial resolution (2× in each dimension) and increase number of channels (1.5×). Reduction blocks are fixed throughout and consists of a depthwise separable convolution 3×3, with the first part performing spatial reduction and the second increasing the number of channels, and a standard 2×2 convolution applied to a residual link. The sequence of blocks is finished with a global average pooling per-channel and a classification layer outputting either 10 or 20/100 logits, which is the only difference between the two models. Each convolution operation is followed by a batch normalization (BN) and ReLU.

In our experiments we considered the following candidate operations:

- standard convolution 1×1 with BN and ReLU,
- depthwise separable convolution 3×3 with expansion ratio (controlling the number of intermediate channels) set to $\{0.5, 1, 2\}$, with each expansion ratio being an independent choice in our search space;
- MobileNet-like block consisting of a convolution with kernel size k and expansion ratio e, followed by a Squeeze-and-Excitation layer, followed by a 1×1 convolution reverting the channels expansion, we considered $\{(k = 1, e = 2), (k = 3, e = 0.5), (k = 3, e = 1), (k = 3, e = 2)\}$;
- identity operation.

The stem layer was set to a 3×3 convolution outputting 64 channels.

Speech Commands. The model follows the one used for the CIFAR datasets but is made more lightweight – to roughly match what can be found in the literature – by reducing stem channels to 16 and including only 1 (resp. 2) searchable operations in the first (resp. last) two blocks. Additionally, reduction block only includes a single 1×1 convolution, that changes the number of channels, followed by a 3×3 average pooling that reduces spatial dimensions. We also include additional candidate operations:

- standard convolution 3×3,
- depthwise separable convolution with kernel size 1 and expansion ratio 2.

All other operations from the CIFAR model are also included.

Shakespeare. We base our design on the model used by FjORD. (Horvath et al., 2021). Specifically, the model is a recurrent network that begins with a fixed embedding layer outputting 8-dimensional feature vector per each input character. This is then followed by a searchable operation #1 that increases dimensionality from 8 to 128; in parallel, we have a searchable skip connection #1 operation whose output is added to the output of the operation #1. Later there is another searchable operation #2 with its own skip connection

Table 8: This table summarises how each search space is split into tiers. Parameters ρ_L and ρ_H are used to conveniently split the FLOPs axis for each dataset and present challenging scenarios for `FedorAS`. The last column refers to the percentage of the total clients that are assigned to each tier.

Dataset	$[\rho_L, \rho_H]$	Tier	FLOPs range	# Models	Portion(%)	Clients (%)
CIFAR-10	[0.0, 0.95]	T1	[0, 120.9M]	$239.4 \cdot 10^{12}$	12.92%	25%
		T2	(120.9M, 223.2M]	$1089.4 \cdot 10^{12}$	58.80%	25%
		T3	(223.2M, 325.4M]	$477.2 \cdot 10^{12}$	25.75%	25%
		T4	(325.4M, 716.0M]	$46.9 \cdot 10^{12}$	2.53%	25%
CIFAR-100	[0.0, 0.9]	T1	[0, 111.5M]	$168.4 \cdot 10^{12}$	9.09%	25%
		T2	(111.5M, 204.3M]	$975.4 \cdot 10^{12}$	52.64%	25%
		T3	(204.3M, 297.1M]	$609.5 \cdot 10^{12}$	32.89%	25%
		T4	(297.1M, 716.0M]	$99.7 \cdot 10^{12}$	5.38%	25%
CIFAR-100 (multi-task setting of Appendix H)	[0.0, 0.9]	T1	[0, 111.5M]	$168.4 \cdot 10^{12}$	9.09%	50%
		T2	(111.5M, 204.3M]	$975.4 \cdot 10^{12}$	52.64%	25%
		T3	(204.3M, 297.1M]	$609.5 \cdot 10^{12}$	32.89%	12.5%
		T4	(297.1M, 716.0M]	$99.7 \cdot 10^{12}$	5.38%	12.5%
SpeechCommands	[0.3, 0.925]	T1	[0, 5.0M]	$827.5 \cdot 10^{3}$	46.71%	80%
		T2	(5.0M, 7.5M]	$617.4 \cdot 10^{3}$	34.85%	1.25%
		T3	(7.5M, 10.1M]	$260.2 \cdot 10^{3}$	14.69%	1.25%
		T4	(10.1M, 20.0M]	$66.4 \cdot 10^{3}$	3.75%	17.5%
Shakespeare	[0.1, 0.77]	T1	[0, 7.8M]	316	13.48%	20%
		T2	(7.8M, 12.8M]	577	24.54%	20%
		T3	(12.8M, 17.8M]	787	33.48%	20%
		T4	(17.8M, 22.8M]	473	20.14%	20%
		T5	(22.8M, 33.8M]	196	8.37%	20%

#2, both keeping the hidden dimension at 128. Again, their outputs are added and passed to the final classification layer.

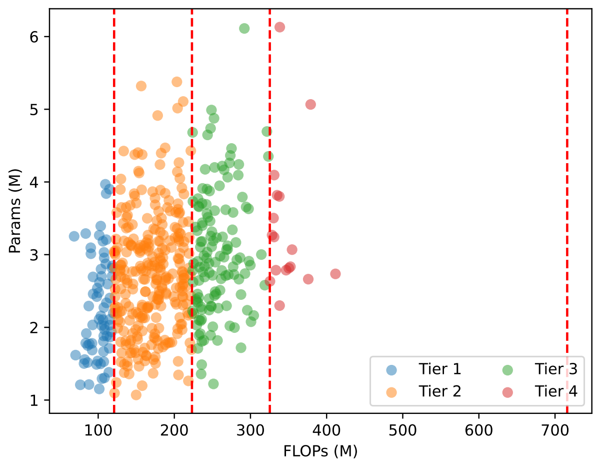
Candidate operations for each of the four searchable layers are mainly the same, with minor adjustments made to make sure a valid model is always constructed. These include:

- an LSTM layer (Hochreiter & Schmidhuber, 1997),
- a GRU layer (Cho et al., 2014),
- a LiGRU layer with `tanh` activation (Ravanelli et al., 2018),
- a QuasiRNN layer (Bradbury et al., 2017),
- a simple Linear layer (no recurrent connection) followed by a `sigmoid` activation,
- a 1D convolution with kernel 5 spanning time dimension (looking at the current character and 4 previous ones), followed by a `sigmoid` activation (no normalisation);
- identity operation (only included in later operation, after feature dimension has been increased to 128);
- zero operation, outputting zeros of the correct shape (only included in skip connection layers).

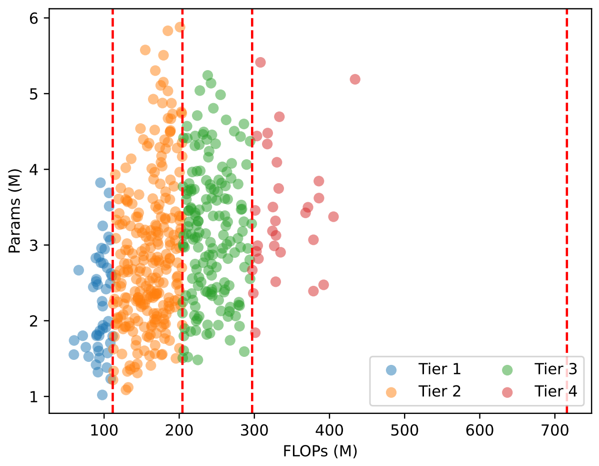
For LiGRU and QuasiRNN we used implementations provided by the Speechbrain project (Ravanelli et al., 2021), after minor modifications. For others, we used standard operations from PyTorch. All operations were used as unidirectional. We did not use any normalisation throughout the model.

E.4 Tiers: Definition and Client Assignment

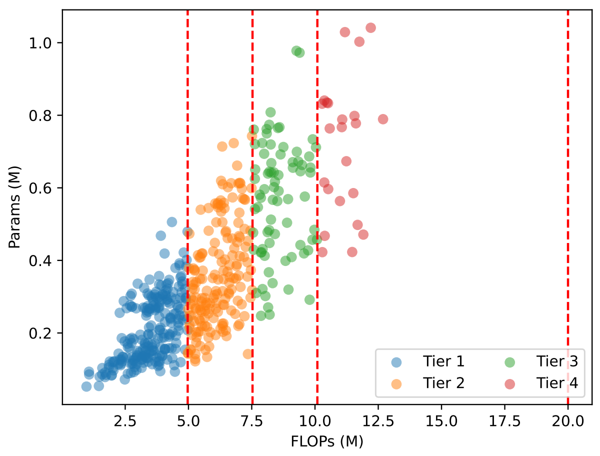
With `FedorAS` the discovery and training of architectures happens in a tier-aware fashion as a federated process. In this work we considered splitting each search space along the FLOPs dimensions (but other splits are possible, e.g: energy, peak-memory usage, etc – or a combination of these). Fig. 8 illustrates the span in terms of model parameters and FLOPs of each search space presented in E.3 and the split along the FLOPs dimensions for each of them. These search spaces vary considerably in terms of size and span, which motivated us to use different number of tiers or client-to-tier assignment strategies. Tab 8 shows the



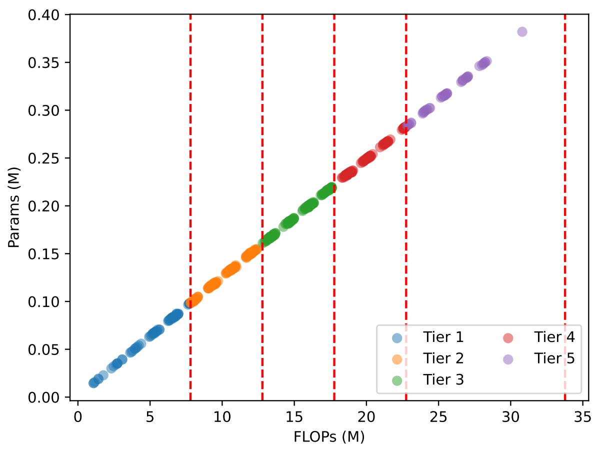
(a) CIFAR-10



(b) CIFAR-100



(c) SpeechCommands



(d) Shakespeare

Figure 8: For each search space, we randomly sample 500 architectures and color code them based on the tier they belong to. Vertical dashed lines represent the boundaries between device tiers. For Shakespeare, the majority of the candidate operators in the searchspace include just linear layers or no-op layers, as a results the number of FLOPs grows almost linearly with the model of parameters.

the FLOPs ranges considered for each tier, the number of models in those sub-spaces exposed to `FedorAS` as well as the ratio of models in the entire search space that fall onto each tier.

Identifying FLOPs ranges for each tier. Regardless of the dataset, we follow a common approach to divide the FLOPs dimension that each dataset spans. The main aim is to finely control how many models fall onto the smallest/largest tier, given that these are considerably sparse regions of the entire search space. If we were to evenly split the FLOPs dimension, almost no architecture would fall onto the largest tier. To simplify the process of defining where each tier's boundaries lie, we follow these steps which require two hyperparameters: (1) we construct a long array of FLOPs from models sampled from the search space and this array is then sorted from lowest to highest; then, (2) we normalise this array of FLOPs and compute the cumulative sum of such normalised array; finally, (3) we identify the lower limit of the highest tier as maximum FLOPs value found in the first $\rho N, \rho \in [0, 1]$, elements of the array where N is the number of samples taken (we found 100K to work well). Essentially, we find the FLOPs for which the approximated PDF over the FLOPS of a given search space doesn't surpass ρ. Once that FLOPs value is identified, the FLOPs range is evenly split for the tiers below and the higher limit for the largest tier becomes the maximum FLOPs a model in the search space can have. This is the approach for CIFAR-10 and CIFAR-100

Table 9: Hyperparameters used for the federated supernet training stage in `FedorAS`, as described in Sec. 3.1. The learning rate is kept fixed during this stage. In all datasets, the aggregation strategy followed the proposed histogram-informed methodology OPA, first presented in Sec. 3.1.

Dataset	# Federated Rounds	# Clients per round	# Local Epochs	Local Optimizer	Batch Size	LR	Momentum	Gradient Clipping
CIFAR-10	500	10	50	SGD	128	0.1	0.9	N
CIFAR-100	500/750	10	25	SGD	64	0.1	0.9	N
SpeechCommands	750	21	25	SGD	64	0.1	0.9	N
Shakespeare	500	16	5	SGD	4	1.0	0.0	Y

Table 10: Hyperparameters used by `FedorAS` to search for the best model in the supernet and finetune them in a tier-aware fashion as described in Sec. 3.2 and Sec. 3.3 respectively. Searching iterations shown are allocated per-tier (i.e. CIFAR-10 has 4 tiers so a total of 4K valid models would be considered during the search). *Cosine* LR scheduling gradually reduces the initial LR (shown in the table) by an order of magnitude over the span of the the fine-tuning process. For Shakespeare, *step* LR decay worked best. This is applied at rounds 50 and 75, each decreasing the LR by a factor of 10×. SpeechCommands assigns different search iterations based on the ratio of clients assigned to each tier. `FedorAS` scales the amount of search for valid models within the tier accordingly. For Shakespeare, the sub-searchspaces that yield Tier-1 and Tier-5 models are smaller than for the other tiers, we therefore consider fewer search iterations.

Dataset	# Tiers	# Search Iterations	# Finetune Rounds	# Clients per round	Local Epochs	Batch Size	LR	LR Scheduling	Other Hyperparams
CIFAR-10	4	1000	100	6	1	32	0.01	cosine	momentum=0.9
CIFAR-100	4	1000	100	6	1	32	0.01	cosine	momentum=0.9
SpeechCommands	4	500/2000	100	21	1	32	0.01	cosine	momentum=0.9
Shakespeare	5	100/150	100	16	1	4	1.0	step	g.clipping=5

as illustrated in Fig. 8 (a) and (b). For SpeechCommands and Shakespeare we introduce a similar approach to give a wider range to the smallest tier. We refer to ρ_L and ρ_H as the PDF ratios to identify the boundaries splitting Tier1&2 and Tier3&4 respectively. The concrete values for each dataset as well as the FLOPs values for each tier boundary are shown in Tab 8.

Clients to tiers assignment. For CIFAR-10/100 clients are uniformly assigned to a cluster of devices or tier. For these datasets we consider four tiers, so each ends up containing 25% the clients resulting in 25 clients for CIFAR-10 and 125 clients for CIFAR-100. Similarly, for Shakespeare clients are also uniformly assigned to a tier, resulting in 143 clients per tier. For SpeechCommands, we designed a more challenging setup and divide the clients into tiers as follows: 80% of clients are assigned to be tier-1 devices, 17.5% are Tier-4 devices and the rest is evenly split into Tier-2 and Tier-3. This distribution better represents the the types of systems in the wild that perform keyword-spotting (Zhang et al., 2018), where the majority of the commercially deployed systems run these applications on low end CPUs or microcontrollers due to their *always-on* nature. Across datasets, the client-to-tier assignments was done irrespective of the amount of data these contained or distribution over the labels. An alternative client to tier allocation is provided in Sec. J.3.

E.5 Hyperparameters

Here we present the hyperparameters used across all datasets and tasks to generate the results presented in Sec. 4. In Tab. 9 we show the hyperparamters utilised for the first stage in the `FedorAS` framework: federated training of the supernet. Noticeably we require more local epochs in this stage as training each client effectively (pre-)trains multiple models and we need sufficiently many forward passes in order to sample a large enough number of paths on each client. In the context of `FedorAS`, the combination of batch size and number of local epochs define the exploration vs. exploitation trade-off in the communicated subspace. It is natural that if not enough paths are explored or if paths overfit the local data, the overall NAS results and thus model ranking would be severely affected. As such, we increase the number of local epochs together with a batch size, to ensure that the total number of forward passes remains constant for the same number of

Table 11: The architectures discovered by `FedorAS` can also be trained from scratch. This table contains the hyperparameters utilised to generate the *rand-init* results shown in Tab. 6 and 16. Training of these baselines is also performed in a tier-aware fashion. *Cosine* LR scheduling gradually reduces the initial LR (shown in the table) by an order of magnitude over the span of the the fine-tuning process. For Shakespeare, *step* LR decay worked best. This is applied at rounds 250 and 375, each decreasing the LR by a factor of 10×.

Dataset	# Federated Rounds	# Clients per round	Local Epochs	Batch Size	LR	LR Scheduling	Other Hyperparams
CIFAR-10	500	10	1	32	0.1	cosine	momentum=0.9
CIFAR-100	500	10	1	32	0.1	cosine	momentum=0.9
SpeechCommands	500	21	1	32	0.1	cosine	momentum=0.9
Shakespeare	500	16	1	4	1.0	step	g.clipping=5

Table 12: Cross-device federated NAS on CIFAR-10. We compare `FedorAS` and FedNAS while normalising aspects that are critical to on-device training: namely the number of FLOPs clients do in a given round and the memory peak seen over the course of such training. The later directly impacts on which devices a particular model could be trained on. We normalise memory peak by lowering the batch size that FedNAS uses (from 32 to 16) and reduce the number of local epochs in `FedorAS` down to 20 to match the FLOPs of typical FedNAS client. For further context, we maintain the results of `FedorAS` with 50 local epochs – the setting used throughout the majority of the experiments in this paper.

Dataset	**Method**	**Local Epochs**	**Batch**	**Mem. Peak (GB)**	**GFLOPs/client**	**Perf. (%)**
CIFAR10$_{\alpha=1}$	FedNAS	10	32	3837	1431	**90.02**
	FedNAS	10	16	**1919**	1431	85.45
	`FedorAS`	20	128	1996	**1402**	87.21±0.15
	`FedorAS`	50	128	1996	3504	86.46±0.32
CIFAR10$_{\alpha=0.1}$	FedNAS	10	32	3837	1431	65.28
	FedNAS	10	16	**1919**	1431	54.84
	`FedorAS`	20	128	1996	**1402**	79.41±0.31
	`FedorAS`	50	128	1996	3504	**81.53**±0.29

training examples on a client. After this stage, the best model for each tier is extracted from the supernet and then, they get finetuned in a per-tier aware manner (i.e. clients in tier T and above can finetune a model that belong to tier T). The hyperparameters of these two consecutive stages are shown in Tab. 10.

Regarding the *rand-init* results (i.e. models discovered by `FedorAS` but trained from scratch – discarding the weights given by the supernet) we present the hyperparameters in Tab. 11. These hyperparameters were the ones used to generate the results in Tab. 6 and 16 as well as Fig. 3. In Sec. 4.2, we present a CIFAR-100 result that largely outperforms existing federated baseline of (Reddi et al., 2021). This was achieved by a model discovered by `FedorAS` but trained in the *rand-init* setting following the setup as in (Reddi et al., 2021): 10 clients per round for 4k rounds using batch 20, starting learning rate of 0.1 decaying to 0.01 following a *cosine* scheduling.

E.6 Cross-device Federated NAS Evaluation

In Tab. 3 of Sec. 4.2 we compared `FedorAS` against FedNAS in the cross-sile setting with 100 clients and 10 clients randomly sampled on each round. Both methods follow a substantially different approach as far as NAS is concern and, as a result, FedNAS has a significantly higher memory peak than FedorAS for the same batch size. For example, for a batch size of 64 images, FedorAS sees a memory peak of 998MB whereas FedNAS requires 7674MB. Similarly, both methods translate in different compute footprints for each client. For example, FedNAS requires on average 716 GFLOPs per client (assuming each client), while clients in FedorAS need 280 GFLOPs for the same amount of local epochs and data in the client. Due to these differences, we aimed at normalising these aspects to make the comparison fair. These results are shown in Tab. 12, which extends the content of Tab. 3.

E.7 Baselines

In this section we faithfully describe what each baseline represents in the experiments of the main paper and the appendix to clarify with what we are comparing in each section.

Tier-unaware (Fig.3): This baseline represents model architectures that have been trained end-to-end in a federated manner without any awareness of client eligibility. We are using FedAvg with hyperparameters similar to those presented in Tab. 11.

Tier-aware (Fig.3): This baseline adds client eligibility awareness to the previous baseline. This means that models of certain footprint can be trained only on clients of the eligible cluster and above.

FjORD (Horvath et al., 2021) (Fig.3, Tab. 5): FjORD is a baseline that is tackling system heterogeneity by means of Ordered Dropout. It assumes a uniform dropout rate across layers, essentially keeping the control variable one-dimensional and thus offering fewer degrees of architectural freedom. Nevertheless, it enables the dynamic extraction of candidate submodels without the need to retrain or finetune. This particular variant is assuming the experimental setup of the original paper which shards the CIFAR-10 dataset per client without LDA. The Shakespeare setup remains the same as ours.

FjORD-LDA (Horvath et al., 2021) (Fig.3, Tab. 5): For this baseline, we implemented FjORD and ran it on the same number of clusters as `FedorAS`, with LDA for CIFAR-10.

SPIDER (Mushtaq et al., 2021) (Tab. 4): SPIDER is another paper performing personalised Federated NAS in the cross-silo setting. As there is no publicly available codebasse, we assume their setting with `FedorAS` and compare on CIFAR-10.

ZeroFL (Qiu et al., 2022) (Tab. 5): For this baseline, we borrow the results of respective paper for CIFAR-10$_{\alpha=\{1,1000\}}$, which assumes the same setup as ours. We present the results for sparsity level of 90% and annotate its footprint as the original model FLOPS and the number of non-zero parameters.

Oort (Lai et al., 2021) (Tab. 5): The Oort framework proposes a participation sampling criterion by which clients are sampled based on their utility (i.e. how much their data can contribute to the global model) while also taking the device capabilities into consideration. Over the course of FL training, the sampling of clients with high data and system utility is prioritised to reduce the time needed to convergence. For SpeechCommands (see Tab. 5), Oort made use of a ResNet-34 model.

PyramidFL (Li et al., 2022)(Tab. 5): At a high-level, PyramidFL proposes a framework similar to that in Oort. The core difference between these two methods is that PyramidFL leverages more fine-grained statistics when assessing the contribution potential (i.e. utility) of the selected clients. It also uses ResNet-34 for SpeechCommands.

rand-init (Tab. 6, 16, 21): This baseline refers to the concept of running the search, coming up with an architecture and subsequently re-initialising the weights and running a conventional federated training setup. It shows vanilla scaling that would be obtained if we simply trained identified models in a standard way, using random initialization and full FL training using standard practices. Model architectures are kept the same between `FedorAS` and "*rand-init*" experiments. Models belonging to higher tiers, as a whole, are never trained on data belonging to devices from lower tiers.

F Comparison with Alternative NAS Algorithms

Note: this section considers replacing `FedorAS` as a whole with a different approach to searching for the best performing model in the same search space etc. On the other hand, for additional details about the searching stage of `FedorAS` (Stage 2) please see Section K.

F.1 Comparison against Random Search

In the main paper we compare our `FedorAS` to the state-of-the-art methods from the literature (Sec. 4.2) and also investigate the impact of initializing models with weights from a supernet (Sec. 4.3). Here we present

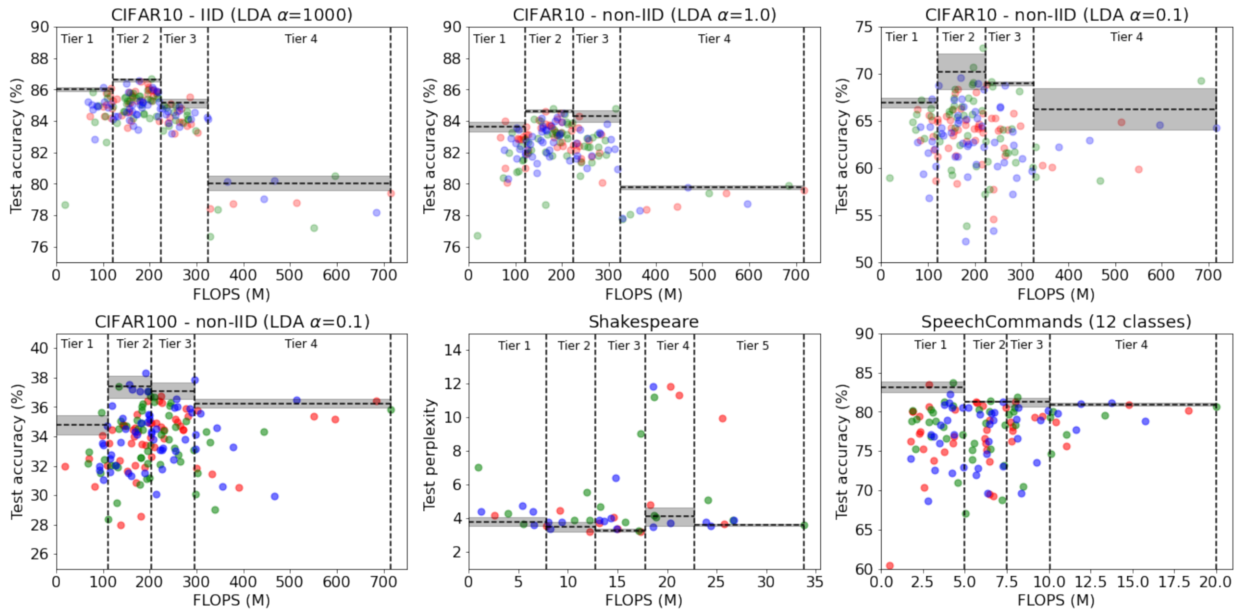


Figure 9: For each dataset, we randomly sample their search space and train these models end-to-end. We repeat this process (with new samples) three times, and overlap the scatter plots of different runs (i.e. red, green, blue). Then, for each tier we average the best performing models across each of the three runs and plot the result as a dashed horizontal line with grey area representing ± standard deviation. The compute budget allocated to generate the data for each plot is equivalent to the cost of running `FedorAS`, which also yields one model per tier.

Table 13: Comparison of performance of models found with `FedorAS` and with a simple random search.

Dataset	`FedorAS`					Random search				
	T1	T2	T3	T4	T5	T1	T2	T3	T4	T5
CIFAR-10 ($\alpha = 1000$)	**89.40**±**0.19**	**89.60**±**0.15**	**89.64**±**0.22**	**89.24**±**0.29**	-	86.03±0.13	86.39±0.28	85.18±0.26	79.84±0.75	-
CIFAR-10 ($\alpha = 1.0$)	**85.99**±**0.13**	**86.30**±**0.41**	**86.34**±**0.19**	**85.58**±**0.55**	-	83.68±0.25	84.50±0.31	84.30±0.38	79.65±0.18	-
CIFAR-10 ($\alpha = 0.1$)	**81.01**±**0.46**	**81.53**±**0.29**	**80.64**±**0.66**	**80.85**±**0.28**	-	66.75±0.74	70.85±1.53	69.00±0.18	66.08±2.29	-
CIFAR-100	**45.25**±**0.13**	**45.84**±**0.18**	**45.42**±**0.39**	**45.07**±**0.71**	-	34.41±0.90	37.64±0.46	36.98±0.64	35.45±0.90	-
SpeechCommands	80.19±1.78	80.47±1.69	81.00±1.58	80.56±0.40	-	**82.84**±**1.08**	**81.17**±**0.26**	**81.10**±**0.74**	**80.61**±**0.61**	-
Shakespeare	**3.43**±0.01	**3.39**±0.04	**3.38**±0.03	**3.40**±0.01	**3.42**±0.01	3.86±0.33	3.54±0.28	3.45±0.32	3.80±0.30	3.60±0.39

one more important baseline for any NAS algorithm - comparison to a random search. Specifically, while we have already shown that models found by `FedorAS` in most cases benefit from supernet training, this does constitute a conclusive argument justifying the necessity of the entire process in a broader context. Perhaps there exist models that do not need to be initialized from a supernet but, since the focus of our work is on supernet training, we missed them due to biased conditioning?

In order to check this hypothesis we run a simple random search algorithm, to get an estimate of the expected *best case* performance of models from our search spaces when trained following standard FL procedure. This simple baseline was allowed to train random models until the total training cost exceeded the cost of running `FedorAS` – this turned out to be equivalent to roughly 40 fully-trained models. After we run out of the training budget, we simply get the best model for each tier as our solution to NAS. We repeated the entire process 3 times for each search space and report average and standard deviation compared to the average performance of models found by `FedorAS` in Tab. 13. Additionally, we also plot detailed performance of each model trained during this process in Fig. 9 for the sake of completeness.

Noticeably, `FedorAS` performs significantly better at an average of +5.11pp, +3.24pp, +12.84pp, +9.27pp, +0.25p across tiers for CIFAR-10$_{\alpha=\{1000,1,0.1\}}$, CIFAR-100 and Shakespeare, respectively. Only in the case of SpeechCommands did our search result in -0.87pp of accuracy on average. We suspect this is due to problem with fine-tuning rather than architectures themselves – specifically, we witness accuracy of the discovered

Table 14: Comparison of FedorAS vs centralised NAS setting.

Method	**Dataset**	**Tier 1**	**Tier 2**	**Tier 3**	**Tier 4**
Centralised NAS	CIFAR-10$_{\text{centralised}}$	93.77$\pm$0.20	94.14$\pm$0.34	94.04$\pm$0.22	94.33$\pm$0.09
FedorAS	CIFAR-10$_{\alpha=1000}$	89.40$\pm$0.19 (**-4.37**)	89.60$\pm$0.15 (**-4.54**)	89.64$\pm$0.22 (**-4.40**)	89.24$\pm$0.29 (**-5.09**)

Table 15: The effect of Stage-III of FedorAS: tier-aware federated fine-tuning of models extracted from the supernet. This table reports the increase in validation accuracy (measured in percentage points) for classification tasks of the model after fine-tuning and the decrease in perplexity for next word prediction (i.e. Shakespeare).

Dataset	**Tier 1**	**Tier 2**	**Tier 3**	**Tier 4**	**Tier 5**
CIFAR-10$_{\alpha=1000}$	1.23$\pm$0.11	0.80$\pm$0.16	1.18$\pm$0.02	1.11$\pm$0.19	N/A
CIFAR-10$_{\alpha=1.0}$	3.99$\pm$0.22	3.86$\pm$1.36	3.55$\pm$0.68	2.63$\pm$0.49	N/A
CIFAR-10$_{\alpha=0.1}$	8.27$\pm$1.34	7.44$\pm$1.12	7.78$\pm$1.39	7.76$\pm$2.57	N/A
CIFAR-100$_{\alpha=0.1}$	8.56$\pm$0.04	8.69$\pm$0.33	7.20$\pm$0.34	7.74$\pm$1.04	N/A
Speech Commands	4.66$\pm$0.06	5.68$\pm$0.55	7.98$\pm$1.11	6.71$\pm$1.32	N/A
Shakespeare	0.273$\pm$0.025	0.220$\pm$0.028	0.227$\pm$0.046	0.210$\pm$0.020	0.190$\pm$0.020

models can vary significantly as we repeat the fine-tuning process (this is also visible in the case of full training, although the extend is smaller). Consequently, although on average FedorAS performs slightly worse, in many cases the best results surpass that of random search – because of that we suspect that fine-tuning for longer would improve FedorAS's performance. We leave this for future work, considering that a single shortcoming like that does not seem significant in the light of the rest of our results.

F.2 Comparison against Centralised NAS

To study the impact that FL has in supernet-based NAS, we train the CIFAR-10 supernet in a similar centralised setting to FedorAS's. Similarly to Stage I in FedorAS, we do 500 training epochs following the SPOS paradigm but, because it is centralised, we do not impose any FLOPs budgets when sampling paths along the supernet. After training, we use NSGA-II to search for the best performing architecture/path and select the best among 1000 valid models for each tier. Finally, these models are trained from scratch for 200 epochs (again in a centralised setting, i.e., having access to all training data). The resulting test accuracies per tier are shown in the Tab. 14 and compared against the federated IID setting of CIFAR-10 (the simplest FL setup).

G Convergence and FedorAS per-stage Analysis

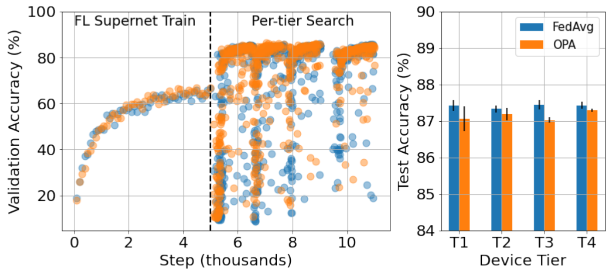
In this section, we complement the results presented in Sec. 4.6, by measuring the impact of stage III of FedorAS as well as the repercussions of supernet sampling when allocating tasks downstream.

G.1 Impact of Stage III: Federated Fine-tuning

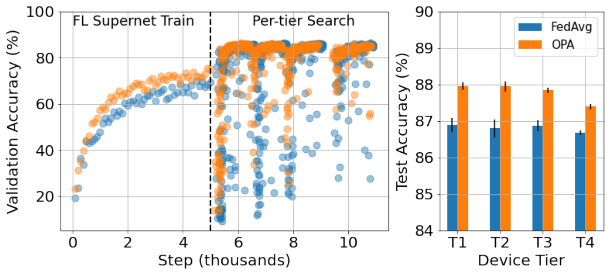
After Stage I in FedorAS, models extracted from the supernet can directly be used for the target task (e.g. image classification). However, performance on this target task can be improved further by fine-tuning these models. This is the purpose of Stage III. In Table 15 we show the increase in model quality (measured on the global validation set) before and after Stage III.

G.2 Convergence and Impact of Supernet Sampling

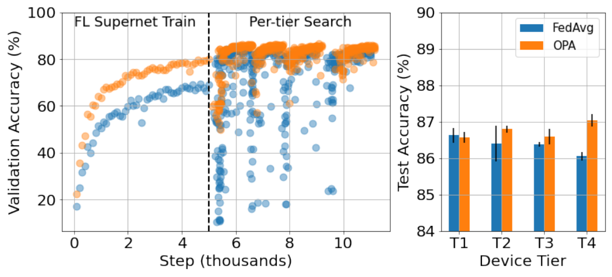
The proposed OPA aggregation scheme converges faster than an alternative aggregation method that does FedAvg of the updated supernets. We show this in Fig. 10. When supernets from each client participating in the round get aggregated with OPA, the resulting supernet is consistently better in terms of validation accuracy than when aggregation is done with FedAvg. This difference becomes more evident as $B_{\Phi_{\text{comm}}}$ is



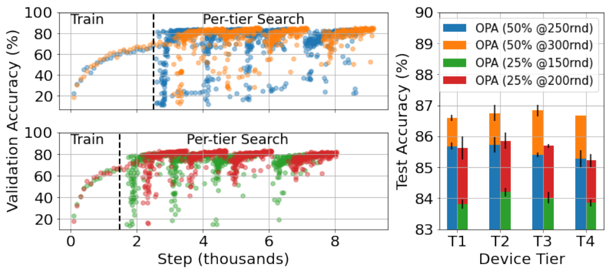
(a) Sending whole of the supernet ($B_{\Phi_{\text{comm}}} = 22.3M$)



(b) Sending 50% of the supernet ($B_{\Phi_{\text{comm}}} = 11.15M$)



(c) Sending 25% of the supernet ($B_{\Phi_{\text{comm}}} = 5.58M$)



(d) Impact of fewer rounds

Figure 10: Results using OPA compared to FedAvg in **`FedorAS`** for CIFAR-10 non-IID ($\alpha = 1.0$) with fixed hyperparameters. Each sub-plot contains two plots: first a scatter plot that visualises the federated supernet training (Sec. 3.1) in the first 5k steps and the per-tier search stage (Sec. 3.2) the remaining steps; and, a bar plot that shows the average test accuracies of each tier after federated fine-tuning (Sec. 3.3). For (a)-(c), the training setups are identical with the exception of using either OPA or FedAvg. Both settings perform 500 rounds of federated supernet training using 10 clients per round (i.e. 5k steps). Every 10 rounds, the supernet is evaluated on the global validation set by randomly sampling paths and a dot is added to the plot. In the context of reduced communication budget, OPA largely outperforms FedAvg, requiring fewer federated rounds to reach the same validation accuracy. This difference is more noticeable when sending 50% of the supernet (roughly corresponding to the size of a ResNet18 model). In (d) we measure the quality of the models found when the supernet training phase ends once 70% validation accuracy is reached – the highest reached by FedAvg in (a)-(c) – and show competitive performance of models derived from OPA-aggregated supernets. By extending the training phase by 50 rounds, we observe large improvements in the quality of the final models.

reduced, i.e., as smaller portions of the supernet are sent to the clients. In Fig. 10 (d) we asses the feasibility of reducing the number of rounds and end the federated supernet training stage when validation accuracy reaches approximately 70%. Those points correspond to 250 rounds and 150 rounds for the setting when 50% and 25% of the supernet is sent to the clients, respectively. The results that these settings yield show a significant loss (compared to their respective settings but over 500 rounds) and we therefore also run the same settings but when allowing for 50 additional runs. We observe a significant jump in per-tier model performance. We leave as future work investigating alternative metrics to more accurately (but without incurring into heavy computational costs) measure the quality of the supernet at any given training iteration and leverage this to better inform an early stopping mechanism to further reduce communication costs.

H Impact of Weight-sharing on Different Task Training

In this section, we expand on the analysis of Sec. 4.3. Specifically for CIFAR-100 and SpeechCommands, we also create a second scenario, where not all clients train in the same domain of labels. For CIFAR-100, there are 20 superclasses that are coarse-grained categories of the 100 standard labels. In contrast, for SpeechCommands, there are 12 and 35-class label sets, with the latter annotating the "other" class more specifically. In both cases, we assume a non-uniform distribution of clients to clusters, assuming that most data (75-80%) reside on lower-tier devices, whereas the two higher tier-devices (holding 20-25% of data) can

Table 16: Models discovered by `FedorAS` benefit from weight sharing across tiers compared to models using the same architecture but trained end-to-end on clients that support them. Models derived from a `FedorAS` supernet outperform their baselines by large margins in most cases. First part: accuracy, second part: perplexity.

Dataset	**Clients**	**Setting**	**Partitioning**	**Mode**	**Classes**	**Tier 1**	**Tier 2**	**Tier 3**	**Tier 4**
CIFAR-100	500	Multi-task	non-IID$_{\alpha=0.1}$	`FedorAS`	$[20, 20, 100, 100]$	**57.93**±0.31	**57.86**±0.7	**38.63**±0.74	**37.68**±0.73
				rand-init		41.64±0.26	42.45±0.58	27.11±0.64	21.28±1.26
Speech Commands	2112	Transfer	*given*	`FedorAS`	$12 \rightarrow 35$	**67.06**±2.12†	**66.87**±1.85†	**67.65**±1.59	**68.49**±1.47
				rand-init		64.99±1.41†	65.03±0.69†	66.55±2.2	66.84±0.87

† Trained only on clients belonging to Tier 3 and 4.

Table 17: Communication costs of supernet pretraining for different methods. FL rounds refers to the number of FL rounds dedicated to supernet training. Supernet size is the size of the model that is communicated to the clients.

Experiment	Method	FL Rounds	Supernetsize (M)	Num clients	Comm. Costs (GB)
Table 2 – CIFAR-10	FedNAS	500	1.93	10	75.39
Table 4 – CIFAR-10	FjORD†	500	5.23	10	204.30
Table 2,4 – CIFAR-10	FedorAS	500	11.5*	10	449.22
Table 4 – Shakespeare	FjORD†	500	0.10	10	3.83
Table 4 – Shakespeare	FedorAS	500	0.83*	16	51.88
Table 4 – SpeechCommands	FedorAS	750	1.5*	21	184.57

‡ Supernet size is size of avg. model communicated to clients.
* We are actually only communicating 1/2 of the supernet to participating clients.

train on the fine-grained label set. Our aim is to test whether few data allocation on the high tier devices can benefit from the knowledge and feature extraction learned from the coarse-grained label set. We adopt two different setups. For CIFAR-100, we train both tasks simultaneously, having essentially two distinct linear layers across tiers (1,2) and (3,4). We call this setup multi-task. On the other hand, for SpeechCommands, we train all clients on the same coarse domain and subsequently transfer learn a fine-grained 35 classes linear layer for the client of tiers (3,4). We fine-tune the linear layer keeping the rest of the network frozen for 25 epochs and then allow for fine-tuning of the whole network for another 100. We present the results in Tab. 16.

In both cases `FedorAS` learns better models and transfer knowledge from the low-tiers to high-tiers and vice-versa through weight sharing of the supernet. Indicatively, for CIFAR-100 we are able to achieve +14.91 pp (percentage points) of accuracy compared to training the same architectures end-to-end on eligible devices. Similarly, we achieve +1.6 pp higher accuracy for SpeechCommands.

I Communication Cost of FedorAS and Baselines

Here we quantify the communication cost of federated training with `FedorAS` and other selected baselines, depicted in Tab. 17 and Tab. 18. The former compares the total communications costs of supernet-based federated methods, namely FjORD and FedNAS. While our method evidently assumes a larger search space in terms of parameters – leading to higher communication costs – `FedorAS` is able to keep the peak memory of searching at a significantly lower level and achieve higher accuracies to the selected baselines (see Tab. 3 in Sec. 4.2) and 12. Given that most cross-device FL is happening when devices are plugged in and connected to unmetered connections (Bonawitz et al., 2019), we believe it does not affect the deployability of our solution. The costs of supernet training in `FedorAS` can be ammortised as more architectures are finetuned for different device tiers or tasks. The latter table compares communications costs but focuses on FL training (or fine-tuning) of individual standard models and for a single tier (if device tiers are applicable). Here we notice that the communication cost of `FedorAS`' fine-tuning is relative small and on par with the most-efficient baselines.

Table 18: Communication costs of federated fine-tuning (federated training) a model. (note: methods that are not tier-aware are included here for a single tier).

Experiment	Method	Standard FL Rounds	Avg. Model Size (M)	Num. clients	Comm. Costs (GB)	Notes
Table 2 – CIFAR-10	FedNAS	?	?	?	?	Unclear how much fine-tuning they do after NAS
Table 2– CIFAR-10	FedorAS	100	2.6	6	12.19	Average per tier
Table 4– CIFAR-10	ZeroFL	500	11.7	10	436.33	
Table 4– CIFAR-10	FjORD	0	0	0	0	FjORD does not consider fine-tuning
Table 4– CIFAR-10	FedorAS	100	3.34*	6	15.64	Average per tier
Table 4 – Shakespeare	FjORD	0	0	0	0	FjORD does not consider fine-tuning
Table 4 – Shakespeare	FedorAS	100	0.20	16	2.46	Average per tier
Table 4 – SpeechCommands	Oort	400	21.29	1300 / 100	46571.88	Oort asks 1300 clients to do training but only the first 100 to finish are aggregated (we therefore only include 100 models for uplink communication)
Table 4 – SpeechCommands	PyramidFL	400	21.29	50	3326.56	
Table 4 – SpeechCommands	FedorAS	100	0.42	21	6.96	Average per tier

* Average model size is calculated as the average parameter size across tiers in fine-tuning stage.

J Behaviour under Alternative Tier Clustering

J.1 Supernet with Wider FLOPs Range

We design a larger supernet for CIFAR-10 by stacking more blocks (4 blocks with 4 searchable layers followed by two blocks of 8 searchable layers) in the supernet while not requiring the reduction block between supernet blocks to always downsample the input (see E.3). This is done to prevent the width and height of activations to decrease too rapidly (i.e. after each block). The MFLOPs limits for each tier are: [101.06, 714.01, 1326.95, 4450.72], with the smallest model in Tier 1 (i.e. 53.77 MFLOPs) representing the fixed costs of the supernet (FLOPs involving non-searchable layers). With this setup, Tier 4 models can have a 82× larger compute footprint than Tier 1 models. Results are depicted in Table 19. These were obtained using the exact same hyperparemeter configuration (see Table 9 and Table 10 for hyperparameters of FedorAS Stage I and Stage-II & III respectively) as the CIFAR-10 experiments shown in Table 6, with the exception of now having a larger supernet. Still, the candidate operators in the supernet remain the same and so is the proportion of clients assigned to each tier. For these experiments we followed the same procedure when establishing the communication budget and therefore we set it to be half of the supernet size. We report the results of the fine-tuning stage of FedorAS and show improvement over those obtained with the smaller (original) supernet. We also report the resulting performance when networks are randomly initialised (i.e. *rand-init* setup).

The results in Table 19 tell us that even in the scenario where the span of model footprints is larger (82× in this experiment): small models (Tier 1 – up to 101 MFLOPs) trained with FedorAS can very well take advantage of larger models trained on other more capable clients (Tier 2 and above); and, better models are obtained across Tiers showing large gains compared to the original supernet (considering models up to 700MFLOPs approximately).

J.2 Scalability to More Clusters

So far, we have shown results on a set number of clusters. In this section, we scale up the number of clusters from 4 to 8 and show the behaviour of our system in the case of CIFAR-10$_{\alpha=\{1.0,0.1\}}$. Results are depicted in Tab. 20.

Table 19: Performance of `FedorAS` when increasing the FLOPs range of models in the supernet. Models in Tier 4 are up to 82× larger than dose in Tier 1.

Dataset	Tier 1	Tier 2	Tier 3	Tier 4
CIFAR-10$_{\alpha=1.0}$	89.54$\pm$0.12 (+3.55pp)	89.89$\pm$0.10 (+3.59pp)	89.99$\pm$0.19 (+3.65pp)	89.34$\pm$0.41 (+2.88pp)
CIFAR-10$_{\alpha=0.1}$	84.42$\pm$0.41 (+3.41pp)	84.94$\pm$0.42 (+3.41pp)	84.11$\pm$1.37 (+3.47pp)	83.63$\pm$1.13 (+2.78pp)

Table 20: Performance of `FedorAS` when scaled to 8 tiers.

Dataset	Tier 1	Tier 2	Tier 3	Tier 4	Tier 5	Tier 6	Tier 7	Tier 8
CIFAR-10$_{\alpha=1.0}$	87.53$\pm$0.29	87.81$\pm$0.21	87.59$\pm$0.57	87.84$\pm$0.26	87.65$\pm$0.20	87.76$\pm$0.40	87.21$\pm$0.55	87.11$\pm$0.45
CIFAR-10$_{\alpha=0.1}$	80.34$\pm$1.70	81.56$\pm$0.46	81.50$\pm$1.19	81.54$\pm$0.69	81.12$\pm$0.81	81.14$\pm$0.29	81.28$\pm$1.13	79.92$\pm$1.94

Table 21: `FedorAS` performance under (Lai et al., 2021) device clustering. We see that performance still scales well, albeit taking an impact due to more extreme system heterogeneity compared to results from Tab. 6. This hints that weight-sharing through our supernet works well under varying device allocation settings.

Dataset	#clients	Method	Perf.
CIFAR10$_{\alpha=1}$	100 (10)	`FedorAS`$_{\text{per tier}}$	[**87.77$\pm$0.09**, **87.43$\pm$0.28**, **87.15$\pm$0.42**, **87.01$\pm$0.11**]
	100 (10)	*rand-init*	[85.83$\pm$1.24, 83.91$\pm$0.98, 83.21$\pm$0.60, 79.10$\pm$1.63]
CIFAR100$_{\alpha=0.1}$	500 (10)	`FedorAS`$_{\text{per tier}}$	[**44.33$\pm$0.81**, **43.83$\pm$0.89**, **43.72$\pm$0.82**, **43.22$\pm$1.02**]
	500 (10)	*rand-init*	[34.88$\pm$0.28, 33.76$\pm$2.53, 32.72$\pm$0.68, 30.77$\pm$2.30]

J.3 Different Allocation of Clients to Tiers

In this experiment, contrary to what was described in Appendix E.4, we adopt the device capabilities trace from Oort (Lai et al., 2021) for our device to cluster allocation and performed training on CIFAR-10 and CIFAR-100. Results are depicted on Tab. 21. It can be witnessed that `FedorAS` is able to operate even under harsher heterogeneity conditions and still output competitive models in the federated setting.

K Evaluation of the Search Phase

K.1 Sensitivity to Size of Validation Set

In order to more meaningfully examine the impact of the validation set size, we run experiments on all datasets with 20% and 50% of the size of the initial global validation set. This means that during Stage-II of `FedorAS`, architectures sampled from the supernet are scored using a fraction of the global validation set. These validation subsets are extracted uniformly. After obtaining the best performing models for each tier, these are fine-tuned in a federated fashion (Stage-III in `FedorAS`) for 100 rounds just like it was done for Tab. 6. We maintain the same hyperparameters as those used to generate Tab. 6. Results are depicted in Tab 22 for all datasets.

K.2 Federated Search

In this Section, we provide additional details and results concerning our federated variant of NSGA-II, discussed in Sec. 4.7 of the main paper. Below, we provide some context about how the algorithm works, how we setup the federated experiments and commentary of the results on CIFAR-10$_{\{1000,1,0.1\}}$ and CIFAR-100.

Algorithm details. Federation is achieved by delegating evaluation of models to individual clients. This means that evaluation is done stochastically on clients' local datasets. This setting is similar to Federated Evaluation in (Paulik et al., 2021). In order to avoid sending a large number of models to clients, thus saving communication cost, we leverage the fact that NSGA-II operates in "batches" of models – in each iteration, a number of models (i.e. *sample size*) from the *population* is selected to produce new models that replace the ones not selected. Since models from a single "batch" are all selected at the same time, we do

Table 22: Test accuracy for different sample size of validation set. Results are shown as relative change in final test accuracy for each tier compared to the scenario where the whole validation set is used. Results for each dataset are averaged over three runs.

Dataset	Clients	Partitioning	Val set prct.	Classes	Tier 1	Tier 2	Tier 3	Tier 4
CIFAR-10	100	IID$_{\alpha=1000}$	1.0	10	89.40±0.19	89.60±0.15	89.64±0.22	89.24±0.29
			0.5		-0.31	-0.10	-0.14	**+0.03**
			0.2		-0.06	-0.17	-0.19	-0.02
		non-IID$_{\alpha=1.0}$	1.0	10	85.99±0.13	86.30±0.41	86.34±0.19	85.58±0.55
			0.5		**+0.06**	-0.18	**+0.19**	-0.19
			0.2		-0.05	+0.02	-0.28	-0.29
		non-IID$_{\alpha=0.1}$	1.0	10	81.01±0.46	81.53±0.29	80.64±0.66	80.85±0.28
			0.5		**+0.63**	-0.5	-0.29	-0.14
			0.2		**+0.78**	**+0.13**	-1.55	-0.09
CIFAR-100	500	non-IID$_{\alpha=0.1}$	1.0	100	45.25±0.13	45.84±0.18	45.42±0.39	45.07±0.71
			0.5		-0.10	**+0.01**	-0.27	-0.38
			0.2		**+0.07**	**+0.11**	**+0.32**	-0.60
Speech Commands	2112	*given*	1.0	12	80.19±1.78	80.47±1.69	81.0±1.58	80.56±0.40
			0.5		**+0.40**	**+0.64**	**+0.25**	**+1.76**
			0.2		-0.98	-0.40	-1.49	**+1.08**

Dataset	Clients	Partitioning	Val set prct.	Classes	Tier 1	Tier 2	Tier 3	Tier 4	Tier 5
Shakespeare	715	*given*	1.0	90	3.43±0.01	3.39±0.04	3.38±0.03	3.40±0.01	3.42±0.01
			0.5		-0.004	-0.000	-0.014	-0.015	-0.004
			0.2		-0.002	-0.07	-0.003	**+0.003**	-0.008

not need to evaluate them sequentially. Instead, considering that weights between architectures are shared, we can construct a minimal supernet that encapsulates all selected models and send it to relevant clients. This way, we can achieve parallel evaluation of all selected models and optimal[7] communication cost for a given "batch".

Experimental setting. The experimental setting is following exactly what was used in other experiments. Specifically, population and sample sizes were the same as the ones used in centralised NSGA-II (128 and 64, respectively), and federated evaluation (FE) rounds were analogous to rounds during federated training of the supernet from which models to evaluate were extracted. Concretely, number of clients, allocation of clients to tiers, allocation of data to clients, number of available clients per round, and client selection mechanism were all exactly the same for FE as the ones used during the relevant Stage 1 of `FedorAS`.

However, to present more meaningful results, we assumed that as we fluctuate the number of clients used to evaluate models (number of evaluation rounds) additional clients are always unique. In other words, reshuffling and "forgetting" of clients is happening only between federated NSGA-II iterations. Consequently, it is possible that: *i)* if all clients are used to evaluate a model (e.g. 10 evaluation rounds for CIFAR-10), FE is equivalent to centralised evaluation; *ii)* with a relatively small number of FE rounds, it is possible that one set of models produced by NSGA-II is evaluated on a completely disjoint set of data to another "batch"; *iii)* similarly, we do not enforce in any way that all clients have to be used (unless we use all clients in a single iteration of NSGA-II). Finally, keeping consistent with the rest of the paper, we performed evaluation in a tier-aware manner, meaning that models belonging to higher tiers could only access a subset of all validation data based on client eligibility. This is the reason why relevant curves finish at a different maximum number of FE rounds, that is a smaller number of FE rounds (i.e. Federated NSGA-II iterations) is needed to exhaust the set of eligible clients.

Results. Fig. 11 complements Fig. 6 and presents results across three different experimental settings, with varying level of non-IID-ness ($\alpha \in \{1, 0.1\}$) and number of clients ($\{100, 500\}$). As conjectured in the main paper, we can observe that the cost of achieving the same fidelity of FE increases with both the level of non-IID-ness and the number of clients. Specifically, Kendall-τ of 0.8 is achieved at approximately three FE rounds for $\{\alpha = 1, \text{clients} = 100\}$, and increases to four and six for $\{\alpha = 0.1, \text{clients} = 100\}$ and

[7] Optimal in the sense of communicating paths once and not accounting for orthogonal techniques such as compression, etc.

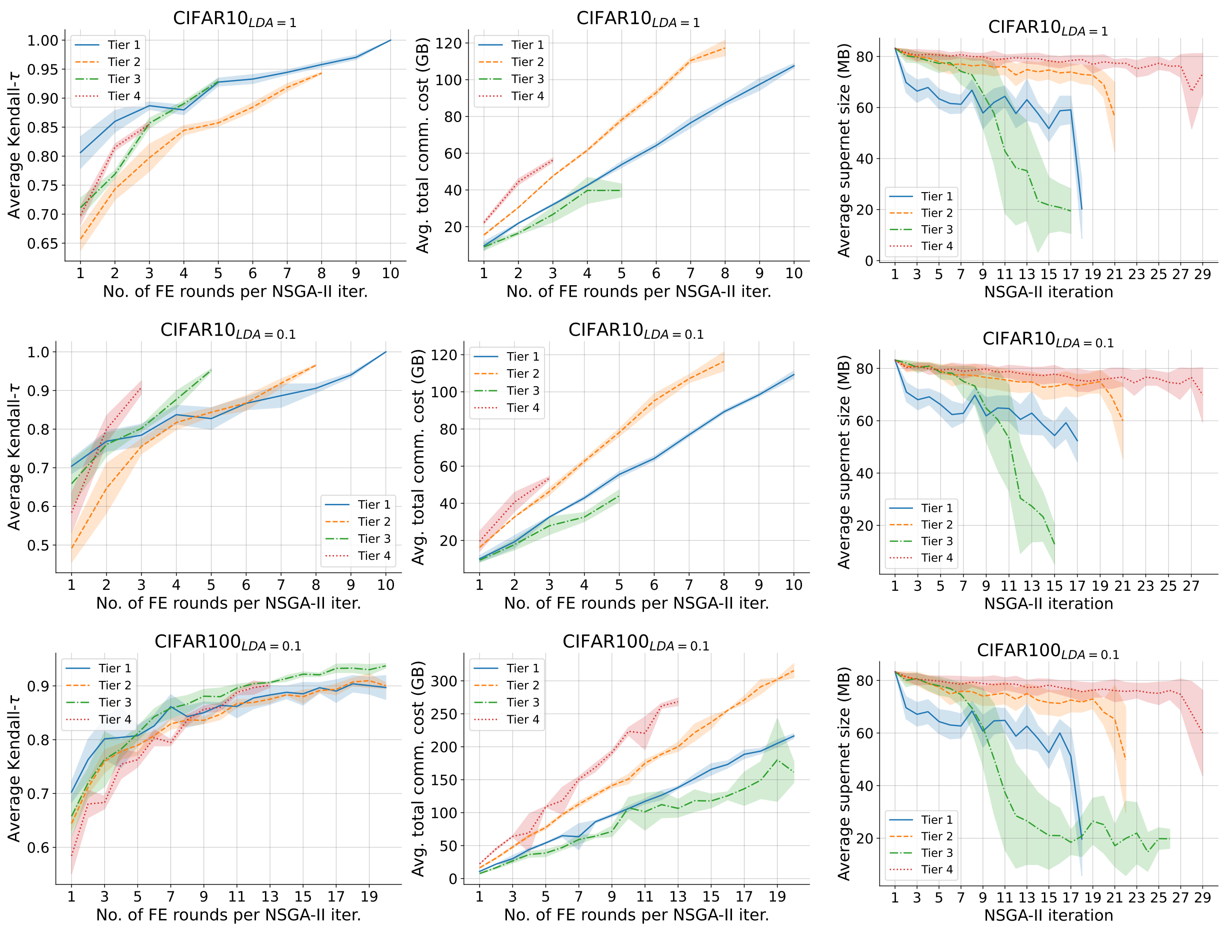


Figure 11: Ranking quality & cost of federated evaluation (FE) of models during federated search. Each time a new population of models is evaluated, a minimal supernet encompassing selected models is sent to a sample of clients; **left)** ranking correlation between scores produced by FE & centralised evaluation (CE), as a function of FE rounds ($\uparrow$ rounds = $\uparrow$ clients); **middle)** total communication cost of sending all necessary supernets to all clients, to run a full search; **right)** changes in the supernet size as NSGA-II progresses.

$\{\alpha = 0.1, \text{clients} = 500\}$, respectively. At the same time, we can see that regardless of the setting NSGA-II tends to produce smaller supernets as the search progresses, suggesting that searching for longer does not have to incur proportional increase in the communication cost.

K.3 Correlation with Post Fine-tuning Accuracy

The main objective of the NSGA-II search in Stage 2 is to find the most accurate models, where accuracy is understood as validation accuracy immediately after extracting a relevant subnet from a pretrained supernet. The assumption is that the better a model performs in a case like that, the better it will be after the following fine tuning performed in Stage 3. However, weight-sharing NAS is well-known to fail to meet this assumption in many cases Yu et al. (2020); Zela et al. (2020); Zhang et al. (2020); White et al. (2021); Ning et al. (2021). Therefore, we additionally quantify the fidelity of out searching objective by measuring ranking correlation between validation accuracy of 160 random models from a search space, when taken directly from a supernet after Stage 2, to their final test accuracy after fine tuning in Stage 3. Results are presneted in Figure 12.

We can see that in general ranking models based on their accuracy when directly using weights from a supernet is not very faithful to how good a model can be with extra fine-tuning, which is aligned with the aforementioned observations made in the centralised setting. While this shows that further improvements

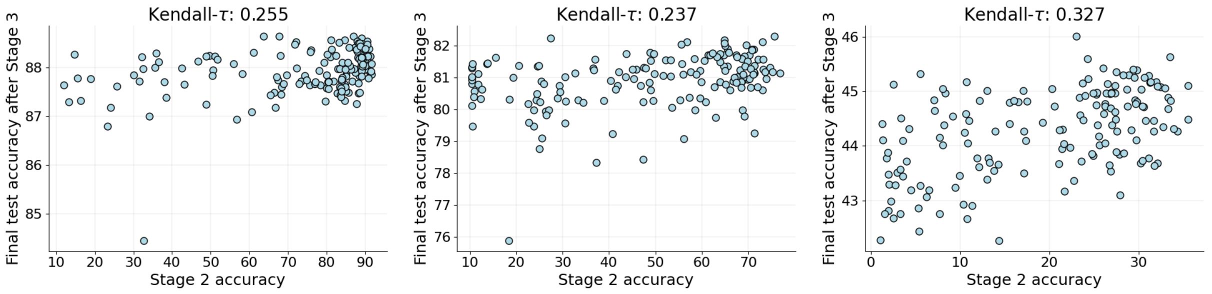


Figure 12: Correlation between validation accuracy as measured during Stage 2 and the final test accuracy of a model after fine-tuning in Stage 3, for 160 models randomly selected from a relevant search space. **Left)** CIFAR-$10_{\alpha=1.0}$, **middle)** CIFAR-$10_{\alpha=0.1}$, **right)** CIFAR-$100_{\alpha=0.1}$. Note: unlike the rest of the paper, CIFAR-10 supernets were trained for 750 epochs here, not 500.

could be achieved with more work on the searching algorithm, we would argue that to obtain more indicative performance of models, one would need to perform additional training, which in our settings involves following the entire FL procedure, making the cost of doing so even more non-negligible. On the other hand, while the best models identified after Stage 2 are unlikely to be the best after fine-tuning, they tend to lean towards better performing ones. Therefore, based on our strong empirical results and observed rankings in Figure 12, we would conclude that they constitute a safe choice which, at the same time, involves minimal searching cost. Due to those reasons, we leave the challenge of improving the fidelity of validation scores for future work.