Cross-device Federated Architecture Search

Abstract

Federated learning (FL) has recently 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. However, such centralised datasets may not be always available. 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 environment 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 better performance compared to state-of-the-art federated solutions, while maintaining resource efficiency.

1 Introduction

As smart devices become omnipresent where we live, work and socialise, the ML-powered services that these provide grow in sophistication. This ambient intelligence has undoubtedly been sustained by recent advances in Deep Learning (DL) across a multitude of tasks and modalities. Parallel to this race for state-of-the-art performance in various in DL benchmarks, mobile and embedded devices also became more capable to accommodate new DNN designs [31], some even integrating accelerators to their SoCs (e.g. NPUs) to efficiently run DL workloads [2]. These often come in various configurations in terms of their compute/memory capabilities and power envelopes [3] and co-exist in the wild as a rich multi-generational ecosystem (*system heterogeneity*) [63]. These devices bring intelligence through users' interactions, also innately heterogeneous amongst them, leading to non-independent or identically distributed (non-IID) data in the wild (*data heterogeneity*).

Powered by the recent advances in SoCs' capabilities and motivated by privacy concerns [59] over the custody of data, Federated Learning (FL) [48] has emerged as a way of training on-device on user data without it 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 [34]. Not only may this not be the case, but it also can 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 [44].

Parallel to these trends, Neural architecture search (NAS) has become the *de facto* mechanism to automate the design on 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

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

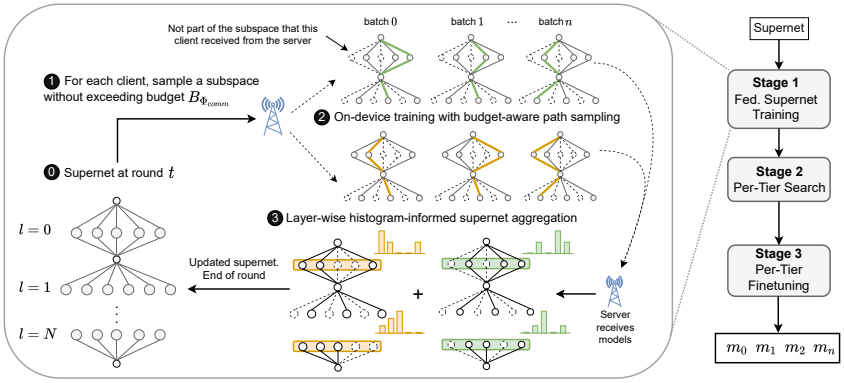


Figure 1: Training process workflow with `FedorAS`.

datasets [37, 20, 17, 33, 50, 50] are available. However, this also imposes two major limitations on the current NAS approaches: i) *privacy*, i.e. these methods were 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 [21, 58, 39], which has mainly focused on inference performance hitherto.

Motivated by the aforementioned phenomena and limitations of the existing NAS methods, we propose `FedorAS`, a system that performs NAS over *heterogeneous devices* holding *heterogeneous data* in a resource-aware and federated manner. To this direction, we cluster clients in tiers based on their capabilities and design supernet comprising of operations covering the whole spectrum of compute complexities. This supernet acts both as search space and a weight-sharing backbone. Upon federation, the supernet is only partially and stochastically shared to clients, respecting their computational and bandwidth capabilities. In turn, we leverage resource-aware one-shot path sampling [23] to facilitate lightweight on-device NAS. 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. Once federated training of the supernetwork has completed, usable pretrained networks can be extracted even before performing fine-tuning or personalising per device, 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 propose a system for resource efficient federated NAS that can be applied in *cross-device* settings, where partial device participation, device and data heterogeneity are innate characteristics.
- We implement a system called `FedorAS` (Federated nAS) that leverages a server-resident supernet enabling weight sharing for efficient knowledge exchange between clients, without directly sharing common model architectures with one another.
- We propose a novel aggregation method named OPA (OPerator Aggregation) for weighing updates from multiple "single-path one-shot" client updates in a frequency-aware manner.
- We empirically evaluate the performance and convergence of our system under IID and non-IID settings across different datasets, tasks and modalities, spanning different device distributions and compare our system's behaviour against state-of-the-art FL techniques.

2 The FedorAS Framework

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

Workflow. `FedorAS`'s workflow consists of three stages (Fig. 1): (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. 2.1). We then search for models from the supernet with the goal of finding the best architecture per tier (**Stage 2**, Sec. 2.2). Models are effectively sampled, validated on a global validation set and ranked per tier. These architectures and their associated weights act as initialisation to the next phase, where each model is fine-tuned in a per-tier manner (**Stage 3**. Sec. 2.3). The end goal of our system is to have the best possible model per each cluster of devices.

Design rationale. 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 from 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. In order to ensure efficient training of a supernet, we chose to base our approach on SPOS [23] as its training procedure is one of the most lightweight, introducing very little overhead in terms of memory consumption and/or required floating-point operations, especially compared to [47]. Last, we opted for clustering devices into "tiers" according to their computational capabilities and search for an architecture for each tier as a balance between having one architecture to fit all needs [25] and having a different architecture per client [49].

2.1 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.

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 [34, 44]. To this direction, `FedorAS` adopts a uniform parameter size budget, $B_{\Phi_{\text{comm}}}$, and samples[1] the search space for operators until this limit is hit (Eq. 1). Setting the limit to half the size of a typical network deployed for a task worked sufficiently well in our experiments and in fact accelerated convergence (Sec. 3.4).

$$\sum_{l=0}^{L} \sum_{o \in \mathbb{O}_l} \mathbb{1}_{\hat{\mathbb{O}}_l}(o) \Phi_{\text{comm}}(o) < B_{\Phi_{\text{comm}}}, \hat{\mathbb{O}}_l \subseteq \mathbb{O}_l \tag{1}$$

where L is the number of layer in the supernet, $\mathbb{O}_l$ the candidate and $\hat{\mathbb{O}}_l$ the selected operations in layer l, $\mathbb{1}$ a unit vector of valid operations and Φ_{comm} a measure of DNN size (e.g. #parameters).

In terms of sampling strategies, we experimented with uniform operator sampling, which we found to work sufficiently well and provide uniform coverage over the search space. It is also worth noting that a different subspace (not necessarily mutex) is selected for each participating client in each round.

❷ Client-side sampling & local training. Participating clients receive the subspace sampled on the server, $\{\hat{\mathbb{O}}_l\}_{l \in [0;L]}$, 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. This FLOPs limit is defined per tier so that a network does not exceed the capabilities of the target device. Path sampling is performed randomly across the valid (locally available) operations of each layer to ensure uniform coverage of the search space (Eq. 2). Alternative strategies, such as probabilistic sampling are left as future work.

$$p_L = \bigcup_{l \in [0;L]} o_l \sim \mathcal{U}\{\hat{\mathbb{O}}_l\} \;\; \text{s.t.} \;\; \sum_{l=0}^{L} \Phi_{\text{train}}(o_l) < B^{\text{Tier}}_{\Phi_{\text{train}}} \tag{2}$$

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

After having sampled the path, a model is created and gets a forward and backward pass from the input sample at hand, in a single-path one-shot manner [23]. 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. Operator updates coming from single clients are dropped for privacy reasons.

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 a client (FedAvg [48]) would not reflect the training experience during a round. For this reason, we propose OPA, OPerator Aggregation, an aggregation method that weighs updates based on the relative experience of an operator across all clients that have updated that operator. Concretely, for each operation we track the number of examples updating its weights for each client. Then, upon aggregation, for searchable operation o_l, we normalise over all the updates of that specific operator across clients. Formally put:

$$\omega_g^{(t+1)}(o_l) = \sum_{i \in K}^{|k|} \frac{|D_{(i,o_l,t)}|}{\sum_{\forall j \in [0,k-1]} |D_{(j,o_l,t)}|} \omega_i^{(t)}(o_l) \tag{3}$$

where $\omega_g^{(t)}(\cdot)$ are global weights, $\omega_i^{(t)}(\cdot)$ are local weights of client i at global step t and $|D(i, o_l, t)|$ is the number of samples having backpropagated through an operator o_l for client i in round t.

2.2 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 [19] and evaluated on a global validation set. Our rationale behind selecting the NSGA-II 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. 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.

2.3 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. Our hypothesis here is that compared to regular FL where one needs to trade off capacity of participation, `FedorAS` allows for better generalisation due to knowledge sharing through its supernet structure. Here we use conventional FedAvg and partial client participation per round, again with the goal of training a single network per tier.

3 Evaluation

This section provides a thorough evaluation of `FedorAS` across different tasks to show its performance and generality. Extra results (*e.g.* convergence, fairness) can be found in the Appendix.

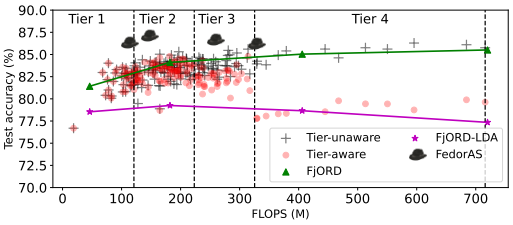
3.1 Experimental Setup

Datasets & Baselines. Datasets are summarised in Tab. 1. More information in Appendix D.

Table 1: High level overview of the datasets considered for evaluating `FedorAS`. More details about datasets and search spaces in the Appendix.

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

Figure 2: `FedorAS` outperforms other approaches (details in Appendix D.6). CIFAR-10 (non-IID, α=1.0).



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.

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 D.4.

3.2 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 [25] and SPIDER [49] 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. 2. At the same time, while running in *cross-silo* setting is outside the scope of `FedorAS`, we have adapted our experimental setting to match that of FedNAS and SPIDER. Results are shown in Tab. 3.

Table 2: Cross-device federated NAS on CIFAR-10.

Dataset	Method	Mem. Peak (MB)	Perf. (%)
$\text{CIFAR10}_{\text{LDA}=1}$	FedNAS	3837	**90.02**
	FedNAS	**1919**	85.45
	$\texttt{FedorAS}_{\text{best}}$	1996	$86.34_{\pm 0.19}$
$\text{CIFAR10}_{\text{LDA}=0.1}$	FedNAS	3837	65.28
	FedNAS	**1919**	54.84
	$\texttt{FedorAS}_{\text{best}}$	1996	$\mathbf{81.53}_{\pm 0.29}$

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

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

Federated Learning baselines. Next, we compare the performance of `FedorAS` with different federated baselines, including *homogeneous* [56] and system *heterogeneous* systems [28, 38, 52]. In the former setting, we compare with results from [56] on $\text{CIFAR100}_{\text{LDA=0.1}}$. `FedorAS` performs 1 pp (percentage point) better than the FedAvg baseline, at 45.84%, but at a fraction of the cost[2]. At the same time, training end-to-end the resulting `FedorAS` architecture from scratch – in a similar homogeneous setting as the baseline – results in 11.43pp higher accuracy at 52.5%, again at a lower cost. Comparisons with heterogeneous baselines are summarised in Tab. 4 and also partly depicted in Fig. 2.

Table 4: Comparison with Federated baselines. `FedorAS` performs significantly better across datasets.

Dataset	Method	MFLOPs	Params (M)	Perf. (%)
$\text{CIFAR10}_{\text{LDA}=1000}$	FjORD [28]	[**35.05**, **139.30**, 312.76, 555.62]	[**0.70**, **2.79**, 6.28, 11.16]	[$78.19_{\pm 1.20}$, $78.63_{\pm 1.31}$, $78.25_{\pm 1.06}$, $77.19_{\pm 0.85}$]
	$\text{ZeroFL}_{s=90\%}$ [52]	$557.22^{\ddagger}$	43.7	$82.82_{\pm 0.64}$
	$\texttt{FedorAS}_{\text{per tier}}$	[111.20, 139.67, **256.33**, **329.10**]	[2.96, 2.93, **3.35**, **4.32**]	[$\mathbf{89.40}_{\pm \mathbf{0.19}}$, $\mathbf{89.60}_{\pm \mathbf{0.15}}$, $\mathbf{89.64}_{\pm \mathbf{0.22}}$, $\mathbf{89.24}_{\pm \mathbf{0.29}}$]
$\text{CIFAR10}_{\text{LDA}=1}$	FjORD [28]	[**35.05**, **139.30**, 312.76, 555.62]	[**0.70**, **2.79**, 6.28, 11.16]	[$78.54_{\pm 0.12}$, $79.25_{\pm 0.51}$, $78.66_{\pm 0.29}$, $77.35_{\pm 0.44}$]
	$\text{ZeroFL}_{s=90\%}$ [52]	$557.22^{\ddagger}$	43.7	$81.04_{\pm 0.28}$
	$\texttt{FedorAS}_{\text{per tier}}$	[115.67, 183.00, **262.29**, **329.57**]	[2.59, 2.90, **3.55**, **4.31**]	[$\mathbf{85.99}_{\pm \mathbf{0.13}}$, $\mathbf{86.30}_{\pm \mathbf{0.41}}$, $\mathbf{86.34}_{\pm \mathbf{0.19}}$, $\mathbf{85.58}_{\pm \mathbf{0.55}}$]
$\text{CIFAR10}_{\text{LDA}=0.1}$ (Acc. (%) ↑ is better)	FjORD [28]	[**35.05**, **139.30**, 312.76, 555.62]	[**0.70**, **2.79**, 6.28, 11.16]	[$61.43_{\pm 0.39}$, $60.81_{\pm 1.42}$, $59.72_{\pm 5.19}$, $57.44_{\pm 3.53}$]
	$\texttt{FedorAS}_{\text{per tier}}$	[117.33, 159.27, **238.48**, **345.17**]	[2.17, 3.13, **2.49**, **2.61**]	[$\mathbf{81.01}_{\pm \mathbf{0.46}}$, $\mathbf{81.53}_{\pm \mathbf{0.29}}$, $\mathbf{80.64}_{\pm \mathbf{0.66}}$, $\mathbf{80.85}_{\pm \mathbf{0.28}}$]
Shakespeare (Acc. (%) ↑ is better)	FjORD [28]	[**0.01, 0.04, 0.08, 0.14, 0.22**]	[**0.01, 0.04, 0.08, 0.14, 0.22**]	[$4.44_{\pm 0.07}$, $3.91_{\pm 0.10}$, $3.87_{\pm 0.13}$, $3.87_{\pm 0.13}$, $3.87_{\pm 0.13}$]
	$\texttt{FedorAS}_{\text{per tier}}$	[7.09, 12.00, 14.71, 20.54, 23.93]	[0.09, 0.15, 0.18, 0.26, 0.30]	[$\mathbf{3.43}_{\pm \mathbf{0.01}}$, $\mathbf{3.39}_{\pm \mathbf{0.04}}$, $\mathbf{3.38}_{\pm \mathbf{0.03}}$, $\mathbf{3.40}_{\pm \mathbf{0.01}}$, $\mathbf{3.42}_{\pm \mathbf{0.01}}$]
SpeechCommands (Perplexity (%) ↓ is better)	Oort [38]†	**4.76**	21.29	62.20
	PyramidFL [42]†	**4.76**	21.29	63.84
	$\texttt{FedorAS}_{\text{best}}$	10.00	**0.63**	**70.10**

† [38, 42] perform client selection based on system heterogeneity and are provided for context. ‡ [52] speeds-up training w/ highly-sparse convs, attainable only with specialised h/w.

3.3 Supernet weight-sharing with FedorAS

In this section, we evaluate the impact of weight sharing through `FedorAS`' supernet. To this end, 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. Results are presented in Tab. 5.

The standard setting across datasets is that we first train with `FedorAS` across the three stages described in Sec. 2. 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 (tier and above). 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. Indeed, we can see in all cases models belonging to the most

[2] 1.11 vs 0.16 GFLOPS, 11.4M vs 1.62M parameters, 4000 vs 850 global rounds

Table 5: Models discovered by `FedorAS` benefit from weight sharing across tiers compared to models using the same architecture but trained end-to-end on eligible clients.

Dataset	Clients	Setting	Partitioning	Mode	Classes	Tier 1	Tier 2	Tier 3	Tier 4
CIFAR-10 (Acc. (%) ↑ is better)	100	Standard	$\text{IID}_{\alpha=1000}$	`FedorAS`	10	**89.40**±0.19	**89.60**±0.15	**89.64**±0.22	**89.24**±0.29
				End2End		89.05±0.17	87.84±0.38	86.18±0.38	81.27±0.81
			$\text{non-IID}_{\alpha=1.0}$	`FedorAS`	10	85.99±0.13	**86.30**±0.41	**86.34**±0.19	**85.58**±0.55
				End2End		**87.12**±0.44	86.29±0.86	85.10±0.44	80.10±1.92
			$\text{non-IID}_{\alpha=0.1}$	`FedorAS`	10	**81.01**±0.46	**81.53**±0.29	**80.64**±0.66	**80.85**±0.28
				End2End		70.61±2.16	70.30±1.90	68.29±0.49	64.87±1.48
CIFAR-100 (Acc. (%) ↑ is better)	500	Standard	$\text{non-IID}_{\alpha=0.1}$	`FedorAS`	100	**45.25**±0.13	**45.84**±0.18	**45.42**±0.39	**45.07**±0.71
				End2End		36.30±0.96	39.26±1.21	39.06±1.32	36.77±1.32
Speech Commands (Acc. (%) ↑ is better)	2112	Standard	*given*	`FedorAS`	12	80.19±1.78	**80.47**±1.69	**81.0**±1.58	**80.56**±0.40
				End2End		**81.92**±1.32	79.94±0.84	78.57±1.42	79.36±1.67

Dataset	Clients	Setting	Partitioning	Mode	Classes	Tier 1	Tier 2	Tier 3	Tier 4	Tier 5
Shakespeare (Perplexity (%) ↓ is better)	715	Standard	*given*	`FedorAS`	90	**3.43**±0.01	**3.39**±0.04	**3.38**±0.03	**3.40**±0.01	**3.42**±0.01
				End2End		3.44±0.03	3.50±0.02	3.47±0.08	3.52±0.07	3.60±0.04

restricted tiers (least amount of capable clients) benefit from being initialised from a supernet. The improvement is further magnified as non-IID-ness increases, leading to increases of up to 15 p.p. over random initialisation. Interestingly, we also see that in many cases low-tier models seem to benefit; we attribute this to the fact that, by being a part of the supernet, their weights are also updated in a context of much larger models which allows them to eventually reach better minima. We present results on different tasks of the same dataset in Appendix G.

3.4 Ablation and sensitivity study

Here 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 CIFAR-10 (non-IID, α=1.0). Last, we assess the impact of differently sized validation sets to the quality of the models, as well as the behaviour of `FedorAS` to different clients to tiers allocation.

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

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. Thus, not only is the overhead of communication reduced with `FedorAS`, but 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. More details in Appendix F.

Validation set size. In order to assess the quality of models during search (stage 2), we need 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. 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 the quality of the final models. Only in the case of SpeechCommands, we notice an 1pp drop on average when scaling down to 20%, which can be attributed to the imbalanced nature of the dataset. Detailed results are presented in the Appendix H.

Different allocation of clients to tiers. To showcase that `FedorAS` can work efficiently under realistic settings, we also adopted the client capabilities from Oort [38]. We witnessed similar scaling in the quality of the end models. Detailed results are presented in the Appendix I.

4 Conclusion

In this work, we have presented `FedorAS`, a system that performs resource-efficient federated NAS in the cross-device setting. We have shown that our technique leads to better results across modalities and tasks compared to existing federated solutions. Specifically, `FedorAS` is the first method to enable cross-device NAS offering significantly lower overhead compared to existing federated NAS techniques while at the same time achieving state-of-the-art performance compared to heterogeneous FL solutions by means of effective weight sharing and flexible resource-aware training.

References

[1] 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, pages 96–103, New York, NY, USA, April 2021. Association for Computing Machinery.

[2] Mario Almeida, Stefanos Laskaridis, Ilias Leontiadis, Stylianos I Venieris, and Nicholas D Lane. EmBench: Quantifying Performance Variations of Deep Neural Networks across Modern Commodity Devices. In *The 3rd International Workshop on Deep Learning for Mobile Systems and Applications (EMDL)*, pages 1–6, 2019.

[3] 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, page 658–672, New York, NY, USA, 2021. Association for Computing Machinery.

[4] 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.

[5] Anonymous. Personalised neural architecture search for federated learning.

[6] 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*, pages 1253–1269. Association for Computing Machinery, New York, NY, USA, October 2020.

[7] Axel Berg, Mark O'Connor, and Miguel Tairum Cruz. Keyword Transformer: A Self-Attention Model for Keyword Spotting. In *Proc. Interspeech 2021*, pages 4249–4253, 2021.

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

[9] 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, pages 1175–1191, New York, NY, USA, October 2017. Association for Computing Machinery.

[10] 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)*, pages 1–6, 2021.

[11] James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. Quasi-recurrent neural networks. In *International Conference on Learning Representations (ICLR)*, 2017.

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

[13] 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.

[14] 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.

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

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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*, pages 248–255, 2009.

[21] 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, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 10480–10490. Curran Associates, Inc., 2020.

[22] 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.

[23] 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*, pages 544–560. Springer, 2020.

[24] 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.

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

[26] 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, 2020.

[27] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, nov 1997.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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*, pages 1–13, 2022.

[33] William M. Fisher John S. Garofolo, Lori F. Lamel et al. Timit acoustic-phonetic continuous speech corpus. 1983.

[34] 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.

[35] 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.

[36] 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.

[37] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[38] 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)*, pages 19–35. USENIX Association, July 2021.

[39] 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.

[40] 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, pages 420–437, New York, NY, USA, October 2021. Association for Computing Machinery.

[41] 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)*, pages 68–79, 2021.

[42] Chenning Li, Xiao Zeng, Mi Zhang, and Zhichao Cao. Pyramidfl: A fine-grained client selection framework for efficient federated learning.

[43] 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.

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

[45] Tian Li, Maziar Sanjabi, Ahmad Beirami, and Virginia Smith. Fair Resource Allocation in Federated Learning. In *International Conference on Learning Representations (ICLR)*, 2020.

[46] Or Litany, Haggai Maron, David Acuna, Jan Kautz, Gal Chechik, and Sanja Fidler. Federated learning with heterogeneous architectures using graph hypernetworks, 2022.

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

[48] 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*, pages 1273–1282. PMLR, 2017.

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

[50] 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*, pages 5206–5210. IEEE, 2015.

[51] 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)*, pages 8026–8037, 2019.

[52] 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.

[53] 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.

[54] 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.

[55] 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.

[56] 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.

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

[58] 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.

[59] 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*, page 102402, 2021.

[60] Roman Vygon and Nikolay Mikhaylovskiy. Learning efficient representations for keyword spotting with triplet loss. In *Speech and Computer*, pages 773–785. Springer International Publishing, 2021.

[61] 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.

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

[63] C. Wu, D. Brooks, K. Chen, D. Chen, S. Choudhury, M. Dukhan, K. Hazelwood, E. Isaac, Y. Jia, B. Jia, T. Leyvand, H. Lu, Y. Lu, L. Qiao, B. Reagen, J. Spisak, F. Sun, A. Tulloch, P. Vajda, X. Wang, Y. Wang, B. Wasti, Y. Wu, R. Xian, S. Yoo, and P. Zhang. Machine Learning at Facebook: Understanding Inference at the Edge. In *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 331–344, 2019.

[64] 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*, 2021.

[65] 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)*, pages 830–840, July 2021.

[66] 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. February 2022.

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

A Related Work

Data and system heterogeneity. Traditionally, works on heterogeneous FL have been focused on tackling the statistical data heterogeneity [57, 43, 29, 22, 45] or minimising the upstream communication cost [40, 36, 61, 24, 4] of sending updates from client to servers, as the primary bottleneck in the federated training process. However, it has become apparent that computational disparity becomes an equally important barrier for contributing knowledge in FL. As such, lately, there has been a line of work focusing on this very problem, where the discrepancy between the dynamics of different clients affects the convergence rate or fairness of the deployed system. Specifically, such solutions draw from efficient ML and attempt to dynamically alter the footprint of local models my means of structured (PruneFL [32]), unstructured (Adaptive Federated Dropout [10], LotteryFL [41]) or importance-based pruning (FjORD [28]), quantisation (AQFL [1]), low-rank factorisation (FedHM [64]), sparsity-inducing training (ZeroFL [53]) or distillation (GKT [26]). However, each approach has limitations, either because they involve extra training overhead [26] and residence of multiple DNN copies in memory [1], or because they require specialised hardware for performance gains ([53, 10]). Last, some of the architectural changes proposed may not offer the degrees of freedom that NAS exposes [28, 64, 41].

Federated NAS. The concept of performing NAS in a federated setting has been visited before in the literature [25, 49, 65, 66, 46]. However, current solutions vary greatly in their approaches and goals, with most of them being applied and applicable to the cross-silo setting, where full participation and small number of participants are expected. Specifically, one of the first works in the area was FedNAS [25], 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 that make it largely inapplicable for cross-device setups spanning many clients of limited capabilities partially participating in training. To mitigate this requirement, [65] proposes an RL-based approach for FL-based NAS, where they sample a series of models to be trained on each round on clients Despite the intention, their three-phase process (pre-train, search, train) still incurs significant overheads. A somewhat different approach is adopted by HAFL [46], which leverages 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. On the front of personalisation, FedPNAS [5] searches for architectures with a shared based component across clients and a personalised component. However, this work 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 [49] 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 non-negligible overheads make porting this approach to the cross-device setting non-trivial.

In contrast, `FedorAS` brings Federated NAS to the cross-device setting and is designed with resource efficiency 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 and generalise across different modalities and granularity of tasks.

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 [53] – 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 [35].

C Limitations & Future Work

Despite the challenges address 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 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. Last but not least, we have considered privacy-enhancing techniques, such as Differential Privacy [12] or Secure Aggregation [9, 6], 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 Detailed Experimental Setup

D.1 Implementation

FedorAS was implemented on top of the Flower (v0.18) [8] framework and PyTorch (v1.11.0) [51]. We run all our experiments on a private cloud cluster in a simulated manner, across four iterations each and report averages and standard deviations.

D.2 Datasets

We partition CIFAR-10/100 following the Latent Dirichlet Allocation (LDA) partitioning [30], 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 [56]. 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 [62] 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 [7, 16, 60] 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 [18] features from each audio clip [67, 7]. 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 $1 \times 40 \times 51$ input that can be passed to a CNN.

Shakespeare. This dataset is built [14, 48] 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.

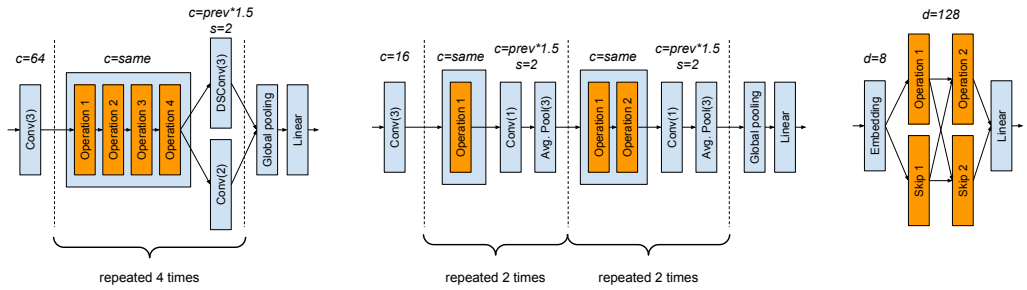


Figure 3: Summary of models used in our experiments: *left)* CIFAR-10 and CIFAR-100, *middle)* Speech Commands, *right)* Shakespeare. Blocks highlighted in blue are fixed, orange blocks represent searchable layers – see Sec.D.3 for details. c – output channels, s – stride, d – output feature dimension, 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.

The vocabulary considered has 86 English characters as well as four special tokens: start and end of sentence, padding and out-of-vocabulary tokens.

D.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.

CIFAR-10 and CIFAR-100. We use a ResNet-like search space similar to the one used in, for example, Cai et al. [13]. 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 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 ($2\times$ in each dimension) and increase number of channels ($1.5\times$). 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 Horvath et al. [28]. 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 #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 [27],
- a GRU layer [15],
- a LiGRU layer with `tanh` activation [54],
- a QuasiRNN layer [11],
- 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 [55], 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.

D.4 Tiers: definition and client assignment

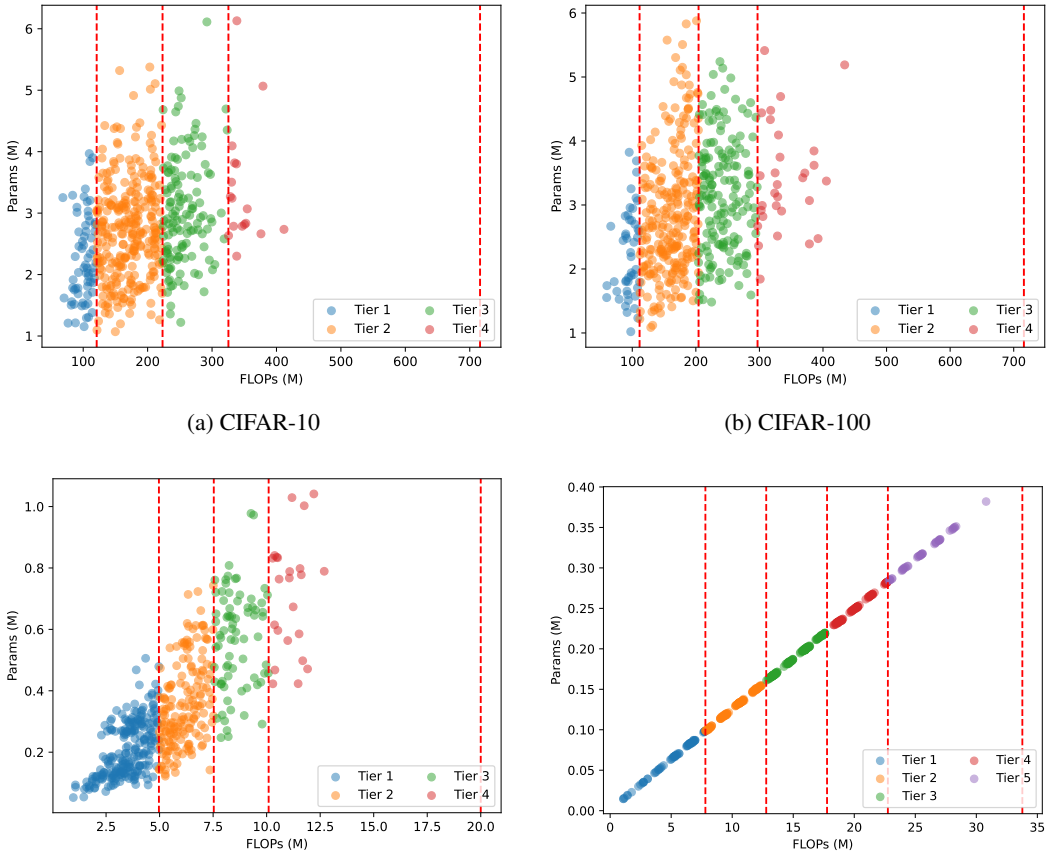
With `FedorAS` the discovery and training of architectures happens in a tier-aware fashion as a federated process. In this work we considered splitting each search space along the FLOPs dimensions (but other splits are possible, e.g: energy, peak-memory usage, etc – or a combination of these). Fig. 4 illustrates the span in terms of model parameters and FLOPs of each search space presented in D.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 6 shows the 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.

Table 6: 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 G)	[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%

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 as illustrated in Fig. 4 (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 6.

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 [67], 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.



(a) CIFAR-10

(b) CIFAR-100

(c) SpeechCommands

(d) Shakespeare

Figure 4: 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.

D.5 Hyperparameters

Here we present the hyperparameters used across all datasets and tasks to generate the results presented in Sec. 3. In Tab. 7 we show the hyperparamters utilised for the first stage in the `FedorAS` framework: federated training of the supernet. 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. 8.

Table 7: Hyperparameters used for the Federated supernet training stage in `FedorAS`, as described in Sec. 2.1. 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. 2.1.

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 8: 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. 2.2 and Sec. 2.3 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 finetuning process. For Shakespeare, *step* LR decay worked best. This is applied at rounds 50 and 75, each decreasing the LR by a factor of $10\times$. 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

Regarding the End2End results (i.e. models discovered by `FedorAS` but trained from scratch – discarding the weights given by the supernet) we present the hyperparameters in Tab. 9. These hyperparameters were the ones used to generate the results in Tab. 5 and 12 as well as Fig. 2. In Sec. 3.2, we present a CIFAR-100 result that largely outperforms existing federated baseline of [56]. This was achieved by a model discovered by `FedorAS` but trained in End2End setting following the setup as in [56]: 10 clients per round for 4k rounds using batch 20, starting learning rate of 0.1 decaying to 0.01 following a *cosine* scheduling.

Table 9: The architectures discovered by `FedorAS` can also be trained from scratch. This table contains the hyperparameters utilised to generate the End2End results shown in Tab. 5 and 12. 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 finetuning process. For Shakespeare, *step* LR decay worked best. This is applied at rounds 250 and 375, each decreasing the LR by a factor of $10\times$.

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

In Table 2 we compare `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. This results in the table below, which extends the content of Tab. 2.

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

Dataset	**Method**	**Local Epochs**	**Batch**	**Mem. Peak (MB)**	**Perf. (%)**
$\text{CIFAR10}_{\text{LDA}=1}$	FedNAS	10	32	3837	**90.02**
	FedNAS	10	16	**1919**	85.45
	$\texttt{FedorAS}_{\text{best}}$	20	128	1996	86.34 ± 0.19
$\text{CIFAR10}_{\text{LDA}=0.1}$	FedNAS	10	32	3837	65.28
	FedNAS	10	16	**1919**	54.84
	$\texttt{FedorAS}_{\text{best}}$	20	128	1996	$\mathbf{81.53}\pm0.29$

D.6 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.2): 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. 9.

Tier-aware (Fig.2): 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 [28] (Fig.2, Tab. 4): FjORD is a baseline that is tackling system heterogeneity by means of Ordered Dropout. This particular variant is assuming the experimetnal 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 [28] (Fig.2, Tab. 4): For this baseline, we took FjORD and ran it on the same number of clusters as `FedorAS`, with LDA for CIFAR-10.

ZeroFL [52] (Tab. 4): For this baseline, we borrow the results of respective paper for CIFAR-10 with $\text{LDA} = \{1000, 1\}$, 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.

FedNAS [25] (Tab. 2, 3): FedNAS was one of the first papers attempting to perform NAS in a federated setting. Its search is DARTS-based and it was primarily aimed for the cross-silo setting. Cross-device FL NAS of limited scale (20 total clients with 4 clients sample per round) was also showcased in the original paper. We adjusted the code of FedNAS to perform cross-silo and cross-device FL NAS with setups of varying training footprints by adjusting the amount of local epochs and batch size. We have kept the original search space.

SPIDER [49] (Tab. 3): 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.

End2End (Tab. 5, 12, 14): 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 "End2End" experiments. Models belonging to higher tiers, as a whole, are never trained on data belonging to devices from lower tiers.

E Comparison against random search

In the main paper we compare our `FedorAS` to the state-of-the-art models from the literature (Sec. 3.2) and also investigate the impact of initializing models with weights from a supernet (Sec. 3.3). Here we present 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?

Table 11: 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

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. 11. Additionally, we also plot detailed performance of each model trained during this process in Fig. 5 for the sake of completeness.

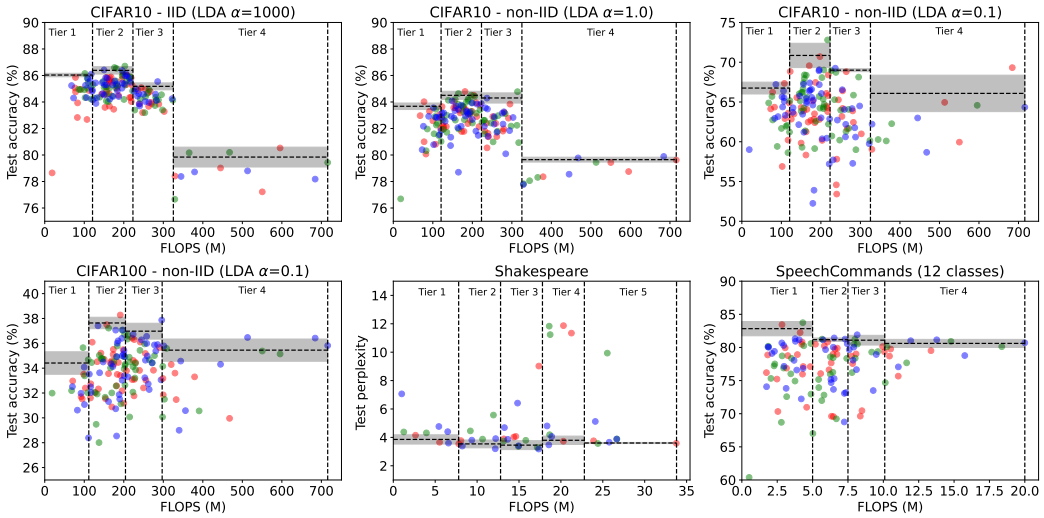
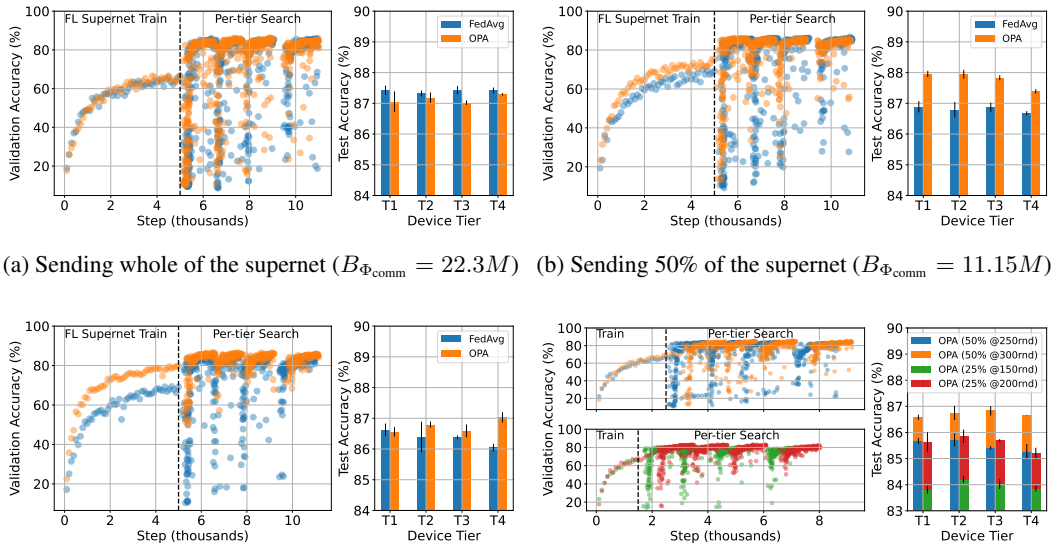


Figure 5: For each dataset, we randomly sample their search space and train these models end-to-end. We repeat this process (with new samples) three times, and overlap the scatter plots of different runs (i.e. red, green, blue). Then, for each tier we average the best performing models across each of the three runs and plot the result as a dashed horizontal line with grey area representing $\pm$ 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.

F 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. 6. 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 reduced, i.e., as smaller portions of the supernet are sent to the clients. In Fig. 6 (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.



(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 6: 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. 2.1) in the first 5k steps and the per-tier search stage (Sec. 2.2) the remaining steps; and, a bar plot that shows the average test accuracies of each tier after federated finetuning (Sec. 2.3). 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.

G Impact of weight-sharing on different task training

In this section, we expand on the analysis of Sec. 3.3. 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 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 finetuning of the whole network for another 100. We present the results in Tab. 12.

Table 12: 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`	$20 \rightarrow 100$	**57.93**$\pm$**0.31**	**57.86**$\pm$**0.7**	**38.63**$\pm$**0.74**	**37.68**$\pm$**0.73**
				End2End		41.64$\pm$0.26	42.45$\pm$0.58	27.11$\pm$0.64	21.28$\pm$1.26
Speech Commands	2112	Transfer	*given*	`FedorAS`	$12 \rightarrow 35$	**67.06**$\pm$**2.12**†	**66.87**$\pm$**1.85**†	**67.65**$\pm$**1.59**	**68.49**$\pm$**1.47**
				End2End		64.99$\pm$1.41†	65.03$\pm$0.69†	66.55$\pm$2.2	66.84$\pm$0.87

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

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 $+14.91$ p.p (percentage points) of accuracy compared to training the same architectures end-to-end on eligible devices. Similarly, we achieve $+1.6$ p.p higher accuracy for SpeechCommands.

H 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. 5. We maintain the same hyperparameters as those used to generate Tab. 5. Results are depicted in Tab 13 for all datasets.

Table 13: 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$\pm$0.19	89.60$\pm$0.15	89.64$\pm$0.22	89.24$\pm$0.29
			0.5		-0.31	-0.10	-0.14	+0.03
			0.2		-0.06	-0.17	-0.19	-0.02
		non-IID$_{\alpha=1.0}$	1.0	10	85.99$\pm$0.13	86.30$\pm$0.41	86.34$\pm$0.19	85.58$\pm$0.55
			0.5		+0.06	-0.18	+0.19	-0.19
			0.2		-0.05	+0.02	-0.28	-0.29
		non-IID$_{\alpha=0.1}$	1.0	10	81.01$\pm$0.46	81.53$\pm$0.29	80.64$\pm$0.66	80.85$\pm$0.28
			0.5		+0.63	-0.5	-0.29	-0.14
			0.2		+0.78	+0.13	-1.55	-0.09
CIFAR-100	500	non-IID$_{\alpha=0.1}$	1.0	100	45.25$\pm$0.13	45.84$\pm$0.18	45.42$\pm$0.39	45.07$\pm$0.71
			0.5		-0.10	+0.01	-0.27	-0.38
			0.2		+0.07	+0.11	+0.32	-0.60
Speech Commands	2112	*given*	1.0	12	80.19$\pm$1.78	80.47$\pm$1.69	81.0$\pm$1.58	80.56$\pm$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$\pm$0.01	3.39$\pm$0.04	3.38$\pm$0.03	3.40$\pm$0.01	3.42$\pm$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

I Different allocation of clients to tiers

In this experiment, contrary to what was described in Appendix D.4, we adopt the device capabilities trace from Oort [38] for our device to cluster allocation and performed training on CIFAR-10 and CIFAR-100. Results are depicted on Tab. 14. It can be witnessed that `FedorAS` is able to operate even under harsher heterogeneity conditions and still output state-of-the-art models in the federated setting.

Table 14: `FedorAS` performance under [38] device clustering. We see that performance still scales well, albeit taking an impact due to more extreme system heterogeneity compared to results from Tab. 5. This hints that weight-sharing through our supernet works well under varying device allocation settings.

Dataset	**#clients**	**Method**	**Perf.**
$\text{CIFAR10}_{\text{LDA}=1}$	100 (10)	$\texttt{FedorAS}_{\text{per tier}}$	[**87.77**$\pm$**0.09**, **87.43**$\pm$**0.28**, **87.15**$\pm$**0.42**, **87.01**$\pm$**0.11**]
	100 (10)	End2End	[85.83$\pm$1.24, 83.91$\pm$0.98, 83.21$\pm$0.60, 79.10$\pm$1.63]
$\text{CIFAR100}_{\text{LDA}=0.1}$	500 (10)	$\texttt{FedorAS}_{\text{per tier}}$	[**44.33**$\pm$**0.81**, **43.83**$\pm$**0.89**, **43.72**$\pm$**0.82**, **43.22**$\pm$**1.02**]
	500 (10)	End2End	[34.88$\pm$0.28, 33.76$\pm$2.53, 32.72$\pm$0.68, 30.77$\pm$2.30]

J Fairness

Up to this point, we have reported on the global test accuracy, which is a union over the test sets of the clients for all datasets but SpeechCommands, which defines a separate set of clients mutually exclusive to the ones participating in federated training. In this section we want to measure how the ability of our system to aggregate knowledge across tiers during supernet training affects the fairness in participation. We quantify this through variance and worst-case statistical measures of client performance on their respective test set. These test sets are considered to be following the same distribution (LDA or pre-split) of their train and validation sets.

Table 15: 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**$\pm$**8.58**	[**82.43**$\pm$**7.87**, **81.41**$\pm$**9.34**, **81.78**$\pm$**8.49**, **79.80**$\pm$**8.42**]	**47.00**
	End2End		65.70 $\pm$13.11	[67.43$\pm$12.29, 66.10$\pm$14.05, 66.14$\pm$13.02, 63.12$\pm$12.76]	11.00
CIFAR-100 (Acc. (%) ↑ is better)	Fedoras	500	**45.61**$\pm$13.15	[**44.46**$\pm$13.81, **45.51**$\pm$12.28, **45.19**$\pm$13.15, **47.29**$\pm$13.15]	**5.00**
	End2End		31.08$\pm$**12.12**	[30.72$\pm$**13.16**, 30.65$\pm$**11.27**, 30.43$\pm$**11.25**, 32.52$\pm$**12.59**]	0.00
Shakespeare (Perplexity ↓ is better)	Fedoras	715	**2.93**$\pm$**1.01**	[**2.93**$\pm$**0.97**, **2.94**$\pm$**0.99**, **2.88**$\pm$**1.03**, **2.98**$\pm$**1.04**, **2.94**$\pm$**1.01**]	8.43
	End2End		3.07$\pm$1.10	[3.07$\pm$1.05, 3.08$\pm$1.10, 3.02$\pm$1.13, 3.11$\pm$1.13, 3.07$\pm$1.10]	**8.36**

Results are shown on Tab. 15 for the different datasets offering per-client test data splits for all clients. It can be seen that `FedorAS` leads consistently 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. 7 for CIFAR-10, where we show performance of Fedoras vs. end-to-end FL-trained models per client.

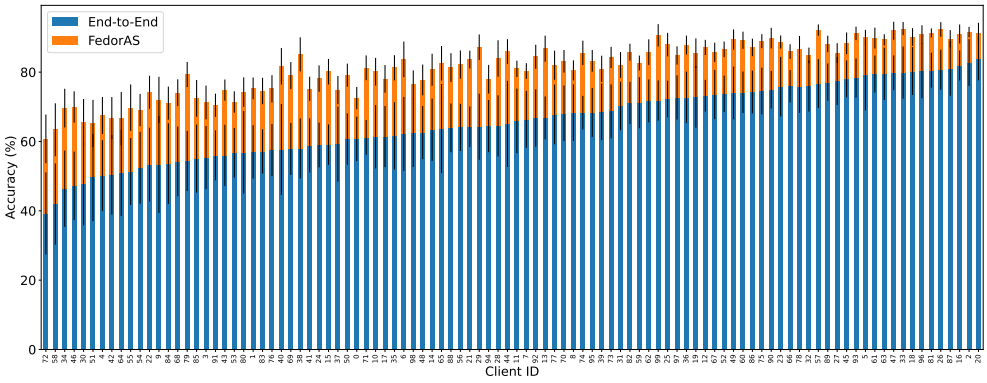


Figure 7: Accuracy per client of `FedorAS` vs. end-to-end FL-trained models (same architecture) on CIFAR-10. Accuracy is quantified on each client's dataset from the model associated with that device's tier and error bars represent the standard deviation of accuracy between different runs. Across runs, the allocation of data to clients does not change. Ordered by ascending end-to-end accuracy.