Supernet Training for Federated Image Classification under System Heterogeneity

Taehyeon Kim [1] **Se-Young Yun** [1]

Abstract

Efficient deployment of deep neural networks across many devices and resource constraints, especially on edge devices, is one of the most challenging problems in the presence of data-privacy preservation issues. Conventional approaches have evolved to either improve a single global model while keeping each local training data decentralized (i.e., data-heterogeneity) or to train a once-for-all network that supports diverse architectural settings to address heterogeneous systems equipped with different computational capabilities (i.e., model-heterogeneity). However, little research has considered both directions simultaneously. In this work, we propose a novel framework to consider both scenarios, namely Federation of Supernet Training (FedSup), where clients send and receive a supernet whereby it contains all possible architectures sampled from itself. Specifically, in the FedSup framework, a weight-sharing approach widely used in the training single shot model is combined with the averaging of Federated Learning (FedAvg). Under our framework, we present an efficient algorithm (E-FedSup) by sending the sub-model to clients in the broadcast stage for reducing communication costs and training overhead.

1. Introduction

Deep neural networks (DNN) have achieved remarkable empirical success in many machine learning applications. As a next evolution, there has been an increasing demand for training a model by using local data from mobile devices and the Internet of Things (IoT) because billions of local machines worldwide can bring more computational power and amounts of data than those of the center server machine (Lim et al., 2020; El-Sayed et al., 2018). However, it is still challenging to deploy them efficiently on diverse hardware platforms whose specification (i.e., latency, TPU) is significantly various (Cai et al., 2019) and to train a global model without sharing local data. Federated learning (FL) is one of the most popular paradigms of collaborative machine learning (McMahan et al., 2017; Li et al., 2019; Han et al., 2020; Li et al., 2018; Karimireddy et al., 2019; Mohri et al., 2019; Lin et al., 2020; Acar et al., 2021). In general, to train the central server (e.g., service manager) in the FL framework, each client (e.g., mobile devices or whole organization) updates its local model via their private data by itself; all local updates are aggregated to the global model; after which the procedure is repeated until convergence. Most notably, federated averaging (FedAvg) (McMahan et al., 2017) uses averaging as its aggregation method over the local learned models on clients. Such FL framework ensures us to occlude many of the systematic privacy leakages (Voigt & Von dem Bussche, 2017).

Recent FL works have been evolving into designing new objective functions for the aggregation of each model (Acar et al., 2021; Karimireddy et al., 2019; Li et al., 2018; Wang et al., 2020; Felix et al., 2020; Yuan & Ma, 2020; Li et al., 2021a), using auxiliary data in the center server (Lin et al., 2020; Zhang et al., 2022), encoding the weight for an efficient communication stage (Wu et al., 2022; Hyeon-Woo et al., 2022; Xu et al., 2021), or recruiting helpful clients for more accurate global model (Li et al., 2019; Cho et al., 2020; Nishio & Yonetani, 2019). On the other side, there has been tremendous recent interest in deploying the FL algorithms for real-world applications such as mobile devices and the Internet of Things (IoT) (Nishio & Yonetani, 2019; Diao et al., 2021; Horvath et al., 2021; Hyeon-Woo et al., 2022). However, less has been tackled on the issue of delivering compact models specialized for the edge device having different hardware platforms and efficiency constraints (Figure 1 (a)). It is known that the inference time of a neural network varies greatly depending on the specification of devices (Yu et al., 2018). It would become a significant bottleneck at every aggregation round in FL's synchronous training if a same-size model is distributed to clients without considering local resources (Li et al., 2020).

This paper presents a novel framework to consider both

[1]KAIST AI, Seoul, South Korea. Correspondence to: Taehyeon Kim <potter32@kaist.ac.kr>, Se-Young Yun <yunseyoung@kaist.ac.kr>.

DyNN workshop at the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, 2022.

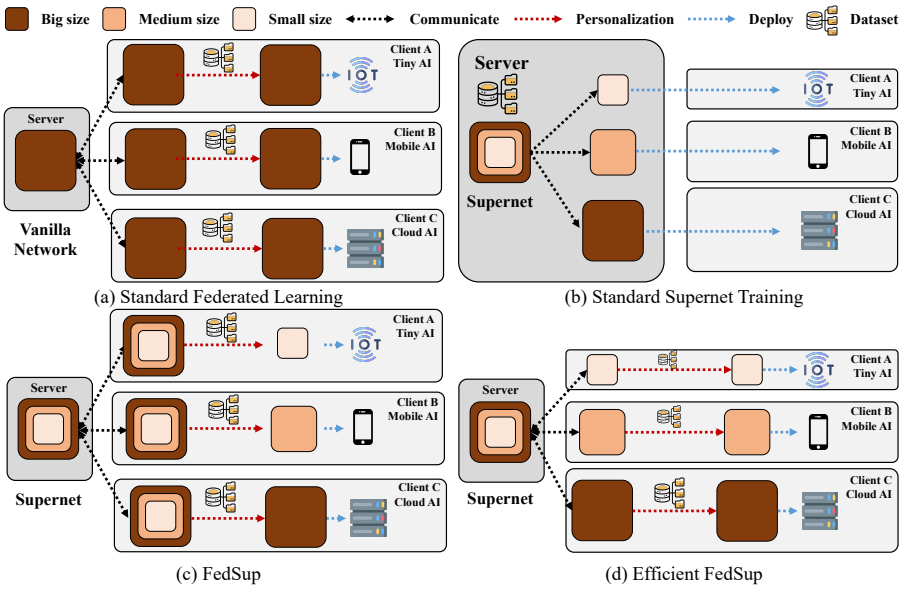


Figure 1. Overview of (a) standard FL framework, (b) supernet training in a standard datacenter optimization (i.e., centralized settings), (c) our proposed federation of supernet training framework (FedSup), and (d) efficient FedSup algorithm (E-FedSup).

scenarios, namely *Federation of Supernet Training* (FedSup) which encompasses the training of sub-models nested in a supernet under system heterogeneity. Using the weight-sharing in supernet training, FedSup forwards a supernet to each local client and ensembles the training of sub-models sampled from itself at each client (Figure 1 (c)). Referred to (Diao et al., 2021), we manifest a *Efficient FedSup* (E-FedSup) broadcasting a sub-model to local in lieu of a full supernet (Figure 1 (d)). For the evaluation of both methods, we focus on improving the global accuracy (of the server; universality) and the personalized accuracies (of on-device fine-tuned models; personalization).

Organization. The remainder of this paper is organized as follows. In Section 2, we discuss the recent literature on model-heterogeneity in FL and supernet training in NAS. In Section 3, we address our motivation for collaborating the federated learning with supernet training and provide our main methods, the Federation of Supernet Training (FedSup) and efficient FedSup (E-FedSup). In Section 4, we exhibit experimental results. Finally, Section 5 concludes the paper.

2. Related Work

Model Heterogeneity in FL. Model heterogeneity in FL, the problem of training heterogeneous local models with varying computation complexities, has remained largely under-explored in comparison with statistical data heterogeneity. Recently, a few works have been proposed in the following direction: generating a set of sub-models through a hypernetwork that outputs parameters for other neural networks (Shamsian et al., 2021), using a pruned model from a global model (Bouacida et al., 2020; Horvath et al., 2021; Luo et al.), and distilling the knowledge from local to global by using either extra proxy datasets or generator (Lin et al., 2020; Afonin & Karimireddy, 2022). However, pruning approaches are not really cost-effective in terms of inference time, and distillation-based methodologies require additional training overhead. Using a hypernetwork (Shamsian et al., 2021) or sampling a sub-model from the global model (Diao et al., 2021) may avoid such issues, but the sub-model scale is limited to only a single direction such as width or kernel size. Furthermore, such optimization is simple so that the accuracy gap among sub-models should be bridged through some advanced training techniques. Newly, (Mushtaq et al., 2021) apply continuous differentiable relaxation and gradient descent, but it is significantly sensitive to hyperparameter choices.

3. Method

3.1. Problem Settings: Federated Learning

The main goal of FL (McMahan et al., 2017) is to solve the following optimization problem:

$$\min_{\boldsymbol{w}} f(\boldsymbol{w}) \triangleq \min_{\boldsymbol{w}} \sum_{k \in S} p_k F_k(\boldsymbol{w}) \tag{1}$$

where S is the set of total clients, p_k is the weight of client k, such as $p_k \geq 0$, and $\sum_k p_k = 1$. The local objective of client k is to minimize $F_k(\boldsymbol{w}) = \mathbb{E}_{\boldsymbol{x}_k \sim \mathcal{D}_k}[\ell_k(\boldsymbol{x}_k, \boldsymbol{y}_k; \boldsymbol{w})]$ parameterized by $\boldsymbol{w}$ on the local data $(\boldsymbol{x}_k, \boldsymbol{y}_k)$ from local data distribution $\mathcal{D}_k$. FedAvg (McMahan et al., 2017), the canonical algorithm for FL, involves *local update*, which learns a local model $\boldsymbol{w}_k^t$ (Eq. 2) with learning rate η and synchronizing $\boldsymbol{w}_k^t$ with $\boldsymbol{w}^t$ every E steps,

$$\boldsymbol{w}_k^t \triangleq \begin{cases} \boldsymbol{w}_k^{t-1} - \eta \nabla F_k(\boldsymbol{w}_k^{t-1}) & \text{if } t \bmod E \neq 0 \\ \boldsymbol{w}^t & \text{if } t \bmod E = 0 \end{cases} \tag{2}$$

and *global aggregation*, which learns the global model $\boldsymbol{w}^t$ (Eq. 3) by averaging all $\boldsymbol{w}_k^t$ with regard to the client $k \in S^t$ uniformly sampled at random where p_k is used as $\frac{|\mathcal{D}_k|}{|\mathcal{D}|}$.

$$\boldsymbol{w}^t = \sum_{k \in S^t} p_k \boldsymbol{w}_k^t \tag{3}$$

3.2. Weight Sharing in Client System Heterogeneity

Weight-sharing NAS is an effective technique for assembling all the architectures as its sub-networks and jointly trains the supernet (Cai et al., 2019; Yu et al., 2020; Wang et al., 2021b;a). With the access permission to local data from the center server, it can be used for tackling the client system heterogeneity, the diversity in the processing capabilities and network bandwidth of clients, but not vice versa. Assuming the weights of the supernet as $\boldsymbol{w}$ and the sub-models $\boldsymbol{w}_{arch}$, then the problem is generally formulated as follows:

$$\min_{\boldsymbol{w}} \sum_{\boldsymbol{w}_{arch} \subset \boldsymbol{w}} \mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}}[\ell(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{w}_{arch})] \quad \text{where } \mathcal{D} = \cup_k \mathcal{D}_k$$

To preserve the data privacy, the above can be reformulated with a simple double summation:

$$\begin{aligned} &= \min_{\boldsymbol{w}} \sum_{\boldsymbol{w}_{arch} \subset \boldsymbol{w}} \sum_k p_k \mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}_k}[\ell_k(\boldsymbol{x}_k, \boldsymbol{y}_k; \boldsymbol{w}_{arch})] \\ &= \min_{\boldsymbol{w}} \sum_k p_k \sum_{\boldsymbol{w}_{arch} \subset \boldsymbol{w}} F_k(\boldsymbol{w}_{arch}) \end{aligned} \tag{4}$$

After exchanging the order of two summations, a new objective (Eq. 4) is obtained, termed as Federation of Supernet Training (FedSup) (Algorithm 1). Obviously, it is highly non-trivial due to the distinct learning dynamics of various child models under data heterogeneity; training strategies are mainly discussed in subsection 3.3.

3.3. Training Strategies for FedSup

Architecture Space The details of our search space are presented by referring to the previous neural architecture search (NAS) and FL approaches (Cai et al., 2019; Yu et al., 2020; Oh et al., 2021). Our network architecture consists of a stack with the MobileNet V1 blocks (Howard et al., 2017), and the detailed search space is summarized in Appendix. The arbitrary numbers of layers, channels, and kernel sizes can be sampled from our network. Following previous settings (Yu et al., 2018; 2020), *lower-index layers* in each network stage are always kept. Both kernel size and channel numbers can be adjusted in a layer-wise manner.

Inplace Distillation. During the FedSup training, a sub-model can be distilled with the soft labels predicted by the full model (biggest child), called *inplace distillation* (Yu et al., 2018). Without any additional models, it can supervise a sub-model's representation aligning into the same direction (i.e., *representation alignment*; a concept from (Kim et al., 2021)). The temperature hyperparameter and the balancing hyperparameter between distillation and target loss (Hinton et al., 2015) are not used in our experiments.

Algorithm 1 Generic Framework for FedSup

INPUT : Supernet $\boldsymbol{w}$, the number of sampled child models M, weight update function UPDATE

Initialize SuperNet $\boldsymbol{w}_0$
for $t \leftarrow 0, \ldots, T-1$ **do**
  $S^t \leftarrow$ SAMPLECLIENTS
  for each client $k \in S^t$ in parallel **do**
    $\boldsymbol{w}_k^{t,0} \leftarrow \boldsymbol{w}^t$
    for $e \leftarrow 0, \ldots, E-1$ **do**
      for $m = 1, \ldots, M$ **do**
        $\boldsymbol{w}_{arch_{k,m}}^{t,e} \leftarrow$ SAMPLEMODEL$(\boldsymbol{w}_k^{t,e})$
        $\boldsymbol{w}_{arch_{k,m}}^{t,e+1} \leftarrow$ OPTIMIZE$(\boldsymbol{w}_{arch_{k,m}}^{t,e})$
      end for
      $\boldsymbol{w}_k^{t,e+1} \leftarrow$ UPDATE$(\boldsymbol{w}_{arch_{k,1}}^{t,e+1}, \ldots, \boldsymbol{w}_{arch_{k,M}}^{t,e+1})$
    end for
  end for
  $\boldsymbol{w}^{t+1} \leftarrow \sum_{k \in S^t} p_k \boldsymbol{w}_k^{t,E}$
end for

M Sampled Child Models and Sandwich Rule. At every local training iteration, the gradients are aggregated from M sampled child models. If $M \geq 3$, the smallest child and the biggest child are included where the gradients are clipped (i.e., *sandwich rule* (Yu et al., 2018; 2020)). Through these aggregated gradients, a supernet's weight is updated where the "smallest" child denotes the model having the thinnest width, shallowest depth, and smallest kernel size under the pre-defined architecture space.

3.4. Efficient FedSup (E-FedSup)

If inplace distillation and sandwich rule are not applied during the local training, an alternative objective function can be used instead of Eq. 4:

$$\underbrace{\min_{\boldsymbol{w}} \sum_k p_k \sum_{\boldsymbol{w}_{arch} \subset \boldsymbol{w}} F_k(\boldsymbol{w}_{arch})}_{\text{FedSup}} \geq \underbrace{\min_{\boldsymbol{w}} \sum_k p_k F_k(\boldsymbol{w}_{arch_k})}_{\text{E-FedSup}}$$

In the broadcast stage, a sub-model is sent to each local client to achieve the efficiency on network bandwidth. A chief difference in the optimization is that FedSup samples a new sub-model every iteration, but E-FedSup trains a pre-fixed sub-model received from communication at local. E-FedSup saves communication cost by sending the child locally, and also curtails training overhead because it trains one model per iteration in lieu of several child models.

4. Experiment

4.1. Experimental Settings

Heterogeneous Distribution of Client Data. We conduct heterogeneous data distribution settings referring to the lit-

Table 1. Personalized accuracy on CIFAR-100 with 100 clients, $s = 50$, $f = 0.1$, and $m = 0.5$. A full supernet (Big) with dynamic width is utilized while the static version is applied on other algorithms.

Algorithm	Local-Only	FedAvg [2017]	Ditto [2021b]	LG-FedAvg [2020]	Per-FedAvg [2020]	FedSup	E-FedSup
Personalized Acc.	$27.98_{\pm 4.12}$	$49.61_{\pm 5.10}$	$44.10_{\pm 5.75}$	$39.92_{\pm 5.02}$	$44.21_{\pm 6.23}$	$\mathbf{56.51}_{\pm \mathbf{5.15}}$	$\mathbf{55.75}_{\pm \mathbf{5.61}}$

Table 2. Initial and personalized accuracy on CIFAR100 under various FL settings with 100 clients. The initial and personalized accuracy indicate the evaluated performance without fine-tuning and after five fine-tuning epochs for each client, respectively.

FL Settings			s=50				s=10			
f	τ	A	FedSup		E-FedSup		FedSup		E-FedSup	
			Initial	Personalized	Initial	Personalized	Initial	Personalized	Initial	Personalized
0.1	1	B	$38.94_{\pm 5.30}$	$53.55_{\pm 4.81}$	$39.37_{\pm 5.40}$	$54.88_{\pm 4.79}$	$22.43_{\pm 5.11}$	$65.12_{\pm 5.95}$	$22.42_{\pm 5.32}$	$64.75_{\pm 6.38}$
		M	$38.09_{\pm 5.40}$	$53.31_{\pm 5.26}$	$39.33_{\pm 5.10}$	$54.81_{\pm 5.22}$	$22.40_{\pm 5.23}$	$64.95_{\pm 6.21}$	$22.23_{\pm 5.66}$	$64.73_{\pm 6.55}$
		S	$36.01_{\pm 5.50}$	$52.29_{\pm 4.91}$	$37.34_{\pm 5.26}$	$53.08_{\pm 5.20}$	$21.17_{\pm 5.67}$	$64.18_{\pm 6.52}$	$21.32_{\pm 5.59}$	$64.29_{\pm 6.69}$
	5	B	$43.83_{\pm 6.20}$	$56.51_{\pm 5.15}$	$43.53_{\pm 6.22}$	$55.75_{\pm 5.61}$	$26.07_{\pm 6.58}$	$68.09_{\pm 6.22}$	$24.56_{\pm 7.35}$	$67.68_{\pm 6.49}$
		M	$42.21_{\pm 5.78}$	$55.42_{\pm 5.35}$	$42.24_{\pm 5.72}$	$55.38_{\pm 5.51}$	$24.81_{\pm 6.99}$	$67.93_{\pm 6.32}$	$24.76_{\pm 7.23}$	$67.87_{\pm 6.50}$
		S	$37.94_{\pm 5.15}$	$52.15_{\pm 5.28}$	$37.19_{\pm 5.10}$	$52.33_{\pm 5.38}$	$20.27_{\pm 6.98}$	$64.41_{\pm 6.30}$	$20.28_{\pm 6.71}$	$64.35_{\pm 5.99}$

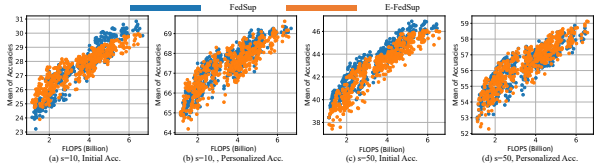


Figure 2. Attributes (e.g., initial acc., personalized acc., and FLOPS) of training process. Each dot represents a sub-model and 500 child models are sampled from the supernet.

erature (McMahan et al., 2017; Oh et al., 2021; Lin et al., 2020). Data is divided into the same-sized shards by considering its label distribution. Because there is no overlapping data between shards, the size of a shard is defined by $\frac{|D|}{N \times s}$, where $|D|$ is the data set size, N is the total number of clients, and s is the number of shards per user.

Training Details. To evaluate the personalized accuracy, each client has the same label distribution on local training and test data. Referred to (Oh et al., 2021), we control FL environments with following hyperparameters: client fraction ratio f, local epochs τ, shards per user s, and Dirichlet concentration parameter β. f is the number of participating clients out of the total number of clients in every round and a small f is natural in the FL settings because the total number of clients is numerous. For the FedSup training, we use the number M of randomly sampled child model as equal to 3 and appy the inplace distillation. "Big (B)", "Medium (M)", "Small (S)" indicate the amount of FLOPS of sub-model sampled from the supernet located in the Pareto-frontier. We calculate the accuracy of global model and local models following the federated personalization evaluation procedure proposed in (Wang et al., 2019) and (Oh et al., 2021). Details are provided in Appendix.

4.2. Evaluation on the Common FL Settings

Personalization. As mentioned in literature (Oh et al., 2021; Luo et al., 2021), it is shown that updating only head has slightly better performance than the others including local-only training (Table 1).

Compounding Dimensions. Table 2 describes the initial and personalized accuracies when combining the dimensions for architecture space. In most cases, FedSup has slightly less generalization error than E-FedSup in the global model (initial accuracy) while the personalized accuracy becomes almost similar when the five fine-tuning epochs are applied. Both FedSup and E-FedSup show a slight decrease in performance as a model size gets smaller.

Pareto Frontier. We compare the accuracy vs. FLOPS Pareto that finds the set of solutions where improving one objective will degrade another in the Multi-Objective Optimization (Eriksson et al., 2021). Here, we randomly sample 500 sub-models from the supernet and estimate their initial and personalized accuracies. As Figure 2 shows, FedSup has better Pareto frontier than E-FedSup for the initial accuracy while those for personalized accuracy are almost similar.

Inference Time. Figure 3 shows the results of the boxplot plotting sub-models' inference time comparing with other model heterogeneity methods FjORD (Horvath et al., 2021) and HeteroFL (Diao et al., 2021). We demonstrate that FedSup spawns more efficient models in terms of local inference time. Because FjORD and HeteroFL are unstructured pruning or channel pruning-based methods, ours have lower processing time benefitted from dynamic depth. We measure the inference time on the NVIDIA 2080-Ti GPU.

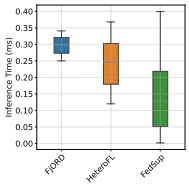


Figure 3. Comparison of other model heterogeneity methods about the inference time per image of each sub-model.

5. Conclusion

In this paper, we propose a novel branch of approaches, Federation of Supernet Training (FedSup) under system heterogeneity. Our work engages in the solutions of FL for both data-heterogeneity and model-heterogeneity. Specifically, FedSup aggregates a large number of sub-networks with different capabilities into one global model at once. In addition, we further develop an efficient version of FedSup (E-FedSup) which reduces the model size transferred per round as well as local training overhead.

Acknowledgement

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by Korea government (MSIT) [No. 2021-0-00907, Development of Adaptive and Lightweight Edge-Collaborative Analysis Technology for Enabling Proactively Immediate Response and Rapid Learning, 90%] and [No. 2019-0-00075, Artificial Intelligence Graduate School Program (KAIST), 10%].

References

Acar, D. A. E., Zhao, Y., Navarro, R. M., Mattina, M., Whatmough, P. N., and Saligrama, V. Federated learning based on dynamic regularization. In *International Conference on Learning Representations*, 2021.

Afonin, A. and Karimireddy, S. P. Towards model agnostic federated learning using knowledge distillation. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=lQI_mZjvBxj.

Bouacida, N., Hou, J., Zang, H., and Liu, X. Adaptive federated dropout: Improving communication efficiency and generalization for federated learning. *CoRR*, abs/2011.04050, 2020.

Briggs, C., Fan, Z., and Andras, P. Federated learning with hierarchical clustering of local updates to improve training on non-iid data. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–9. IEEE, 2020.

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

Cai, H., Gan, C., and Han, S. Once for all: Train one network and specialize it for efficient deployment. *CoRR*, abs/1908.09791, 2019.

Chen, H.-Y. and Chao, W.-L. On bridging generic and personalized federated learning for image classification. In *International Conference on Learning Representations*, 2021.

Cho, Y. J., Wang, J., and Joshi, G. Client selection in federated learning: Convergence analysis and power-of-choice selection strategies. *arXiv preprint arXiv:2010.01243*, 2020.

Diao, E., Ding, J., and Tarokh, V. Hetero{fl}: Computation and communication efficient federated learning for heterogeneous clients. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=TNkPBBYFkXg.

El-Sayed, H., Sankar, S., Prasad, M., Puthal, D., Gupta, A., Mohanty, M., and Lin, C.-T. Edge of things: The big picture on the integration of edge, iot and the cloud in a distributed computing environment. *IEEE Access*, 6: 1706–1717, 2018. doi: 10.1109/ACCESS.2017.2780087.

Eriksson, D., Chuang, P. I.-J., Daulton, S., Aly, A., Babu, A., Shrivastava, A., Xia, P., Zhao, S., Venkatesh, G., and Balandat, M. Latency-aware neural architecture search with multi-objective bayesian optimization. *arXiv preprint arXiv:2106.11890*, 2021.

Fallah, A., Mokhtari, A., and Ozdaglar, A. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. *Advances in Neural Information Processing Systems*, 33, 2020.

Felix, X. Y., Rawat, A. S., Menon, A. K., and Kumar, S. Federated learning with only positive labels. *arXiv preprint arXiv:2004.10342*, 2020.

Gao, Y., Bai, H., Jie, Z., Ma, J., Jia, K., and Liu, W. Mtl-nas: Task-agnostic neural architecture search towards general-purpose multi-task learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11543–11552, 2020.

Han, P., Wang, S., and Leung, K. K. Adaptive gradient sparsification for efficient federated learning: An online learning approach. *arXiv preprint arXiv:2001.04756*, 2020.

Han, Y., Huang, G., Song, S., Yang, L., Wang, H., and Wang, Y. Dynamic neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

Hinton, G., Vinyals, O., Dean, J., et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015.

Horvath, S., Laskaridis, S., Almeida, M., Leontiadis, I., Venieris, S., and Lane, N. Fjord: Fair and accurate federated learning under heterogeneous targets with ordered dropout. *Advances in Neural Information Processing Systems*, 34, 2021.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Hsu, T.-M. H., Qi, H., and Brown, M. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*, 2019.

Huang, G., Chen, D., Li, T., Wu, F., Van Der Maaten, L., and Weinberger, K. Q. Multi-scale dense networks for resource efficient image classification. *arXiv preprint arXiv:1703.09844*, 2017.

Huang, Y., Gupta, S., Song, Z., Li, K., and Arora, S. Evaluating gradient inversion attacks and defenses in federated learning. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=0CDKgyYaxC8.

Hyeon-Woo, N., Ye-Bin, M., and Oh, T.-H. Fedpara: Low-rank hadamard product for communication-efficient federated learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=d71n4ftoCBy.

Jiang, Y., Konečnỳ, J., Rush, K., and Kannan, S. Improving federated learning personalization via model agnostic meta learning. *arXiv preprint arXiv:1909.12488*, 2019.

Karimireddy, S. P., Kale, S., Mohri, M., Reddi, S. J., Stich, S. U., and Suresh, A. T. Scaffold: Stochastic controlled averaging for on-device federated learning. *arXiv preprint arXiv:1910.06378*, 2019.

Kather, J. N., Krisam, J., Charoentong, P., Luedde, T., Herpel, E., Weis, C.-A., Gaiser, T., Marx, A., Valous, N. A., Ferber, D., et al. Predicting survival from colorectal cancer histology slides using deep learning: A retrospective multicenter study. *PLoS medicine*, 16(1):e1002730, 2019.

Kim, T., Ko, J., Cho, S., Choi, J., and Yun, S.-Y. FINE samples for learning with noisy labels. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=QZpx42n0BWr.

Kornblith, S., Norouzi, M., Lee, H., and Hinton, G. Similarity of neural network representations revisited. In *International Conference on Machine Learning*, pp. 3519–3529. PMLR, 2019.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Kuen, J., Kong, X., Lin, Z., Wang, G., Yin, J., See, S., and Tan, Y.-P. Stochastic downsampling for cost-adjustable inference and improved regularization in convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7929–7938, 2018.

Li, Q., He, B., and Song, D. Model-contrastive federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10713–10722, 2021a.

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., and Smith, V. Federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127*, 2018.

Li, T., Sahu, A. K., Talwalkar, A., and Smith, V. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020.

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

Li, X., Huang, K., Yang, W., Wang, S., and Zhang, Z. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189*, 2019.

Liang, P. P., Liu, T., Ziyin, L., Allen, N. B., Auerbach, R. P., Brent, D., Salakhutdinov, R., and Morency, L.-P. Think locally, act globally: Federated learning with local and global representations. *arXiv preprint arXiv:2001.01523*, 2020.

Lim, W. Y. B., Luong, N. C., Hoang, D. T., Jiao, Y., Liang, Y.-C., Yang, Q., Niyato, D., and Miao, C. Federated learning in mobile edge networks: A comprehensive survey. *IEEE Communications Surveys Tutorials*, 22(3): 2031–2063, 2020. doi: 10.1109/COMST.2020.2986024.

Lin, J., Rao, Y., Lu, J., and Zhou, J. Runtime neural pruning. *Advances in neural information processing systems*, 30, 2017.

Lin, T., Kong, L., Stich, S. U., and Jaggi, M. Ensemble distillation for robust model fusion in federated learning. *arXiv preprint arXiv:2006.07242*, 2020.

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

Liu, L. and Deng, J. Dynamic deep neural networks: Optimizing accuracy-efficiency trade-offs by selective execution. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Luo, M., Chen, F., Li, Z., and Feng, J. Architecture personalization in resource-constrained federated learning.

Luo, M., Chen, F., Hu, D., Zhang, Y., Liang, J., and Feng, J. No fear of heterogeneity: Classifier calibration for federated learning with non-iid data. *Advances in Neural Information Processing Systems*, 34, 2021.

Mansour, Y., Mohri, M., Ro, J., and Suresh, A. T. Three approaches for personalization with applications to federated learning. *arXiv preprint arXiv:2002.10619*, 2020.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pp. 1273–1282, 2017.

Mohri, M., Sivek, G., and Suresh, A. T. Agnostic federated learning. *arXiv preprint arXiv:1902.00146*, 2019.

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

Nishio, T. and Yonetani, R. Client selection for federated learning with heterogeneous resources in mobile edge. In *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pp. 1–7. IEEE, 2019.

Oh, J., Kim, S., and Yun, S.-Y. Fedbabu: Towards enhanced representation for federated image classification. *arXiv preprint arXiv:2106.06042*, 2021.

Pasunuru, R. and Bansal, M. Continual and multi-task architecture search. *arXiv preprint arXiv:1906.05226*, 2019.

Sabour, S., Frosst, N., and Hinton, G. E. Dynamic routing between capsules. *Advances in neural information processing systems*, 30, 2017.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018.

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

Strubell, E., Ganesh, A., and McCallum, A. Energy and policy considerations for deep learning in nlp. *arXiv preprint arXiv:1906.02243*, 2019.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

T Dinh, C., Tran, N., and Nguyen, J. Personalized federated learning with moreau envelopes. *Advances in Neural Information Processing Systems*, 33:21394–21405, 2020.

Voigt, P. and Von dem Bussche, A. The eu general data protection regulation (gdpr). *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, 10(3152676): 10–5555, 2017.

Wang, D., Gong, C., Li, M., Liu, Q., and Chandra, V. Alphanet: Improved training of supernets with alpha-divergence. In *International Conference on Machine Learning*, pp. 10760–10771. PMLR, 2021a.

Wang, D., Li, M., Gong, C., and Chandra, V. Attentivenas: Improving neural architecture search via attentive sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6418–6427, 2021b.

Wang, J., Liu, Q., Liang, H., Joshi, G., and Poor, H. V. Tackling the objective inconsistency problem in heterogeneous federated optimization. *arXiv preprint arXiv:2007.07481*, 2020.

Wang, K., Mathews, R., Kiddon, C., Eichner, H., Beaufays, F., and Ramage, D. Federated evaluation of on-device personalization. *arXiv preprint arXiv:1910.10252*, 2019.

Wang, X., Yu, F., Dou, Z.-Y., Darrell, T., and Gonzalez, J. E. Skipnet: Learning dynamic routing in convolutional networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 409–424, 2018.

Wu, C., Wu, F., Lyu, L., Huang, Y., and Xie, X. Communication-efficient federated learning via knowledge distillation. *Nature communications*, 13(1):1–8, 2022.

Wu, Z., Nagarajan, T., Kumar, A., Rennie, S., Davis, L. S., Grauman, K., and Feris, R. Blockdrop: Dynamic inference paths in residual networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8817–8826, 2018.

Xu, H., Kostopoulou, K., Dutta, A., Li, X., Ntoulas, A., and Kalnis, P. Deepreduce: A sparse-tensor communication framework for federated deep learning. *Advances in Neural Information Processing Systems*, 34:21150–21163, 2021.

Yang, B., Bender, G., Le, Q. V., and Ngiam, J. Condconv: Conditionally parameterized convolutions for efficient inference. *Advances in Neural Information Processing Systems*, 32, 2019.

Yang, J., Shi, R., Wei, D., Liu, Z., Zhao, L., Ke, B., Pfister, H., and Ni, B. Medmnist v2: A large-scale lightweight benchmark for 2d and 3d biomedical image classification. *arXiv preprint arXiv:2110.14795*, 2021.

Yu, J. and Huang, T. S. Universally slimmable networks and improved training techniques. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1803–1811, 2019.

Yu, J., Yang, L., Xu, N., Yang, J., and Huang, T. S. Slimmable neural networks. *CoRR*, abs/1812.08928, 2018.

Yu, J., Jin, P., Liu, H., Bender, G., Kindermans, P.-J., Tan, M., Huang, T., Song, X., Pang, R., and Le, Q. Bignas:

Scaling up neural architecture search with big single-stage models. In *European Conference on Computer Vision*, pp. 702–717. Springer, 2020.

Yuan, H. and Ma, T. Federated accelerated stochastic gradient descent. *arXiv preprint arXiv:2006.08950*, 2020.

Yurochkin, M., Agarwal, M., Ghosh, S., Greenewald, K., Hoang, T. N., and Khazaeni, Y. Bayesian nonparametric federated learning of neural networks. *arXiv preprint arXiv:1905.12022*, 2019.

Zhang, L., Shen, L., Ding, L., Tao, D., and Duan, L.-Y. Fine-tuning global model via data-free knowledge distillation for non-iid federated learning. *arXiv preprint arXiv:2203.09249*, 2022.

Zoph, B., Vasudevan, V., Shlens, J., and Le, Q. V. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8697–8710, 2018.

A. Overview of Appendix

In this supplementary material, we present additional details, results, and experiments that are not included in the main paper due to the space limit.

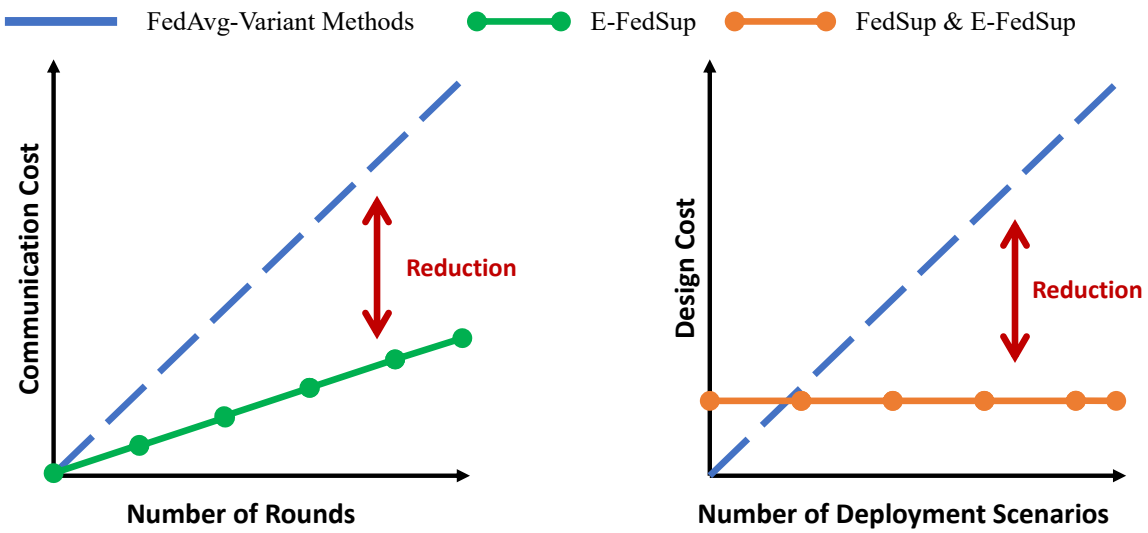
B. Ethics Statement

To address potential concerns, we describe the ethical aspect in respects to privacy, security, infrastructure level gap, and energy consumption.

Privacy and Security. Despite the promise of FL, owing to the presence of malicious users or the stragglers in the network, some workers may disturb the protocols and send arbitrary/adversarial messages that disturbs the generalization during FL. Recently, to tackle the system heterogeneity, some works allows the server to use proxy data or transmit encrypted data from local to server, but it may infringes on privacy. FedSup is also able to have such potential risks during communication. However, because FedSup can enable the training of models under heterogeneous system without using any proxy dataset, our methods could be uses as a general solution to personalize the model, having less risks of privacy and security under system heterogeneity. Under adversarial attacks, it would be a nice direction to investigate the defense methods regarding the robustness against such adversarial risks.

Infrastructure Level Gap. In real-world applications, there is a bandwidth issues between clients and the server. More precisely, because of some limited-service access to areas where communication is rarely possible. Sending a model of the same size can greatly affect the synchronize training of FL with such infrastructure level gap. Because our work is efficient in terms of communication cost, we can deploy the model resource-adaptively. In addition, it is possible to use the model adaptively enough within the local according to the model resource and situation.

Energy Consumption. Our methods is more efficient than other methods in the respect of energy consumption: (1) communication efficiency and (2) design costs. Firstly, if E-FedSup is used, the sub-model is transferred to local as a substitute for the full supernet. Therefere, noticeable energy-saving effects can be obtained. On the other side, since our methods can design various architectures rather than specialized neural networks, our approach reduces the cost of specialized deep learning deployment from $O(N)$ to $O(1)$ (Cai et al., 2019). Even, our methods has less generalization errors than other FedAvg-variant methods while total communication costs are the same, so further energy-savings can happen in the respect of convergence speed.



(a) Communication Efficiency (b) Design Efficiency

Figure 4. Savings of energy consumption in the respects to communication costs and design costs.

C. Limitations and Future Directions

In this sections, we describe the limitations of our work and future directions for further development.

Limitations. Although illustrating the superiority of our proposed methods over state of the art, the bottleneck lies in the presence of arbitrary device unavailability or adversarial clients that disturbs the training. We only consider vision-centric classification tasks on smaller datasets (CIFAR-10, CIFAR-100, CINIC, PathMNIST). We do not investigate a large-scale datasets (namely ImageNet); FL framework gets computationally more prohibitive as the number of clients and local training iterations are increasing.

Future Directions. In future work, we aim to explore more efficient training strategies in the presence of stragglers and adversarial users. Furthermore, we improve the robustness of FedSup families in more resource-intensive settings. We intend to investigate our methods on other applications such as object-detection, semantic segmentation, and natural language processing models. Lastly, we plan to explore why FedSup and E-FedSup has similar personalization accuracies while the global accuracy has slight gap between FedSup and E-FedSup.

D. Conceptual Comparison to Prior Works

Neural architecture search (NAS) studies have also suffered from such model heterogeneity issues in deploying resource-adaptive models to clients, but have resolved this challenge by training a single set of shared weight from one-shot models (Cai et al., 2019; Yu et al., 2020; Wang et al., 2021b) (Figure 1 (b)). However, it has been under-explored under data-heterogeneity scenarios that can provoke the training instability. Recently, some works have studied the model heterogeneity in FL by sampling/generating a sub-network (Mushtaq et al., 2021; Diao et al., 2021; Shamsian et al., 2021) or using a pruned model from a global model (Horvath et al., 2021; Luo et al.). Such methods have limitations on model scaling (e.g., depth (#layers), width (#channels), kernel size), training stability, and personalization of each clients.

Designing specialized DNNs for every scenario is labor-intensive and computationally prohibitive with human-based methods or sample-based NAS (Liu et al., 2018). Such methods require repeating the network design process and replacing the network design process from scratch in each case, and therefore their total cost increases linearly as the number of deployment scenarios increases. Furthermore, in the presence of a federated environment, such design cost is also significantly affected by the number of participating clients and the amount of each local data, even considering the network bandwidth of clients. To mitigate such issues, some works attempt to examine the single-shot model under data heterogeneity. We do not consider the works that assumed the availability of proxy data in the server (Lin et al., 2020; Zhang et al., 2022).

Recently, numerous studies have been studied to address the problem of either data heterogeneity or model heterogeneity. As discussed in Section 2, literature can be categorized into several groups with FedSup and E-FedSup: data heterogeneity, model heterogeneity, and their hybrid. Table 3 systematically compares related methods in the respect to flexibility, data-privacy, efficiency on design, and efficiency on communication. More detailed explanations are described in Section 2. To the best of our knowledge, our methods are the first method to satisfy the two conflicting factors: *compounding model scales* (depth, width, kernel size) and *personalized models*, by taking advantage of both categories.

Table 3. Comparison with related training methods: each method is grouped into three categories. In the first row, "Flexibility": need not be tied with a specific architecture; "Data Privacy": keep the data privacy on each client; "Efficiency on Design": can design the architecture efficiently; "Efficiency on Communication": can reduce the communication cost between clients and the server.

Category	Data Heterogeneity			Model Heterogeneity		Hybrid
Method	FedAvg [2017]	AFD [2020]	FedDF [2020]	OFA [2019]	BigNAS [2020]	Ours
Flexibility	X	X	O	O	O	O
Data-Privacy	O	O	O	X	X	O
Efficieny on Design	X	X	X	O	O	O
Efficiency on Communication	X	O	X	X	X	O

Dynamic Neural Network. Compared to static neural networks, dynamic neural networks can adapt their structures or parameters to the input during inference considering the quality-cost trade-off (Han et al., 2021). To adaptively allocate computations on demand at inference, some works selectively activate model components (e.g., layers (Huang et al., 2017),

channels (Lin et al., 2017; Sabour et al., 2017)); a controller or gating modules are learned to dynamically choose which layers of a deep network (Wu et al., 2018; Liu & Deng, 2018; Wang et al., 2018); Kuen et al. (Kuen et al., 2018) introduce stochastic downsampling points to adaptively reduce the size of the feature map. By extending the capabilities of already-developed human-designed neural networks like the MobileNet series (Howard et al., 2017; Sandler et al., 2018), Slimmable nets (Yu et al., 2018; Yu & Huang, 2019) train a model to support multiple width multipliers (for instance, 4 different global width multipliers).

Supernet. Supernet, a category of dynamic neural networks, assembles all candidate architectures into a weight-sharing network where each architecture corresponds to one sub-network. It is an emerging research topic in deep learning, specifically neural architecture search. It dramatically reduces the huge cost of searching, training, or fine-tuning each architecture individually whose child models can be directly deployed. Despite its strength, supernet training is highly challenging (Yu et al., 2018; Yu & Huang, 2019). For the stability of training optimization, it requires many training techniques such as (1) inplace knowledge distillation (Yu & Huang, 2019) leveraging the soft prediction of the largest sub-network for the supervision of other sub-networks, (2) modified batch normalization to synchronize the batch statistics of all child models (Yu et al., 2018; Yu & Huang, 2019), (3) sampling strategy of child models from the supernet (Cai et al., 2019; Wang et al., 2021b), and (4) modified loss/gradient function (Yu et al., 2020; Wang et al., 2021a).

Benefits of Supernet. Many applications present the use cases of supernet for real-world scenarios. One of the most notable advantages is that they are able to allocate the user-customized network in consideration of their capabilities on edge devices (e.g., smartphones, the internet of things) (Cai et al., 2018). Next, the supernet seemingly produces better representation power than the static version of the network (Yang et al., 2019; Cai et al., 2019). In addition, supernet alleviates the issue of excessive energy consumption and CO_2 emission caused by designing specialized DNNs for every scenario (Strubell et al., 2019; Cai et al., 2019). Lastly, supernet has superior transferability across different datasets (Zoph et al., 2018) and tasks (Pasunuru & Bansal, 2019; Gao et al., 2020). All these advantages seem like a double line that will work well in a federated environment, to our best knowledge, but there are few studies applied in FL yet. Recently, Diao et al. (Diao et al., 2021) show the possibility of coordinatively training local models by using a weight-sharing concept while it limits the degree of flexibility (e.g., only width multiplier can adapt), analysis of model behavior, the examination for a collection of training refinements, and the investigation towards personalization.

Personalized FL. Machine learning-based personalization has emerged for keeping privacy and fairness as well as recognizing the local particular character. Personalized federated learning has been proposed as one of them to learn personalized local models. To improve the performance, the methods for the personalized FL models have been evolving in such directions (T Dinh et al., 2020; Mansour et al., 2020; Oh et al., 2021): user clustering, designing new loss functions, meta-learning, and model interpolation. With meta-features from all clients, each local client is clustered by measuring the data distribution and sharing separate models for each cluster without inter-cluster federation (Briggs et al., 2020; Mansour et al., 2020). Adding a regularizer to the loss function can be a recommended scheme to prevent local models from overfitting their own local data (T Dinh et al., 2020; Li et al., 2021b). Bi-level optimization between clients and servers can be interpreted as *meta-learning* (i.e., Model-Agnostic Meta-Learning (MAML)). This approach aims to obtain a well-initialized shared global model that facilitates personalized generalization with a few fine-tuning (Jiang et al., 2019; Oh et al., 2021). Lastly, decoupling the base and personalized layers in a network is used; both types of layers are trained by clients in addition to the server's base layers to create a model that is unique to each user (Oh et al., 2021; Chen & Chao, 2021). On the other hand, few studies have been on personalized FL performance under the client system heterogeneity, which denotes the clients with different computational capabilities.

Until this work, considerable efforts have been devoted to solving the efficiency problems of client system heterogeneity, which can be categorized into three dimensions as shown in Figure 5. We aim to design a robust framework to combine the federated averaging scheme with weight sharing on mobile-friendly architectures. We attempt to apply several training techniques to bridge the gap among three dimensions (Figure 5).

E. Implementation Details for section 4

We build our methods and reproduce all experimental results referring to other official repositories [1, 2, 3].

[1] https://github.com/facebookresearch/AttentiveNAS
[2] https://github.com/jhoon-oh/FedBABU
[3] https://github.com/pliang279/LG-FedAvg

Table 4. MobileNetV1-based search space.

Stage	Operator	Resolution	#Channels	#Layers	Kernel Sizes
	Conv	32x32	32	1	3
1	MBConv	16x16	32-64	1-1	3,5,7
2	MBConv	16x16	64-128	1-2	3,5,7
3	MBConv	8x8	128-256	1-2	3,5,7
4	MBConv	4x4	256-512	3-6	3,5,7
5	MBConv	2x2	512-1024	1-2	3,5,7

E.1. Architectural Space

In this section, we present the details of our search space. Our network architectures consist of a stack with MobilenetV1 blocks (MBConv) (Howard et al., 2017). The detailed search space is summarized in Table 4. For the depth dimension, our network has five stages (excluding the first convolutional layer (also called Stem)).Each stage has multiple choices of the number of layers, the number of channels and kernel size.

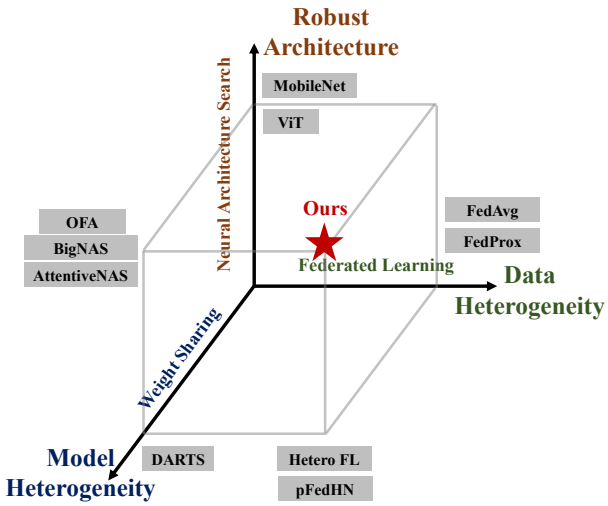


Figure 5. Illustration for the problem settings.

E.2. Static Batch Normalizaiton.

We follow the batch normalization settings used in the previous works (Yu et al., 2018; Diao et al., 2021). Specifically, because the running statistics of batch normalization layers can not be accumulated during training owing to the violence of data privacy as well as different model size (Huang et al., 2021), track running statistics are not tracked and simply normalized through the batch data. For the evaluation, BN statistics are updated as the local data is sequentially queried.

E.3. Experimental Settings

Data Preprocessing. We use the same settings in (Oh et al., 2021). We apply normalization and simple data augmentation techniques (random crop and horizontal flip) on the training sets of all datasets. The size of the random crop is set to 32 for all datasets referred to previous works (Oh et al., 2021; Liang et al., 2020; McMahan et al., 2017).

Evaluation. We analyze the algorithms at the client level: (1) the learned global model is broadcast to all clients and is then evaluated on the test data set of each client D_i^{ts} (referred to as the *initial accuracy*), (2) the learned global model is personalized using the training data set of each client D_i^{tr} by fine-tuning with the fine-tuning epochs of τ_f ; the personalized models are then evaluated on the test data set of each client D_i^{ts} (referred to as the *personalized accuracy*). The values ($X_{\pm Y}$) in all tables indicate the mean$_{\pm \text{std}}$ of the accuracies across all clients, not across multiple seeds.

Dirichlet Distribution. To simulate a wide range of non-IIDness, we designed representative heterogeneity settings based on widely used techniques (Yurochkin et al., 2019). A dataset is partitioned by following $\mathbf{p}_c \sim Dir_N(\beta \cdot \vec{1})$ that involves allocating $p_{k,c}$ proportion of data examples for class c to client k where $\vec{1}$ is the vector of ones.

CIFAR-10. CIFAR-10 (Krizhevsky et al., 2009) is the popular classification benchmark dataset. CIFAR-10 consists of 32 × 32 resolution images in 10 classes, with 6,000 images per class. We use 50,000 images for training and 10,000 images for testing.

CIFAR-100. CIFAR-100 (Krizhevsky et al., 2009) is the popular classification benchmark dataset. CIFAR-100 consists of 32×32 resolution images in 100 classes, with 6,000 images per class. We use 50,000 images for training and 10,000 images for testing.

PathMNIST. PathMNIST (Kather et al., 2019) is a collection of 10 pre-processed medical open datasets. It is standaridzed to perform classification tasks on light weight 28 x 28 images, which requires no background knowledge, while we apply the image size as 32 x 32. PathMNIST has 9 classes and three subsets: training, validation, and test. Each has 89,996 data whose label distribution is near balanced, but unbalanced, and we do not use the validation subset for training. Figure 6 shows several images from the training dataset.

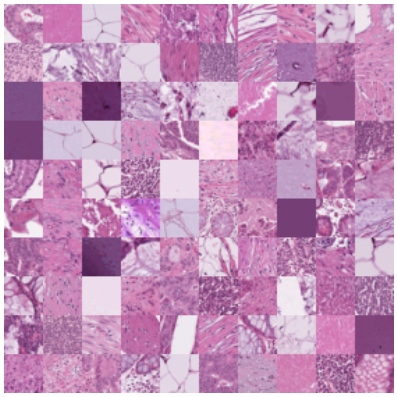
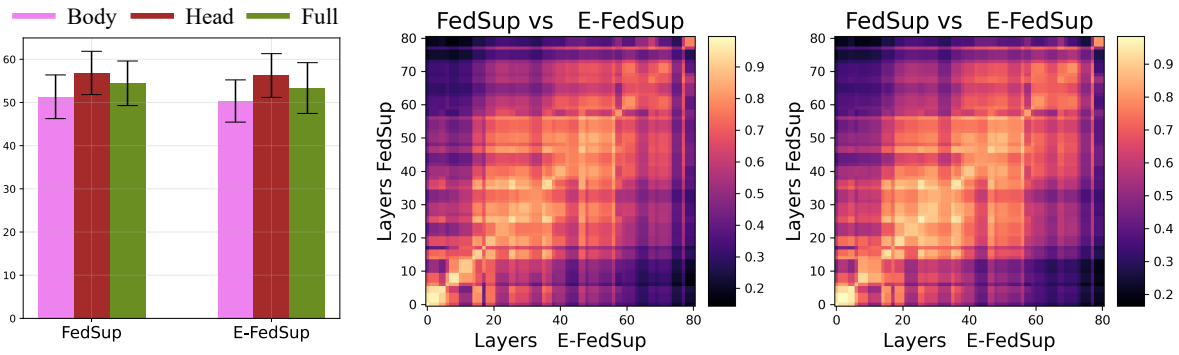


Figure 6. PathMNIST Images.



(a) Updates for personalization (b) CKA, s=10 (c) CKA, s=50

Figure 7. (a) Personalization accuracy of FedSup and E-FedSup on CIFAR-100 according to the fine-tuned part by referring to (Oh et al., 2021) (Other parts are freezed); (b), (c): Centered Kernel Alignments (CKA) similarities of two different global models trained with FedSup and E-FedSup (Kornblith et al., 2019).

Specification. We describe the detailed specification regarding 'Big', 'Medium', 'Small' models. Deservedly, other medium-size models also are able to be sampled from the supernet while the trade-off between resources and accuracies happens (Figure 2).

Table 5. Specification for the child models sampled from the supernet. We report inference time in milliseconds, model size in million (M) units, and FLOPS in billions (B) units of parameters.

Child Model	Big (B)	Medium (M)	Small (S)
Inference Time	0.37 (ms)	0.20 (ms)	0.06 (ms)
Model Size	0.40 (M)	2.47 (M)	3.08 (M)
FLOPS	2.07 (B)	6.01 (B)	13.36 (B)

F. Additional Experimental Results

F.1. Additional Results on the Common FL Settings

Personalization. Referring to recent personalized FL experimental settings (Oh et al., 2021), we compare the performance according to the fine-tuned part (Figure 7 (a)). Child models are fine-tuned with five epochs based on the local training data. In the paper, we thus updates only the head for the personalization unless otherwise mentioned.

CKA Similarities (Kornblith et al., 2019). We vividly compare how the representations of neural networks are changed through the FedSup and E-FedSup. To be specific, Centered Kernel Alignment (CKA) is leveraged to analyze the features learned by two architectures trained with FedSup and E-FedSup under different heterogeneous settings, given the same input testing samples (Figure 7 (b) and (c)). Regardless of the degree of heterogeneity, CKA visualizations show that the representations of two neural networks trained with FedSup and E-FedSup seem similar during the propagation.

F.1.1. Ablation Studies: Momentum and Label Smoothing

Momentum. Table 7 describes the initial and personalzied accuracy according to the momentum. The momentum is not applied during the fine-tuning of personalization. In most cases, appropriate momentum improves the performance.

Label Smoothing (LS) (Szegedy et al., 2016). Table 8 describes the initial and personalized accuracy according to the LS. LS is popularly used in the existing weight-sharing methods (Cai et al., 2019; Yu et al., 2020; Wang et al., 2021b), but in our environment, it rather degrades both initial and personalized performance.

F.1.2. Experiments on Medical Dataset

As Table 9 shows, FedSup and E-FedSup work fairly well on the PathMNIST dataset and have the similar tendency presented in Table 2.

F.2. Global Accuracies

Unlike the main section, we evaluate a global accuracy of each server model with original test dataset (Table 10, Table 11, Table 12, Table 13, Table 14, Table 15).

Table 6. Additional results: initial and personalized accuracy on CIFAR100 under various FL settings with 100 clients. The initial and personalized accuracy indicate the evaluated performance without fine-tuning and after five fine-tuning epochs for each client, respectively.

FL Settings			s=50				s=10			
f	τ	A	FedSup		E-FedSup		FedSup		E-FedSup	
			Initial	Personalized	Initial	Personalized	Initial	Personalized	Initial	Personalized
1.0	1	B	$42.83_{\pm 5.05}$	$55.03_{\pm 4.95}$	$42.46_{\pm 5.60}$	$55.95_{\pm 6.03}$	$25.96_{\pm 6.47}$	$65.75_{\pm 6.05}$	$26.33_{\pm 6.37}$	$66.44_{\pm 6.83}$
		M	$41.39_{\pm 5.33}$	$55.33_{\pm 4.53}$	$42.15_{\pm 5.57}$	$55.91_{\pm 5.57}$	$26.04_{\pm 6.28}$	$65.59_{\pm 6.00}$	$26.50_{\pm 6.70}$	$66.50_{\pm 7.01}$
		S	$39.19_{\pm 4.77}$	$53.17_{\pm 4.77}$	$39.78_{\pm 5.29}$	$54.35_{\pm 5.88}$	$25.06_{\pm 5.94}$	$64.81_{\pm 6.12}$	$25.20_{\pm 6.00}$	$64.53_{\pm 6.52}$
	5	B	$47.08_{\pm 5.14}$	$58.15_{\pm 6.14}$	$46.18_{\pm 5.68}$	$58.04_{\pm 5.62}$	$31.24_{\pm 5.66}$	$69.42_{\pm 6.69}$	$29.77_{\pm 6.22}$	$69.47_{\pm 6.35}$
		M	$43.34_{\pm 4.89}$	$57.01_{\pm 5.36}$	$44.13_{\pm 5.51}$	$57.25_{\pm 6.17}$	$28.81_{\pm 6.14}$	$69.37_{\pm 5.39}$	$30.22_{\pm 6.31}$	$69.38_{\pm 6.09}$
		S	$40.33_{\pm 5.02}$	$52.78_{\pm 5.66}$	$40.22_{\pm 5.28}$	$53.24_{\pm 5.42}$	$25.01_{\pm 5.11}$	$66.49_{\pm 6.36}$	$26.41_{\pm 5.97}$	$66.30_{\pm 6.34}$

Table 7. Initial and personalized accuracy of FedSup and E-FedSup on CIFAR-100 according to the change of the momentum magnitude. The fine-tuning epochs is 5, f is 0.1, N is 100, and s is 10.

Settings		B		M		S	
Alg.	m	Initial	Personalized	Initial	Personalized	Initial	Personalized
FedSup	0.0	$31.76_{\pm 7.07}$	$68.92_{\pm 6.18}$	$31.73_{\pm 7.54}$	$68.43_{\pm 6.26}$	$31.24_{\pm 7.47}$	$67.39_{\pm 7.28}$
	0.1	$32.52_{\pm 6.45}$	$68.03_{\pm 5.95}$	$32.09_{\pm 6.94}$	$68.51_{\pm 5.96}$	$31.41_{\pm 6.62}$	$66.92_{\pm 5.91}$
	0.5	$34.54_{\pm 7.04}$	$69.98_{\pm 6.89}$	$34.40_{\pm 6.29}$	$70.83_{\pm 6.11}$	$33.40_{\pm 6.25}$	$68.34_{\pm 6.89}$
E-FedSup	0.0	$32.67_{\pm 6.80}$	$67.61_{\pm 6.81}$	$32.69_{\pm 6.80}$	$67.74_{\pm 7.05}$	$32.19_{\pm 6.62}$	$66.30_{\pm 7.22}$
	0.1	$32.78_{\pm 6.88}$	$67.69_{\pm 6.19}$	$32.93_{\pm 7.05}$	$67.33_{\pm 6.28}$	$32.23_{\pm 6.64}$	$66.02_{\pm 6.65}$
	0.5	$32.49_{\pm 6.10}$	$68.50_{\pm 7.38}$	$32.79_{\pm 6.30}$	$68.77_{\pm 6.83}$	$31.96_{\pm 6.12}$	$66.81_{\pm 6.88}$

Table 8. Initial and personalized accuracy of FedSup and E-FedSup on CIFAR-100 with and without label smoothing. The fine-tuning epochs is 5, f is 0.1, N is 100, and s is 10.

Architecture Size		B		M		S	
Architecture	LS	Initial	Personalized	Initial	Personalized	Initial	Personalized
FedSup	0.0	$34.54_{\pm 7.04}$	$69.98_{\pm 6.89}$	$34.40_{\pm 6.29}$	$70.83_{\pm 6.11}$	$33.40_{\pm 6.25}$	$68.34_{\pm 6.89}$
	0.1	$31.44_{\pm 6.89}$	$68.92_{\pm 7.48}$	$31.34_{\pm 7.03}$	$68.56_{\pm 7.25}$	$30.81_{\pm 7.28}$	$66.90_{\pm 7.60}$
E-FedSup	0.0	$32.49_{\pm 6.10}$	$68.50_{\pm 7.38}$	$32.79_{\pm 6.30}$	$68.77_{\pm 6.83}$	$31.96_{\pm 6.12}$	$66.81_{\pm 6.88}$
	0.1	$31.89_{\pm 6.98}$	$68.71_{\pm 6.45}$	$31.70_{\pm 6.48}$	$68.35_{\pm 6.88}$	$30.64_{\pm 6.91}$	$65.77_{\pm 6.71}$

Table 9. Initial and personalized accuracy on PathMNIST (Yang et al., 2021) under various FL settings with 100 clients. We implement data heterogeneity through Dirichlet distribution (β) (Yurochkin et al., 2019; Hsu et al., 2019; Lin et al., 2020). FedAvg algorithm has 2-3% lower initial and personalzied acc. on average than E-FedSup (Appendix).

FL Settings		$\beta = 100.0$				$\beta = 1.0$			
f	A	FedSup		E-FedSup		FedSup		E-FedSup	
		Initial	Personalized	Initial	Personalized	Initial	Personalized	Initial	Personalized
1.0	B	$75.02_{\pm 4.95}$	$74.67_{\pm 4.56}$	$73.04_{\pm 4.39}$	$73.56_{\pm 4.74}$	$71.70_{\pm 8.01}$	$79.67_{\pm 6.34}$	$70.33_{\pm 8.10}$	$79.17_{\pm 6.62}$
	M	$74.33_{\pm 4.45}$	$74.33_{\pm 4.40}$	$74.30_{\pm 4.47}$	$73.69_{\pm 4.66}$	$70.19_{\pm 7.91}$	$78.63_{\pm 6.84}$	$69.40_{\pm 8.43}$	$78.48_{\pm 7.33}$
	S	$74.03_{\pm 4.41}$	$73.12_{\pm 4.54}$	$70.66_{\pm 5.50}$	$70.00_{\pm 5.60}$	$68.59_{\pm 8.17}$	$77.60_{\pm 7.17}$	$66.95_{\pm 8.14}$	$76.38_{\pm 7.65}$
0.1	B	$74.76_{\pm 4.23}$	$74.38_{\pm 4.20}$	$73.91_{\pm 4.87}$	$73.22_{\pm 4.95}$	$70.07_{\pm 8.65}$	$79.30_{\pm 7.29}$	$69.40_{\pm 8.05}$	$79.08_{\pm 6.74}$
	M	$73.97_{\pm 4.90}$	$73.48_{\pm 4.54}$	$74.08_{\pm 4.91}$	$73.26_{\pm 4.54}$	$69.46_{\pm 9.23}$	$78.80_{\pm 6.33}$	$69.13_{\pm 9.07}$	$78.76_{\pm 6.24}$
	S	$73.23_{\pm 4.84}$	$71.79_{\pm 4.63}$	$73.88_{\pm 4.44}$	$72.97_{\pm 4.94}$	$68.05_{\pm 8.77}$	$77.37_{\pm 7.63}$	$67.27_{\pm 8.97}$	$76.99_{\pm 7.58}$

Table 10. Performance of FedSup on CIFAR-10 test dataset with supernet having only dynamic depth. (last accuracy / best accuracy) is written in order ($f = 0.1, N = 100$).

		Dirichlet		Shard	
τ	m	$\beta = 0.01$	$\beta = 1.0$	s=2	s=10
1	0.0	21.28 / 29.05	71.04 / 71.56	48.53 / 52.70	69.10 / 71.59
	0.1	28.96 / 32.08	70.07 / 72.41	52.57 / 55.84	72.47 / 73.24
	0.5	23.47 / 31.62	70.89 / 74.13	49.54 / 60.09	75.18 / 76.19
5	0.0	33.59 / 42.82	75.61 / 76.80	54.22 / 62.78	76.57 / 77.79
	0.1	39.59 / 42.94	75.05 / 76.02	47.26 / 61.67	77.43 / 78.35
	0.5	39.76 / 45.44	75.92 / 77.18	51.94 / 62.30	74.03 / 77.21

Table 11. Performance of FedSup on CIFAR-10 test dataset with supernet having only dynamic kernel. (last accuracy / best accuracy) is written in order ($f = 0.1, N = 100$).

		Dirichlet		Shard	
τ	m	$\beta = 0.01$	$\beta = 1.0$	s=2	s=10
1	0.0	20.57 / 31.77	76.69 / 78.88	36.84 / 55.48	77.20 / 78.84
	0.1	10.78 / 30.05	77.50 /78.78	54.46 / 58.06	78.29 / 78.29
	0.5	18.94 / 34.25	77.83 / 79.99	42.34 / 56.48	78.26 / 80.46
5	0.0	42.79 / 48.76	80.76 / 81.57	35.77 / 62.16	82.57 / 83.06
	0.1	18.89 / 44.96	79.85 / 80.88	45.81 / 63.21	80.96 / 82.84
	0.5	29.59 / 51.12	81.52 / 82.35	48.79 / 59.92	80.00 / 82.76

Table 12. Performance of FedSup on CIFAR-10 test dataset with supernet having only dynamic width. (last accuracy / best accuracy) is written in order ($f = 0.1, N = 100$).

		Dirichlet		Shard	
τ	m	$\beta = 0.01$	$\beta = 1.0$	s=2	s=10
1	0.0	24.72 / 31.07	73.67 / 75.61	38.99 / 50.04	75.56 / 76.44
	0.1	27.30 / 33.77	74.85 / 76.34	38.95 / 54.33	75.22 / 76.04
	0.5	26.04 / 30.24	77.65 / 77.85	33.66 / 50.03	77.89 / 78.12
5	0.0	32.98 / 40.88	78.48 / 79.34	51.84 / 57.85	80.03 / 80.14
	0.1	31.39 / 42.89	78.29 / 78.49	43.28 / 56.86	78.48 / 80.10
	0.5	32.53 / 43.01	79.56 / 79.82	51.06 / 59.33	77.86 / 80.37

Table 13. Performance of E-FedSup on CIFAR-10 test dataset with supernet having only dynamic depth. (last accuracy / best accuracy) is written in order ($f = 0.1, N = 100$).

		Dirichlet		Shard	
τ	m	$\beta = 0.01$	$\beta = 1.0$	s=2	s=10
1	0.0	25.76 / 31.89	71.16 / 72.80	37.18 / 53.33	71.38 / 72.59
	0.1	25.84 / 35.11	73.17 / 73.21	42.01 / 53.11	73.13 / 73.27
	0.5	17.75 / 32.58	75.26 / 75.73	52.95 / 57.48	74.19 / 76.03
5	0.0	34.85 / 41.96	79.28 / 79.59	51.94 / 63.70	80.37 / 81.01
	0.1	28.96 / 40.77	80.19 / 80.19	60.48 / 65.21	81.41 / 81.60
	0.5	24.34 / 43.42	80.53 / 81.10	40.22 / 62.49	81.51 / 81.78

Table 14. Performance of E-FedSup on CIFAR-10 test dataset with supernet having only dynamic kernel. (last accuracy / best accuracy) is written in order ($f = 0.1, N = 100$).

		Dirichlet		Shard	
τ	m	$\beta = 0.01$	$\beta = 1.0$	s=2	s=10
1	0.0	22.08 / 31.71	75.79 / 76.97	46.71 / 54.07	74.68 / 77.30
	0.1	24.20 / 32.36	76.66 / 76.93	43.80 / 57.80	75.22 / 77.35
	0.5	26.75 / 34.95	77.94 / 78.59	32.61 / 54.86	79.28 / 79.28
5	0.0	38.27 / 42.90	79.56 / 80.75	46.11 / 60.77	81.33 / 82.54
	0.1	24.08 / 43.91	79.51 / 80.78	54.85 / 61.71	81.16 / 82.32
	0.5	31.97 / 50.82	80.74 / 81.93	21.26 / 43.72	82.01 / 82.61

Table 15. Performance of E-FedSup on CIFAR-10 test dataset with supernet having only dynamic width. (last accuracy / best accuracy) is written in order ($f = 0.1, N = 100$).

		Dirichlet		Shard	
τ	m	$\beta = 0.01$	$\beta = 1.0$	s=2	s=10
1	0.0	17.70 / 27.93	72.20 / 72.63	39.45 / 48.52	73.93 / 73.93
	0.1	20.07 / 30.38	74.64 / 74.73	36.08 / 51.81	72.99 / 74.30
	0.5	16.68 / 30.32	77.90 / 77.90	36.63 / 51.81	75.71 / 76.29
5	0.0	32.18 / 38.54	78.42 / 80.15	43.34 / 59.88	79.27 / 80.83
	0.1	23.56 / 39.97	78.35 / 78.86	37.13 / 59.98	80.81 / 80.81
	0.5	32.80 / 38.32	79.84 / 80.29	51.40 / 59.92	80.35 / 81.16

F.3. Learning Curve of Global Accuracy

We visualize the learning curves of the networks trained with FedSup and E-FedSup (Figure 8). As Figure 8 shows, FedSup has slightly better performance than E-FedSup. Here, the cosine learning rate scheduler is used, and the detailed explanations are noted in Section 4.

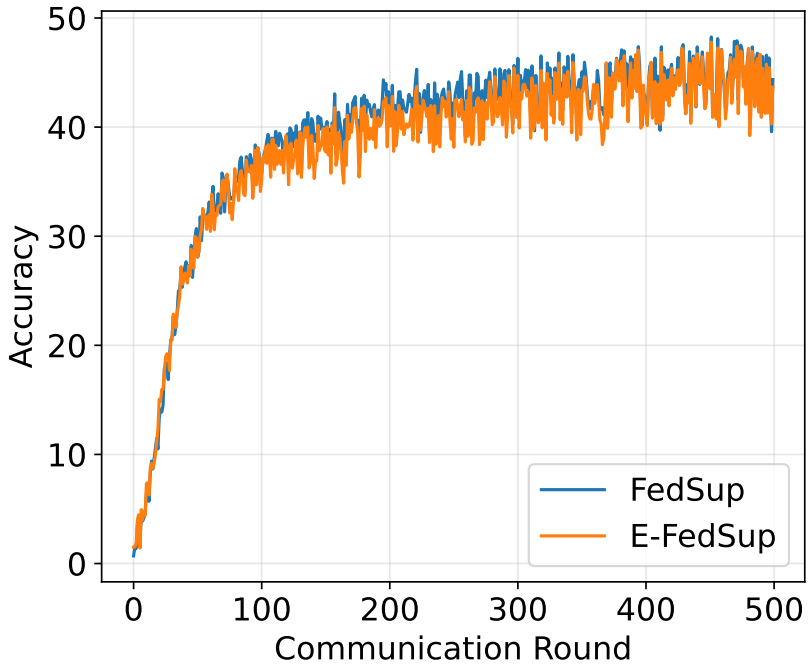


Figure 8. Learning curve of the networks trained with FedSup and E-FedSup. Both networks are trained with $s = 50, \tau = 5, f = 0.1, N = 100$.

F.4. PathMNIST Results

As mentioned in Table 9, FedSup and E-FedSup works better than FedAvg algorithm. Most performances in Table 16 are lower than the values in Table 16.

Table 16. FedAvg performance on PathMNIST ($N = 100, \tau = 5$).

		$\beta = 100.0$		$\beta = 1.0$	
f	m	Initial	Personalized	Initial	Personalized
1.0	0.0	$72.21_{\pm 4.67}$	$72.09_{\pm 4.30}$	$69.95_{\pm 6.94}$	$77.00_{\pm 6.26}$
0.1	0.0	$71.15_{\pm 4.43}$	$72.12_{\pm 4.69}$	$67.87_{\pm 6.78}$	$76.77_{\pm 7.34}$

F.5. Inplace Distillation: Representation Divergence

Table 17 describes the initial and personalzied accuracy according to the inplace distillation. The inplace-distillation is not applied during the fine-tuning of personalization. In most cases, applying inplace distillation improves the performance.

F.6. FedProx: Weight Divergence

Table 18 describes the initial and personalized accuracy according to the FedProx. The FedProx is not applied during the fine-tuning of personalization. In most cases, there remains little changes in performance after applying FedProx. Here, we use the value of hyperparameter λ in FedProx as 0.001.

Table 17. Initial and personalized accuracy of FedSup on CIFAR-100 with and without inplace distillation. The fine-tuning epochs is 5, f is 0.1, N is 100, and s is 10.

Architecture Size		B		M		S	
Architecture	In-Distill	Initial	Personalized	Initial	Personalized	Initial	Personalized
FedSup	True	$34.54_{\pm 7.04}$	$69.98_{\pm 6.89}$	$34.40_{\pm 6.29}$	$70.83_{\pm 6.11}$	$33.40_{\pm 6.25}$	$68.34_{\pm 6.89}$
	False	$33.12_{\pm 7.19}$	$68.80_{\pm 7.01}$	$32.54_{\pm 6.75}$	$68.73_{\pm 7.11}$	$30.52_{\pm 7.02}$	$67.24_{\pm 7.44}$

Table 18. Initial and personalized accuracy of FedSup on CIFAR-100 with and without FedProx. The fine-tuning epochs is 5, f is 0.1, N is 100, and s is 10.

Architecture Size		B		M		S	
Architecture	FedProx	Initial	Personalized	Initial	Personalized	Initial	Personalized
FedSup	X	$34.54_{\pm 7.04}$	$69.98_{\pm 6.89}$	$34.40_{\pm 6.29}$	$70.83_{\pm 6.11}$	$33.40_{\pm 6.25}$	$68.34_{\pm 6.89}$
	O	$34.44_{\pm 6.55}$	$69.91_{\pm 6.79}$	$34.46_{\pm 6.31}$	$70.79_{\pm 6.15}$	$33.49_{\pm 6.50}$	$68.01_{\pm 6.77}$

F.7. Training Time Analysis on Synchronized Training Settings

Our methods are much more efficient in terms of time than the synchronous training of FedAvg-Variant methods. Consider an example for real-world applications. Since IoT, Edge Device, and Cloud Server have different resource performance, the time it takes for local training is different for each machine. We assume the local training time for every round in Table 19. If you need to train with FedAvg-Variant Model, the time it takes to synchronize every round is 30 (sec) + network bandwidth time. On the other hand, in the case of E-FedSup, the model is distributed in consideration of the resource, S for IoT, M for Edge Device, and B for Cloud Server, FL can be implemented so that 10 (sec) + network bandwidth time is required. FedSup can also be implemented much more effectively than FedAvg-variant methods if sub-models are selected well in local training.

Table 19. Assuming that the local training time of the Big model in the IoT device is 30 seconds, the training time in different machines of different models is assumed based on this.

	B	M	S
IoT	30(sec)	20(sec)	10(sec)
Edge Device	20(sec)	10(sec)	6(sec)
Cloud Server	10(sec)	5(sec)	3(sec)

F.8. Communication Cost Analysis

Figure 9 is a graph depicting the communication cost required for the neural network to reach 36% accuracy on CIFAR-100 ($N = 100, f = 0.1, s = 10$). E-FedSup is the most efficient, and FedSup has the same communication cost as FedAvg for each round, but the performance converges faster.

F.9. Number of Sampled Architectures for Training in FedSup

We study the number of sampled architectures M per training iterations. It is important because larger n leads to more training time. We train the models with n equal to 1,2,3, or 4 where the sandwich rule is not applied when $n \leq 2$.

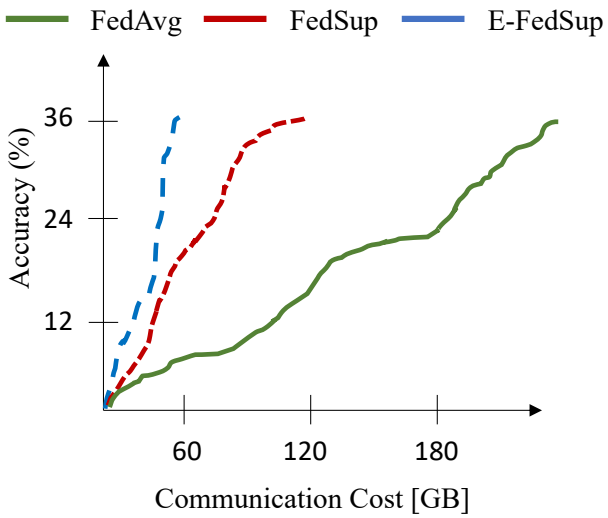


Figure 9. Comparison of communication costs with FedAvg, FedSup, and E-FedSup. The communication cost paid until reaching the same accuracy is compared.

Table 20. Performance of FedSup on CIFAR-10 test dataset with supernet having dynamic operations on depth, kernel, and width. (accuracy on Dirichlet distribution having $\beta = 0.01$ / accuracy on Dirichlet distribution having $\beta = 1.0$) is written in order ($f = 0.1, N = 100, \tau = 5, m = 0.5$).

M	1	2	3	4
W/ Sandwich Rule	-	-	45.44 / 77.79	45.12 / 76.78
W/O Sandwich Rule	43.64 / 77.68	41.58 / 77.27	43.08 / 76.61	43.83 / 77.27