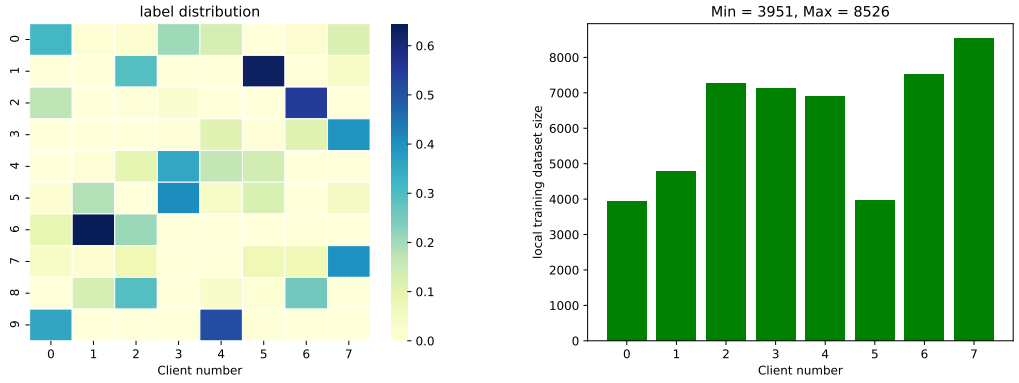
A Appendix

A Data Heterogeneity

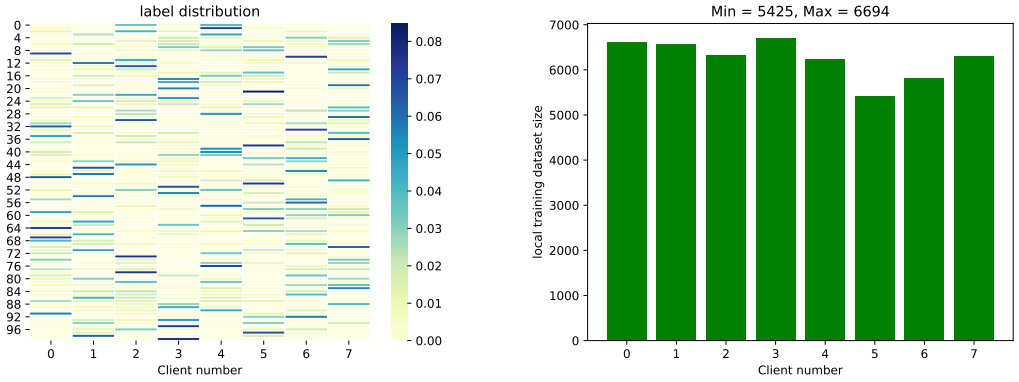
A.1 Data Heterogeneity

Figure 4 and 5 represents data distribution across 8 clients for CIFAR10 and CIFAR100 datasets, respectively. Sub-figure 4a and 5a represent the label distribution across clients, where a darker color indicates more images of that class/label. It can be seen from these sub-figures that the data distribution is heterogeneous label-wise, e.g, Client with ID 0 has class 0 and 9 in excess, whereas, Client with ID 7 has more samples of class 3 and 7 for the CIFAR10 dataset. We observe a similar trend of heterogeneity for CIFAR100 as well, where we have 100 class labels distributed across 8 clients via LDA distribution with α parameter = 0.2. Furthermore, the sub-figures 4b and 5b represent the total number of data samples preset at each client. These figures illustrate that the number of samples is also varying across silos. However, the variation in the total number of samples by silo is more prominent for the CIFAR10 dataset.



(a) Label distribution per client *(b) Image distribution per client*

Figure 4: CIFAR10: LDA distribution (α=0.2) across 8 clients (Seed 9). We can see that the data distribution is heterogeneous in both label and image distribution across clients.



(a) Label distribution per client *(b) Image distribution per client*

Figure 5: CIFAR100: LDA distribution across 8 clients (Seed 9). We can see that the data distribution is heterogeneous in both label and image distribution across clients.

B Hyper-Parameter and Architecture Search Space

B.1 Hyper-parameter Search

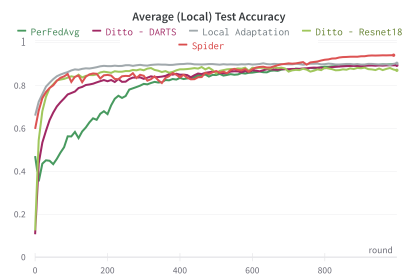
For empirical results of CIFAR10, CIFAR100, and CINIC10, we use a batch size of 32 for all our experiments. We use LDA distribution with a 0.2 α parameter value. We conduct experiments with two seeds for all methods and report the average values in Table 1. For SPIDER, we use the first 30 rounds as warmup rounds, for SPIDER-Searcher, we use a recovery period of 20. Furthermore, we use a learning rate in the search range of $\{0.01, 0.03\}$ for SPIDER. For SPIDER, we used λ search from the set of $\{0.01, 0.1, 1\}$. For the other personalized schemes such as Ditto, perFedAvg, KNN-Per, FedMN, and local adaptation with Resnet18, we searched learning rate over the set $\{0.1, 0.3, 0.01, 0.03, 0.001, 0.003\}$. The reason for having a larger set of learning rates for these methods is that we found 0.001 and 0.003 work better for some of these methods. For Ditto, we used λ from the set $\{0.01, 0.1, 0.5, 1, 2\}$. For PerFedAvg, we used the local lr as a factor of $\{1, 3, 5\}$ of the global lr. We used 1000 rounds of communication for the reported results and observed that they were sufficient to achieve convergence as shown in Figure 6. For FedMN, we used 500 rounds of communication as pretraining. After pretraining, the next 500 for federated local training. We report the best average local test accuracy during the federated local training phase in Table 1. For KNN-Per, we evaluate the global model on the λ hyper-parameter selected from the set $\{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$ and report the best average local test accuracy in Table 1. We used stochastic gradient descent (SGD) optimizer for all the methods.

B.2 Architecture Search Space

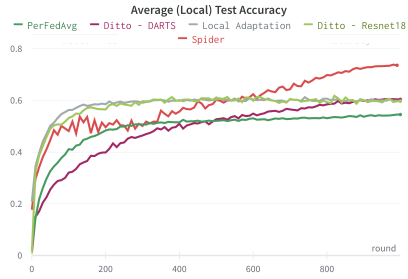
As mentioned in Section 4.2, we have used DARTS (s2) search space (Wang et al., 2021) in our proposed work. During the search, we are using a total of 8 cells as shown by Figure 7. Each type of cell consists of 14 edges; each edge connects two intermediate representations (node) by a mixture of two operations frequently used in various modern CNNs, sep convolution 3x3 and skip connection, hence the name 's2 DARTS search space' (Wang et al., 2021). In this search space, there are two types of cells: normal and reduction cells. Hence, our supernet $\mathcal{A}$ consists of search space of normal cells and reduction cells, each with 14 edges and 2 operations at each edge. As the search progresses via progressive NAS, some operations and edges are removed based on the perturbation criterion as explained in Algorithm 2. It is important to note that the same cells follow the same construction; therefore, if one operation is selected for one normal cell at a particular edge, the same operation will be selected for the same edge at all normal cells. More specifically, we have 14 learnable edges and 2 operations at each edge in a cell, there are a total of 2^{14} possible configurations. Since we have two types of cells, a normal cell, and a reduction cell, there are a total of $(2^{14})^2$ possible architecture designs. The other DARTS search spaces, s3 and s4 (Wang et al., 2021) have a larger search space, e.g., a larger operation candidate set, and therefore, require more computational resources.

C Gradient-based NAS as SPIDER-Searcher

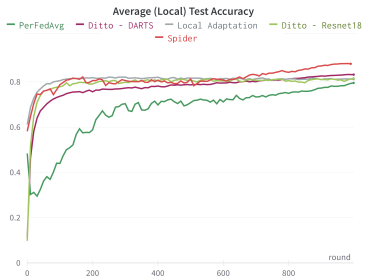
SPIDER is formulated as a bilevel optimization problem as shown in Equation 2 and 3, where the lower-level optimization optimizes the global model parameters w^*, and the upper-level optimization optimizes the local architecture $\mathcal{A}_k$ and its weight parameters v_k for a client k. For the proposed lower-level optimization of SPIDER, we exploit the perturbation-based neural architecture search method as it simplifies the architecture search process by selecting architectures based on the evaluation criterion only. This searcher picks optimal operations based on the impact of its absence (a.k.a perturbation) on the local validation accuracy as explained in detail in Algorithm 2. On the other hand, differentiable NAS is another widely known method of architecture selection which is often formulated in itself as a bilevel optimization problem with architecture parameters α as an upper-level variable and w as lower-level variables. As an ablation study, we replace perturbation-based NAS Searcher with MileNAS Searcher (He et al., 2020d) which is a gradient-based NAS Searcher. We use its first-order approximation which essentially applies softmax operation to the architecture parameters α for each edge and updates them by the stochastic gradient step $\mathcal{A}_k^{t+1} = \mathcal{A}_k^t - \eta_\alpha \left(\nabla_\alpha F^{\text{tr}}(v_k, \mathcal{A}_k^t) + \lambda_\alpha \nabla_\alpha F_{\text{val}}(v_k, \mathcal{A}_k^t)\right)$ at each iteration of local epochs. F^{tr} and F^{val} are



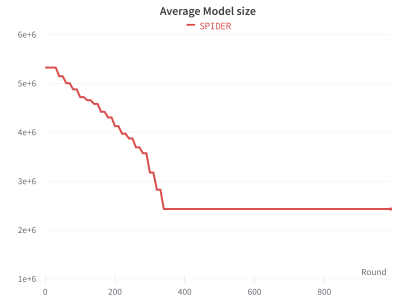
(a) Average Validation Accuracy comparison on CIFAR10 Dataset



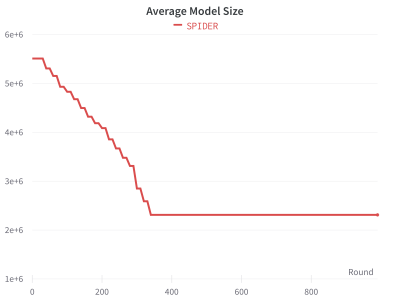
(b) Average Validation Accuracy comparison on CIFAR100 Dataset



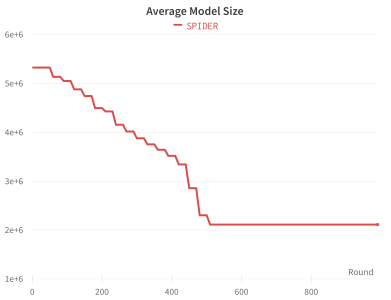
(c) Average Validation Accuracy comparison on CINIC10 Dataset



(d) Change in Average Model Size of the local models on CIFAR10 Dataset



(e) Change in Average Model Size of the local models on CIFAR100 Dataset



(f) Change in Average Model Size of the local models on CINIC10 Dataset

(g) Average Local Training Loss on CIFAR10 Dataset

(h) Average Local Training Loss on CIFAR100 Dataset

(i) Average Local Training Loss on CINIC10 Dataset

Figure 6: Figure 6a, 6b, 6c represent the average validation accuracy comparison between SPIDER and PerFedAvg, Ditto, and Local Adaptation on CIFAR10, CIFAR100, and CINIC10 datasets, respectively. SPIDER outperforms the representative baselines on all three datasets. Figures 6d, 6e and 6f illustrate the change in Average Model Size of the local models as SPIDER Search progresses through three phases (Phase 1 (Local Architecture Pre-training), Phase 2 (Local Architecture Search), Phase 3 (Local Architecture Training)) on CIFAR10, CIFAR100, and CINIC10 Datasets, respectively. Likewise, Figures 6g, 6h and 6i illustrate the change in Average local training loss of the local models as SPIDER Search progresses through three phases (Phase 1 (Local Architecture Pre-training), Phase 2 (Local Architecture Search), Phase 3 (Local Architecture Training)) on CIFAR10, CIFAR100, and CINIC10 Datasets, respectively

calculated at the training and validation data, respectively. For simplicity, $\mathcal{A}_k$ here represents a collection of all architecture parameters in the local model of client k, and η_α is the learning rate of architecture parameters. We select η_α to be 0.01 and the regularization parameter λ_α as 1. We apply MileNAS solver for the first 400 communication rounds and find the local architectures based on the magnitude of the architecture parameters α at each edge, and then train them for the next 600 rounds by following SPIDER formulation given in Equation 4 and 5. As shown in Table 2, the performance gap between the two methods is negligible for the CIFAR100 dataset. However, perturbation-based NAS is a simpler method as it does not require a gradient step for the architecture search/update at each step, and therefore, takes less amount of time in comparison to gradient-based NAS.

Table 2: Average (local) test Accuracy comparison of Perturbation-based NAS with gradient-based NAS in SPIDER on CIFAR100 dataset.

Method	Average Accuracy
Perturbation-based NAS (Wang et al., 2021)	72.36
MiLeNAS (He et al., 2020d)	72.40

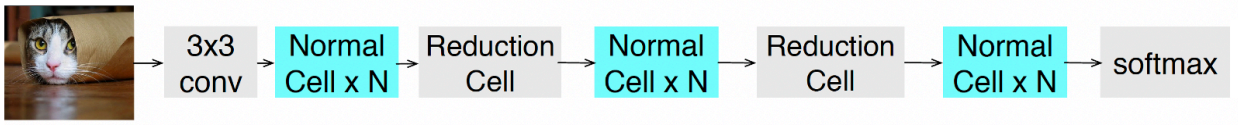


Figure 7: Supernet $\mathcal{A}$ search space. Note that since we have 8 cell based search space, $N = 2$ for our construction.

D Federated Neural Architecture Search versus SPIDER

In this section, we compare the proposed Personalized Neural Architecture search method SPIDER with its two base cases, centralized learning where $\lambda = 0$ and architecture personalization base case where $\lambda \to \infty$ which essentially refers to the setting where local model weights are identical to the global model weights. We also compare SPIDER with the federated neural architecture search method, which searches a global architecture in a federated manner and then trains it via FedAvg. This investigation can help us appreciate the benefits of architecture personalization and federated learning in a cross-silo setting.

D.1 Federated Neural Architecture Search

We analyze NAS performance to search for a global architecture in a federated setting as proposed by previous works (He et al., 2020b). To keep the algorithmic comparison fair, we have used the NAS setting as same as SPIDER. Essentially, only one Supernet (DARTS) is maintained at each silo. Each silo trains the Supernet following the FedAvg algorithm for 30 rounds of warmup. Next, the neural architecture search is performed following perturbation-based NAS at the server using the server's test/validation data. After NAS, the architecture is obtained and trained in a federated manner by the FedAvg algorithm. This can be considered a slight variation of FedNAS (He et al., 2020b) where we replace MiLeNAS He et al. (2020d) with a state-of-the-art NAS method, Perturbation-based NAS (Wang et al., 2021) to keep the comparison fair. Figure 8 represents the average validation accuracy across each silo. We obtained a total of 68.03% average local test accuracy with global NAS. Although this accuracy is still higher than various PFL baselines, it is lower than SPIDER (72.36%). This shows that a global architecture search may not capture the data heterogeneity in FL with non-I.I.D data.

D.2 SPIDER: Personalized Federated Neural Architecture Search

We can observe from Figure 8 that SPIDER with $\lambda = 0.01$ hyper-parameter setting outperforms the Federated NAS. However, it is important to explore two base cases of SPIDER. The first setting is where $\lambda = 0$ and therefore, has no learning between the local and global models. It is where each silo performs a local neural architecture search. Each silo uses only one model, Supernet, and deploys the perturbation-based NAS Searcher locally **without any collaboration from the other silos**. The Local Supernet has the same hyper-parameters of warmup (30 local epochs) and recovery period of 20 local epochs after every pruning step, followed by local training with a total of 1000 epochs. We obtained 72.36% average local test accuracy with SPIDER, and 69.31% average local test accuracy with Centralized NAS on CIFAR100 dataset. This highlights the significance of the proposed regularization. This also shows that tailoring architectures to the silo's data distributions even for centralized training can be very powerful. However, collaboration with other silos as is performed by SPIDER, can enhance the prediction performance of personalized architectures at individual silos.

Further, we experiment with the $\lambda \to \infty$ setting of SPIDER where we set $v_k = w^*_{share}$ for each round and finetune the local model parameters v_k for 5 local epochs before evaluating its performance. We observe

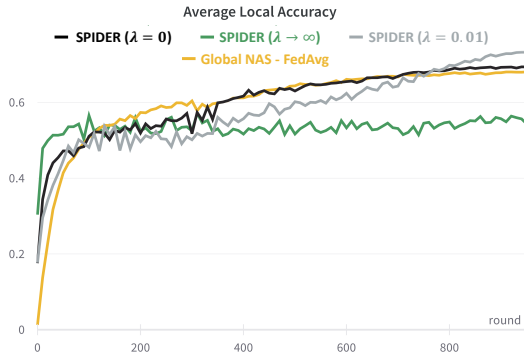


Figure 8: Average Validation Accuracy comparison of SPIDER with the Federated NAS on CIFAR100 dataset. For SPIDER, we show the results of three different values of λ hyper-parameter. First setup of special case of SPIDER is centralized learning where $\lambda = 0$. The second setting is $\lambda \to \infty$ where local model weight parameters are set identical to global model weight parameters. The third setting of SPIDER is $\lambda = 0.01$, which outperforms the other two special cases of SPIDER ($\lambda = 0$ and $\lambda \to \infty$) as well as Federated NAS.

that the local average accuracy for this hyper-parameter setting is 57.14. This hyper-parameter setting underperforms FedNAS. This indicates that personalization of the local architecture weight parameters v_k is also a critical element in the proposed formulation. We know that architecture search, optimization of $\mathcal{A}_k$, is achieved by perturbing the local Supernet architecture (v_k, $\mathcal{A}_k$) and evaluating its performance on the local validation dataset. Therefore, restricting $v_k = w^*_{share}$ may restrict the quality of personalization from the selected architecture since they are evaluated/selected based on the global model weights w_*(general/global knowledge).

E Computational Cost Analysis of Training

We propose SPIDER as a cross-silo framework where clients have ample computational resources and data heterogeneity is a main concern (Huang et al., 2022). As shown in Table 3, we achieve substantial performance improvement in terms of average local accuracy over the state-of-the-art personalization methods. For research purposes, it is essential to compare the computational cost and wall clock time of the proposed method with the other representative personalized federated learning (PFL) methods. Therefore, we report peak memory costs and total wall clock time for all methods. Note that we obtain the peak memory cost values for the maximum number of compute required for the model training, which includes forward/backward pass size, and parameter size of all the models, global and local model, for a batch size of 32 on the NVIDIA RTX 2080Ti GPU card.

We observe the peak compute cost per silo to be 4.18GB, 2.14GB, 1.45GB, 1.45GB, 2.90GB, 1.45GB, and 1.98GB for SPIDER, FedNAS, Local adaptation, KNN-Per, Ditto, perFedAvg, and FedMN, respectively. Though SPIDER requires relatively higher peak memory for maintaining two models, a global supernet and a local model, at each silo, it outperforms the state-of-the-art personalization methods substantially. For example, SPIDER yields a 6.18% performance gain relative to the second-best method, KNN-Per, on the CIFAR100 dataset which can be a significant gain for organizations that require high accuracy without the constraint of small computing cost.

Further, we report the end-to-end time of each method for a total of 1,000 rounds on the CIFAR100 dataset. For implementation, we used 8 GPU NVIDIA RTX 2080Ti GPU cards where each GPU represents a physical node with an NVIDIA RTX 2080Ti GPU card on the fedml platform implementation for FL (He et al., 2020c). Please note that to report FedMN and KNN-Per method results, we used their GitHub code directly where they perform federated learning on one GPU in a sequential manner. Therefore, the time comparisons of these two methods might not be fair. Overall, for wall-clock time comparisons, we observe that there are two factors contributing to the wall-clock overhead.

Overhead of Neural Architecture Search (NAS): One notable observation is that methods that use a predefined architecture tend to have significantly lower wall clock times. However, this comes at the cost of lower prediction performance. It is important to note that these low wall clock times may not hold true in practical scenarios, especially when dealing with the data-invisibility challenge in Federated Learning (FL).

The data invisibility challenge in FL refers to the distributed learning setup where clients' private data remain invisible to the server. As a result, from the server's perspective, it becomes unclear how to select a pre-defined architecture from a pool of all available candidates. For instance, in our work, we have reported results for the Ditto method using two predefined architectures, resnet18 and DARTS. Considering both architectural experiments, the wall clock time has doubled to 8 hours and is even higher if we include the hyper-parameter search time for both architectures, yet the performance remains significantly lower (12% lower than SPIDER).

Given the data heterogeneity and data invisibility challenges of FL, manually searching for a neural architecture that works optimally for all clients can become prohibitively expensive. This challenge has been one of the motivations for developing SPIDER as a personalized neural architecture search framework.

Table 3: Accuracy versus Computational Cost Tradeoff on CIFAR100 dataset for SPIDER versus the other representative personalized federated learning techniques such as Local Adaptation, Ditto, KNN-Per, perFedAvg, FedMN.

Method	Accuracy	Peak Memory Cost	Wall Clock Time
SPIDER	72.36%	4.18GB	15 hour
FedNAS	68.03%	2.14GB	7 hour
Local Adaptation - ResNet18	60.92%	1.45GB	2 hour
KNN-Per - ResNet18	67.05%	1.45GB	10 hour
Ditto - ResNet18	61.06%	2.90GB	4 hour
perFedAvg - ResNet18	54.56%	1.45GB	5 hour
FedMN - ResNet18	61.92%	1.98GB	16 hour

Overhead of Bilevel Optimization: Another factor contributing to overhead is the Bilevel Optimization where the lower-level optimization optimizes the global model parameters and the upper-level optimization optimizes the local architecture and its weights parameters. For example, we observe the wall clock time of SPIDER to be almost double the wall clock time of FedNAS (NAS-based method that does not deploy bilevel optimization for personalization). However, SPIDER demonstrates approximately a 5% increase in prediction performance. Likewise, we observe the wall clock of Ditto to be double the wall clock time of Local Adaptation.

It is important to highlight that our main focus has been to address the data-heterogeneity challenge, especially in cross-silo FL settings where silos can have higher computational resources and non-IID data distributions are the main challenge. We show via extensive empirical experiments that the proposed method yields higher predictive performance with lower standard deviation (a proxy metric for fairness) across silos as compared to other state-of-the-art methods at the cost of higher computational power.