Neural Architecture Search Driven by Locally Guided Diffusion for Personalized Federated Learning (Supplementary Material)

Peng Liao[1*], Xilu Wang[3*], Yaochu Jin[2,1†], Wenli Du[1], Han Hu[1]
[1] Key Laboratory of Smart Manufacturing in Energy Chemical Process, Ministry of Education, ECUST, Shanghai, China
[2] Trustworthy and General AI Lab, School of Engineering, Westlake University, Hangzhou, China
[3] Computer Science, University of Surrey, Surrey, UK
pengliao@mail.ecust.edu.cn, wangxilu@surrey.ac.uk, jinyaochu@westlake.edu.cn, wldu@ecust.edu.cn, han.hu@mail.ecust.edu.cn

The Personalized Federated Stochastic Differential Equation-Based NAS (PerFedSDE-NAS) framework consists of two stages: (1) Pretraining a conditional diffusion-based architecture generation model and (2) personalized federated search using the diffusion-based candidate architecture generation strategy. We validate the proposed PerFedSDE-NAS across three search spaces, with their detailed introduction provided in Sec. A.

The pretraining of the conditional diffusion model is discussed in the main paper of Sec. 3.2. Additional details include the diffusion process (Sec. B), training of the score network (Sec. C), and the mapping of architectural and data features to accuracy predictions (Sec. D).

The personalized federated search based on the diffusion-generated candidate architectures is also elaborated in the main paper of Sec. 3, with supplementary pseudocode provided in Sec. E.

The visualization of non-i.i.d. data distribution is presented in Sec. F, and the extended ablation study is in Sec. G.

Finally, we conduct a quantitative analysis of GPU days consumption between client-side and server-side NAS-based personalized federated learning (PFL) approaches, revealing the inherent reasons why client-side NAS-based PFL is theoretically more computationally expensive (Sec. H). Notably, although the inference time for architecture diffusion is higher than that of crossover and mutation operators, it does not increase the overall computational cost, as shown in Table D. This is because architecture diffusion is parallelized with supernet training, and the time required to generate 100 architectures via diffusion is shorter than that needed for a single epoch of the supernet training. All experiments used one A100 GPU.

* Equal contribution. † Corresponding author.

A. Search space

A.1. DARTS

DARTS [8] is one of the most widely used search spaces in NAS. The searched network consists of stacked Normal Cells, which maintain the same feature map size between input and output, and Reduction Cells, which reduce the feature map size by half. Each cell comprises two input nodes, one output node, and four intermediate nodes. The edges represent different operations, including 3×3 and 5×5 separable convolutions, 3×3 and 5×5 dilated convolutions, 3×3 max pooling, 3×3 average pooling, identity mapping, and zero operation.

A.2. MobileNetV3

MobileNetV3 [2] is one of the largest search spaces in NAS, covering approximately 10^{19} possible architectures. The search space is designed hierarchically, consisting of five stages, where the number of building blocks per stage varies between 2 and 4. Consequently, the maximum possible depth of a MobileNetV3 architecture is calculated as $5 \times 4 = 20$. For each building block, the kernel size can be selected from the set $\{3, 5, 7\}$, and the expansion ratio can be chosen from $\{3, 4, 6\}$. This results in a total of $3 \times 3 = 9$ possible operation types per layer.

A.3. NASViT

NASViT [4] defines a hybrid search space that integrates both convolutional and transformer-based components. The

Table A. A summary of hyper-parameter settings for training score networks.

Parameters	Settings
Batch size for training	256
Dropout rate	0.1
Initial learning rate	2.00E-05
Optimizer	Adam
β_1 in Adam optimizer	0.9
β_2 in Adam optimizer	0.999
Warmup period	1000
Gradient clipping	1.0

architecture is constructed sequentially with a convolutional block, three MBConv blocks, four transformer blocks, and an MBPool block. Each block allows for specific architectural choices, such as width, depth, kernel size, and expansion ratio. Since the search space definitions vary across different blocks, we refer readers to the original work [4] for further details.

B. Diffusion process

In Sec. 3.2 of the main paper, we describe the construction of a conditional diffusion model, specifically employing the Variance Exploding SDE [10]. During the forward process, architectures are perturbed by Gaussian noise. To enable the reverse generative process, the score network $\boldsymbol{s_\theta}$ is trained to approximate the score function $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$. By incorporating information from the predictor, the generation process is guided accordingly.

C. Score network

In Sec. 3.2 of the main paper, we introduce the score network for capturing architectural structures and topological relationships. Each architecture is represented as a directed acyclic graph (DAG) with a node set and an adjacency matrix. We employ Computation-Aware Transformer-based Encoding (CATE) [13], which learns DAG representations by processing similar architectures, masking operations, and minimizing reconstruction error. However, CATE alone may generate invalid architectures, such as disconnected paths disrupting computational flow. To mitigate this, positional embeddings [11] enhance topological awareness [1].

The score network is trained only once per search space. Due to computational constraints, for each search space, the score network is trained unsupervised on a subset of architectures from the search space. Note that we extend our work to the DARTS and NASViT search spaces. Specifically, we sample 500,000 architectures from DARTS, MobileNetV3, and NASViT, respectively, for unsupervised training. Table A summarizes the hyperparameter settings for the score network.

Table B. A summary of hyper-parameter settings for training predictors.

Parameters	Settings
Batch size for training	256
Dropout rate	0.1
Initial learning rate	1.00E-03
Optimizer	Adam
β_1 in Adam optimizer	0.9
β_2 in Adam optimizer	0.999
Warmup period	1000
Gradient clipping	1.0

D. Predictor

In Sec. 3.2 of the main paper, a predictor [6] is introduced to map architectures and dataset features to the classification accuracy of an architecture on a given dataset. The predictor takes two inputs: architectural information and dataset features. These inputs are fused to predict the classification accuracy.

The architecture is represented as a directed graph and processed through graph convolutional layers based on DiGCN [12], capturing structural dependencies. The resulting graph embeddings are then aggregated and passed through a fully connected layer to obtain the architecture's latent representation.

For dataset features, images are first processed by ResNet-18 [5] to extract image-level features, which are then grouped by class. A Transformer-based Set Transformer [7] is employed to extract class prototypes and capture inter-class relationships via attention mechanisms, producing the dataset's latent representation.

The architecture and dataset latent vectors are concatenated and fed into a multilayer perceptron to predict classification accuracy. The predictor is trained using Mean Squared Error loss with a sufficient amount of data.

Training data generation for each search space is as below:

- DARTS: We save the supernet trained by FairNAS [3] and evaluate the validation accuracy of 500,000 randomly sampled subnets on ImageNet [9].
- MobileNetV3: We use the publicly available pre-trained supernet weights (`ofa_mbv3_d234_e346_k357_w1.0`) from GitHub and evaluate the validation accuracy of 500,000 randomly sampled subnets on ImageNet.
- NASViT: We leverage [2]'s released checkpoint from GitHub and evaluate the validation accuracy of 500,000 subnets on ImageNet.

The hyperparameters for training the predictor are summarized in Table B.

E. Pseudocode

Algorithm 1 Pseudocode of PerFedSDE-NAS

Input: The number of clients N, number of participating clients per round n, number of searches performed by the client n_e, dimension of the discrete architecture encoding D, pretrained conditional diffusion model, number of generated candidate architectures N_{gen}, offspring size s, archive size k.

1: Server initializes the supernet weights w_0;
2: **for** each round $r = 1$ to R **do**
3: **On Server:**
4: $clients \leftarrow$ Randomly sample n clients from $\{1, 2, \ldots, N\}$;
5: Send supernet weights w_r to selected clients $clients$;
6: **On Client:**
7: **for** each client $i \in clients$ in parallel **do**
8: **for** $j = 1$ to n_e **do**
9: **Preliminary evaluation:**
10: Initialize or inherit population P_j of size $3D$;
11: **Architecture diffusion generation (parallel to supernet training):**
12: Construct an RBF model based on P_j;
13: Generate N_{gen} candidate architectures using the conditional diffusion model;
14: $offspring \leftarrow$ Select the top s candidates from N_{gen} candidate architectures using the RBF model;
15: **Supernet training (parallel to architecture diffusion generation):**
16: $archive \leftarrow$ Select the top k architectures from population P_j;
17: Sample architectures from $archive$ to train the supernet, with only one epoch of training;
18: **Candidate evaluation:**
19: Evaluate $offspring$ architectures using the updated supernet;
20: $P_{j+1} \leftarrow$ Combine population P_j and $offspring$, and then select the top $3D$ architectures;
21: **end for**
22: Record the final population P_j and the test accuracy of the best individual;
23: Send weight updates Δw_i to the server;
24: **end for**
25: **On Server:**
26: $w_r \leftarrow$ Aggregate client updates using Equation (5) in the main text;
27: **end for**

Output: The best architecture and its weights for each client.

Algorithm 1 outlines the proposed PerFedSDE-NAS, and the key parameters used are defined as below: The total number of clients N participating in personalized FL training, the total communication rounds R between the server and clients, the number of clients n selected per round, the number of local search iterations n_e per client, the dimension D of the discrete architecture encoding, the pre-trained conditional diffusion model, the number of architectures generated N_{gen} by diffusion model per local search iteration, the number of selected offspring architectures s, and the archive size k.

The PerFedSDE-NAS framework begins when the server distributes the pre-trained diffusion model to all clients. The server initializes the supernet parameters (line 1) and executes R communication rounds with the clients (line 2).

On the server side (line 3), the server selects n clients from N total clients to participate in the current round (line 4) and transmits the supernet weights to them (line 5). After receiving the weight updates from all participating clients (line 26), the server aggregates these updates to refine the global supernet parameters (line 27).

On the client side (line 6), the selected clients perform a local architecture search in parallel (line 7). Each client conducts n_e local searches, training for only one epoch per search (line 8). After completing the search process, each client records the current population P_j, the best found architecture along with its corresponding weights and test accuracy (line 22), and transmits the supernet weight updates to the server (line 23).

The search process consists of four main stages:

(1) **Preliminary Evaluation:** When $j = 1$, population P_j is randomly initialized and evaluated, and when $j > 1$ P_j is inherited from the client's previous search history (line 10). Note that the population size is $3D$.

(2) **Architecture Diffusion Generation (Parallel to Supernet Training):** A Radial Basis Function (RBF) model is constructed based on the current population (line 12), and N_{gen} candidate architectures are generated using the conditional diffusion model (line 13). The top s architectures are selected based on RBF model predictions (line 14).

(3) **Supernet Training (Parallel to Architecture Diffusion Generation):** The top k architectures from P_j are retained as the archive (line 16), and supernet training is conducted by sampling architectures from the archive (line 17).

(4) **Candidate Evaluation:** The updated supernet weights are used to evaluate the s offspring architectures (line 19). The evaluated offspring is combined with the current population, from which the next generation of population is selected (line 20).

Finally, each client obtains a personalized solution, including the model architecture and its corresponding weights.

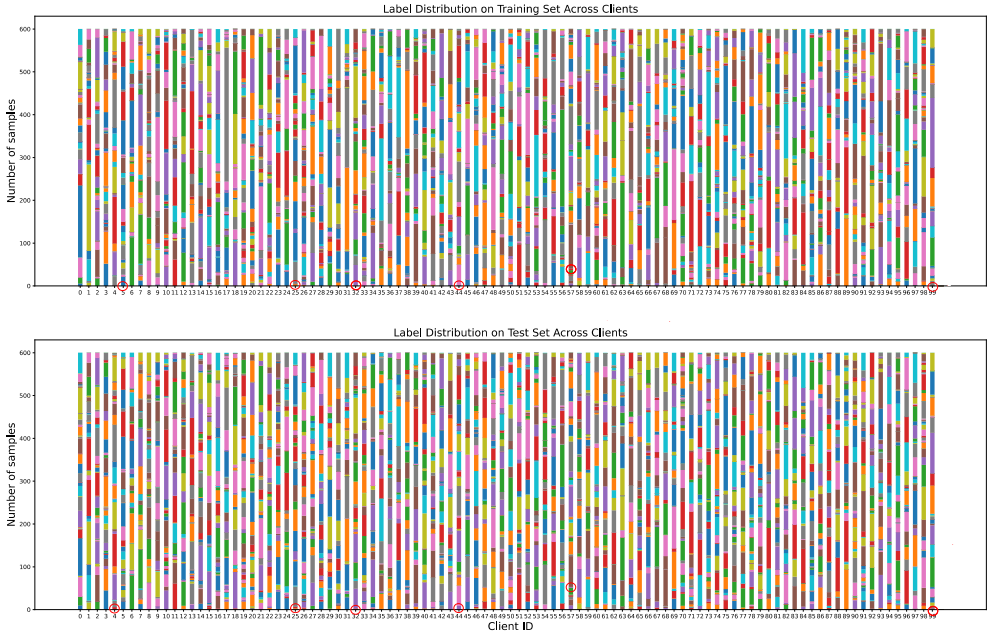

Figure A. Label distribution on CIFAR-100 (Dirichlet $\alpha = 0.1$) across 100 clients used in our setup.

Table C. Extended ablation study on MobileNetV3.

Chal-lenge	Method	CIFAR-10 ($\alpha = 0.3$)		CIFAR-100 ($\alpha = 0.1$)	
		Acc (%)	Time (h)	Acc (%)	Time (h)
C1	Ours w/o (archive + diffusion)	80.06±5.38	17.53	60.98±5.16	17.33
C2	Ours w/o (archive + Eq.5)	79.82±5.36	17.55	60.88±5.31	17.30
C3	Ours w/o (Eq.5 + diffusion)	79.42±5.59	17.51	59.64±5.13	17.41
All	**PerFedSDE-NAS (Ours)**	**82.62±5.21**	**17.61**	**64.15±4.91**	**17.37**

F. Visualization of Non-i.i.d. Data Distribution

Recently, Dirichlet distributions have been generally used to replace manual splits for allocating data to each client by varying α to simulate real-world scenarios, i.e., non-i.i.d. data distributions (**C1**). Following this standard practice, we simulate challenging non-i.i.d. settings with $\alpha = 0.1$, and the label distribution on CIFAR-100 is shown in Fig. A. We can see extreme heterogeneity in both label space and sample size across 100 clients. Notably, some client test sets contain unseen labels (highlighted in red circles), which is an extreme yet realistic non-i.i.d. challenge and underscoring the need for cross-client knowledge transfer in PFL.

G. Extended Ablation Study

We reported ablation studies by removing each component in Table 5. To further validate the effectiveness, Table C provides an additional analysis of each component.

H. Computational Cost and Communication Cost

We quantitatively analyze the computational and communication costs of client-side and server-side NAS-based PFL. Specifically, we adopt PerFedRLNAS [14] for server-side NAS-based PFL, while client-side NAS-based PFL methods are the baseline in the main paper and PerFedSDE-NAS. The results are summarized in Table D.

H.1. Computational cost

A recap on the approaches under comparison is given below:

(1) PerFedRLNAS [14]: Reinforcement learning is used at the server to search for personalized architectures for each client. Once an architecture is selected, the client directly trains it and reports the trained network weights along with the test accuracy.

(2) PerFedSDE-NAS (Ours): A conditional diffusion model is employed for local architecture search on clients. During the iterative search, a number of candidate architectures are evaluated to identify better architectures. Only the supernet weights are transmitted between the server and clients, ensuring no additional privacy leakage.

(3) Baseline: PerFedSDE-NAS is modified by replacing the diffusion model with crossover and mutation operations for candidate architecture generation.

The federated learning settings are kept consistent across all methods: 1500 communication rounds, with 5 clients

Table D. Time analysis.

Type	Method	Prelim Eval (s)	Parallelization (s)		Eval (s)	Sum (s)	Comms (M)
			Training	Generation			
Server	PerFedRLNAS [14]	–	13.197±0.159	–	0.156±0.11	13.353±0.807	0.0159±0.0008
Client	Baseline	23.893±0.124	12.993±0.218	1.029±0.021	8.849±0.122	45.736±2.488	0.2319±0.0000
	PerFedSDE-NAS (Ours)	20.410±0.008	12.745±0.134	10.130±1.328	7.560±0.008	40.714±2.011	0.2319±0.0000

participating per round, each performing 5 local search iterations, where each iteration involves one epoch of supernet training.

For systematic analysis, we decompose the client-side NAS-based PFL process into three stages, i.e., preliminary evaluation, search process, and candidate evaluation, denoted as "Prelim Eval (s)," "Parallelization (s)," and "Eval (s)" in Table D, respectively. The total computation time is summarized in the column "Sum (s)," all measured in seconds.

(1) Preliminary evaluation: This stage is absent in PerFedRLNAS. In PerFedSDE-NAS and Baseline, upon initializing the population, the accuracy of 135 architectures needs to be re-evaluated due to updated supernet weights, with each evaluation taking approximately 0.16 seconds per architecture.

(2) Search process: PerFedRLNAS requires training only the assigned subnet for 5 epochs without generating new candidates. In contrast, PerFedSDE-NAS and Baseline sample diverse subnets to update the supernet weights while concurrently generating candidate architectures. Although the inference time for generating 100 architectures using the diffusion model is longer than that of crossover and mutation operators, it remains computationally cheaper than supernet training. As the data volume and training configurations are identical, the overall training time across methods remains comparable.

(3) Candidate evaluation: PerFedRLNAS evaluates only a single architecture, whereas PerFedSDE-NAS and Baseline evaluate 50 architectures across 5 iterations, i.e., 10 architectures per iteration. Consequently, the evaluation time for PerFedSDE-NAS and Baseline is approximately 50 times that of PerFedRLNAS.

(4) Total computation cost: The total runtime of client-side NAS-based PFL is approximately three times that of server-side NAS-based personalized FL, primarily due to the repeated evaluation of candidate architectures.

Additionally, the computational overhead at the server side is significantly higher in server-side NAS-based PFL compared with client-side NAS. While client-side NAS-based PFL only requires aggregating supernet weights, server-side NAS-based PFL must perform architecture searches for each client, further increasing computational costs.

H.2. Communication cost

We also measure the communication cost between the server and clients, defined as the total transmitted data in bytes. The results are reported in Table D under the column "Comms (M)." The communication cost is computed as the total transmitted bytes divided by the total communication rounds and the number of participating clients per round. The results are presented in megabytes (M), including mean and standard deviation.

Notably, the communication cost of server-side NAS-based PFL is only 7% of that in client-side NAS-based PFL. This is because server-side NAS transmits only subnetworks instead of the full supernet. However, this approach introduces significant privacy risks, as the server gains access to client-specific solutions, including architectures, weights, and evaluation metrics.

References

[1] Sohyun An, Hayeon Lee, Jaehyeong Jo, Seanie Lee, and Sung Ju Hwang. Diffusionnag: Predictor-guided neural architecture generation with diffusion models. In *ICLR*, 2024. 2

[2] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once for all: Train one network and specialize it for efficient deployment. In *International Conference on Learning Representations*, 2020. 1, 2

[3] Xiangxiang Chu, Bo Zhang, and Ruijun Xu. Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search. In *Proceedings of the IEEE/CVF International Conference on computer vision*, pages 12239–12248, 2021. 2

[4] Chengyue Gong and Dilin Wang. Nasvit: Neural architecture search for efficient vision transformers with gradient conflict-aware supernet training. *ICLR Proceedings 2022*, 2022. 1, 2

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2

[6] Hayeon Lee, Eunyoung Hyung, and Sung Ju Hwang. Rapid neural architecture search by learning to generate graphs from datasets. In *9th International Conference on Learning Representations, ICLR 2021*, 2021. 2

[7] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural net-

works. In *International conference on machine learning*, pages 3744–3753. PMLR, 2019. 2

[8] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable Architecture Search. In *International Conference on Learning Representations*, 2018. 1

[9] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 2

[10] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. 2

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 2

[12] Wei Wen, Hanxiao Liu, Yiran Chen, Hai Li, Gabriel Bender, and Pieter-Jan Kindermans. Neural predictor for neural architecture search. In *European conference on computer vision*, pages 660–676. Springer, 2020. 2

[13] Shen Yan, Kaiqiang Song, Fei Liu, and Mi Zhang. Cate: Computation-aware neural architecture encoding with transformers. In *International Conference on Machine Learning*, pages 11670–11681. PMLR, 2021. 2

[14] Dixi Yao and Baochun Li. Perfedrlnas: One-for-all personalized federated neural architecture search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 16398–16406, 2024. 4, 5