Geometry-Aware Subspace Perturbation for Heterogeneous Federated Learning

Xiangtao Zhang[1] **Hailong Yan**[1] **Obed Irihose**[1] **Joey Tianyi Zhou**[3]
Chee Seng Chan[4] **Ce Zhu**[1] **Le Zhang**[2] ✉
[1] University of Electronic Science and Technology of China, Chengdu, China
[2] College of Computer Science and Software Engineering, Hohai University, Nanjing, China
[3] Singapore Management University, Singapore
[4] Universiti Malaya, Malaysia
xiangtaozhang@std.uestc.edu.cn, lezhang@uestc.edu.cn

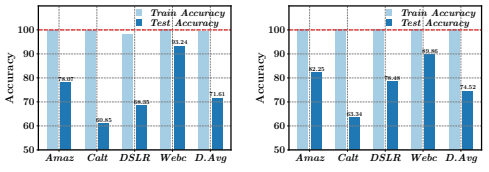
Abstract

We propose a geometry-aware perturbation framework that explicitly models the low-rank, anisotropic, and evolving structure of gradient dynamics to improve generalization in heterogeneous federated learning (HFL). Our approach is built upon three key components. First, we represent perturbations within a low-dimensional subspace that captures the dominant directions of gradient trajectories. Second, we generate perturbations through a structure-aware sampling strategy that aligns with the covariance of projected gradients, enabling distribution-aware exploration. Third, we introduce an adaptive mechanism that dynamically adjusts both the magnitude and the structure of perturbations to match the evolving optimization process. Extensive experiments demonstrate that the proposed method consistently improves generalization across diverse HFL benchmarks, achieving up to a 4.49% improvement on the *Office10* dataset with minimal computational overhead. The code will be available at https://github.com/AVC2-UESTC/SPAS.git.

1 Introduction

Federated learning (FL) has attracted significant attention as a distributed machine learning paradigm [26, 29, 50]. Traditional federated learning (TFL) preserves data privacy by collaboratively training a global model across multiple clients, but it is challenged by data heterogeneity [1, 12, 16, 60]. Personalized federated learning (PFL) alleviates statistical heterogeneity by introducing client-specific parameters, however, it is still restricted to homogeneous model architectures [57, 56, 49, 48, 34].

To overcome this limitation, heterogeneous federated learning (HFL) enables clients to adopt different model architectures, addressing both data and model heterogeneity [52, 59, 43, 31]. Although existing HFL frameworks improve performance from various perspectives, they share a common drawback: due to limited local data and insufficient global aggregation, local models tend to overfit to their respective client data distributions, making them more prone to overfitting than in traditional federated learning [51, 61, 15]. As illustrated in

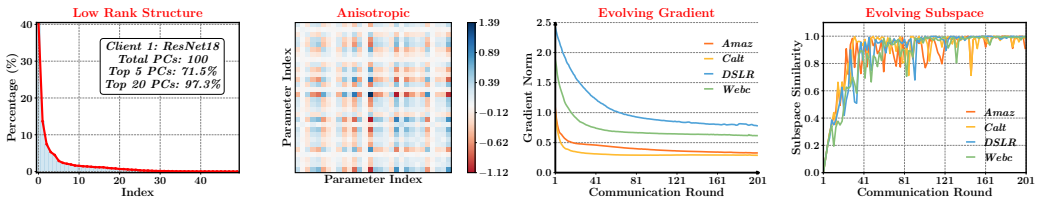


(a) Model performance (b) Model performance

Figure 1: **Challenges** in existing HFL frameworks. **(a)** and **(b)** show the training and test accuracy of different clients under FedTGP [58] and FedMRL [53], respectively.

✉ Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).



(a) Singular value distribution (b) Gradient covariance (c) Gradient norm evolution (d) Principal component similarity

Figure 2: **Empirical analysis** of gradient dynamics in HFL. **(a)** The singular value distribution shows that gradient trajectories exhibit a low-rank structure, where most energy is concentrated in a few dominant components. **(b)** The gradient covariance reveals strong anisotropy, indicating that different directions contribute unequally to optimization. **(c)** The gradient norm evolves over communication rounds, reflecting changes in optimization states during training. **(d)** The similarity of principal components across rounds shows that the underlying gradient subspace is not fixed but evolves over time. These observations collectively highlight the low-rank, anisotropic, and evolving nature of gradient dynamics, motivating the design of geometry-aware perturbations.

Fig. 1, we present the training and testing performance across different clients in existing HFL methods. The results show that while training accuracy has largely reached its peak, testing accuracy remains improvable.

Improving generalization remains a fundamental challenge due to limited local data and insufficient global aggregation. Existing HFL methods primarily focus on information sharing mechanisms [47, 53, 17], such as knowledge distillation [15, 25] or prototype alignment [58, 45], yet still suffer from noticeable overfitting. Perturbation-based optimization [5, 27] improves generalization by encouraging convergence to flat minima through local exploration in the parameter space. For such exploration to be effective, the perturbations should align with the underlying geometry of gradient dynamics while maintaining computational efficiency. However, existing perturbation strategies fall short in this regard: random perturbations [27, 64, 2] ignore the structural properties of gradients, while adversarial approaches such as Sharpness-Aware Minimization (SAM) [9, 33] incur significant computational overhead. Moreover, in the federated learning setting that we consider, most prior work incorporates SAM into traditional federated frameworks, which nearly doubles the computational cost and is difficult to extend to model-heterogeneous scenarios [42].

To better understand the optimization dynamics in HFL, we conduct an empirical analysis under a commonly-used HFL setting. Specifically, we examine model updates from FedMRL [53] across communication rounds on the *Office10* dataset, and treat them as gradient trajectories to investigate their underlying structure. As illustrated in Fig. 2a, the singular values of gradient trajectories are highly concentrated, indicating that most of the variation lies in a few dominant directions, which reveals a pronounced low-rank structure. Furthermore, Fig. 2b shows that the gradient covariance exhibits strong anisotropy, suggesting that different directions contribute unequally to optimization. Such geometric structure implies that effective perturbations should align with these dominant and informative directions, rather than being isotropic or structure-agnostic. In addition, Fig. 2c and Fig. 2d illustrate that both the gradient magnitude and the underlying subspace evolve over time, indicating a non-stationary optimization process. However, existing perturbation strategies do not explicitly exploit this structured geometry, resulting in either inefficient exploration (in the case of random perturbations) or high computational overhead (in adversarial approaches). This raises a fundamental question: *how can we design efficient perturbations that respect the underlying geometry of gradient dynamics?*

To address this challenge, we propose ***S**ubspace **P**erturbation with **A**daptive **S**ampling* (***SPAS***), a geometry-aware perturbation framework that explicitly models the structured dynamics of gradients in HFL. Our approach is built upon three key components: *representation*, *generation*, and *adaptation*, each designed to capture a key property of gradient dynamics in HFL. First, to account for the low-rank structure of gradient trajectories, we represent perturbations within a low-dimensional subspace that captures the dominant directions of gradients. This representation restricts perturbations to informative directions and avoids unnecessary exploration in the full parameter space. Second, to model the anisotropic nature of gradient dynamics, we generate perturbations through a structure-aware sampling strategy in the latent space, where the sampling distribution is aligned with the covariance of projected gradients. This enables perturbations to preserve directional importance by allocating larger variance to more informative directions. Third, to capture the evolving structure of gradient dynamics, we introduce an adaptive mechanism that dynamically adjusts both the magnitude and the structure

of perturbations during training. Specifically, the perturbation scale is modulated according to the optimization state, while the subspace is incrementally updated to track the changing geometry of gradients. Extensive experiments demonstrate that SPAS consistently improves generalization across diverse heterogeneous settings, achieving gains of up to 4.49% and 2.39% under *feature-skew* and *label-skew* scenarios, respectively.

Our main contributions are summarized as follows:

- We identify that gradient dynamics in heterogeneous federated learning exhibit a low-rank, anisotropic, and evolving structure, providing a geometric perspective for understanding generalization and motivating structure-aware perturbation design.
- We propose a geometry-aware perturbation framework consisting of three components: *representation*, *generation*, and *adaptation*. Specifically, perturbations are represented within a low-dimensional subspace to restrict exploration to informative directions; generated via covariance-aligned sampling in the latent space to preserve directional importance; and dynamically adapted through magnitude scaling and subspace updates to track evolving optimization dynamics.
- We conduct extensive experiments across diverse heterogeneous settings, demonstrating consistent and robust improvements over strong baselines and validating the effectiveness of the proposed approach.

2 Related Work

Due to space limitations, we provide a brief overview here, details are deferred to the appendix.

Traditional Federated Learning. FedAvg [37] first introduced server-side aggregation of local updates, but performs poorly on non-IID data. Subsequently, FedProx [30], SCAFFOLD [21], and FedNTD [1] mitigate this issue. To further improve generalization, FedSAM [42] and MoFedSAM [42] introduce adversarial perturbations, but incur additional gradient computations. In contrast, FedSWA [32] and FedGloss [3] reduce overhead, but are limited to homogeneous models and cannot support personalized clients. We refer to these methods as traditional federated learning (TFL).

Heterogeneous Federated Learning. Existing methods fall into three categories: knowledge distillation [15, 25], which relies on public server-side data that is often difficult to obtain in practice, prototype-based methods [58, 45, 17], which share semantic information via global prototypes, and auxiliary model-based methods [47, 43, 53], which exchange information through auxiliary models. Although effective, these methods tend to produce client-specific models that are prone to overfitting.

Generalization and Flat Minimum. Improving generalization via flat minima was first emphasized in [5, 14, 19]. Existing methods fall into two branches. Sharpness-Aware Minimization (SAM)-based methods [9] introduce adversarial perturbations via a minimax formulation but roughly double the computational cost [18, 7]. Random perturbation methods [64, 2, 20] inject noise into model weights, but often underperform due to ignoring gradient geometry and statistics [27, 62, 40]. In contrast, our method simulates gradient-aligned perturbations, achieving comparable benefits with lower cost.

3 Methodology

3.1 Problem Definition

We consider a heterogeneous federated learning (HFL) setting with K clients, where both data distributions and model architectures may vary across clients. Each client k holds a private dataset $\mathcal{D}_k = \{(x_i, y_i)\}_{i=1}^{N_k}$ and trains a local model accordingly. Data heterogeneity can be characterized by the client-specific distribution $P_k(x, y) = P_k(x|y)P_k(y)$, which typically manifests as *label skew* or *feature skew*. In this scenario, the global objective is defined as:

$$\arg\min_{w} L(w) \triangleq \frac{1}{K} \sum_{k\in[K]} L_k(w_t^k), \tag{1}$$

where $L_k(w_t^k) = \mathbb{E}_{(x_k,y_k)\sim\mathcal{D}_k}\left[\ell(w_t^k; (x_k, y_k))\right]$. However, due to model heterogeneity, global aggregation is generally infeasible, and local models are optimized in a largely decoupled manner. Combined with limited local data, this leads to severe overfitting in HFL.

A common approach to improve generalization is to introduce parameter perturbations during training:

$$w_{t+1}^k = w_t^k - \eta \nabla L_k(w_t^k + \epsilon_t^k). \tag{2}$$

Existing perturbation strategies can be broadly categorized into adversarial and random methods. Adversarial perturbations approximate the worst-case direction:

$$\epsilon_t^k = \underset{\|\epsilon_t^k\| \le \sigma}{\arg\max}\, L_k(w_t^k + \epsilon_t^k) \approx \sigma \frac{\nabla L_k(w_t^k)}{\|\nabla L_k(w_t^k)\|}, \tag{3}$$

which aligns with gradient directions but requires additional gradient evaluations, leading to high computational cost. In contrast, random perturbations sample isotropic noise:

$$\epsilon_t^k = \sigma \cdot \texttt{norm}(q_t^k), \quad q_t^k \sim \mathcal{N}(0, I), \tag{4}$$

which is computationally efficient but ignores the structural properties of gradients, resulting in suboptimal performance.

3.2 Subspace Perturbation with Adaptive Sampling

We present a geometry-aware subspace perturbation framework for HFL that is structured along three key aspects: *representation*, *generation*, and *adaptation*.

3.2.1 Representation

First, we represent parameter perturbations within a low-dimensional subspace that captures the intrinsic geometry of gradient dynamics. Specifically, the perturbation is expressed as:

$$\epsilon_t^k = U_t^k z_t^k, \tag{5}$$

where $U_t^k \in \mathbb{R}^{d \times r}$ denotes a gradient subspace with $r \ll d$, and $z_t^k \in \mathbb{R}^r$ represents latent perturbation coefficients. This representation is motivated by the observation that gradient trajectories exhibit a low-rank structure, as shown in Fig. 2a. By restricting perturbations to this subspace, we focus exploration on the most informative directions of the optimization landscape. Moreover, Eq. (5) decouples the structural component (captured by U_t^k) from stochastic generation (governed by z_t^k), providing a flexible foundation for perturbation design.

3.2.2 Generation

Given the subspace representation, perturbation generation reduces to sampling the latent variable z_t^k. A naive choice is Gaussian random sampling:

$$z_t^k \sim \mathcal{N}(0, I), \tag{6}$$

which assumes equal importance across all directions. However, this contradicts the anisotropic nature of gradient dynamics, as shown in Fig. 2b. Therefore, in order to seek a sampling distribution that preserves gradient statistics, we analyzed how perturbations affect the optimization process.

Second-order Statistics. Under a second-order Taylor expansion of the loss, we obtain:

$$L(w + \epsilon) \approx L(w) + \nabla L(w)^\top \epsilon + \frac{1}{2} \epsilon^\top H(w)\, \epsilon, \tag{7}$$

where $H(w)$ denotes the Hessian matrix. For stochastic perturbations with zero mean, (*i.e.*, $\mathbb{E}[\epsilon] = 0$), the expected first-order term vanishes, and the dominant effect is governed by second-order statistics:

$$\mathbb{E}[L(w + \epsilon)] \approx L(w) + \frac{1}{2} \operatorname{Tr}\left(H(w)\ \operatorname{Cov}(\epsilon)\right). \tag{8}$$

This shows that the expected effect of perturbations is fully characterized by their covariance. Therefore, matching second-order statistics provides a principled and sufficient criterion for designing effective perturbations.

Principled Sampling via Covariance Matching. Motivated by the above observation, we design perturbations by matching their covariance to that of gradients. Once the perturbation is represented as $\epsilon_t^k = U_t^k z_t^k$, the key question becomes how to choose the latent distribution of z_t^k. We formalize this principle as the following theorem.

Theorem 1 (Optimal subspace sampling)**.** *Let $g_t^k \in \mathbb{R}^d$ be a random gradient vector with covariance matrix $C_g = \mathbb{E}[g_t^k (g_t^k)^\top]$. Let $U_t^k \in \mathbb{R}^{d \times r}$ be an orthonormal subspace basis. Consider perturbations of the form $\epsilon_t^k = U_t^k z_t^k$, where $z_t^k \in \mathbb{R}^r$ has covariance $\Sigma_z = \mathbb{E}[z_t^k (z_t^k)^\top]$. The covariance mismatch*

between gradients and perturbations is defined as:

$$\mathcal{E}(\Sigma_z) = \left\| C_g - U_t^k \Sigma_z (U_t^k)^\top \right\|_F . \tag{9}$$

Then the optimal latent covariance minimizing this mismatch is

$$\Sigma_z^* = (U_t^k)^\top C_g U_t^k. \tag{10}$$

In particular, when U_t^k spans the top-r eigenspace of C_g, we obtain $\Sigma_z^* = \mathrm{diag}(\lambda_t^k)$, where λ_t^k denotes the corresponding top-r eigenvalues.

Theorem 1 shows that, under the subspace perturbation representation, the optimal latent distribution is obtained by matching the projected covariance of gradients. This leads to the sampling rule:

$$z_t^k \sim \mathcal{N}\left(0, \sigma^2 \,\mathrm{diag}(\lambda_t^k)\right), \tag{11}$$

which assigns higher variance to directions with greater gradient energy, preserving directional importance while avoiding the overhead of adversarial perturbations.

3.2.3 Adaptation

To ensure effectiveness across different training stages, we introduce an adaptive mechanism that operates along two complementary aspects: *magnitude adaptation* and *structure adaptation*. Magnitude adaptation modulates the overall perturbation energy, while structure adaptation maintains alignment with the evolving gradient geometry, together enabling both effective exploration and consistent directional guidance.

Magnitude Adaptation. We adapt the *magnitude* of perturbations according to the optimization state. As the gradient norm evolves (see Fig. 2c), large norms indicate high-curvature or under-explored regions, where stronger perturbations promote exploration and help escape sharp minima. In contrast, small norms suggest near convergence, where excessive perturbations can destabilize training. Therefore, a fixed perturbation scale is suboptimal across different training stages.

Using the orthonormality of U_t^k, we have:

$$\|\epsilon_t^k\|^2 = \|z_t^k\|^2. \tag{12}$$

This allows us to directly control the perturbation magnitude through the distribution of z_t^k. We therefore incorporate gradient magnitude into sampling by introducing a global scaling factor:

$$z_t^k \sim \mathcal{N}\left(0, \sigma^2 \frac{\|g_{t-1}^k\|^2}{\sum_{j=1}^r \lambda_t^{k,j}} \,\mathrm{diag}(\lambda_t^k)\right), \tag{13}$$

which adaptively adjusts the perturbation energy to match the current optimization state.

Importantly, this scaling preserves the relative covariance structure from Theorem 1, introducing only a global factor $\alpha_t^k = \sigma^2 \|g_{t-1}^k\|^2 / \sum_{j=1}^r \lambda_t^{k,j}$, so that $\mathrm{Cov}(z_t^k) = \alpha_t^k \,\mathrm{diag}(\lambda_t^k)$. For any directions i and j, the variance ratio remains $\mathrm{Var}(z_{t,i}^k)/\mathrm{Var}(z_{t,j}^k) = \lambda_t^{k,i}/\lambda_t^{k,j}$. Therefore, perturbations remain aligned with the directional importance of gradients while dynamically controlling exploration strength. In this way, `SPAS` decouples the *shape* of the perturbation distribution, determined by gradient statistics, from its *magnitude*, adapted to training dynamics.

Structure Adaptation. Besides the perturbation magnitude, the subspace itself should also adapt to the evolving optimization trajectory. Although the gradient dynamics exhibit a low-rank structure, the corresponding dominant directions are not fixed throughout training. As local models move across the loss landscape, the principal directions of gradients may gradually shift, as illustrated in Fig. 2d. Therefore, using a fixed subspace may lead to stale perturbation directions that no longer reflect the current gradient geometry. To address this issue, we maintain a client-specific dynamic subspace and update it online using streaming gradients.

For each client k, let $U_t^k \in \mathbb{R}^{d\times r}$ denote the local subspace at step t, initialized as a random orthogonal matrix with rank r. Given the current gradient g_t^k, we first project it onto this subspace:

$$p_t^k = (U_t^k)^\top g_t^k \in \mathbb{R}^r. \tag{14}$$

The projection coefficient p_t^k quantifies the alignment of the current gradient with each basis direction, and is used to update directional importance via an exponential moving average:

$$\lambda_{t+1}^k = \lambda_t^k + \eta_\lambda \left(p_t^k \odot p_t^k - \lambda_t^k\right), \tag{15}$$

where $\odot$ denotes the Hadamard product and η_λ is the update rate. The resulting λ_t^k estimates the energy associated with each subspace direction and is used to guide adaptive sampling. To update the subspace basis, we adopt an online Oja-style rule:

$$\tilde{U}_{t+1}^k = U_t^k + \eta_s \left(g_t^k (p_t^k)^\top - U_t^k p_t^k (p_t^k)^\top \right), \tag{16}$$

where η_s controls the subspace update speed. The first term incorporates the current gradient into the subspace, while the second term removes the component already explained by the current basis, preventing redundant updates and encouraging the basis to track new gradient directions.

Since the updated basis $\tilde{U}_{t+1}^k$ may lose orthonormality, we apply orthonormalization to maintain a stable subspace:

$$u_1 = \frac{\tilde{u}_1}{\|\tilde{u}_1\|_2}, \quad v_j = \tilde{u}_j - \sum_{i=1}^{j-1} (u_i^\top \tilde{u}_j) u_i, \quad u_j = \frac{v_j}{\|v_j\|_2}. \tag{17}$$

The updated subspace is then formed as:

$$U_{t+1}^k = [u_1, u_2, \ldots, u_r] \in \mathbb{R}^{d \times r}. \tag{18}$$

This incremental update avoids storing historical gradients or performing expensive batch SVD, while allowing the subspace to continuously track the evolving geometry of gradient trajectories. As a result, the sampled perturbations remain aligned with informative directions throughout training. The overall framework is summarized in Alg. 1.

Algorithm 1 Subspace Perturbation with Adaptive Sampling (`SPAS`)

Input: Incremental subspace U_t^k, subspace rank r, update interval s, perturbation magnitude σ, gradient g_{t-1}^k, model parameter w_t^k, and perturbation layer p.

1: Low dimensional latent space adaptive sampling z_t^k by Eq. (13).
2: Parameter perturbation generation $\epsilon_t^k = U_t^k z_t^k$.
3: **if** $\texttt{mod}(t, s) == 0$ **then**
4: Incremental update of orthogonal subspace $U_{t+1}^k = [u_1, u_2, \ldots, u_r]$.
5: Update the squared singular values λ_{t+1}^k of the subspace by Eq. (15).
6: **end if**

Output: Local model parameter perturbation ϵ_t^k.

3.3 Discussion

Through this design, `SPAS` generates perturbations that are both statistically aligned and dynamically scaled, improving local generalization without the double gradient computation required by adversarial perturbations. The additional cost mainly comes from subspace updates and orthonormalization, with complexity $\mathcal{O}(dr^2)$. Since r is typically small and the update does not involve backpropagation, this overhead is negligible compared with local model training. To further reduce cost, we update the subspace periodically and apply perturbations only to selected layers.

4 Theoretical Analysis

To further understand the effectiveness of the proposed method, we provide theoretical analysis from optimization and generalization perspectives.

Assumption 1. *The loss is L-smooth,* $\|\nabla L_k(w_t^k) - \nabla L_k(v_t^k)\|_2 \leq L\|w_t^k - v_t^k\|_2, \ \forall w_t^k, v_t^k \in \mathbb{R}^d$.

Assumption 2. *The stochastic gradient for each client* $g_k(w_t^k) = \nabla L_k(w_t^k, \xi)$ *has variance bounded by* S, $\mathbb{E}[\|g_k(w_t^k) - \nabla L_k(w_t^k)\|_2^2] \leq S, \forall w_t^k \in \mathbb{R}^d$.

Assumption 3. *The stochastic gradient is bounded by* γ, $\mathbb{E}[\|g_k(w_t^k)\|_2^2] \leq \gamma^2, \forall w_t^k \in \mathbb{R}^d$.

Theorem 2. *Under Assumptions 1–3, consider* `SPAS` *with SGD as the base optimizer, using learning rate* η *to update the model parameters in Eq. (2). By setting* $\eta = \frac{\eta_0}{\sqrt{T}} \leq 1/L$ *and the perturbation radius* $\sigma_t = \frac{\sigma_0}{\sqrt{t}}$, *we obtain:*

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\|\nabla L_k(w_t^k)\|^2 \leq \frac{2\Delta}{\eta_0\sqrt{T}} + \frac{(2LS + L^3\sigma_0^2\gamma^2)\eta_0}{\sqrt{T}} + \frac{L^2\sigma_0^2\gamma^2 \log T}{\sqrt{T}}, \tag{19}$$

where $\Delta = \mathbb{E}[L_k(w_0^k) - L_k(w_*^k)]$, *and* T *denotes the total number of local update steps.*

Theorem 3. *Assume that the loss function is bounded by* L. *Additionally, we consider a virtual global data domain* $\mathcal{T}$ *and a local data domain* $\mathcal{T}_k$. *Let* $\mathcal{H}$ *be the hypothesis space with VC dimension* v *and parameter space* Ω, *and denote the domain divergence by* $d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{T}, \mathcal{T}_k)$. *Assume* $\epsilon \sim \mathcal{N}(0, \sigma^2 R)$ *with* $R = U\,\mathrm{diag}(\lambda)U^\top$. *Then, for any* $\delta \in (0,1)$, *with probability at least* $1-\delta$, *for all* $w \in \Omega$,

$$L_{\mathcal{T}_k}(w) \le L_{\mathcal{T}}(w) + \frac{\sigma^2\beta}{2}\tilde{\lambda}_1 + \frac{L}{2\sqrt{n}}\sqrt{r\log\left(\frac{\beta}{r} + \frac{\|x\|^2}{r\sigma^2}\right) - \alpha + 2\log\frac{1}{\delta} + \mathcal{O}(\log(n+r))} + C_k, \tag{20}$$

where $C_k = \sqrt{4/n(v\log(2en/v) + \log 4/\delta)} + d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{T}, \mathcal{T}_k) + \lambda_k$, λ_k *is the optimal combined risk on* $\mathcal{T}_k$ *and* $\mathcal{T}$ *over* Ω, $\alpha = \log\det(R)$, $\beta = \mathrm{tr}(R)$, n *is the number of samples,* $r < d$ *is the intrinsic perturbation dimension, and* $\tilde{\lambda}_1$ *is the largest eigenvalue of* $\nabla^2 L_{\mathcal{T}}(w)$.

From Theorem 2, it can be seen that when $\eta = \eta_0/\sqrt{T} \le 1/L$, SPAS can converge at the speed of $\mathcal{O}(\log T/\sqrt{T})$. Theorem 3 indicates that the generalization error is bounded by the dominant eigenvalues, the complexity term, the data discrepancy term C_k. In fact, we employ anisotropic weight perturbations that emphasize important directions, enabling effective exploration of informative modes and reducing the influence of dominant eigenvalues, as illustrated in Fig. 5.

5 Experiments

Datasets. For **feature skew**, we use *Office10* [10] and *Digit5* [63]. For **label skew**, we adopt *Cifar10* [23], *Cifar100* [23], *Flowers102* [41], and *Tiny-Imagenet* [24]. We simulate heterogeneity using pathological splitting (random class subsets per client) and Dirichlet splitting (Dir_L).

Model. We use *ResNet18* [13], *ResNet34* [13], *ResNet50* [13] and *ViT-B/16* [6], *ViT-B/32* [6] to simulate model heterogeneity, assigning each client k a model via (k mod 5). We also evaluate variants restricted to *ResNet* or *ViT*, as well as more challenging settings with up to eight architectures.

Evaluation Metrics. We evaluate performance at both the data and client levels, denoted as ***D.Avg*** and ***C.Avg***, respectively, with *D.Avg* as the primary metric. Let N_k^{corr} be the number of correctly classified samples for client k. The metrics are defined as:

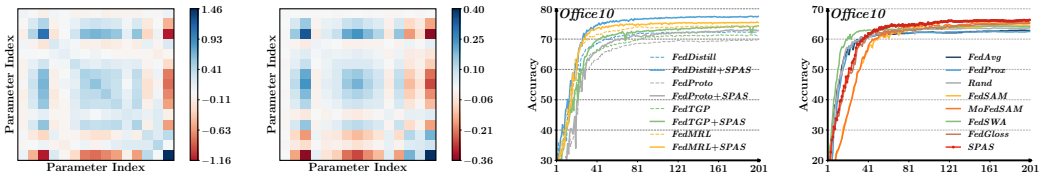
$$D.Avg = \frac{\sum_{k=1}^K N_k^{corr}}{\sum_{k=1}^K N_k}, \quad C.Avg = \frac{1}{K}\sum_{k=1}^K \frac{N_k^{corr}}{N_k}. \tag{21}$$

Implementation Details. We adopt SGD with a learning rate of 0.01, set the number of local epochs to 5 and clients to 20, and run 200 communication rounds, after which most methods show little or no improvement. The appendix details the dataset, model, and configuration.

5.1 Comparison to State-of-the-Arts

Method	*Office10*						*Digit5*						
	Amaz	*Calt*	*DSLR*	*Webc*	***C.Avg***	***D.Avg***	*SYN*	*USPS*	*SVHN*	*MNTM*	*MNT*	***C.Avg***	***D.Avg***
FML [43]	78.91	59.96	78.48	91.89	77.31	72.00	92.94	99.88	75.67	87.45	98.28	90.85	90.45
+SPAS	80.37	62.81	82.27	92.56	**79.51**	**74.13**	92.71	99.88	78.33	87.77	98.31	**91.40**	**91.03**
FedKD [47]	77.45	58.36	79.74	93.24	77.22	70.98	96.41	99.78	85.60	89.08	98.50	93.88	93.62
+SPAS	79.33	61.20	79.74	94.59	**78.72**	**73.11**	96.52	99.82	85.60	89.90	98.60	**94.09**	**93.84**
FedDistill [17]	79.54	60.14	82.27	91.89	78.46	72.56	93.25	99.85	77.62	86.98	98.21	91.19	90.81
+SPAS	83.71	66.19	83.54	93.24	**81.67**	**77.05**	93.44	99.90	80.05	87.08	98.29	**91.75**	**91.40**
FedProto [44]	77.24	56.40	73.41	95.94	75.75	69.95	92.69	99.79	75.29	86.77	98.02	90.52	90.11
+SPAS	81.41	60.32	74.68	93.92	**77.59**	**73.11**	93.81	99.87	77.30	87.15	97.98	**91.22**	**90.84**
FedMRL [53]	82.25	63.34	78.48	89.86	78.49	74.52	94.63	99.90	80.37	88.37	98.32	92.33	92.00
+SPAS	84.96	65.30	83.54	89.86	**80.92**	**76.74**	94.95	99.64	82.48	89.54	98.48	**93.02**	**92.73**
FedTGP [58]	78.07	60.85	68.35	93.24	75.13	71.61	94.34	99.77	77.88	87.37	98.01	91.48	91.11
+SPAS	83.71	62.45	70.88	91.21	**77.07**	**74.37**	94.88	99.86	79.57	87.37	98.09	**91.96**	**91.61**

Table 1: Performance comparison in the **feature-skew** scenarios. The highest average accuracy is in **bold**, and the best per-client accuracy is underlined. Please see details in Sec. 5.1.



(a) Gradient covariance (b) Perturbation covariance (c) Convergence curves of HFL (d) Convergence curves of TFL

Figure 3: **Visualization of Results: (a), (b)** Covariance of gradients and perturbations, respectively, with consistent relative magnitudes across positions. **(c)** Comparison of HFL with and without SPAS. **(d)** Comparison of FedAvg+SPAS with representative TFL methods. See details in Sec. 5.1.

Plug and Play Performance. In Tabs. 1 and 5, we evaluate various HFL methods enhanced with SPAS under *feature-skewed* and *label-skewed* settings. We use default parameters $\sigma = 1$, $s = 5$, and $r = 3$. The results show that most methods benefit from SPAS, achieving improvements of up to 4.49% and 2.39% in the two settings, respectively. This indicates that SPAS alleviates overfitting, helps escape local minima, and enhances generalization. We further study the effects of local training epochs E and model architecture groups, with detailed results provided in the appendix.

Method	*C.Avg*	*D.Avg*	FLOPs	Time
Baseline	78.46±0.20	72.56±0.22 (↑ 0.00)	1×	33.50
Rand	79.97±0.25	75.47±0.31 (↑ 2.91)	1×	35.02
SAM [9]	81.93±0.24	77.52±0.18 (↑ 4.96)	2×	52.43
SPAS	81.87±0.21	77.72±0.25 (↑ 5.16)	1×	38.86

Table 2: **Different perturbation strategies:** FLOPs denote the theoretical cost of gradient evaluation, while Time represents the average runtime per round. See details in Sec. 5.1.

Influence of Perturbation Strategies. Tab. 2 reports the performance and computational cost of different perturbation strategies. SPAS achieves the best performance of 77.72%, a 5.16% improvement over the baseline, and is comparable to SAM. Moreover, SPAS avoids double gradient computations, leading to significantly lower cost than adversarial perturbation. This suggests that its sampled perturbations effectively exploit the statistical structure of gradients.

Extension to Traditional FL. In Tab. 3 and Fig. 3d, we incorporate SPAS into FedAvg [37] and compare it with representative TFL methods for improving generalization. In this setting, we use a global subspace U_t constructed from model updates, rather than the local subspace U_t^k in HFL, to reduce the inconsistency between local perturbations and global gradients. The results show gains of 4.87% and 3.24% in client-level and data-level metrics, respectively, while reducing computational cost from 46.62s to 29.21s. These findings demonstrate that SPAS effectively mitigates the discrepancy between global and local flatness while efficiently enhancing generalization.

Method	*Amaz*	*Calt*	*DSLR*	*Webc*	*C.Avg*	*D.Avg*	*Time*
FedAvg [37]	73.69	51.24	58.22	75.00	64.54	62.93	25.01
FedSAM [42]	74.53	53.20	67.08	75.67	67.62	64.74	46.23
FedSWA [32]	74.53	52.13	60.75	76.35	65.94	63.96	27.18
FedGloss [3]	76.40	54.80	62.02	72.29	66.38	65.46	29.11
MoFedSAM [42]	75.57	52.84	73.41	77.02	**69.72**	65.54	46.62
SPAS	76.41	53.91	69.62	77.70	69.41	**66.17**	29.21

Table 3: **Extension to TFL:** We evaluate the generalization performance of TFL methods enhanced with SPAS. See details in Sec. 5.1.

Visual Analysis. Figs. 3a and 3b illustrate that the generated perturbations align with the anisotropic structure of gradient dynamics. Furthermore, Figs. 3c and 3d show that SPAS enables stable convergence and improves the performance of various methods.

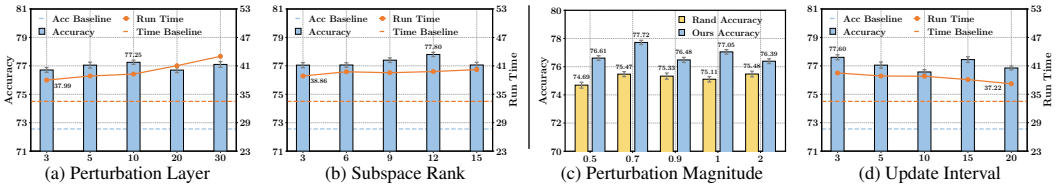
5.2 Ablation Study

Influence of Components. We conduct ablation studies in Tab. 4 to evaluate the effectiveness of low-rank representation (LRR), weighted generation (WG), and adaptive scaling (AS).

As shown in Tab. 4, incorporating low-rank representation and random sampling improves accuracy from 72.56% to 75.39%. Assigning different weights to sampling directions preserves the statistical structure of gradients and further enhances generalization. Adaptive scaling leverages the gradient norm to address the limited sensitivity of constant-amplitude perturbations to the model's convergence state, further improving performance from 76.66% to 77.05%.

Module			*Test Accuracy*					
LRR	WG	AS	*Amaz*	*Calt*	*DSLR*	*Webc*	*C.Avg*	*D.Avg*
			79.54	60.14	82.27	91.89	78.46	72.56
✓			82.04	64.41	84.81	90.54	80.45	75.39
✓	✓		82.04	66.72	83.54	93.24	81.39	76.66
✓		✓	83.29	65.30	82.27	91.21	80.52	76.18
✓	✓	✓	83.71	66.19	83.54	93.24	**81.67**	**77.05**

Table 4: **Ablation study** on the effectiveness of different components. Please see details in Sec. 5.2.



(a) Perturbation Layer (b) Subspace Rank (c) Perturbation Magnitude (d) Update Interval

Figure 4: **Hyperparameter Study:** We evaluate the effects of the perturbation layer p, rank r, and subspace update interval s on the performance and efficiency of SPAS, and further compare performance under different σ settings. Please refer to Sec. 5.2 for a detailed discussion.

Method	*Cifar10*				*Cifar100*				*Tiny-Imagenet*			
	Pat	$Dir_{0.1}$	$Dir_{0.3}$	$Dir_{0.5}$	Pat	$Dir_{0.01}$	$Dir_{0.1}$	$Dir_{0.5}$	Pat	$Dir_{0.01}$	$Dir_{0.1}$	$Dir_{0.5}$
FML [43]	82.44	84.51	69.74	58.46	52.19	62.03	39.11	22.92	30.20	42.76	28.31	14.53
+SPAS	**83.22**	**85.16**	**70.82**	**59.75**	**53.42**	**63.44**	**40.28**	**23.77**	**31.05**	**44.29**	**29.30**	**14.86**
FedKD [47]	84.47	86.29	71.77	61.39	52.71	63.73	40.37	23.60	30.46	43.12	28.42	15.26
+SPAS	**84.82**	**86.52**	**72.37**	**61.82**	**53.51**	**64.15**	**40.80**	**24.25**	**31.86**	**45.00**	**30.11**	**15.42**
FedDistill [17]	82.18	84.07	68.93	57.82	52.60	63.04	38.80	22.92	30.26	42.51	27.72	**14.70**
+SPAS	**82.66**	**84.92**	**70.28**	**58.60**	**54.28**	**64.23**	**39.90**	**23.34**	**31.24**	**44.60**	**28.64**	14.68
FedProto [44]	70.52	69.22	60.91	52.05	49.65	52.92	29.00	17.73	**25.73**	30.49	15.13	10.62
+SPAS	**71.56**	**70.93**	**62.34**	**53.44**	**51.05**	**54.02**	**30.37**	**18.29**	25.32	**30.70**	**15.48**	**10.77**
FedMRL [53]	82.40	84.49	69.18	57.93	52.33	63.57	38.27	21.71	28.72	42.91	28.43	12.20
+SPAS	**83.36**	**85.11**	**69.24**	**59.25**	**54.72**	**65.18**	**40.01**	**23.13**	**29.38**	**44.71**	**28.68**	**14.53**
FedTGP [58]	80.33	83.67	68.61	57.80	50.48	62.68	39.97	22.91	**26.20**	34.50	20.51	**11.10**
+SPAS	**80.83**	**84.20**	**69.80**	**57.90**	**52.14**	**63.27**	**40.63**	**23.81**	25.60	**35.07**	**20.81**	10.96

Table 5: Performance of SPAS combined with different algorithms under **label-skew** scenarios, where Pat denotes pathological heterogeneity. Please see details in Sec. 5.1.

Influence of Hyperparameters. Fig. 4 shows the impact of hyperparameters on the proposed method. Increasing the number of perturbation layers p or subspace rank r slightly raises computational cost compared to the $33.50s$ baseline, with the best performance of 77.80% achieved at $r = 12$. This suggests that p and r should be chosen to balance accuracy and efficiency.

Fig. 4c shows that under different perturbation levels σ, SPAS consistently outperforms random perturbation, indicating that it preserves the statistical structure of gradients. Fig. 4d shows that smaller update intervals improve performance at the cost of higher computation, as more frequent subspace updates capture richer gradient patterns, validating our *structure adaptation* strategy.

The Effect of Perturbation Layer Location. We evaluate perturbations at different model positions in Tab. 6. Results show consistent performance gains, with tail perturbations yielding the largest improvements, increasing accuracy from 72.56% and 68.93% to 77.05% and 70.28% on *Office10* and *Cifar10*, respectively. This is likely because the head is more sensitive to perturbations, while tail perturbations promote more robust and generalizable representations. Thus, we apply perturbations to the tail by default in subsequent experiments.

Method	*Office10*						*Cifar10*				*Cifar100*			
	Amaz	*Calt*	*DSLR*	*Webc*	***C.Avg***	***D.Avg***	Pat	$Dir_{0.1}$	$Dir_{0.3}$	$Dir_{0.5}$	Pat	$Dir_{0.01}$	$Dir_{0.1}$	$Dir_{0.5}$
Baseline	79.54	60.14	82.27	91.89	78.46	72.56	82.18	84.07	68.93	57.82	52.60	63.04	38.80	22.92
Random	78.91	62.45	83.54	93.91	79.71	73.66	82.17	84.33	68.39	57.32	52.66	63.16	39.57	22.60
Head	81.62	62.98	81.01	93.91	79.89	74.76	82.41	84.34	69.07	57.72	52.80	63.42	39.00	22.36
Tail	83.71	66.19	83.54	93.24	**81.67**	**77.05**	**82.66**	**84.92**	**70.28**	**58.60**	**54.28**	**64.23**	**39.90**	**23.34**

Table 6: Performance comparison of **perturbation locations**, where Random, Head, and Tail denote random selection, head-layer selection, and tail-layer selection, respectively. See details in Sec. 5.2.

Hessian Spectrum. Fig. 5 shows the Hessian spectrum of different strategies. SPAS exhibits smaller dominant eigenvalues, indicating convergence to flatter minima and improved generalization.

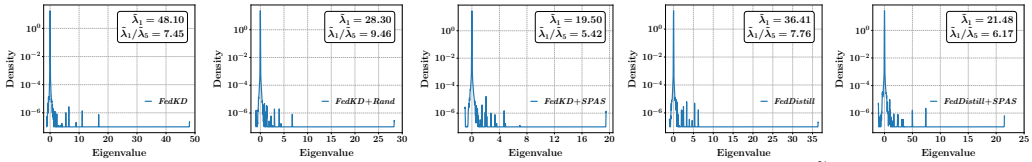


Figure 5: **Hessian spectrum** of local models under different methods. $\tilde{\lambda}_1$ denotes the largest eigenvalue, reflecting the sharpness of the model. Please see details in Sec. 5.2.

6 Conclusion

In this paper, we propose SPAS, a geometry-aware perturbation framework for mitigating overfitting in heterogeneous federated learning. Our approach leverages the low-rank and anisotropic structure of gradient dynamics to construct perturbations that are both statistically aligned and computationally efficient. By representing perturbations in a compact subspace, generating them via structure-aware sampling, and adapting their magnitude and structure throughout training, SPAS enables effective exploration while maintaining stable convergence. Extensive experiments demonstrate consistent improvements across diverse heterogeneous settings, validating the effectiveness of the proposed approach.

7 Acknowledgements and Disclosures

This research is supported by the National Key R&D Program of China (No. 2024YFE0100700). This research is also supported by the National Research Foundation, Singapore, under its AI Singapore Programme (AISG Award Nos. AISG3-RP-2024-033 and AISG3-RPGV-2025-019). The authors declare that they have no competing interests.

References

[1] Durmus Alp Emre Acar, Yue Zhao, Ramon Matas Navarro, Matthew Mattina, Paul N Whatmough, and Venkatesh Saligrama. Federated learning based on dynamic regularization. *arXiv preprint arXiv:2111.04263*, 2021. 1, 3, 14

[2] Devansh Bisla, Jing Wang, and Anna Choromanska. Low-pass filtering sgd for recovering flat optima in the deep learning optimization landscape. In *International Conference on Artificial Intelligence and Statistics*, pages 8299–8339. PMLR, 2022. 2, 3, 14

[3] Debora Caldarola, Pietro Cagnasso, Barbara Caputo, and Marco Ciccone. Beyond local sharpness: Communication-efficient global sharpness-aware minimization for federated learning. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 25187–25197, 2025. 3, 8, 14, 15, 16

[4] Niladri S Chatterji, Behnam Neyshabur, and Hanie Sedghi. The intriguing role of module criticality in the generalization of deep networks. *arXiv preprint arXiv:1912.00528*, 2019. 21

[5] Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In *International Conference on Machine Learning*, pages 1019–1028. PMLR, 2017. 2, 3, 14

[6] Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 7, 15, 17, 19

[7] Jiawei Du, Hanshu Yan, Jiashi Feng, Joey Tianyi Zhou, Liangli Zhen, Rick Siow Mong Goh, and Vincent YF Tan. Efficient sharpness-aware minimization for improved training of neural networks. *arXiv preprint arXiv:2110.03141*, 2021. 3, 14

[8] Gintare Karolina Dziugaite and Daniel M Roy. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. *arXiv preprint arXiv:1703.11008*, 2017. 21

[9] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. In *International Conference on Learning Representations*, 2020. 2, 3, 8, 14

[10] Boqing Gong, Yuan Shi, Fei Sha, and Kristen Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *2012 IEEE conference on computer vision and pattern recognition*, pages 2066–2073. IEEE, 2012. 7, 14, 15, 18

[11] Pengxin Guo, Runxi Wang, Shuang Zeng, Jinjing Zhu, Haoning Jiang, Yanran Wang, Yuyin Zhou, Feifei Wang, Hui Xiong, and Liangqiong Qu. Exploring the vulnerabilities of federated learning: A deep dive into gradient inversion attacks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025. 14

[12] Pengxin Guo, Shuang Zeng, Yanran Wang, Huijie Fan, Feifei Wang, and Liangqiong Qu. Selective aggregation for low-rank adaptation in federated learning. In *International Conference on Learning Representations*, volume 2025, pages 99003–99027, 2025. 1

[13] 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. 7, 15, 17, 18, 19

[14] Sepp Hochreiter and Jürgen Schmidhuber. Simplifying neural nets by discovering flat minima. *Advances in neural information processing systems*, 7, 1994. 3, 14

[15] Wenke Huang, Mang Ye, and Bo Du. Learn from others and be yourself in heterogeneous federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10143–10153, 2022. 1, 2, 3, 14

[16] Wenke Huang, Mang Ye, Zekun Shi, He Li, and Bo Du. Rethinking federated learning with domain shift: A prototype view. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16312–16322. IEEE, 2023. 1

[17] Eunjeong Jeong, Seungeun Oh, Hyesung Kim, Jihong Park, Mehdi Bennis, and Seong-Lyun Kim. Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data. *arXiv preprint arXiv:1811.11479*, 2018. 2, 3, 7, 9, 14, 15, 16, 17, 18

[18] Weisen Jiang, Hansi Yang, Yu Zhang, and James Kwok. An adaptive policy to employ sharpness-aware minimization. *arXiv preprint arXiv:2304.14647*, 2023. 3, 14

[19] Yiding Jiang, Behnam Neyshabur, Hossein Mobahi, Dilip Krishnan, and Samy Bengio. Fantastic generalization measures and where to find them. *arXiv preprint arXiv:1912.02178*, 2019. 3, 14

[20] Chi Jin, Praneeth Netrapalli, Rong Ge, Sham M Kakade, and Michael I Jordan. On nonconvex optimization for machine learning: Gradients, stochasticity, and saddle points. *Journal of the ACM (JACM)*, 68(2):1–29, 2021. 3, 14

[21] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International conference on machine learning*, pages 5132–5143. PMLR, 2020. 3, 14

[22] Zhiqiang Kou, Yucheng Xie, Hailin Wang, Junyang Chen, Jingq Wang, Ming-Kun Xie, Shuo Chen, Yuheng Jia, Tongliang Liu, and Xin Geng. Rankmatch: A novel approach to semi-supervised label distribution learning leveraging rank correlation between labels. *Advances in Neural Information Processing Systems*, 38:158074–158095, 2026. 14

[23] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 7, 15, 19

[24] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015. 7, 15

[25] Daliang Li and Junpu Wang. Fedmd: Heterogenous federated learning via model distillation. *arXiv preprint arXiv:1910.03581*, 2019. 2, 3, 14

[26] Qinbin Li, Zeyi Wen, Zhaomin Wu, Sixu Hu, Naibo Wang, Yuan Li, Xu Liu, and Bingsheng He. A survey on federated learning systems: Vision, hype and reality for data privacy and protection. *IEEE Transactions on Knowledge and Data Engineering*, 35(4):3347–3366, 2021. 1

[27] Tao Li, Qinghua Tao, Weihao Yan, Zehao Lei, Yingwen Wu, Kun Fang, Mingzhen He, and Xiaolin Huang. Revisiting random weight perturbation for efficiently improving generalization. *arXiv preprint arXiv:2404.00357*, 2024. 2, 3, 14

[28] Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. Ditto: Fair and robust federated learning through personalization. In *International conference on machine learning*, pages 6357–6368. PMLR, 2021. 19

[29] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE signal processing magazine*, 37(3):50–60, 2020. 1

[30] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450, 2020. 3, 14, 15, 16, 19

[31] Paul Pu Liang, Terrance Liu, Liu Ziyin, Nicholas B Allen, Randy P Auerbach, David Brent, Ruslan Salakhutdinov, and Louis-Philippe Morency. Think locally, act globally: Federated learning with local and global representations. *arXiv preprint arXiv:2001.01523*, 2020. 1

[32] Junkang Liu, Yuanyuan Liu, Fanhua Shang, Hongying Liu, Jin Liu, and Wei Feng. Improving generalization in federated learning with highly heterogeneous data via momentum-based stochastic controlled weight averaging. In *Forty-second International Conference on Machine Learning*, 2025. 3, 8, 14, 15, 16

[33] Yong Liu, Siqi Mai, Xiangning Chen, Cho-Jui Hsieh, and Yang You. Towards efficient and scalable sharpness-aware minimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12360–12370, 2022. 2

[34] Jianfeng Lu, Tao Huang, Shuqin Cao, Shujun Yu, Riheng Jia, and Minglu Li. Piece: Incentivizing personalized privacy-preserving for multi-version model marketplace in federated learning. *IEEE Transactions on Information Forensics and Security*, 2025. 1

[35] Haocheng Luo, Tuan Truong, Tung Pham, Mehrtash Harandi, Dinh Phung, and Trung Le. Explicit eigenvalue regularization improves sharpness-aware minimization. *Advances in Neural Information Processing Systems*, 37:4424–4453, 2024. 21

[36] Disha Makhija, Xing Han, Nhat Ho, and Joydeep Ghosh. Architecture agnostic federated learning for neural networks. In *International conference on machine learning*, pages 14860–14870. PMLR, 2022. 19

[37] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017. 3, 8, 14, 15, 16

[38] Peng Mi, Li Shen, Tianhe Ren, Yiyi Zhou, Xiaoshuai Sun, Rongrong Ji, and Dacheng Tao. Make sharpness-aware minimization stronger: A sparsified perturbation approach. *Advances in Neural Information Processing Systems*, 35:30950–30962, 2022. 14

[39] Peng Mi, Li Shen, Tianhe Ren, Yiyi Zhou, Tianshuo Xu, Xiaoshuai Sun, Tongliang Liu, Rongrong Ji, and Dacheng Tao. Systematic investigation of sparse perturbed sharpness-aware minimization optimizer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025. 14

[40] Thomas Möllenhoff and Mohammad Emtiyaz Khan. Sam as an optimal relaxation of bayes. *arXiv preprint arXiv:2210.01620*, 2022. 3, 14

[41] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian conference on computer vision, graphics & image processing*, pages 722–729. IEEE, 2008. 7, 15

[42] Zhe Qu, Xingyu Li, Rui Duan, Yao Liu, Bo Tang, and Zhuo Lu. Generalized federated learning via sharpness aware minimization. In *International conference on machine learning*, pages 18250–18280. PMLR, 2022. 2, 3, 8, 14, 15, 16

[43] Tao Shen, Jie Zhang, Xinkang Jia, Fengda Zhang, Gang Huang, Pan Zhou, Kun Kuang, Fei Wu, and Chao Wu. Federated mutual learning. *arXiv preprint arXiv:2006.16765*, 2020. 1, 3, 7, 9, 14, 15, 16, 17, 18

[44] Yue Tan, Guodong Long, Lu Liu, Tianyi Zhou, Qinghua Lu, Jing Jiang, and Chengqi Zhang. Fedproto: Federated prototype learning across heterogeneous clients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8432–8440, 2022. 7, 9, 14, 15, 16, 17, 18

[45] Yue Tan, Guodong Long, Jie Ma, Lu Liu, Tianyi Zhou, and Jing Jiang. Federated learning from pre-trained models: A contrastive learning approach. *Advances in neural information processing systems*, 35:19332–19344, 2022. 2, 3, 14

[46] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008. 18

[47] Chuhan Wu, Fangzhao Wu, Lingjuan Lyu, Yongfeng Huang, and Xing Xie. Communication-efficient federated learning via knowledge distillation. *Nature communications*, 13(1):2032, 2022. 2, 3, 7, 9, 14, 15, 16, 17, 18

[48] Xinghao Wu, Xuefeng Liu, Jianwei Niu, Guogang Zhu, Mingjia Shi, Shaojie Tang, and Jing Yuan. Tackling feature-classifier mismatch in federated learning via prompt-driven feature transformation. *Advances in Neural Information Processing Systems*, 38:73655–73691, 2026. 1

[49] Xinghao Wu, Xuefeng Liu, Jianwei Niu, Guogang Zhu, and Shaojie Tang. Bold but cautious: Unlocking the potential of personalized federated learning through cautiously aggressive collaboration. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 19318–19327. IEEE, 2023. 1

[50] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19, 2019. 1

[51] Mang Ye, Xiuwen Fang, Bo Du, Pong C Yuen, and Dacheng Tao. Heterogeneous federated learning: State-of-the-art and research challenges. *ACM Computing Surveys*, 56(3):1–44, 2023. 1

[52] Liping Yi, Gang Wang, Xiaoguang Liu, Zhuan Shi, and Han Yu. Fedgh: Heterogeneous federated learning with generalized global header. In *Proceedings of the 31st ACM International Conference on Multimedia*, pages 8686–8696, 2023. 1

[53] Liping Yi, Han Yu, Chao Ren, Gang Wang, Xiaoxiao Li, et al. Federated model heterogeneous matryoshka representation learning. *Advances in Neural Information Processing Systems*, 37:66431–66454, 2024. 1, 2, 3, 7, 9, 14, 15, 16, 17, 18, 19

[54] Xinping Yi, Gaojie Jin, Xiaowei Huang, and Shi Jin. Towards a unified pac-bayesian framework for norm-based generalization bounds. *arXiv preprint arXiv:2601.08100*, 2026. 21

[55] Jianqing Zhang, Yang Hua, Jian Cao, Hao Wang, Tao Song, Zhengui Xue, Ruhui Ma, and Haibing Guan. Eliminating domain bias for federated learning in representation space. *Advances in Neural Information Processing Systems*, 36, 2024. 23

[56] Jianqing Zhang, Yang Hua, Hao Wang, Tao Song, Zhengui Xue, Ruhui Ma, Jian Cao, and Haibing Guan. Gpfl: Simultaneously learning global and personalized feature information for personalized federated learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5041–5051, 2023. 1

[57] Jianqing Zhang, Yang Hua, Hao Wang, Tao Song, Zhengui Xue, Ruhui Ma, and Haibing Guan. Fedala: Adaptive local aggregation for personalized federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 11237–11244, 2023. 1

[58] Jianqing Zhang, Yang Liu, Yang Hua, and Jian Cao. Fedtgp: Trainable global prototypes with adaptive-margin-enhanced contrastive learning for data and model heterogeneity in federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 16768–16776, 2024. 1, 2, 3, 7, 9, 14, 15, 16, 17, 18, 19

[59] Jianqing Zhang, Yang Liu, Yang Hua, and Jian Cao. An upload-efficient scheme for transferring knowledge from a server-side pre-trained generator to clients in heterogeneous federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12109–12119, 2024. 1

[60] Xiangtao Zhang, Eleftherios Kofidis, Ruituo Wu, Ce Zhu, Le Zhang, and Yipeng Liu. Coupled tensor train decomposition in federated learning. *Pattern Recognition*, 170:112067, 2026. 1

[61] Xiangtao Zhang, Sheng Li, Ao Li, Yipeng Liu, Fan Zhang, Ce Zhu, and Le Zhang. Subspace constraint and contribution estimation for heterogeneous federated learning. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 20632–20642, 2025. 1, 15

[62] Yaowei Zheng, Richong Zhang, and Yongyi Mao. Regularizing neural networks via adversarial model perturbation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8156–8165, 2021. 3, 14

[63] Kaiyang Zhou, Yongxin Yang, Timothy Hospedales, and Tao Xiang. Learning to generate novel domains for domain generalization. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVI 16*, pages 561–578. Springer, 2020. 7, 14, 15, 18, 19

[64] Mo Zhou, Tianyi Liu, Yan Li, Dachao Lin, Enlu Zhou, and Tuo Zhao. Toward understanding the importance of noise in training neural networks. In *International Conference on Machine Learning*, pages 7594–7602. PMLR, 2019. 2, 3, 14

Further details on our work and results are provided in the appendix. Its contents are as follows:

- Appendix A: More detailed discussion of related work.
- Appendix B: More details about the configuration in the experiment.
- Appendix C: More comparative experiments and visualization results.
- Appendix D: Broader impacts of our proposed method.
- Appendix E: Limitations of our method.
- Appendix F: Proof of the convergence in Theorem 2.
- Appendix G: Proof of the generalization bound in Theorem 3.
- Appendix H: Pseudocode of the key function in the proposed method.

A Related Work

Traditional Federated Learning. FedAvg [37] first introduced local training with server aggregation but performs poorly under non-IID data. To address this, FedProx [30] adds regularization, SCAFFOLD [21] corrects client drift, and FedNTD [1] reduces bias via non-true distillation. To further improve generalization, FedSAM [42] introduces local adversarial perturbations, while MoFedSAM [42] incorporates global momentum to mitigate inconsistencies between local and global flat minima. Although these methods improve FedAvg, they often incur additional gradient computation costs. In contrast, FedSWA [32] and FedGloss [3] reduce overhead via stochastic weight averaging and global updates, but are limited to homogeneous models and cannot support personalized clients. We refer to these methods as traditional federated learning (TFL).

Heterogeneous Federated Learning. Heterogeneous federated learning addresses personalized client requirements by allowing different model architectures across clients [15, 58, 11]. Existing approaches can be broadly categorized into three groups: knowledge distillation-based [15, 25], prototype-based [44, 58, 45], and auxiliary model-based methods [47, 43]. For instance, FCCL [15] aligns client models by leveraging publicly available server-side data, which is often difficult to obtain in practice. Methods such as FedProto [44], FedTGP [58], and FedDistill [17] share semantic information via global prototypes or logits. Meanwhile, FedKD [47] and FML [43] use meta-learning with auxiliary models for information exchange, and FedMRL [53] further enhances local training with global representations. Although these methods perform well in their respective settings, they lack sufficient global aggregation, causing local models to focus on client-specific tasks and become more prone to local overfitting. We collectively refer to this line of work as heterogeneous federated learning (HFL).

Generalization and Flat Minimum. The effectiveness of improving model generalization by seeking flat minima was first highlighted in [5, 14, 19, 22]. Existing approaches can be broadly divided into two branches. The first is based on Sharpness-Aware Minimization (SAM) [9], which introduces adversarial perturbations and formulates a minimax problem to encourage parameters to reside in uniformly low-loss neighborhoods. However, this approach requires multiple gradient computations, leading to increased computational cost [18, 7]. To mitigate this issue, methods such as [39, 38] and [7] reduce the overhead, but at the expense of introducing bias through strategies like sparse masking and random sample selection. The second branch relies on random perturbations [64, 2, 20], which inject noise into model weights to help escape local optima. Nevertheless, due to the lack of awareness of the geometric and statistical properties of gradients, their generalization performance is often inferior to adversarial methods [27, 62, 40]. In contrast, our approach simulates gradient perturbations via real-time sampling, achieving comparable benefits at a lower computational cost.

B Experimental Details

B.1 Datasets

We evaluate `SPAS` on multiple image classification benchmarks under diverse heterogeneity settings. For feature distribution shift, we conduct experiments on *Office10* [10] and *Digit5* [63], comprising 10 classification tasks distributed across 4 and 5 clients, respectively, to simulate heterogeneous feature distributions. *Office10* consists of four domains: *Amazon (Amaz)*, *Caltech (Calt)*, *DSLR*, and

WebCam (Webc). *Digit5* includes five domains: *SVHN*, *USPS*, *SynthDigits (SYN)*, *MNIST-M (MNTM)*, and *MNIST*.

For label distribution shift, we perform experiments on *Cifar10* [23], *Cifar100* [23], *Flowers102* [41], and *Tiny-ImageNet* [24], involving a total of 20 clients. The data are partitioned to induce heterogeneous label distributions, corresponding to classification tasks with 10, 100, 102, and 200 classes, respectively, thus covering a wide range of task complexities. To simulate label heterogeneity, we consider two partitioning strategies. The first is a pathological setting, where each client is assigned randomly sampled subsets of classes (*e.g.*, 2, 10, 15, or 20 classes for datasets with 10, 100, 102, or 200 categories, respectively). The second employs a Dirichlet distribution (Dir_L) to generate non-IID data splits. Additionally, following [58], we divide each dataset into training and test sets with a ratio of 0.75/0.25. Detailed descriptions of the datasets and experimental configurations are provided in Tab. 7.

Dataset	Scenario	Modality	Class	Sites	Size
Office10 [10]	Feature skew	Image	10	4	5K
Digit5 [63]	Feature skew	Image	10	5	17K
Cifar10 [23]	Label skew	Image	10	20	60K
Cifar100 [23]	Label skew	Image	100	20	60K
Flowers102 [41]	Label skew	Image	102	20	8K
Tiny [24]	Label skew	Image	200	20	100K

Table 7: **Detailed descriptions** of the different datasets. Please see more details in Sec. B.1.

B.2 Heterogeneous Model Architecture

We consider five representative model architectures by default to simulate model heterogeneity across clients, namely *ResNet18* [13], *ResNet34* [13], *ResNet50* [13], *ViT-B/16* [6], and *ViT-B/32* [6], denoted as $\textit{HtM}^{\boldsymbol{I}}_{5M}$. Each client m is assigned a model according to $(m \bmod 5)$. For methods that rely on auxiliary models, we follow [58] and select the smallest model as the auxiliary model to reduce communication and computational overhead. To further evaluate the generality of SPAS across architectural families, we consider two restricted heterogeneous settings: $\textit{HtM}^{\boldsymbol{I}}_{2C}$, where all models are selected from the *ResNet* [13] family, and $\textit{HtM}^{\boldsymbol{I}}_{2T}$, where all models belong to the *ViT* [6] family. Moreover, to assess performance under more complex heterogeneity, we introduce $\textit{HtM}^{\boldsymbol{I}}_{8M}$, which involves a larger pool of model architectures. Detailed configurations of these settings, along with the corresponding datasets, are provided in Tab. 8.

$\textit{HtM}^{\boldsymbol{I}}_{5M}$: *Office10*					$\textit{HtM}^{\boldsymbol{I}}_{5M}$: *Digit5 & Cifar10 & Cifar100 & Flowers102 & Tiny*				
ResNet18	*ResNet34*	*ResNet50*	*ViT-B/16*	–	*ResNet18*	*ResNet34*	*ResNet50*	*ViT-B/16*	*ViT-B/32*
$\textit{HtM}^{\boldsymbol{I}}_{2C}$					$\textit{HtM}^{\boldsymbol{I}}_{2T}$				
ResNet18	*ResNet34*	–	–	–	*ViT-B/16*	*ViT-B/32*	–	–	–
$\textit{HtM}^{\boldsymbol{I}}_{8M}$									
MobileNetv2	*ResNet18*	*ResNet34*	*ResNet50*	*ResNet101*	*ResNet152*	*ViT-B/16*	*ViT-B/32*	–	–

Table 8: **Customized model architectures** and their paired datasets for various heterogeneous federated learning scenarios. Please see details in Sec. B.2.

B.3 Baseline Methods

We introduce SPAS into different federated learning algorithms and evaluate its performance gains, including FML [43], FedKD [47], FedDistill [17], FedProto [44], FedTGP [58], FedMRL [53], and FedSCE [61]. We also evaluate scenarios with data heterogeneity under the model-homogeneous setting and compare against state-of-the-art generalized TFL methods, including FedAvg [37], FedProx [30], FedSAM [42], MoFedSAM [42], FedGloss [3], and FedSWA [32].

First, we provide a brief overview of heterogeneous federated learning baselines.

- FML [43]: By introducing mutual learning, collaboration between auxiliary and local models is encouraged, improving knowledge exchange and local performance.
- FedKD [47]: By decomposing the auxiliary model via singular value decomposition (SVD) and exchanging factors, communication cost is reduced while enabling knowledge transfer.
- FedDistill [17]: Clients extract knowledge from global representations to improve local models, while the server aggregates local representations.

- FedProto [44]: Introduces category-level prototypes, which are aggregated on the server to create global prototypes for improved representation learning performance.
- FedTGP [58]: The server obtains a more generalized global prototype through contrastive learning techniques for improved representation learning.
- FedMRL [53]: Facilitates knowledge sharing by adaptively combining the representations of local and global models, improving collaborative learning.

Next, we introduce representative generalized federated learning baselines under the model-homogeneous setting considered in this work.

- FedAvg [37]: Traditional federated learning achieves knowledge transfer through model aggregation, thereby improving the generalization of the global model.
- FedProx [30]: By introducing regularization terms, the distance between the local model and the global model is constrained during training.
- FedSAM [42]: Adversarial perturbations are applied to local models to improve generalization, and the resulting models are aggregated to obtain the global model.
- MoFedSAM [42]: Building upon FedSAM [42], the global model update is incorporated into local gradients as momentum to accelerate convergence.
- FedGloss [3]: During local training, the global model update is used as a parameter perturbation direction to promote consistency between global and local flatness.
- FedSWA [32]: By incorporating stochastic weight averaging (SWA) into both local training and global aggregation, a more generalized global model can be obtained.

B.4 Detailed Configuration

In addition to the hyperparameter settings outlined in the main paper, we set the subspace and singular value update step sizes in SPAS to $\eta_s = 0.001$ and $\eta_\lambda = 0.01$, respectively. For the baselines, we follow the configurations recommended for each method in [58]. Specifically, for FML [43], we set the knowledge distillation coefficients to $\alpha = 0.5$ and $\beta = 0.5$. For FedKD [47], the learning rate of the auxiliary model is aligned with that of the local model (both set to 0.01), with $T_{\text{start}} = 0.95$ and $T_{\text{end}} = 0.95$. For FedDistill [17], we use $\gamma = 0.1$. For FedProto [44], we set $\lambda = 0.1$. For FedTGP [58], we adopt $\lambda = 0.1$ (consistent with FedProto), $\tau = 100$, and $S = 100$ (server training epochs), and set the feature dimension to 512 following the original implementation. Furthermore, since the configurations of these generalized TFL methods are not specified in [58], we perform a grid search over commonly used ranges to determine the hyperparameters. Specifically, for FedProx [30], we search the regularization coefficient $\mu \in \{0.01, 0.1, 0.2, 0.5\}$ and set $\mu = 0.1$. For FedSAM [42] and MoFedSAM [42], we search the perturbation magnitude over $\{0.2, 0.4, 0.6, 0.8, 1.0\}$ and set $\rho = 0.2$, and search the momentum coefficient $\beta \in \{0.2, 0.4, 0.6, 0.8\}$ and set $\beta = 0.2$. For FedSWA [32], we search $\rho \in \{0.05, 0.1, 0.2, 0.3\}$ and $\alpha \in \{0.5, 1, 1.5, 2\}$, and set $\rho = 0.1$ and $\alpha = 1.5$. For FedGloss [3], we search $\rho_s \in \{0.01, 0.1, 0.15\}$ and $\beta \in \{1, 2, 5, 10\}$, and set $\rho_s = 0.1$, $\beta = 5$, with $T_s = 0$. Additionally, for ease of reference, Tab. 9 summarizes the notations used in this paper along with their brief descriptions.

	Description		Description
K	Number of clients	T	Total update steps
U_t^k	Gradient subspace	λ_t^k	Squared singular values
E	Local epochs	g_t^k	Local gradient
η	Learning rate	η_λ	Singular value learning rate
σ	Perturbation magnitude	z_t^k	Sampling vector
N_k	Samples at client k	r	Subspace rank
s	Subspace update interval	I	Identity matrix
p	Perturbation layers	η_s	Subspace learning rate

Table 9: A list of notations and descriptions.

Implementation Details. For all datasets, we use a batch size of 64, except for *Tiny-Imagenet* (256). We adopt SGD with a learning rate of 0.01, set the number of local epochs to 5 and clients to 20, and run 200 communication rounds, after which most methods show little or no improvement. All experiments are conducted on an NVIDIA GeForce RTX 3090 GPU.

Method	HtM^{I}_{2C}						HtM^{I}_{2T}						HtM^{I}_{8M}	
	Amaz	*Calt*	*DSLR*	*Webc*	***C.Avg***	***D.Avg***	*Amaz*	*Calt*	*DSLR*	*Webc*	***C.Avg***	***D.Avg***	***C.Avg***	***D.Avg***
FML [43]	79.95	60.67	86.07	85.81	78.13	72.48	74.94	60.14	75.94	91.89	75.73	70.43	80.58	84.37
+SPAS	81.00	62.63	83.54	91.21	**79.60**	**74.21**	75.78	61.38	81.01	93.24	**77.86**	**71.77**	**81.35**	**85.17**
FedKD [47]	77.87	59.78	78.48	89.18	76.33	71.21	74.94	56.76	79.74	93.91	76.34	69.40	80.66	84.14
+SPAS	79.33	65.12	81.01	89.86	**78.83**	**74.37**	76.61	59.60	79.74	94.59	**77.64**	**71.37**	**81.78**	**84.86**
FedDistill [17]	79.74	62.45	84.81	87.83	78.71	73.34	75.57	58.18	78.48	91.21	75.86	69.87	80.71	84.43
+SPAS	81.62	63.52	84.81	89.18	**79.79**	**74.68**	75.99	59.07	81.01	92.56	**77.16**	**70.74**	**81.93**	**85.32**
FedProto [44]	77.03	58.89	78.48	85.81	75.06	70.11	74.53	58.36	77.21	91.21	75.33	69.48	80.29	84.18
+SPAS	79.54	61.03	75.94	87.83	**76.09**	**72.08**	75.36	58.54	79.74	92.57	**76.56**	**70.19**	**81.07**	**84.83**
FedMRL [53]	83.71	62.45	82.27	94.59	80.76	75.47	73.06	59.25	75.94	91.89	75.04	69.32	**75.99**	79.41
+SPAS	84.13	63.87	87.34	94.59	**82.49**	**76.58**	77.03	57.82	79.75	91.89	**76.63**	**70.43**	75.64	**80.52**
FedTGP [58]	79.95	61.74	78.48	85.81	76.50	72.48	74.73	59.25	77.21	91.21	75.61	69.95	76.35	81.15
+SPAS	82.25	64.94	78.48	86.48	**78.04**	**74.84**	74.53	60.14	78.48	91.22	**76.09**	**70.35**	**77.21**	**81.62**

Table 10: Performance of SPAS under different heterogeneous model architectures on *Office10* and *Cifar10*. Please see details in Sec. C.1.

C Additional Experiments

C.1 Impact of Model Architectures

Tab. 10 shows the performance of SPAS under different model architectures. Overall, SPAS consistently improves all baseline methods across heterogeneous settings, demonstrating strong generality. Under the HtM^{I}_{2C} and HtM^{I}_{2T} settings, SPAS achieves stable gains in both client average (*C.Avg*) and data average (*D.Avg*), with improvements of up to 3.16%, indicating its effectiveness across both CNN [13], and Transformer [6] architectures. Furthermore, in the more challenging HtM^{I}_{8M} setting, SPAS still yields improvements of up to 1.22% and 0.89%, respectively, highlighting its robustness to more complex model heterogeneity.

C.2 Local Epoch

Tab. 11a shows the impact of different local training epochs E on model performance. When $E = 1$, SPAS already brings notable improvements. For example, on FedTGP [58], performance increases from 67.43% and 66.08% to 69.89% and 69.72% in *C.Avg* and *D.Avg*, respectively. As E increases to 10 and 20, although more extensive local training may lead to overfitting and performance tends to stabilize, SPAS continues to provide consistent improvements, indicating its ability to escape suboptimal local minima. Overall, SPAS consistently enhances all baseline methods across different E, demonstrating strong robustness to varying levels of local computation.

Method	*E=1*		*E=10*		*E=20*	
	C.Avg	***D.Avg***	***C.Avg***	***D.Avg***	***C.Avg***	***D.Avg***
FML [43]	67.06	62.53	77.91	72.31	77.27	72.00
+SPAS	**67.96**	**65.77**	**79.68**	**74.68**	**79.06**	**73.82**
FedKD [47]	73.48	68.92	77.21	71.92	78.56	73.50
+SPAS	**73.65**	**72.63**	**79.59**	**74.61**	**81.34**	**76.10**
FedDistill [17]	74.96	71.21	79.03	73.50	79.71	74.44
+SPAS	**76.11**	**74.76**	**81.46**	**76.18**	**81.74**	**76.89**
FedProto [44]	**66.07**	64.74	76.44	70.42	78.68	72.94
+SPAS	65.20	**65.77**	**77.04**	**71.92**	**80.28**	**75.63**
FedTGP [58]	67.43	66.08	76.97	73.89	79.11	**76.41**
+SPAS	**69.89**	**69.72**	**77.52**	**74.29**	**80.26**	76.18
FedMRL [53]	69.72	61.47	78.48	74.53	78.29	74.45
+SPAS	**72.97**	**72.63**	**80.71**	**76.66**	**80.92**	**76.74**

(a) Different Local Epoch E

Method	*Flowers102*				
	Pat	$Dir_{0.01}$	$Dir_{0.1}$	$Dir_{0.3}$	$Dir_{0.5}$
FML [43]	49.36	71.22	49.09	33.65	24.06
+SPAS	**50.10**	**71.61**	**49.73**	**33.89**	**26.20**
FedKD [47]	49.12	71.37	49.58	33.75	25.42
+SPAS	**51.17**	**73.42**	**50.66**	**33.83**	**26.44**
FedDistill [17]	50.24	70.83	49.59	**33.85**	25.43
+SPAS	**51.36**	**72.59**	**50.32**	33.80	**27.17**
FedProto [44]	36.80	44.74	27.15	16.53	12.93
+SPAS	**38.89**	**46.35**	**28.56**	**17.22**	**14.44**
FedMRL [53]	50.34	70.78	48.71	33.51	25.66
+SPAS	**52.92**	**73.17**	**51.43**	**33.71**	**26.73**
FedTGP [58]	41.77	**55.74**	**34.84**	25.53	16.28
+SPAS	**43.91**	55.55	34.80	**25.86**	**16.92**

(b) Performance on *Flowers102*

Table 11: **Additional experiments: (a)** The impact of different local epochs. **(b)** Performance of different algorithms on *Flowers102*. Please see details in Sec. C.2.

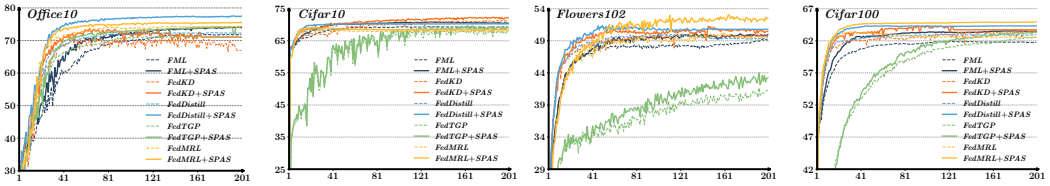


Figure 6: **Convergence curves** of test accuracy for different algorithms with and without the SPAS strategy under heterogeneous data settings. Please see details in Sec. C.3.

Method	*Office10*						*Digit5*						
	Amaz	*Calt*	*DSLR*	*Webc*	***C.Avg***	***D.Avg***	*SYN*	*USPS*	*SVHN*	*MNTM*	*MNT*	***C.Avg***	***D.Avg***
FML [43]	78.91	59.96	78.48	91.89	77.31	72.00	92.94	99.88	75.67	87.45	98.28	90.85	90.45
+SPAS_H	77.45	61.20	82.27	91.90	78.21	72.24	92.92	99.88	76.09	87.51	98.31	90.94	90.55
+SPAS_T	80.37	62.81	82.27	92.56	**79.51**	**74.13**	92.71	99.88	78.33	87.77	98.31	**91.40**	**91.03**
FedKD [47]	77.45	58.36	79.74	93.24	77.22	70.98	96.41	99.78	85.60	89.08	98.50	93.88	93.62
+SPAS_H	78.28	58.37	82.27	93.24	78.04	71.45	96.38	99.80	85.25	89.87	98.68	94.00	93.75
+SPAS_T	79.33	61.20	79.74	94.59	**78.72**	**73.11**	96.52	99.82	85.60	89.90	98.60	**94.09**	**93.84**
FedDistill [17]	79.54	60.14	82.27	91.89	78.46	72.56	93.25	99.85	77.62	86.98	98.21	91.19	90.81
+SPAS_H	81.62	62.98	81.01	93.91	79.89	74.76	93.62	99.88	78.00	87.03	98.19	91.35	90.97
+SPAS_T	83.71	66.19	83.54	93.24	**81.67**	**77.05**	93.44	99.90	80.05	87.08	98.29	**91.75**	**91.40**
FedProto [44]	77.24	56.40	73.41	95.94	75.75	69.95	92.69	99.79	75.29	86.77	98.02	90.52	90.11
+SPAS_H	77.66	58.71	72.15	94.59	75.78	70.90	93.92	99.74	75.89	87.05	98.03	90.93	90.55
+SPAS_T	81.41	60.32	74.68	93.92	**77.59**	**73.11**	93.81	99.87	77.30	87.15	97.98	**91.22**	**90.84**
FedMRL [53]	82.25	63.34	78.48	89.86	78.49	74.52	94.63	99.90	80.37	88.37	98.32	92.33	92.00
+SPAS_H	83.71	64.23	79.74	91.89	79.90	75.79	95.12	99.93	82.11	87.88	98.34	92.67	92.36
+SPAS_T	84.96	65.30	83.54	89.86	**80.92**	**76.74**	94.95	99.64	82.48	89.54	98.48	**93.02**	**92.73**
FedTGP [58]	78.07	60.85	68.35	93.24	75.13	71.61	94.34	99.77	77.88	87.37	98.01	91.48	91.11
+SPAS_H	80.79	61.92	70.88	93.91	76.88	73.34	94.27	99.82	78.60	87.06	98.01	91.55	91.19
+SPAS_T	83.71	62.45	70.88	91.21	**77.07**	**74.37**	94.88	99.86	79.57	87.37	98.09	**91.96**	**91.61**

Table 12: Impact of **parameter perturbation locations** on performance across different algorithms. SPAS_H and SPAS_T denote head and tail perturbations, respectively See details in Sec. C.5.

C.3 Convergence Performance

Fig. 6 illustrates the convergence curves with and without the SPAS strategy across different datasets. The results show that SPAS achieves faster convergence and better performance under both *feature-skew* and *label-skew* scenarios.

C.4 t-SNE Visualization

We visualize the feature representations of SPAS across different training rounds on the *Digit5* [63] dataset using t-SNE [46], as shown in Fig. 7. As training progresses, the learned features become increasingly well-separated and more structured, exhibiting clearer cluster boundaries and improved class separability.

(a) Initial (b) $R = 5$ (c) $R = 10$ (d) $R = 50$ (e) $R = 100$ (f) SPAS Final

Figure 7: t-SNE visualization of features extracted by *ResNet50* [13] on the *Digit5* [63] dataset, where R denotes the communication round.

C.5 Further Evaluation of Perturbation Locations

To examine whether applying perturbations to the tail generally outperforms applying them to the head across different methods, we further evaluate the impact of perturbation locations in Tab. 12. For example, on *Office10* [10] with FedMRL [53], applying perturbations to the head and tail

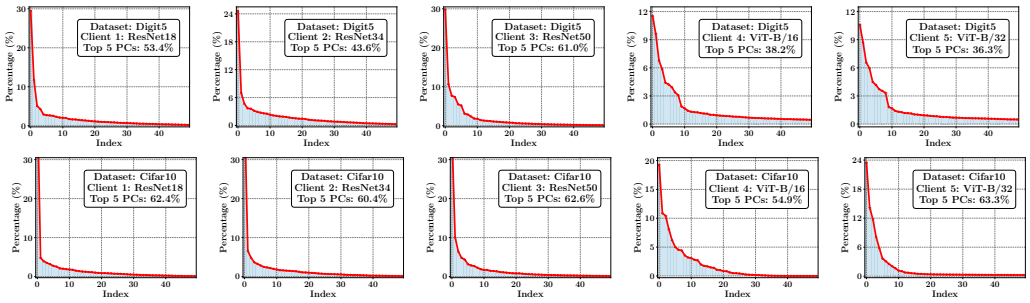


Figure 8: The **distribution** of **singular values** in gradient trajectories under different scenarios. Specifically, we vectorize the gradients, concatenate them across time steps, and compute the singular values.

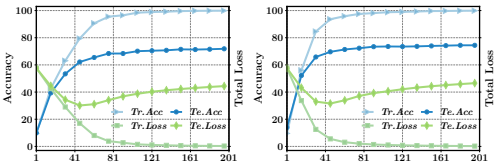
improves performance by 1.27% and 2.22%, respectively, compared to the baseline. Overall, the results indicate that tail perturbations consistently achieve better performance, aligning with our observations. This also supports the intuition that deeper layers of the model tend to be more stable, and introducing perturbations at these layers facilitates the learning of more robust and discriminative feature representations.

C.6 Low rank trajectory

To verify the low-rank nature of model update trajectories across scenarios, we compute the singular values of training trajectories for different architectures on *Digit5* [63] and *Cifar10* [23], as shown in Fig. 8. The results indicate that this low-rank, anisotropic structure consistently appears across settings. Moreover, it holds for both *ResNet* [13], and *ViT* [6] architectures.

D Broader Impacts

In heterogeneous federated learning frameworks, local overfitting remains a persistent challenge, as illustrated in Fig. 9. To address this, we first observe that the optimization trajectory of local models exhibits a low-rank and anisotropic structure. Building on this insight, we employ a subspace-based perturbation representation to efficiently explore critical directions, align with the geometric structure of gradient dynamics, and avoid the suboptimal performance of isotropic perturbations.



(a) Loss and accuracy curve (b) Loss and accuracy curve

Figure 9: Overfitting phenomena in representative methods FedTGP [58] and FedMRL [53].

Moreover, our method integrates seamlessly into traditional federated learning frameworks. By constructing a subspace for global model updates and sampling perturbations within it, it mitigates the mismatch between local and global loss flatness, leading to improved generalization. The method is also versatile. Explicit regularization is widely adopted to mitigate overfitting in federated learning [36, 28, 30], and combining SPAS with such methods can further enhance generalization performance.

E Limitations

Our method efficiently generates geometry-aware perturbations via representation, generation, and adaptation. As a plug-and-play approach, it can be readily integrated into existing FL frameworks to improve generalization. However, as our method mainly modifies the local training process, it does not explicitly address certain inherent challenges in the original framework, such as aggregation bias and knowledge transfer. These aspects could be further explored in future work.

F Proof of Theorem 2

Assumption 1. *Local loss function is L-Lipschitz smooth,* $\|\nabla L_k(w_t^k) - \nabla L_k(v_t^k)\|_2 \leq L\|w_t^k - v_t^k\|_2, \quad \forall w_t^k, v_t^k \in \mathbb{R}^d$.

Assumption 2. *The stochastic gradient for each client* $g_k(w_t^k) = \nabla L_k(w_t^k, \xi)$ *its variance is bounded by* σ^2, $\mathbb{E}[\|g_k(w_t^k) - \nabla L_k(w_t^k)\|_2^2] \leq S, \forall w_t^k \in \mathbb{R}^d$.

Assumption 3. *The expectation of the stochastic gradient is bounded by* γ, $\mathbb{E}[\|g_k(w_t^k)\|_2^2] \leq \gamma^2, \forall w_t^k \in \mathbb{R}^d$.

Theorem 2. *Under Assumptions 1–3, consider* SPAS *with SGD as the base optimizer, using learning rate* η *to update the model parameters in Eq. (2). By setting* $\eta = \frac{\eta_0}{\sqrt{T}} \leq 1/L$ *and the perturbation radius* $\sigma_t = \frac{\sigma_0}{\sqrt{t}}$, *we obtain:*

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\|\nabla L_k(w_t^k)\|^2 \leq \frac{2\Delta}{\eta_0\sqrt{T}} + \frac{(2LS + L^3\sigma_0^2\gamma^2)\eta_0}{\sqrt{T}} + \frac{L^2\sigma_0^2\gamma^2\log T}{\sqrt{T}}, \tag{22}$$

where $\Delta = \mathbb{E}[L_k(w_0^k) - L_k(w_*^k)]$, *and* T *denotes the total number of local update steps.*

Proof. Denote the perturbed iterate as $w_{t+1/2}^k = w_t^k + \epsilon_{r,t}$. We begin by analyzing the one-step descent behavior of the objective function. Under Assumption 1, we have:

$$\begin{aligned}
L_k(w_{t+1}^k) \leq & L_k(w_t^k) + \nabla L_k(w_t^k)^\top(w_{t+1}^k - w_t^k) + \frac{L}{2}\|w_{t+1}^k - w_t^k\|^2 \\
= & L_k(w_t^k) - \eta_t \nabla L_k(w_t^k)^\top g_k(w_{t+1/2}^k) + \frac{\eta_t^2 L}{2}\|g_k(w_{t+1/2}^k)\|^2 \\
= & L_k(w_t^k) - \eta_t \nabla L_k(w_t^k)^\top g_k(w_{t+1/2}^k) \\
& + \frac{\eta_t^2 L}{2}\left(\|g_k(w_{t+1/2}^k) - \nabla L_k(w_t^k)\|^2 - \|\nabla L_k(w_t^k)\|^2 + 2\nabla L_k(w_t^k)^\top g_k(w_{t+1/2}^k)\right) \\
= & L_k(w_t^k) - \frac{\eta_t^2 L}{2}\|\nabla L_k(w_t^k)\|^2 + \frac{\eta_t^2 L}{2}\|g_k(w_{t+1/2}^k) - \nabla L_k(w_t^k)\|^2 \\
& - (1 - L\eta_t)\eta_t \nabla L_k(w_t^k)^\top g_k(w_{t+1/2}^k) \\
\leq & L_k(w_t^k) - \frac{\eta_t^2 L}{2}\|\nabla L_k(w_t^k)\|^2 + \eta_t^2 L\|g_k(w_{t+1/2}^k) - \nabla L_k(w_{t+1/2}^k)\|^2 \\
& + \eta_t^2 L\|\nabla L_k(w_{t+1/2}^k) - \nabla L_k(w_t^k)\|^2 - (1 - L\eta_t)\eta_t \nabla L_k(w_t^k)^\top g_k(w_{t+1/2}^k) \\
\leq & L_k(w_t^k) - \frac{\eta_t^2 L}{2}\|\nabla L_k(w_t^k)\|^2 + \eta_t^2 L\|g_k(w_{t+1/2}^k) - \nabla L_k(w_{t+1/2}^k)\|^2 \\
& + \eta_t^2 L^3\|w_{t+1/2}^k - w_t^k\|^2 - (1 - L\eta_t)\eta_t \nabla L_k(w_t^k)^\top g_k(w_{t+1/2}^k).
\end{aligned} \tag{23}$$

Taking expectation on both sides and using Assumption 2 and 3, we obtain:

$$\begin{aligned}
\mathbb{E}[L_k(w_{t+1}^k)] \leq & \mathbb{E}[L_k(w_t^k)] - \frac{\eta_t^2 L}{2}\mathbb{E}\|\nabla L_k(w_t^k)\|^2 + \eta_t^2 L\mathbb{E}[\|g_k(w_{t+1/2}^k) - \nabla L_k(w_{t+1/2}^k)\|^2] \\
& + \eta_t^2 L^3\mathbb{E}[\|w_{t+1/2}^k - w_t^k\|^2] - (1 - L\eta_t)\eta_t\mathbb{E}[\nabla L_k(w_t^k)^\top g_k(w_{t+1/2}^k)] \\
\leq & \mathbb{E}[L_k(w_t^k)] - \frac{\eta_t^2 L}{2}\mathbb{E}\|\nabla L_k(w_t^k)\|^2 + \eta_t^2 LS + \eta_t^2 L^3\sigma^2\gamma^2 \\
& - (1 - L\eta_t)\eta_t\mathbb{E}[\nabla L_k(w_t^k)^\top g_k(w_{t+1/2}^k)].
\end{aligned} \tag{24}$$

We now focus on the inner product term. By adding and subtracting $g_k(w_t^k)$, we have:

$$\begin{aligned}
\mathbb{E}[\nabla L_k(w_t^k)^\top g_k(w_{t+1/2}^k)] = & \mathbb{E}\left[\nabla L_k(w_t^k)^\top\left(g_k(w_{t+1/2}^k) - g_k(w_t^k) + g_k(w_t^k)\right)\right] \\
= & \mathbb{E}\left[\|\nabla L_k(w_t^k)\|^2\right] + A,
\end{aligned} \tag{25}$$

where $A = \mathbb{E}\left[\nabla L_k(w_t^k)^\top \left(g_k(w_{t+1/2}^k) - g_k(w_t^k)\right)\right]$. Using Cauchy–Schwarz inequality and Assumption 1, we can bound A as:

$$\begin{aligned} A \leq & \mathbb{E}\left[\frac{1}{2}\|\nabla L_k(w_t^k)\|^2 + \frac{1}{2}\|\nabla L_k(w_{t+1/2}^k) - \nabla L_k(w_t^k)\|^2\right] \\ \leq & \frac{1}{2}\mathbb{E}\|\nabla L_k(w_t^k)\|^2 + \frac{L^2}{2}\|w_{t+1/2}^k - w_t^k\|^2 \\ \leq & \frac{1}{2}\mathbb{E}\|\nabla L_k(w_t^k)\|^2 + \frac{1}{2}L^2\sigma^2\gamma^2. \end{aligned} \tag{26}$$

Plugging Eq. (25) and (26) into Eq. (24), we obtain:

$$\begin{aligned} \mathbb{E}[L_k(w_{t+1}^k)] \leq & \mathbb{E}[L_k(w_t^k)] - \frac{\eta_t^2 L}{2}\mathbb{E}\|\nabla L_k(w_t^k)\|^2 + \eta_t^2 LS + \eta_t^2 L^3\sigma^2\gamma^2 \\ & - (1 - L\eta_t)\eta_t\mathbb{E}\left[\|\nabla L_k(w_t^k)\|^2\right] + (1 - L\eta_t)\eta_t\left(\frac{1}{2}\mathbb{E}\|\nabla L_k(w_t^k)\|^2 + \frac{L^2\sigma^2\gamma^2}{2}\right) \\ \leq & \mathbb{E}[L_k(w_t^k)] - \frac{\eta_t}{2}\mathbb{E}\|\nabla L_k(w_t^k)\|^2 + \eta_t^2 LS + \frac{1}{2}(1 + L\eta_t)\eta_t L^2\sigma^2\gamma^2. \end{aligned} \tag{27}$$

Taking summation over T steps, we have:

$$\begin{aligned} \frac{\eta_0}{2\sqrt{T}}\sum_{t=1}^{T}\mathbb{E}\|\nabla L_k(w_t^k)\|^2 \leq & \mathbb{E}[L_k(w_0^k)] - \mathbb{E}[L_k(w_t^k)] + (LS + \frac{1}{2}L^3\sigma_0^2\gamma^2)\eta_0^2\sum_{t=1}^{T}\frac{1}{T} \\ & + \frac{1}{2}L^2\sigma_0^2\gamma^2\eta_0\sum_{t=1}^{T}\frac{1}{t}. \end{aligned} \tag{28}$$

Finally, we have:

$$\begin{aligned} \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\|\nabla L_k(w_t^k)\|^2 \leq & \frac{2\left(\mathbb{E}[L_k(w_0^k)] - \mathbb{E}[L_k(w_T^k)]\right)}{\eta_0\sqrt{T}} + \frac{(2LS + L^3\sigma_0^2\gamma^2)\eta_0}{\sqrt{T}} + \frac{L^2\sigma_0^2\gamma^2\log T}{\sqrt{T}} \\ \leq & \frac{2\left(\mathbb{E}[L_k(w_0^k) - L_k(w_*^k)]\right)}{\eta_0\sqrt{T}} + \frac{(2LS + L^3\sigma_0^2\gamma^2)\eta_0}{\sqrt{T}} + \frac{L^2\sigma_0^2\gamma^2\log T}{\sqrt{T}}, \end{aligned} \tag{29}$$

where w_*^k is the optimal solution.

G Proof of Theorem 3

Proof. Inspired by [35, 4, 54, 8], we start from the PAC-Bayes generalization bound, which holds for any prior distribution P over the parameters and any posterior Q, with probability at least $1 - \delta$:

$$L_{\mathcal{D}}(Q) \leq L_{\mathcal{S}}(Q) + \sqrt{\frac{KL(Q\|P) + \log\frac{1}{\delta}}{2n}} \times L. \tag{30}$$

We consider an anisotropic posterior $Q = \mathcal{N}(x, \sigma^2 R)$ and an isotropic prior $P = \mathcal{N}(0, \sigma_P^2 I)$. The KL divergence between Q and P can be computed in closed form:

$$KL(Q\|P) = \frac{1}{2}\left[\frac{\sigma^2\,\mathrm{tr}(R) + \|x\|^2}{\sigma_P^2} - r + r\log\frac{\sigma_P^2}{\sigma^2} - \log\det(R)\right]. \tag{31}$$

To obtain the tightest bound, we optimize the KL divergence with respect to σ_P^2 for a fixed posterior Q. Taking derivative and solving yields:

$$\sigma_P^2 = \frac{\sigma^2\,\mathrm{tr}(R) + \|x\|^2}{r}. \tag{32}$$

Substituting this optimal value yields the following lower bound under $\log\det(R) \leq 0$:

$$KL(Q\|P) \geq \frac{r}{2}\log\left(\frac{\mathrm{tr}(R)}{r} + \frac{\|x\|^2}{r\sigma^2}\right). \tag{33}$$

Thus, the second term on the right-hand side of the PAC-Bayes generalization bound can be lower bounded as:

$$\sqrt{\frac{KL(Q\|P) + \log\frac{1}{\delta}}{2n}} \times L \geq \sqrt{\frac{r\log\left(1 + \frac{\|x\|^2}{\sigma^2\,\mathrm{tr}(R)}\right)}{4n}} \times L \geq L. \tag{34}$$

This term becomes trivially large (*i.e.*, exceeds L) whenever $\|x\|^2 > \sigma^2\,\mathrm{tr}(R)\big(\exp(4n/r) - 1\big)$. Therefore, in the remainder of the proof, we restrict our analysis to the non-trivial regime where $\|x\|^2 \leq \sigma^2\,\mathrm{tr}(R)\big(\exp(4n/r) - 1\big)$.

The optimal σ_P^2 depends on x, which violates the requirement that the prior must be independent of the data. To address this issue, we construct a countable family of priors and apply a union bound argument. Specifically, define $c = \sigma^2\big(1 + \exp(4n/r)\big), P_j = \mathcal{N}\big(0, c\exp\left(\frac{1-j}{r}\right) I\big), \mathfrak{P} := \big\{P_j : j = 1, 2, \dots\big\}$. We assign confidence levels $\delta_j = \frac{6\delta}{\pi^2 j^2}$ to each prior. Then, for each j, the PAC-Bayes bound holds with probability at least $1 - \delta_j$.

By noting that $\sum_{j=1}^{\infty} \frac{1}{j^2} = \frac{\pi^2}{6}$, a union bound ensures that the bound holds simultaneously for all $P_j \in \mathfrak{P}$ with probability at least $1 - \delta$. We then select the index:

$$j^* := \left\lfloor 1 - r\log\frac{\sigma^2\,\mathrm{tr}(R)/r + \|x\|^2/r}{c} \right\rfloor. \tag{35}$$

Following the same derivation, this choice guarantees the following scale matching property:

$$\sigma^2\frac{\mathrm{tr}(R)}{r} + \frac{\|x\|^2}{r} \;\leq\; \sigma_{j^*}^2 \;\leq\; \exp(1/r)\left(\sigma^2\frac{\mathrm{tr}(R)}{r} + \frac{\|x\|^2}{r}\right). \tag{36}$$

Using the above inequality, we can upper bound the KL divergence:

$$\begin{aligned}
KL(Q\|P_{j^*}) &= \frac{1}{2}\left[\frac{\sigma^2\,\mathrm{tr}(R) + \|x\|^2}{\sigma_{j^*}^2} - r + r\log\frac{\sigma_{j^*}^2}{\sigma^2} - \log\det(R)\right] \\
&\leq \frac{1}{2}\left[r - r + r\log\frac{\exp(1/r)\left(\sigma^2\,\mathrm{tr}(R)/r + \|x\|^2/r\right)}{\sigma^2} - \log\det(R)\right] \\
&= \frac{1}{2}\left[r\log\frac{\exp(1/r)\left(\sigma^2\,\mathrm{tr}(R)/r + \|x\|^2/r\right)}{\sigma^2} - \log\det(R)\right] \\
&= \frac{1}{2}\left[1 + r\log\left(\frac{\mathrm{tr}(R)}{r} + \frac{\|x\|^2}{r\sigma^2}\right) - \log\det(R)\right].
\end{aligned} \tag{37}$$

Meanwhile, the confidence term can be bounded as:

$$\begin{aligned}
\log\frac{1}{\delta_{j^*}} &= \log\frac{(j^*)^2\pi^2}{6\delta} \\
&\leq \log\frac{1}{\delta} + \log\frac{\pi^2}{6} + 2\log\left(1 + r\log\frac{c}{\sigma^2\,\mathrm{tr}(R)/r + \|x\|^2/r}\right) \\
&= \log\frac{1}{\delta} + \log\frac{\pi^2}{6} + 2\log\left(1 + r\log\frac{\sigma^2\big(1 + \exp(4n/r)\big)}{\sigma^2\,\mathrm{tr}(R)/r + \|x\|^2/r}\right) \\
&\leq \log\frac{1}{\delta} + \log\frac{\pi^2}{6} + 2\log\Big(1 + r\log\big(r + r\exp(4n/r)\big)\Big) \\
&\leq \log\frac{1}{\delta} + \log\frac{\pi^2}{6} + 2\log\Big(1 + r\big(1 + \frac{4n}{r} + \log(r)\big)\Big) \\
&\leq \log\frac{1}{\delta} + \log\frac{\pi^2}{6} + 4\log(2n + r).
\end{aligned} \tag{38}$$

Combining the above results, we obtain:

$$L_{\mathcal{D}}(Q) \leq L_{\mathcal{S}}(Q) + \sqrt{\frac{KL(Q\|P_{j^*}) + \log\frac{1}{\delta_{j^*}}}{2n}} \times L$$
$$\leq L_{\mathcal{S}}\Big(Q\Big) + \frac{L}{2\sqrt{n}}\sqrt{r\log\Big(\frac{\mathrm{tr}(R)}{r} + \frac{\|x\|^2}{r\sigma^2}\Big) + O(1) + 2\log\frac{1}{\delta} + 8\log(2n+r) - \log\det(R)}. \tag{39}$$

Next, we bound the expected empirical loss under perturbation:

$$\mathbb{E}_{\epsilon\sim\mathcal{N}(0,\sigma^2 R)}\Big[L_{\mathcal{D}}\Big(x+\epsilon\Big)\Big] \leq \mathbb{E}_{\epsilon}\Big[L_{\mathcal{S}}\Big(x+\epsilon\Big)\Big] + \text{same generalization term}. \tag{40}$$

Using the assumption $L_{\mathcal{D}}(w) \leq \mathbb{E}_{\epsilon}[L_{\mathcal{D}}(w+\epsilon)]$, we have:

$$L_{\mathcal{D}}(w) \leq \mathbb{E}_{\epsilon}[L_{\mathcal{S}}(w+\epsilon)] + \text{generalization term}. \tag{41}$$

Applying a second-order Taylor expansion, we have:

$$L_{\mathcal{S}}(w+\epsilon) \leq L_{\mathcal{S}}(w) + \epsilon^{\top}\nabla L_{\mathcal{S}}(w) + \frac{1}{2}\tilde{\lambda}_1(\nabla^2 L_{\mathcal{S}}(w))\|\epsilon\|^2. \tag{42}$$

Taking expectation over $\epsilon \sim \mathcal{N}(0,\sigma^2 R)$ and noting that the linear term vanishes, we obtain:

$$\mathbb{E}[L_{\mathcal{S}}(w+\epsilon)] \leq L_{\mathcal{S}}(w) + \frac{\sigma^2}{2}\tilde{\lambda}_1(\nabla^2 L_{\mathcal{S}}(w))\,\mathrm{tr}(R). \tag{43}$$

Combining all components, let $\alpha = \log\det(R)$ and $\beta = \mathrm{tr}(R)$, we obtain the final result:

$$L_{\mathcal{D}}(w) \leq L_{\mathcal{S}}(w) + \frac{\sigma^2\beta}{2}\tilde{\lambda}_1 + \frac{L}{2\sqrt{n}}\sqrt{r\log\Big(\frac{\beta}{r} + \frac{\|x\|^2}{r\sigma^2}\Big) + O(1) + 2\log\frac{1}{\delta} + 8\log(2n+r) - \alpha},$$
$$\leq L_{\mathcal{S}}(w) + \frac{\sigma^2\beta}{2}\tilde{\lambda}_1 + \frac{L}{2\sqrt{n}}\sqrt{r\log\Big(\frac{\beta}{r} + \frac{\|x\|^2}{r\sigma^2}\Big) - \alpha + 2\log\frac{1}{\delta} + O(\log(n+r))}. \tag{44}$$

Therefore, we obtain the following theorem.

Theorem 3. *Assume that the loss function is bounded by L. Additionally, we consider a global training dataset $\mathcal{S}$ and local data distributions $\mathcal{D}_k$. We further assume that $\epsilon \sim \mathcal{N}(0,\sigma^2 R)$, $R = U\,\mathrm{diag}(\lambda)U^{\top}$. For any $\delta \in (0,1)$ and $\sigma > 0$, with a probability over $1-\delta$ over the choice of $\mathcal{S} \sim \mathcal{D}$, we have:*

$$L_{\mathcal{D}}(w) \leq L_{\mathcal{S}}(w) + \frac{\sigma^2\beta}{2}\tilde{\lambda}_1 + \frac{L}{2\sqrt{n}}\sqrt{r\log\Big(\frac{\beta}{r} + \frac{\|x\|^2}{r\sigma^2}\Big) - \alpha + 2\log\frac{1}{\delta} + O(\log(n+r))}. \tag{45}$$

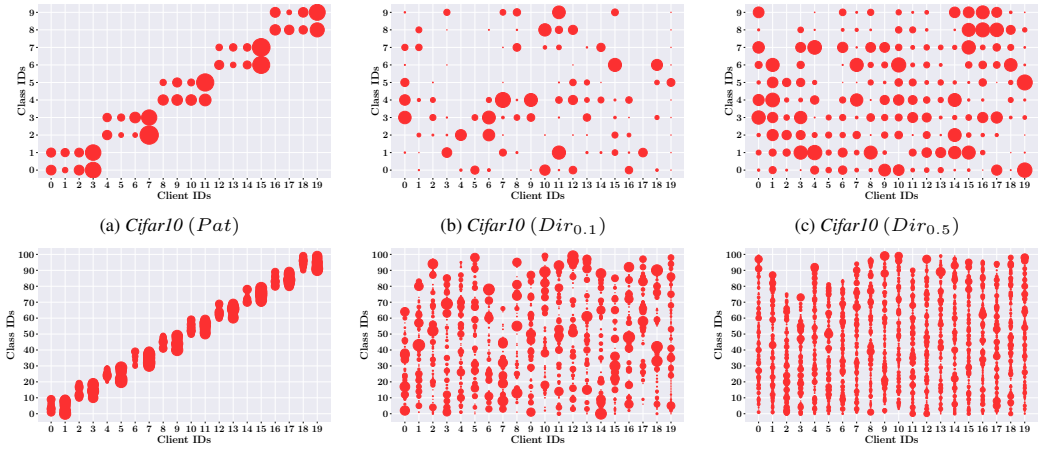
where $\alpha = \log\det(R)$ and $\beta = \mathrm{tr}(R)$, n denotes the number of training samples, $r < d$ represents the intrinsic dimensionality of the perturbation, and $\tilde{\lambda}_1$ denotes the largest eigenvalue of $\nabla^2 L_{\mathcal{S}}(w)$.

In heterogeneous federated learning (HFL), which emphasizes knowledge transfer from global to local models on the client side [55], the above theorem can be restated as follows.

Theorem 4. *Assume that the loss function is bounded by L. Additionally, we consider a virtual global data domain $\mathcal{T}$ and a local data domain $\mathcal{T}_k$. Let $\mathcal{H}$ be the hypothesis space with VC dimension v and parameter space Ω, and denote the domain divergence by $d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{T},\mathcal{T}_k)$. Assume $\epsilon \sim \mathcal{N}(0,\sigma^2 R)$ with $R = U\,\mathrm{diag}(\lambda)U^{\top}$. Then, for any $\delta \in (0,1)$, with probability at least $1-\delta$, for all $w \in \Omega$,*

$$L_{\mathcal{T}_k}(w) \leq L_{\mathcal{T}}(w) + \frac{\sigma^2\beta}{2}\tilde{\lambda}_1 + \frac{L}{2\sqrt{n}}\sqrt{r\log\Big(\frac{\beta}{r} + \frac{\|x\|^2}{r\sigma^2}\Big) - \alpha + 2\log\frac{1}{\delta} + O(\log(n+r))}$$
$$+ \sqrt{\frac{4}{n}\left(v\log\frac{2en}{v} + \log\frac{4}{\delta}\right)} + d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{T},\mathcal{T}_k) + \lambda_k. \tag{46}$$

where λ_k is the optimal combined risk on $\mathcal{T}_k$ and $\mathcal{T}$ over Ω, $\alpha = \log\det(R)$, $\beta = \mathrm{tr}(R)$, n is the number of samples, $r < d$ is the intrinsic perturbation dimension, and $\tilde{\lambda}_1$ is the largest eigenvalue of $\nabla^2 L_{\mathcal{T}}(w)$.



(a) *Cifar10* (Pat) (b) *Cifar10* ($Dir_{0.1}$) (c) *Cifar10* ($Dir_{0.5}$)

(d) *Cifar100* (Pat) (e) *Cifar100* ($Dir_{0.1}$) (f) *Cifar100* ($Dir_{0.5}$)

Figure 10: Data distribution of each client on *Cifar10* and *Cifar100* under different settings. The size of the circle indicates the number of samples.

H Pseudocode of SPAS

Algorithm 2 Pseudocode of SPAS

```
import torch
import torch.nn as nn
import numpy as np
import random
import math

   class Client(object):

      def __init__(self, args, id, train_samples, test_samples, **kwargs):
         # Initialize hyperparameters
         self.gap=args.gap
         self.layer_idx=args.layer_idx
         self.rank=args.rank
         self.sigma = args.sigma
         self.mode = args.mode
         # Select which layers to apply operations on
         self.select_layers()

      def select_layers(self):
         # Load model to obtain total number of layers
         model = load_item(self.role, 'model', self.save_folder_name)
         params = list(model.parameters())
         n_layers = len(params)

         # Select the last k layers (equivalent to [-k:])
         if self.mode == "head":
            self.layer_select = list(range(n_layers - self.layer_idx, n_layers))
         # Select the first k layers (equivalent to [:k])
         elif self.mode == "tail":
            self.layer_select = list(range(self.layer_idx))
         # Randomly select k layers
         elif self.mode == "random":
            self.layer_select = sorted(random.sample(range(n_layers), self.layer_idx))
         # Unknown mode: raise error
         else:
            raise ValueError(f"Unknown mode")

         return self.layer_select

      def initialize_projection_mat(self, model):
         params = list(model.parameters())
         total_dim = 0

         # Use selected layers instead of slicing
         for idx in self.layer_select:
            param = params[idx]
            total_dim += param.numel()
```

```
        # Random initialization
        random_matrix = torch.randn(total_dim, self.rank, device=self.device)
        # QR orthogonalization
        projection, _ = torch.linalg.qr(random_matrix)

        return projection

    def initialize_singular_values(self):
        # Initialize singular values
        return torch.ones(self.rank, device=self.device)

    def update_projection(self, model, prev_params, projection, singular_values, lr=1e-3, eta=0.01):
        with torch.no_grad():
            model_params = list(model.parameters())
            delta_list = []

            for idx in self.layer_select:
                param = model_params[idx]
                prev_param = prev_params[idx]

                # Approximate gradient
                delta = (param - prev_param).reshape(-1) / self.learning_rate
                delta_list.append(delta)

            # Concatenate into a single vector
            d = torch.cat(delta_list, dim=0)
            # Oja update
            U = projection
            y = U.T @ d
            U = U + lr * (torch.outer(d, y) - U @ torch.outer(y, y))
            # Re-orthogonalization
            # U = gram_schmidt(U)
            U, _ = torch.linalg.qr(U)
            # Update singular values
            singular_values = (singular_values + eta * (y * y - singular_values))

        return U, singular_values

    def gen_subspace_noise(self, model, prev_params, projection, singular_values, eps: float = 1e-12):
        # Weighted and scaled low-rank noise
        with torch.no_grad():

            params = list(model.parameters())
            delta_norm = 0
            for idx in self.layer_select:
                param = params[idx]
                prev_param = prev_params[idx]
                delta = (param - prev_param) / self.learning_rate
                delta_norm += delta.norm().item() ** 2

            # Compute scaling factor
            scale_factor = (delta_norm / (singular_values.sum() + eps))
            # Generate noise
            z = torch.randn(projection.shape[1], device=singular_values.device) * torch.sqrt(self.sigma**2
                 * scale_factor * singular_values)
            delta_w_subspace = projection @ z

        return delta_w_subspace

    def apply_subspace_noise(self, model, delta_w_subspace, sign: int = +1):
        # sign=+1 add noise; sign=-1 remove noise
        with torch.no_grad():

            all_params = list(model.parameters())
            params = [all_params[idx] for idx in self.layer_select if all_params[idx].requires_grad]

            off = 0
            for p in params:
                n = p.numel()
                p.add_(sign * delta_w_subspace[off:off + n].view_as(p))
                off += n
```

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The Introduction part summarizes our contribution, and the Related Work part discusses the relationship between our work and the related work.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss these limitations in the Sec. 3.3 and Appendix E.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: Some assumptions are explicitly stated in Sec. 4. All our proofs are in Appendix F and G.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: All the details about the experimental setup are described in Appendix B.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The pseudocode and Python implementations of the key functions are provided in the supplementary materials.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: All details have been stated in Appendix B.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We design some experiments by using three different random seeds and calculate their mean and standard deviation, as shown in Tab. 2.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide experimental equipment details and run times in Appendix B and Fig. 4.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: The research conducted in this paper complies with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We provide Broader Impacts in Appendix D.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We used existing datasets and cited them in Appendix B.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.