FEDSDG: STRUCTURE DECOUPLED GATING FOR PERSONALIZED FEDERATED FINETUNING

NODI Lab., Jilin University

ABSTRACT

Federated learning enables collaborative model training across distributed clients without sharing raw data, but its performance often degrades under strong data heterogeneity and limited communication budgets. Personalized federated learning addresses this challenge by allowing client specific adaptation, yet existing approaches typically rely on static model partitions, additional regularization, or complex optimization, and lack a structured and flexible mechanism to control personalization across different parts of the model. To bridge this gap, we propose FedSDG, a federated finetuning framework that models personalization as a structured and learnable deviation from a shared model. FedSDG decouples model adaptation into shared and client specific components and introduces lightweight, block level gates to regulate their interaction. This enables different layers to exhibit different degrees of personalization, providing fine grained control over specialization while keeping the number of trainable and communicated parameters small. FedSDG is compatible with standard parameter efficient finetuning methods such as LoRA and can be implemented on top of existing federated learning pipelines with minimal changes. Experiments under heterogeneous data distributions show that FedSDG improves client level performance and training stability compared to existing baselines, particularly in strongly non-IID settings, while maintaining low communication and computation overhead.

1 INTRODUCTION

Federated learning enables multiple clients to collaboratively train a model without sharing raw data, which is essential for privacy sensitive and large scale applications such as mobile and edge intelligence (McMahan et al., 2017). In this setting, each client holds data drawn from its own distribution, and these distributions are often heterogeneous across clients. Such heterogeneity leads to slow convergence, unstable optimization, and degraded accuracy for standard federated methods that assume a single shared model (Li et al., 2020). Although prior work has proposed modified objectives, aggregation rules, and robustness mechanisms to mitigate these issues (Li et al., 2020; Blanchard et al., 2017), a central challenge remains: how to learn models that generalize across clients while respecting strict privacy and communication constraints.

A natural response to data heterogeneity is personalization, where each client is allowed to adapt the model to its own data while still benefiting from collaboration. Existing personalized federated learning methods explore different ways to balance shared and client specific components, including splitting representations and heads (Collins et al., 2021), coupling global and local models through regularization (Dinh et al., 2020; Li et al., 2021a), or learning initializations that adapt quickly (Fallah et al., 2020). While effective, these approaches typically rely on static model partitions or separate local models, and thus implicitly assume that the structure of personalization is fixed across clients and across layers. This assumption is misaligned with the hierarchical nature of modern deep models, where different layers capture different types of information and may require different degrees of sharing and specialization.

At the same time, modern machine learning increasingly relies on large pretrained transformer models (Vaswani et al., 2017; Devlin et al., 2019). Fully finetuning such models in federated settings is often impractical due to their size and the associated communication overhead. Parameter efficient finetuning methods address this challenge by freezing the backbone and training only small adaptation modules such as Adapters (Houlsby et al., 2019) or low rank updates (Hu et al., 2021).

These techniques significantly reduce computation and communication cost, but they are typically treated as either fully shared or fully local in federated learning, offering no mechanism to adaptively control personalization across the model. This creates a mismatch between the need for flexible personalization under heterogeneity and the rigid parameter sharing schemes.

In this work, we propose **FedSDG** (**Fed**erated **S**tructure-**D**ecoupled **G**ating), a federated finetuning framework that addresses this mismatch by explicitly modeling personalization as a structured and learnable deviation from a shared model. FedSDG decouples model adaptation into shared and client specific components and introduces lightweight, block level gates to regulate their interaction. This design allows different layers to exhibit different degrees of personalization, enabling the model to capture common patterns across clients while adapting where necessary, all under strict communication and privacy constraints. FedSDG is simple, interpretable, and easy to integrate into existing federated learning pipelines. It builds on standard parameter efficient finetuning techniques, introduces only a small number of additional gating variables, and relies on a straightforward client server optimization protocol that transmits only shared updates. This makes it practical for real world federated settings with heterogeneous data and limited communication.

In summary, our contributions are threefold. First, we introduce a structure decoupled parameterization that models personalization as a residual deviation from a shared adaptation and combines these components through learnable block level gates. Second, we propose a federated optimization protocol that jointly learns shared parameters, personalized parameters, and gates while transmitting only the shared updates. Third, we provide a simple and reproducible implementation based on standard parameter efficient finetuning and federated learning tools, and demonstrate that this design improves performance and stability under heterogeneous data.

2 Related Work

Federated learning aggregates local updates to learn a shared model while keeping data decentralized (McMahan et al., 2017). Under strong data heterogeneity, however, a single shared model may be insufficient and local updates may conflict. Methods such as FedProx introduce regularization to stabilize local optimization (Li et al., 2020), while robust aggregation rules such as Krum suppress unreliable or adversarial updates (Blanchard et al., 2017). These approaches primarily aim to protect the global model from harmful updates, but they do not explicitly model client specific deviations or provide a mechanism for controlled personalization.

Personalized federated learning seeks to balance collaboration and client specificity. Approaches such as FedRep explicitly separate shared and local components (Collins et al., 2021), while pFedMe and Ditto couple a global model with client models through regularization (Dinh et al., 2020; Li et al., 2021a). Meta learning based methods such as Per-FedAvg learn initializations that adapt quickly to each client (Fallah et al., 2020). While effective, these methods typically treat personalization as a static partition of parameters or as a separate local model, and do not adapt the degree of personalization across layers or clients.

Large pretrained transformers motivate parameter efficient finetuning to reduce computation and communication costs (Vaswani et al., 2017; Devlin et al., 2019). Techniques such as Adapters (Houlsby et al., 2019) and LoRA (Hu et al., 2021) freeze the backbone and train lightweight modules, making them suitable for federated settings. Some methods localize specific parameters, such as keeping batch normalization local in FedBN (Li et al., 2021b). Gating mechanisms are commonly used to combine multiple components, for example in mixture of experts models (Jacobs et al., 1991). In contrast, FedSDG models personalization as a residual deviation from a shared adaptation and uses learnable, layer wise gates to regulate this deviation, providing a continuous and structured form of personalization.

3 Proposed Design

We consider federated finetuning of large pretrained models under two fundamental constraints: (1) strong statistical heterogeneity across clients, and (2) severely limited communication budgets that prohibit frequent synchronization of full model parameters. Existing personalized federated learning methods typically address heterogeneity either by learning separate models per client or by partially sharing model components, but they often treat personalization as a discrete or static design choice. In contrast, we argue that under heterogeneous data, the degree and the location of personalization should itself be adaptive and client dependent.

Motivated by this observation, we propose **FedSDG**, a federated finetuning framework based on *structure decoupling* and *learnable gating*. The key idea is to explicitly decompose model adaptation into a globally shared component and a client specific component, and to regulate their interaction through lightweight, learnable gates at the structural level of the model. This design introduces an explicit inductive bias that separates common adaptation directions from client specific residuals, while allowing their relative contribution to be adjusted per client and per layer.

3.1 Problem Setup

We consider cross device federated learning with K clients. Client k owns a local dataset $\mathcal{D}_k = \{(x_i, y_i)\}_{i=1}^{n_k}$ drawn from a client specific distribution P_k, which may be non IID across clients. At each communication round t, a subset $S_t \subseteq \{1, \dots, K\}$ of M clients participates in training. This setting reflects practical scenarios where data remain decentralized, privacy constraints prohibit data sharing, and clients observe data from heterogeneous domains or user behaviors.

We focus on federated finetuning of a pretrained transformer with L blocks. Let θ^{bb} denote the frozen backbone parameters and θ^{head} the task head parameters. Instead of synchronizing the full model, we define a communicated trainable parameter vector

$$\theta_g := \theta^{\mathrm{comm}} \in \mathbb{R}^{d_g}, \tag{1}$$

which represents the *shared adaptation component*. This vector may correspond to the task head, a PEFT module such as LoRA or Adapters, or a combination thereof. All other backbone parameters are kept fixed and identical across clients. This restriction enforces a low dimensional shared update space and makes communication cost explicit and controllable. In addition to θ_g, each client k maintains a local parameter vector

$$\theta_{p,k} \in \mathbb{R}^{d_p}, \tag{2}$$

which represents a *client specific adaptation component*. This vector is stored and updated locally and is never transmitted to the server. Conceptually, $\theta_{p,k}$ captures residual directions that are useful for client k but not necessarily aligned with the global objective.

The objective of FedSDG is to jointly learn a shared adaptation θ_g that captures patterns common across clients and a set of client specific adaptations $\{\theta_{p,k}\}$ that model systematic deviations induced by data heterogeneity, while respecting strict communication and privacy constraints. By explicitly separating these two roles and regulating their interaction, FedSDG provides a structured and adaptive mechanism for personalization that goes beyond static partitioning of model components.

3.2 Structure-Decoupled Model with Learnable Gates

We model personalization as a structured and adaptive deviation from a shared model rather than as a separate model or a hard partition of parameters. Concretely, we introduce a structure decoupled parameterization in which each client operates on a combination of a globally shared adaptation and a client specific residual adaptation, regulated by learnable gates.

This design reflects the inductive bias that under heterogeneous data, clients should share a common representational scaffold, while deviating from it only where necessary. The deviation should be both structured and controllable, rather than implicitly absorbed into a monolithic parameter update.

Block-wise learnable gates. We index gates at the transformer block level to allow different layers to exhibit different degrees of personalization. For each block $l \in \{1, \dots, L\}$ and client k, we define a scalar gate $m_{k,l} \in [0, 1]$ with logit $a_{k,l} \in \mathbb{R}$:

$$m_{k,l} = \sigma(a_{k,l}), \tag{3}$$

where $\sigma(\cdot)$ is the sigmoid function. This parametrization yields a smooth, bounded, and differentiable control signal that can be optimized jointly with model parameters.

Residual decomposition of adaptation. Let $\theta_{g,l}$ denote the shared adaptation parameters associated with block l, and $\theta_{p,k,l}$ denote the corresponding client specific adaptation parameters with the same structure. Both are instantiated as parameter efficient modules (e.g., LoRA or Adapters) attached to a frozen pretrained backbone. We define the effective trainable parameters used by client k at block l as

$$\tilde{\theta}_{k,l} = \theta_{g,l} + m_{k,l}\,\theta_{p,k,l}. \tag{4}$$

This additive form models personalization as a residual perturbation of the shared adaptation. The shared component $\theta_{g,l}$ captures directions that are broadly useful across clients, while $\theta_{p,k,l}$ captures client specific deviations from this shared structure. The gate $m_{k,l}$ modulates the magnitude of this deviation, enabling each layer to operate on a continuum between globally aligned behavior and locally specialized behavior.

Forward computation. During the forward pass on client k, each transformer block l uses $\tilde{\theta}_{k,l}$ together with the frozen backbone weights. For example, if a block contains a linear transformation augmented with LoRA, the backbone weight is frozen and the low rank update is given by $\tilde{\theta}_{k,l}$. The task head is included in θ_g by default, but can be moved to $\theta_{p,k}$ when task specific personalization is required. We use one gate per transformer block rather than per parameter to enforce a coarse but stable form of structural control. This choice reflects the assumption that different layers serve different functional roles and therefore require different degrees of personalization, while avoiding the instability and overfitting risks associated with fine grained gating.

Takeaways 3.1. *Equation equation 4 defines a continuous and structured interpolation between purely shared and fully personalized adaptation. When $m_{k,l} = 0$, block l follows only the shared adaptation; when $m_{k,l} = 1$, the full client specific residual is applied. Intermediate values correspond to partial deviation from the shared model. We initialize all gate logits as $a_{k,l} = 0$, yielding $m_{k,l} = 0.5$ and an unbiased starting point between sharing and personalization. The additional client side storage consists of $\theta_{p,k}$ and L scalar gate logits. This structure decoupled formulation provides an explicit, interpretable, and low overhead mechanism for controlling personalization across network depth, and remains compatible with standard federated optimization and communication protocols.*

3.3 Client Update, Server Aggregation, and Communication Protocol

We now describe the optimization procedure and communication protocol. The key principle is to jointly learn a global adaptation direction shared across clients, a set of client specific residual adaptations, and a set of gates that regulate how strongly each client deviates from the shared model.

Local optimization. At communication round t, after receiving the current shared parameters $\theta_g^{(t)}$, each participating client $k \in S_t$ solves the following regularized local problem:

$$\min_{\theta_{p,k},\, a_k,\, \theta_g} \; \frac{1}{|\mathcal{B}_k|} \sum_{(x,y)\in\mathcal{B}_k} \ell(f(x;\tilde{\theta}_k), y) + \lambda_1 \sum_{l=1}^{L} |m_{k,l}| + \lambda_2 \|\theta_{p,k}\|_2^2, \tag{5}$$

where $a_k = \{a_{k,l}\}$ and $\tilde{\theta}_k$ is defined in Eq. equation 4. The ℓ_1 penalty on $m_{k,l}$ encourages clients to use personalization sparingly, activating client specific deviations only when the shared model is insufficient. The ℓ_2 penalty on $\theta_{p,k}$ limits the capacity of the residual adaptation and prevents it from absorbing globally useful information.

Each client performs U local SGD steps starting from $\theta_g^{(t)}$ and its current local state $(\theta_{p,k}, a_k)$:

$$\theta_g \leftarrow \theta_g - \eta_g \nabla_{\theta_g} \mathcal{L}_k, \tag{6}$$

$$\theta_{p,k} \leftarrow \theta_{p,k} - \eta_p \nabla_{\theta_{p,k}} \mathcal{L}_k, \tag{7}$$

$$a_k \leftarrow a_k - \eta_m \nabla_{a_k} \mathcal{L}_k. \tag{8}$$

Only θ_g is synchronized across clients; $\theta_{p,k}$ and a_k remain local. This enforces a functional separation between globally shared structure and client specific deviations.

After local training, client k uploads the shared update

$$\Delta\theta_g^{(k)} = \theta_g^{(k)} - \theta_g^{(t)}. \tag{9}$$

Server aggregation. The server aggregates client updates with weights based on their alignment with the mean update direction. Let $\bar{\Delta} = \frac{1}{M} \sum_{k \in S_t} \Delta\theta_g^{(k)}$. We define

$$\alpha_k = \max\left(0, \frac{\langle \Delta\theta_g^{(k)}, \bar{\Delta} \rangle}{\|\Delta\theta_g^{(k)}\|_2 \cdot \|\bar{\Delta}\|_2 + \epsilon}\right), \qquad w_k = \frac{\alpha_k}{\sum_{j \in S_t} \alpha_j + \epsilon}. \tag{10}$$

This weighting suppresses updates that are poorly aligned with the emerging global consensus direction, which are more likely to correspond to client specific residuals or noise under strong heterogeneity. The shared parameters are then updated as

$$\theta_g^{(t+1)} = \theta_g^{(t)} + \sum_{k \in S_t} w_k \Delta\theta_g^{(k)}. \tag{11}$$

The degenerate case in which all α_k are zero corresponds to a situation where no consistent global update direction can be identified in the current round. In practice this is a measure zero event under continuous updates and is further avoided by the small ϵ term and by initializing from a non degenerate pretrained model.

Takeaways 3.2. *Clients jointly optimize shared parameters, client specific residuals, and gates under structural regularization, and transmit only the shared updates. The server aggregates updates using alignment based weights to suppress conflicting or purely client specific directions. This protocol preserves privacy, limits communication, and enforces a clear separation between shared and personalized components.*

3.4 Implementation Details

Instantiation of shared and personalized components. We freeze the entire pretrained backbone and insert LoRA modules into the attention output projection and feed forward output projection of each transformer block. The shared parameters θ_g consist of all LoRA parameters and the task head. Each client maintains an additional local LoRA branch with identical structure as its personalized parameters $\theta_{p,k}$. This instantiation enforces that both shared and personalized adaptations operate in the same low dimensional subspace, making their interaction interpretable and preventing the personalized branch from introducing arbitrary model capacity. We use one scalar gate per block controlling both attention and feed forward adaptations in that block, resulting in L gate parameters per client. This choice reflects the assumption that personalization is a structural property of layers rather than of individual weights, and provides a coarse but stable form of control.

Hyperparameters and stability. Unless otherwise stated, we use $K \in \{50, 100\}$ clients, sample $M/K = 0.1$ per round, and perform U local updates corresponding to $E \in \{1, 2\}$ local epochs. We use LoRA rank $r = 8$ with scaling $\alpha = 16$, learning rates $\eta_g = \eta_p = 10^{-3}$ and $\eta_m \in \{5 \times 10^{-3}, 10^{-2}\}$, and regularization coefficients $\lambda_1 \in \{10^{-4}, 5 \times 10^{-4}, 10^{-3}\}$ and $\lambda_2 \in \{10^{-4}, 10^{-3}\}$. Data are partitioned using a Dirichlet distribution with concentration $\alpha \in \{0.1, 0.3, 1.0\}$ to control the degree of heterogeneity.

These values are chosen to ensure stable optimization while keeping the magnitude of residual personalization bounded. In particular, the regularization terms prevent the personalized branch from dominating the shared adaptation and isolate the effect of structural gating from that of model capacity. We use Adam with $(\beta_1, \beta_2) = (0.9, 0.999)$ and no warmup for all parameters.

Communication and computation. Each round communicates $2M \cdot d_g$ scalars corresponding to the shared updates. The personalized parameters $\theta_{p,k}$ and gate logits a_k remain local. Since LoRA adds less than one percent parameters to the backbone and gates add only L scalars, the memory and communication overhead is negligible compared to the frozen model size.

The protocol can be implemented as a minimal modification of FedAvg: only θ_g is synchronized, $(\theta_{p,k}, a_k)$ are kept local, and $\tilde{\theta}_{k,l}$ is constructed on the fly during the forward pass. This makes the method compatible with existing federated learning pipelines and easy to reproduce.

Takeaways 3.3. *We initialize $a_{k,l} = 0$ so that $m_{k,l} = 0.5$ at the start of training, apply gradient clipping with norm 1.0, and fix random seeds for client sampling, data partitioning, and initialization. These choices reduce variance across runs and ensure that observed effects are attributable to the proposed design rather than to stochastic artifacts.*

REFERENCES

Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017.

Liam Collins, Hamed Hassani, Aryan Mokhtari, and Sanjay Shakkottai. Exploiting shared representations for personalized federated learning. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, volume 139 of *Proceedings of Machine Learning Research*, pp. 2089–2099. PMLR, 2021.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pp. 4171–4186. Association for Computational Linguistics, 2019.

Canh T. Dinh, Nguyen H. Tran, and Tuan Dung Nguyen. Personalized federated learning with moreau envelopes. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pp. 21394–21405, 2020.

Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized federated learning: A meta-learning approach. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pp. 20343–20354, 2020.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2790–2799. PMLR, 2019.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. In *Neural Computation*, volume 3, pp. 79–87, 1991.

Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. In *Proceedings of Machine Learning and Systems (MLSys)*, 2020.

Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. Fair and robust federated learning through personalization. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, volume 139 of *Proceedings of Machine Learning Research*, pp. 6357–6368. PMLR, 2021a.

Xiaoxiao Li, Meirui Jiang, Xiaofei Zhang, Michael Kamp, and Qi Dou. Fedbn: Federated learning on non-iid features via local batch normalization. In *International Conference on Learning Representations (ICLR)*, 2021b.

H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 54 of *Proceedings of Machine Learning Research*, pp. 1273–1282. PMLR, 2017.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017.