ONE PEFT DOES NOT FIT ALL: HETEROGENEOUS PEFT MIXTURES FOR FEDERATED LLM FINE-TUNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Parameter-efficient fine-tuning (PEFT) enables efficient federated adaptation of large language models, yet most federated PEFT (FedPEFT) methods adopt a single PEFT type for all clients. We observe that the strongest PEFT type varies across client tasks, while naively stacking multiple types does not consistently outperform the best single choice. Learning heterogeneous PEFT compositions introduces a selection-aggregation coupling: local module selection determines both which modules are trained and which clients contribute to their aggregation. To address this coupling, we propose **Hermes**, a heterogeneous FedPEFT framework that moves beyond a globally fixed PEFT choice to learn adaptive, client-specific PEFT compositions. A unified block-level interface makes structurally different PEFT modules composable while preserving their original internal computations. Building on this interface, Hermes coordinates sparse routing with usage-aware aggregation, allowing clients to learn distinct PEFT compositions while collaboratively training a shared heterogeneous PEFT pool. Extensive experiments under task-homogeneous and task-heterogeneous settings demonstrate consistent improvements over strong FedPEFT baselines.

1 INTRODUCTION

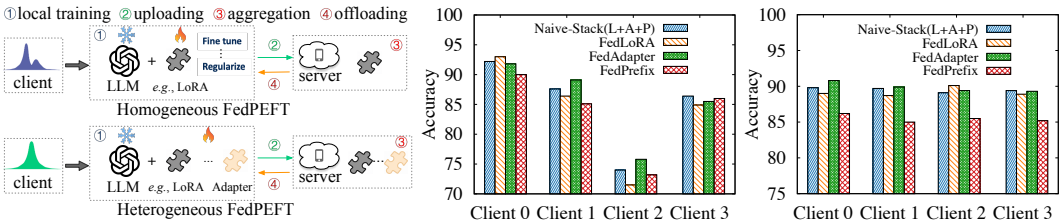
Federated fine-tuning enables multiple clients to collaboratively adapt a pretrained language model while keeping their raw training examples local Li et al. (2020); Chen et al. (2024). However, updating and communicating the full model is often prohibitively expensive for modern language models Kuang et al. (2024). Parameter-efficient fine-tuning (PEFT) reduces this cost by freezing the pretrained backbone and optimizing only lightweight modules, such as LoRA Hu et al. (2022), Adapter Houlsby et al. (2019), and Prefix-tuning Li & Liang (2021). This efficiency has motivated a growing body of federated PEFT (FedPEFT) methods Sun et al. (2024).

Most existing FedPEFT methods focus on how adaptation parameters should be optimized, personalized, or aggregated across clients. Recent approaches introduce client-specific parameters and shared-private decompositions Bian et al. (2025); Peng et al. (2025); Su et al. (2025); Singhal et al. (2024); Liu et al. (2025), or combine global and local adaptation modules Tran et al. (2025); Cui et al. (2024); Xie et al. (2024); Long et al. (2024). Nevertheless, the PEFT type itself is typically chosen before federated training and fixed across clients. These methods therefore generally follow a *homogeneous FedPEFT* design: they personalize parameters within a common PEFT type, rather than allowing clients to choose among structurally different types.

A globally fixed PEFT type can be restrictive when clients differ in their tasks and data distributions. Our motivation study in Fig. 1 shows that the strongest single PEFT type varies across client tasks, while densely stacking multiple types is not consistently better than the strongest single choice. These observations motivate *heterogeneous FedPEFT*, which allows clients to adaptively combine different PEFT types from a shared pool. The goal is not merely to provide more adaptation parameters, but to learn PEFT compositions suited to local data while retaining collaboration across clients.

A simple alternative to a globally fixed type is to select one PEFT type for each client using local validation performance. This *client-static* choice is more flexible, but remains fixed across Transformer layers and local inputs. We instead study *client-conditional PEFT composition*, where a client selects a small subset of PEFT modules at each layer according to its local data during training, rather than committing to one PEFT type through a separate search.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) Homo. vs. hetero. FedPEFT (b) Task heterogeneity (c) Task homogeneity

Figure 1: Motivation for heterogeneous FedPEFT. (a) Homogeneous FedPEFT fixes one PEFT type across clients, whereas heterogeneous FedPEFT permits client-specific compositions of different types. (b–c) Comparison of single-PEFT methods and dense PEFT stacking under task-heterogeneous and task-homogeneous settings, respectively.

Learning these compositions requires coordinating local selection with federated aggregation. PEFT mixtures have been explored in centralized fine-tuning and federated systems with lightweight experts Mao et al. (2022); Cao et al. (2026); Wu et al. (2024); Wang et al. (2025). In the federated setting, a local routing decision affects both which modules are trained and which clients contribute to each module at the server. We refer to this dependency as *selection–aggregation coupling*. On the client, activating all modules incurs extra cost and may cause interference, whereas sparse selection risks leaving some modules under-trained. On the server, aggregation must accommodate module-specific contributors and differences in their local usage. Thus, effective heterogeneous FedPEFT requires client-side composition and server-side coordination to be designed together.

To address these challenges, we propose **Hermes**, a heterogeneous FedPEFT framework for federated LLM fine-tuning. Hermes exposes structurally different PEFT modules through a unified block-level interface while preserving their internal computations. On each client, a sparse router combines current input features with historical gradient information to select the Top-k modules before execution. A loss-free bias correction discourages persistent module starvation. On the server, Hermes aggregates each module separately at each layer, using client weights and normalized module usage to reduce the relative influence of clients that repeatedly update that module. These mechanisms allow clients to learn distinct PEFT compositions while collaboratively training a shared heterogeneous PEFT pool. Our main contributions are summarized as follows:

- We formulate heterogeneous FedPEFT, which allows clients to adaptively combine structurally different PEFT modules, and identify selection–aggregation coupling as a key challenge.
- We propose **Hermes**, which coordinates client-side sparse routing and server-side usage-aware aggregation through a unified block-level PEFT interface.
- Extensive experiments on GLUE and FLAN with RoBERTa and LLaMA backbones demonstrate consistent improvements over strong FedPEFT baselines across task-homogeneous and task-heterogeneous settings.

2 Motivation

We consider FL setting with n clients. Each client i holds a local dataset $\mathcal{D}_i$ drawn from a non-IID distribution. A pre-trained LLM with parameters Θ kept frozen, and clients adopt PEFT modules. Let $\mathcal{M} = \{A, L, P\}$ denote the set of PEFT types, corresponding to Adapter, LoRA, and Prefix-tuning.

Limitation of homogeneous FedPEFT. Most existing FedPEFT methods adopt a homogeneous design, where all clients are constrained to use the same PEFT type. Each PEFT type $m \in \mathcal{M}$ defines a feasible parameter space Θ_m for its trainable parameters. Under homogeneous FedPEFT, all clients are restricted to the same parameter space, $\theta_{m,i} \in \Theta_m, \forall i \in \{1, \ldots, n\}$. For a given PEFT type m, the best local objective attainable by client i within this parameter space is

$$\mathcal{L}_i^{(m)} = \inf_{\theta_{m,i} \in \Theta_m} \mathbb{E}_{(x,y)\sim\mathcal{D}_i} \left[\ell(f(x; \Theta, \theta_{m,i}), y)\right]. \tag{1}$$

Eq. (1) implicitly assumes that a single PEFT parameter space is suitable for all clients. However, heterogeneous federated data often violates this assumption. Fig. 1b shows that under task heterogeneity, *no single PEFT is universally best across clients/tasks*. For example, FedLoRA performs

well on SST-2 (client 0), but is less effective on RTE (client 2), where other PEFT types perform better. The same observation holds when the methods are matched to the same trainable-parameter budget, as reported in Appendix E.1. In comparison, the differences among PEFT types are smaller under task homogeneity in Fig. 1c. These observations indicate that a globally fixed PEFT choice becomes increasingly restrictive as client tasks diverge.

Heterogeneous FedPEFT. A natural relaxation is to make multiple PEFT types available to each client. Accordingly, we define the heterogeneous PEFT parameter space as $\Theta_{\text{hyb}} \triangleq \Theta_A \times \Theta_L \times \Theta_P$. When unselected PEFT modules are bypassed, this space contains every homogeneous PEFT choice as a special case. Therefore, for any client i,

$$\inf_{\theta_{\text{hyb},i} \in \Theta_{\text{hyb}}} \mathcal{L}_i(\theta_{\text{hyb},i}) \leq \min_{m \in \mathcal{M}} \inf_{\theta_{m,i} \in \Theta_m} \mathcal{L}_i(\theta_{m,i}). \tag{2}$$

Eq. (2) shows that heterogeneous FedPEFT is no less expressive than any homogeneous PEFT choice. However, this is an expressiveness result rather than an optimization guarantee: it does not imply that activating and updating all PEFT modules together will improve practical federated training. A detailed proof is provided in Appendix C.1.

Why naive stacking is insufficient. A naive strategy is to always activate and update all PEFT blocks in Θ_{hyb}, as in *Naive-Stack (L+A+P)*. However, Figs. 1b and 1c show that naive stacking does not consistently match the strongest single PEFT choice and can even degrade performance on some clients, especially under task heterogeneity. One possible reason is that, although different PEFT modules have separate trainable parameters, their updates ultimately affect the same hidden representations. If the representation-level updates induced by different modules point in opposing directions, their effects may partially cancel and hinder optimization. A formal analysis of this representation-level interference is provided in Appendix C.2.

Selection-aggregation coupling. Sparse activation can reduce unnecessary joint execution and limit such interference. In federated training, however, module selection also determines which updates are available to the server. Different clients may select different PEFT modules and may use the same module at different frequencies. Consequently, each module-layer pair receives updates from a different and potentially imbalanced set of clients.

Client-side selection therefore affects not only local optimization, but also server-side aggregation. Standard average-based aggregation does not account for differences in module usage, so a shared PEFT module may be influenced mainly by clients that select and update it more frequently. We refer to this dependency as *selection–aggregation coupling*. Addressing this coupling requires a coordinated design for client-side sparse module selection and server-side aggregation of selectively updated modules, which motivates Hermes.

3 METHODOLOGY

To address the selection-aggregation coupling identified in Section 2, Hermes coordinates client-side sparse PEFT routing with server-side module-wise aggregation, as illustrated in Fig. 2. At each communication round, participating clients download the global PEFT pool, select and update a sparse subset of PEFT modules at each Transformer block, and upload the updated modules together with their usage statistics. The server aggregates each module-layer pair separately using client-normalized inverse-frequency weights and broadcasts the updated PEFT pool for the next round; the complete procedure is provided in Appendix B.2.

3.1 CLIENT-SIDE MODULAR DESIGN

3.1.1 BLOCK-LEVEL PEFT INTERFACE

Heterogeneous PEFT modules are inserted at different positions within a Transformer block and operate on different intermediate representations. LoRA introduces low-rank updates into selected linear projections, Adapter adds bottleneck residual transformations, and Prefix-tuning augments the key-value context in self-attention. Thus, their native inputs, outputs, and internal computations are different. To support unified routing over these heterogeneous PEFT modules, Hermes encapsulates their different implementations through a unified *block-level interface*.

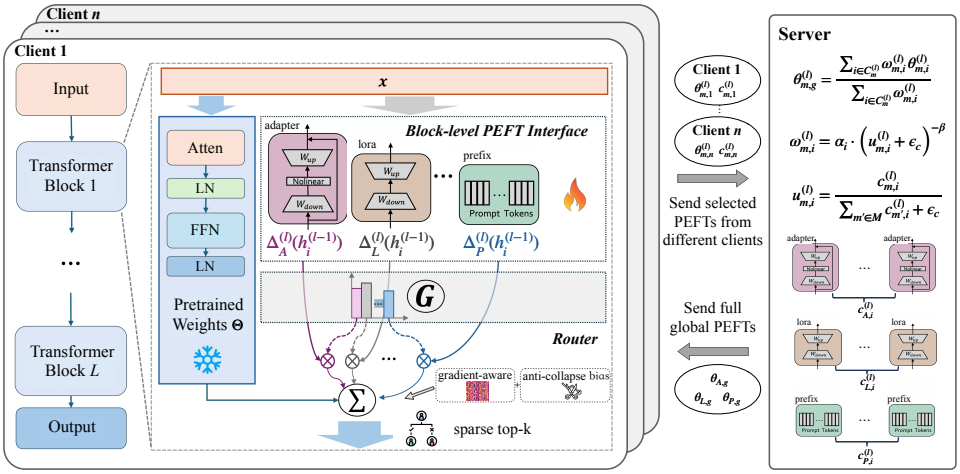


Figure 2: An overview of Hermes.

For client i and layer l, let $F_0^{(l)}(\cdot;\Theta)$ denote the frozen Transformer block and $h_i^{(l-1)}$ its input. When PEFT module $m \in \mathcal{M}$ is activated, its corresponding candidate block output is represented as

$$h_{m,i}^{(l)} = F_0^{(l)}\big(h_i^{(l-1)};\Theta\big) + \Delta_m^{(l)}\big(h_i^{(l-1)};\Theta,\theta_{m,i}^{(l)}\big), \tag{3}$$

where $\Delta_m^{(l)}$ denotes the change in the block output induced by PEFT module m. Although its actual computation may occur at an attention projection, the key-value context, or the FFN sublayer, every PEFT module is exposed through the same external mapping from the block input $h_i^{(l-1)}$ to the candidate output $h_{m,i}^{(l)}$. Eq. (3) thus gives heterogeneous PEFT modules a consistent block-level input and output form, enabling them to be selected and combined through the same routing mechanism.

In this work, $\mathcal{M}$ consists of LoRA, Adapter, and Prefix-tuning, but the block-level interface is not restricted to these three PEFT types. Other PEFT methods can be incorporated in a plug-and-play manner, provided that they can be independently activated within a Transformer block and produce a candidate block output in the same representation space. We further examine this extensibility with an additional PEFT type and a larger PEFT pool in Appendix E.5.

3.1.2 Sparse Router Design

At each local step, Hermes first computes the routing scores and then conditionally executes only the selected PEFT modules. For client i at layer l, the router selects an active subset $\mathcal{S}_i^{(l)} \subseteq \mathcal{M}$ with $|\mathcal{S}_i^{(l)}| = k$ and assigns mixture weights $\{\alpha_{m,i}^{(l)}\}_{m\in\mathcal{S}_i^{(l)}}$.

For clarity, we omit the local-step index when no ambiguity arises. The routing decision combines three signals: the current block context, historical gradient information, and an anti-collapse bias.

Feature-aware score. The feature-aware score is computed from the common pre-block hidden state before any PEFT module is executed. Given the input hidden states $h_i^{(l-1)}$ at layer l, a lightweight gating network $G_f^{(l)}$ produces

$$s_i^{(f,l)} = G_f^{(l)}\big(\mathrm{Pool}(h_i^{(l-1)})\big), \qquad s_{m,i}^{(f,l)} = \big[s_i^{(f,l)}\big]_m, \quad m \in \mathcal{M}. \tag{4}$$

Here, $\mathrm{Pool}(\cdot)$ converts the pre-block hidden states of the current mini-batch into a block-level context representation, *e.g.*, by mean pooling over valid, non-padding tokens. The resulting score provides a feature-based routing signal for all PEFT types before their internal computations are executed.

Historical gradient-aware score. The feature-aware score captures the current block context but does not reflect the recent local training behavior of each PEFT module. Hermes therefore maintains a historical gradient-aware score for every client, layer, and PEFT module. Since heterogeneous PEFT modules may contain different numbers of trainable parameters, directly comparing their gradient

norms may favor larger modules. We therefore use the parameter-normalized gradient magnitude

$$s_{m,i}^{(g,l)} = \mathrm{EMA}(\|\nabla_{\theta_{m,i}^{(l)}} \mathcal{L}_i\|_2 / (\sqrt{|\theta_m^{(l)}|} + \epsilon_g)), \qquad m \in \mathcal{M}, \tag{5}$$

where $|\theta_m^{(l)}|$ denotes the number of trainable parameters of PEFT module m at layer l, and $\epsilon_g > 0$ is a small constant for numerical stability At routing time, $s_{m,i}^{(g,l)}$ denotes the stored historical EMA before the current routing decision. After local backpropagation, the EMA is updated only for activated PEFT modules that receive gradients, while inactive modules retain their previous values. Thus, the historical gradient-aware score captures recent local training dynamics without requiring additional forward or backward passes for unselected modules. It complements the feature-aware score rather than serving as a direct utility estimate across different PEFT types.

Anti-collapse bias. Sparse routing can become self-reinforcing: frequently selected PEFT modules receive more updates, whereas rarely selected modules may remain under-trained. To discourage persistent module starvation, Hermes adapts loss-free balancing Wang et al. (2024a) and maintains a client-specific bias $b_{m,i}^{(l)}$ outside the gradient graph. The stored bias is added to the routing logits before selection and updated after the current mini-batch has been processed.

Since Hermes makes one routing decision per mini-batch at each layer, the token-equivalent usage of module m is $n_{m,i}^{(l)} = N_b \mathbf{1}\{m \in \mathcal{S}_i^{(l)}\}$, $\bar{n}^{(l)} = \frac{N_b k}{|\mathcal{M}|}$, where $N_b > 0$ is the number of valid, non-padding tokens in the mini-batch. After each local step, Hermes updates the bias of *every* module:

$$b_{m,i}^{(l)} \leftarrow b_{m,i}^{(l)} + \gamma_b \frac{\bar{n}^{(l)} - n_{m,i}^{(l)}}{\bar{n}^{(l)}}, \qquad \forall m \in \mathcal{M}, \tag{6}$$

where $\gamma_b > 0$ controls the correction strength. Equivalently, a selected module receives the correction $\gamma_b(1 - |\mathcal{M}|/k)$, whereas an unselected module receives $+\gamma_b$. These updates require only the routing decision and do not require gradients or additional PEFT evaluations. Unlike the gradient EMA, the bias of an unselected module is therefore *not* held fixed. The correction discourages persistent exclusion without adding an auxiliary loss; it does not require equal module usage in a finite training.

Final routing weights. Given the feature-aware score, historical gradient-aware score, and anti-collapse bias, Hermes forms the routing logit for PEFT module m at layer l as

$$r_{m,i}^{(l)} = \gamma s_{m,i}^{(f,l)} + (1 - \gamma) s_{m,i}^{(g,l)} + b_{m,i}^{(l)}, \tag{7}$$

where $\gamma \in [0, 1]$ balances the feature-aware and historical gradient-aware scores. Hermes then selects the Top-k PEFT modules according to $\mathcal{S}_i^{(l)} = \mathrm{TopK}(\{r_{m,i}^{(l)}\}_{m \in \mathcal{M}}, k)$. For the selected modules, Hermes computes sparse-softmax weights:

$$\alpha_{m,i}^{(l)} = \frac{\exp(r_{m,i}^{(l)}/\tau)}{\sum_{m' \in \mathcal{S}_i^{(l)}} \exp(r_{m',i}^{(l)}/\tau)}, \qquad m \in \mathcal{S}_i^{(l)}, \tag{8}$$

where $\tau > 0$ is the routing temperature. For $m \notin \mathcal{S}_i^{(l)}$, we set $\alpha_{m,i}^{(l)} = 0$. Using the *block-level interface* in Eq. (3), the routed block output is

$$h_i^{(l)} = F_0^{(l)}(h_i^{(l-1)}; \Theta) + \sum_{m \in \mathcal{S}_i^{(l)}} \alpha_{m,i}^{(l)} \Delta_m^{(l)}(h_i^{(l-1)}; \Theta, \theta_{m,i}^{(l)}). \tag{9}$$

Only modules in $\mathcal{S}_i^{(l)}$ are evaluated and updated in the current local step. Their token-equivalent usage $n_{m,i}^{(l)}$ is accumulated across local steps to obtain the round-level usage statistics $c_{m,i}^{(l)}$ used for server-side aggregation. Consequently, the PEFT-specific forward and backward computation depends on the selected number k, rather than the full pool size $|\mathcal{M}|$.

3.2 Server-side Aggregation

After local training, each client uploads the PEFT modules with nonzero usage during the current communication round, together with their usage statistics. Since clients may activate different PEFT modules at different layers, aggregation is performed separately for each module–layer pair (m, l).

Module-wise aggregation. Let $\mathcal{C}_m^{(l)}$ denote the set of clients that uploaded PEFT module m at layer l in the current round. Equivalently, $\mathcal{C}_m^{(l)} = \{i : c_{m,i}^{(l)} > 0\}$. The server then aggregates:

$$\theta_{m,g}^{(l)} = \frac{1}{Z_m^{(l)}} \sum_{i \in \mathcal{C}_m^{(l)}} \omega_{m,i}^{(l)} \, \theta_{m,i}^{(l)}, \quad Z_m^{(l)} = \sum_{i \in \mathcal{C}_m^{(l)}} \omega_{m,i}^{(l)}, \tag{10}$$

where $\omega_{m,i}^{(l)}$ is the aggregation weight assigned to client i for module-layer pair (m, l). If no client uploads a module-layer pair, the server retains its global parameters from the previous round.

Inverse-frequency weighting. Directly using the raw module-usage count can mix a client's routing preference with its total token volume, sequence lengths, and local data size. Hermes therefore normalizes the usage of each PEFT module within the same client. Let $c_{m,i}^{(l)}$ denote the round-level token-equivalent usage count of PEFT module m at layer l on client i. We define the normalized usage ratio $u_{m,i}^{(l)} = \frac{c_{m,i}^{(l)}}{\sum_{m' \in \mathcal{M}} c_{m',i}^{(l)} + \epsilon_c}$. The aggregation weight is then

$$\omega_{m,i}^{(l)} = \alpha_i \cdot \left(u_{m,i}^{(l)} + \epsilon_c \right)^{-\beta}, \qquad \beta \in [0, 1], \tag{11}$$

where α_i is the standard FedAvg client weight, such as a sample-count or token-count weight, and $\epsilon_c > 0$ ensures numerical stability. When $\beta = 0$, Eq. (11) reduces to standard client-weighted averaging over clients that uploaded the corresponding module. When $\beta > 0$, a client that allocates a larger fraction of its local routing decisions to module m receives a smaller relative aggregation weight for that module. This prevents a shared PEFT module from being shaped primarily by clients that repeatedly select and update it. Because $u_{m,i}^{(l)}$ is normalized within each client, this correction does not penalize a client merely for processing longer sequences or more local tokens.

The correction is applied across clients within each module-layer pair. It does not directly enforce equal importance or equal usage among different PEFT types.

Global broadcast. After aggregating all available module–layer pairs, the server broadcasts the full global PEFT pool $\{\theta_{m,g}^{(l)}\}_{m \in \mathcal{M},\, l=1,\ldots,L}$ to clients participating in the next communication round. All clients therefore receive the same heterogeneous PEFT pool, while their PEFT compositions remain client- and layer-dependent through sparse routing.

3.3 Convergence Guarantees

We analyze the PEFT updates of Hermes under local SGD and module-wise parameter aggregation. The full setup, assumptions, and proof are provided in Appendix D.

Theorem 3.1 (**Convergence to a stationary neighborhood**). *Let F be the reference objective in Appendix D. Under the setup and Assumptions A1–A6 in that appendix, including the uniform routing-error and aggregation-deviation bounds, let K be the common number of local SGD steps per round. If $\eta > 0$ and $L_F \eta K \le c_0$ for a sufficiently small numerical constant $c_0 > 0$, then after T communication rounds,*

$$\frac{1}{T} \sum_{t=1}^{T} \mathbb{E} \|\nabla F(\bar{\theta}^{t-1})\|^2 \le \underbrace{\mathcal{O}(\frac{F(\bar{\theta}^0) - F^\star}{\eta T K})}_{\textit{optimization}} + \underbrace{\mathcal{O}(L_F \eta \sigma^2 \sum_{i=1}^{n} \alpha_i^2)}_{\textit{stochastic noise}} + \underbrace{\mathcal{O}(L_F^2 \eta^2 K^2 G_g^2)}_{\textit{local drift}} + \underbrace{\mathcal{O}(\epsilon_{\text{sel}}^2 + \epsilon_{\text{agg}}^2)}_{\textit{routing and aggregation}}.$$

Interpretation. The bound separates the initial optimization gap, stochastic noise, local-update drift, and routing and aggregation errors. The routing term includes both Top-k truncation and the difference between the current dense-routing gradient and the reference gradient. The aggregation term measures the deviation introduced by the actual module-wise parameter averaging. Their bounds are specified in Lemmas D.3 and D.4.

4 Experiment

4.1 Setup

Dataset. We evaluate our method on two benchmark families. For *natural language understanding (NLU)*, we use eight GLUE tasks Wang et al. (2018): CoLA, SST-2, MRPC, STS-B, QQP, MNLI,

Table 1: Federated GLUE-S1 (task-homogeneous) on RoBERTa-base under Dirichlet(1.0) non-IID splits. Near-IID results under Dirichlet(10.0) are reported in Appendix E.11. Metrics are acc except MRPC=$F1$, CoLA=mcc, STS-B=$spearman$.

Method	GLUE-S1, RoBERTa-base							
	SST-2	MRPC	CoLA	RTE	QNLI	STS-B	MNLI	QQP
***Dirichlet(1.0)* (non-IID)**								
FedAdapter	$91.8_{\pm 0.96}$	$87.6_{\pm 1.12}$	$60.5_{\pm 0.37}$	$67.9_{\pm 1.37}$	$89.4_{\pm 0.73}$	$87.6_{\pm 0.51}$	$84.5_{\pm 0.77}$	$86.9_{\pm 0.23}$
FedLoRA	$92.1_{\pm 1.14}$	$83.0_{\pm 0.86}$	$59.5_{\pm 0.81}$	$65.9_{\pm 1.45}$	$89.3_{\pm 1.33}$	$86.5_{\pm 0.36}$	$83.4_{\pm 0.44}$	$85.9_{\pm 0.76}$
FedPrefix	$90.5_{\pm 0.44}$	$84.3_{\pm 0.84}$	$55.5_{\pm 0.82}$	$56.1_{\pm 1.37}$	$85.1_{\pm 0.62}$	$83.5_{\pm 1.05}$	$79.7_{\pm 0.88}$	$83.2_{\pm 0.85}$
FedUniPELT	$89.8_{\pm 0.81}$	$81.9_{\pm 0.91}$	$55.9_{\pm 1.01}$	$59.8_{\pm 0.24}$	$86.1_{\pm 0.66}$	$83.6_{\pm 0.93}$	$80.1_{\pm 0.89}$	$82.9_{\pm 0.99}$
FedSA	$92.4_{\pm 1.08}$	$85.2_{\pm 0.55}$	$59.9_{\pm 0.45}$	$66.7_{\pm 0.58}$	$89.6_{\pm 1.09}$	$86.9_{\pm 0.31}$	$84.1_{\pm 0.49}$	$86.5_{\pm 1.47}$
FedDPA	$92.2_{\pm 0.55}$	$88.4_{\pm 1.11}$	$61.0_{\pm 1.37}$	$68.5_{\pm 0.30}$	$89.8_{\pm 0.21}$	$88.0_{\pm 0.55}$	$85.0_{\pm 0.78}$	$87.3_{\pm 0.31}$
FedLEASE	$92.9_{\pm 1.09}$	$88.9_{\pm 0.92}$	$61.8_{\pm 0.58}$	$69.0_{\pm 1.03}$	$90.0_{\pm 0.46}$	$88.5_{\pm 1.12}$	$85.3_{\pm 0.28}$	$88.0_{\pm 1.15}$
Hermes	$93.4_{\pm 0.23}$	$90.1_{\pm 1.21}$	$63.0_{\pm 0.82}$	$70.4_{\pm 0.85}$	$90.5_{\pm 0.77}$	$89.2_{\pm 1.00}$	$86.0_{\pm 0.83}$	$89.1_{\pm 1.30}$

QNLI, and RTE datasets. For *natural language generation (NLG)*, we select four representative FLAN Chung et al. (2024) tasks: *structure-to-text* on Webnlg, *summarization* on XSum, *commonsense reasoning* on HellaSwag, and *open-domain QA* on ARC datasets.

Federated scenarios. We consider two scenarios on NLU and one scenario on NLG. *(GLUE-S1) Task-homogeneous.* We use 8 clients in total. All clients solve the same GLUE task while their local datasets are non-IID. Following Lin et al. Lin et al. (2022), we generate heterogeneity by sampling class-prior vectors from a Dirichlet distribution. *(GLUE-S2) Task-heterogeneous.* Each client is assigned to exactly one GLUE task among SST-2, QNLI, RTE, QQP. *(FLAN) Task-heterogeneous.* Each client is assigned to exactly one FLAN task among the four selected tasks. For *(GLUE-S2)* and *(FLAN)*, we use 16 clients in total, with four clients per task, and randomly split each task dataset into four local partitions. In addition, each client has 300 training instances and 200 testing instances.

Baselines. We compare Hermes against baselines we implement under FedAvg (**FedAdapter** Houlsby et al. (2019), **FedLoRA** Hu et al. (2022), **FedPrefix** Li & Liang (2021), **FedUniPELT** Mao et al. (2022)), and recent personalized FedPEFT methods (**FedSA** Guo et al. (2025), **FedDPA** Long et al. (2024), **FedLEASE** Wang et al. (2025)).

Implementation Details. For GLUE experiments, we adopt **RoBERTa** as the frozen backbone. For FLAN experiments, we use two frozen LLM backbones: **LLaMA-3.2-3B** and **LLaMA-2-13B**. Hermes unifies all three PEFT modules as parallel experts in each Transformer block, coordinated by a gradient-aware router with Top-k expert selection ($k = 2$ by default). Clients perform $E_p = 10$ local epochs per round, learning rate 2×10^{-4}, batch size 32, and weight decay 10^{-4}. Router temperature $\tau = 0.1$ and bias step size $\gamma_b = 0.5$ are used for loss-free balancing. On the server, we apply inverse-frequency aggregation with ε=1 and β=0.5. The adapter bottleneck dimension r_a is set to 32, the LoRA rank is $r = 8$, and the prefix length is $l_p = 16$.

4.2 Performance

Exp-1: Results on NLU. Table 1 and Table 2 summarize results on GLUE-S1 (*task-homogeneous*) and GLUE-S2 (*task-heterogeneous*). Hermes consistently outperforms all baselines, with the advantage more pronounced under stronger heterogeneity. Single-PEFT methods show complementary strengths but none dominates across tasks/clients, and they drop notably on harder/low-resource tasks (e.g., CoLA, RTE), highlighting the limitation of a homogeneous PEFT choice. Naive multi-PEFT (e.g., FedUniPELT) yields only marginal gains, suggesting that simply stacking modules introduces interference. In contrast, Hermes selects which experts to activate via sparse routing and coordinates heterogeneous updates via frequency-aware aggregation, preventing expert collapse and yielding robust improvements.

Exp-2: Results on NLG. Table 3 extends evaluation to FLAN with four generation-oriented tasks (Webnlg, XSum, HellaSwag, ARC) under LLaMA-3.2-3B and LLaMA-2-13B. The same ordering largely holds: Hermes achieves the best overall performance, while single-PEFT baselines remain task-sensitive and multi-PEFT baselines without structured selection are less effective. Notably, Hermes improves more on tasks that benefit from heterogeneous adaptation signals (e.g., structured generation and open QA), indicating that its expert selection can better match task-specific requirements even

Table 2: Federated GLUE-S2 (task-heterogeneous) on RoBERTa-base and RoBERTa-large. Each client is assigned to one task from {SST-2, QNLI, RTE, QQP}, with 4 clients per task.

Method	GLUE-S2, RoBERTa-base				GLUE-S2, RoBERTa-large			
	SST-2	QNLI	RTE	QQP	SST-2	QNLI	RTE	QQP
FedAdapter	91.2 ± 1.29	89.0 ± 0.40	67.0 ± 1.01	86.0 ± 1.34	92.4 ± 0.31	90.2 ± 0.76	69.0 ± 0.60	87.2 ± 1.43
FedLoRA	92.0 ± 0.88	89.5 ± 0.31	64.5 ± 1.47	85.5 ± 0.48	93.1 ± 0.56	90.8 ± 0.79	66.5 ± 0.54	86.8 ± 0.68
FedPrefix	89.0 ± 1.41	86.0 ± 1.39	56.5 ± 1.22	83.0 ± 1.30	90.4 ± 0.63	87.2 ± 1.02	58.5 ± 0.83	84.3 ± 0.22
FedUniPELT	90.0 ± 1.10	87.0 ± 0.62	60.0 ± 0.27	84.0 ± 0.30	91.5 ± 0.69	88.4 ± 0.30	62.0 ± 0.22	85.2 ± 0.91
FedSA	92.3 ± 0.99	89.8 ± 0.49	66.5 ± 0.64	86.5 ± 0.71	93.5 ± 0.48	91.1 ± 1.17	68.5 ± 0.53	87.9 ± 0.34
FedDPA	92.1 ± 0.74	90.2 ± 1.42	68.5 ± 0.98	87.0 ± 0.23	93.3 ± 0.88	91.5 ± 0.90	70.8 ± 0.27	88.5 ± 0.88
FedLEASE	92.8 ± 1.05	90.5 ± 1.36	69.2 ± 1.13	87.8 ± 0.45	94.0 ± 1.27	91.8 ± 0.45	71.5 ± 0.46	89.2 ± 1.01
Hermes	93.6 ± 0.97	91.2 ± 0.63	71.0 ± 0.53	88.8 ± 1.33	94.7 ± 1.21	92.6 ± 0.47	73.5 ± 1.33	90.3 ± 0.97

Table 3: Federated FLAN (task-heterogeneous) on LaMA-3.2-3B and LLaMA-2-13B. Each task has 4 clients, following Long et al. (2024), we report ROUGE-1 metric.

Method	FLAN, LLaMA-3.2-3B				FLAN, LLaMA-2-13B			
	Webnlg	XSum	HellaSwag	ARC	Webnlg	XSum	HellaSwag	ARC
FedAdapter	70.0 ± 1.39	25.0 ± 0.28	61.0 ± 0.77	77.0 ± 0.67	73.0 ± 1.01	27.5 ± 0.51	64.0 ± 0.25	82.0 ± 1.31
FedLoRA	71.2 ± 0.84	25.8 ± 1.24	62.0 ± 0.86	78.5 ± 0.99	74.2 ± 0.76	28.3 ± 0.55	65.0 ± 1.42	83.5 ± 0.73
FedPrefix	71.0 ± 0.66	26.2 ± 1.16	60.2 ± 1.31	77.5 ± 1.39	74.0 ± 0.41	28.6 ± 1.25	63.0 ± 1.06	83.0 ± 0.59
FedUniPELT	70.5 ± 1.06	25.0 ± 1.49	60.0 ± 0.94	76.8 ± 0.59	73.6 ± 0.91	27.7 ± 0.87	63.4 ± 0.47	82.2 ± 0.92
FedSA	72.0 ± 0.59	26.5 ± 1.18	63.5 ± 0.79	80.0 ± 0.25	75.0 ± 0.96	29.0 ± 1.18	66.5 ± 1.16	85.0 ± 1.44
FedDPA	72.5 ± 0.69	27.0 ± 0.44	64.5 ± 1.32	82.0 ± 0.45	75.5 ± 1.21	29.5 ± 0.55	67.8 ± 0.94	87.5 ± 1.47
FedLEASE	73.5 ± 0.67	27.8 ± 0.35	66.0 ± 0.73	83.5 ± 1.42	76.5 ± 0.84	30.2 ± 0.90	69.0 ± 0.65	89.0 ± 0.40
Hermes	75.0 ± 0.42	28.8 ± 0.78	68.0 ± 0.33	85.0 ± 0.98	78.0 ± 0.83	31.0 ± 0.34	71.0 ± 0.42	91.0 ± 0.28

when clients are distributed by task. The gains persist when scaling from 3B to 13B, showing that the proposed routing-and-aggregation design is stable across backbone sizes.

Exp-3: Ablation of components. Table 4 reports ablations on GLUE-S1, GLUE-S2, and FLAN. Removing any component degrades performance across all benchmarks, confirming its necessity. In particular, w/o-F yields the largest drop, showing that scores are critical for reliable expert selection. w/o-K and w/o-G further hurt routing quality, while w/o-A weakens robustness by increasing update dominance under heterogeneity. Overall, Hermes consistently performs best, verifying the complementarity of the design. We additionally compare Hermes with a homogeneous-expert variant (Hermes-3×LoRA), which keeps the same router/aggregation but uses LoRA for all experts. Hermes consistently outperforms Hermes-HomoLoRA on both GLUE-S2 and FLAN, showing that the gains come from heterogeneous experts rather than MoE routing alone.

Exp-4: Impact of the number of activated PEFT. We further examine the effect of varying the number of activated experts k in Fig. 3a and 3b. Activating a single expert (k=1) underutilizes model capacity, while selecting three experts (k=3) introduces redundancy without additional benefits. The best performance is observed at k=2, which strikes a favorable balance between expert diversity and training stability. This result indicates that sparse but non-trivial expert activation is the most effective setting for Hermes in FedPEFT.

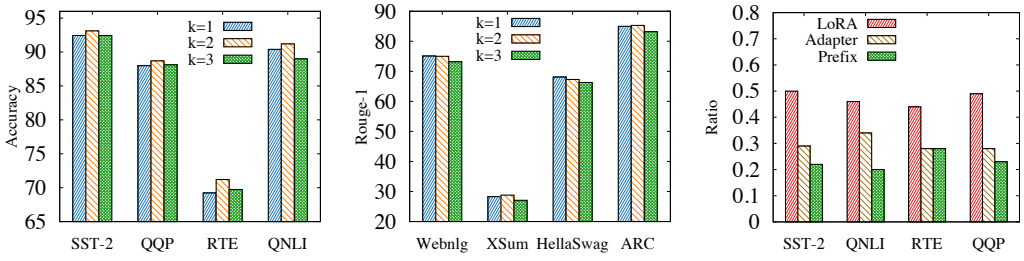
Exp-5: PEFT ratio analysis. Figure 3c reports the task-wise PEFT usage. The router assigns different ratios of LoRA/Adapter/Prefix across tasks, indicating task-dependent expert preference under the balancing constraint.

Exp-6: Communication analysis. Figure 4c reports the communication cost per round. Hermes maintains a cost comparable to single-PEFT baselines since only the selected experts are uploaded. FedUniPELT incurs much higher overhead because it synchronizes all modules regardless of their contribution. This validates that sparse MoE-style activation not only improves accuracy but also preserves communication efficiency.

Exp-7: Convergence analysis. Fig. 4a and 4b compares the convergence behaviors on GLUE-S2 and FLAN. On GLUE-S2 (50 rounds), Hermes converges faster and stabilizes at the highest performance, while baselines improve more slowly and fluctuate more under non-IID training. Single-PEFT methods saturate earlier, and FedUniPELT slows after early gains, consistent with cross-module

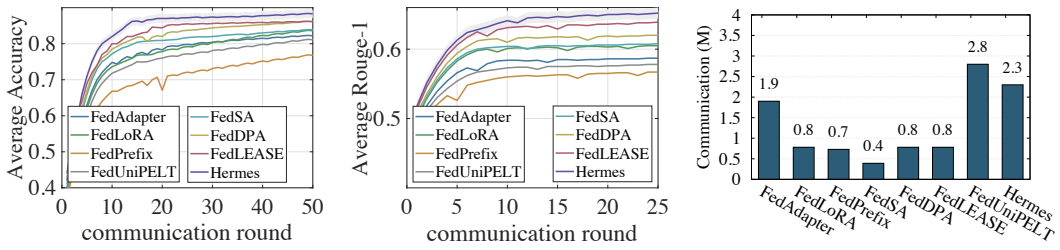
Table 4: The effectiveness of key components (A = aggregation balance, K = Top-k router, G = gradient-aware gating, F = scores).

Ablation	GLUE-S1, RoBERTa-base							
	SST-2	MRPC	CoLA	RTE	QNLI	STS-B	MNLI	QQP
***Dirichlet(1.0)* (non-IID)**								
Hermes-w/o-A	92.7 ± 0.93	89.9 ± 1.25	62.75 ± 1.28	69.9 ± 1.12	90.1 ± 1.17	88.7 ± 1.37	85.5 ± 1.46	88.8 ± 0.30
Hermes-w/o-K	92.7 ± 0.21	89.1 ± 0.35	62.27 ± 0.37	69.5 ± 0.36	89.9 ± 0.93	88.5 ± 1.22	85.4 ± 1.40	88.3 ± 0.49
Hermes-w/o-G	92.1 ± 0.29	88.9 ± 0.30	62.04 ± 1.40	69.4 ± 0.38	89.2 ± 0.91	87.8 ± 0.47	84.9 ± 1.16	87.9 ± 1.37
Hermes-w/o-F	92.2 ± 1.05	88.9 ± 1.10	61.65 ± 0.81	69.5 ± 1.31	85.4 ± 1.48	86.3 ± 0.81	85.3 ± 1.17	87.7 ± 1.39
Hermes	93.4 ± 0.23	90.1 ± 1.21	63.0 ± 0.82	70.4 ± 0.85	90.5 ± 0.77	89.2 ± 1.00	86.0 ± 0.83	89.1 ± 1.30
Ablation	**GLUE-S2, RoBERTa-large**				**FLAN, LLaMA-3.2-3B**			
	SST-2	**QNLI**	**RTE**	**QQP**	**Webnlg**	**XSum**	**HellaSwag**	**ARC**
Hermes-w/o-A	94.0 ± 0.86	92.2 ± 1.16	73.0 ± 1.18	90.0 ± 0.54	74.2 ± 0.90	28.4 ± 0.46	67.4 ± 1.50	84.1 ± 0.25
Hermes-w/o-K	94.0 ± 1.03	92.0 ± 1.34	72.6 ± 1.37	89.5 ± 0.43	73.8 ± 0.76	28.0 ± 0.81	67.0 ± 1.20	83.5 ± 0.55
Hermes-w/o-G	93.4 ± 0.66	91.3 ± 0.46	72.5 ± 0.82	89.1 ± 1.38	73.2 ± 0.36	27.6 ± 1.21	66.6 ± 0.52	83.0 ± 1.22
Hermes-w/o-F	93.5 ± 0.73	87.5 ± 0.53	72.6 ± 1.22	88.9 ± 0.26	72.0 ± 0.85	27.0 ± 0.43	65.8 ± 0.67	81.5 ± 0.93
Hermes	94.7 ± 1.21	92.6 ± 0.47	73.5 ± 1.33	90.3 ± 0.97	75.0 ± 0.42	28.8 ± 0.78	68.0 ± 0.33	85.0 ± 0.98
Expert design ablation (homogeneous experts)								
Hermes-(3×LoRA)	94.2 ± 0.79	92.1 ± 1.39	72.8 ± 1.14	89.8 ± 0.21	74.2 ± 0.23	28.2 ± 1.03	67.2 ± 1.00	83.9 ± 0.79



(a) GLUE-S2 (b) FLAN (c) GLUE-S2

Figure 3: (a-b) The impact of varying the number of activated experts k on GLUE-S2 and FLAN scenarios. (c) Task-wise expert usage and on GLUE-S2.



(a) GLUE-S2 (b) FLAN (c) GLUE-S2

Figure 4: (a-b) The convergence comparison on GLUE-S2 and FLAN. (c) Communicated trainable parameters per round (M) on GLUE-S2.

interference. On FLAN (25 rounds), all methods converge in fewer rounds, yet Hermes maintains steady improvement and reaches higher performance within fewer rounds.

Further studies (budget-matched motivation, routing sanity checks, sensitivity, extensibility, scalability, partial participation, routing-time details) are in Appendix E.

5 Conclusion

We introduce Hermes, a heterogeneous FedPEFT framework that unifies multiple PEFT modules via a sparse mixture-of-experts design. By combining gradient-aware routing, loss-free bias adjustment, and inverse-frequency aggregation, Hermes achieves superior performance compared to SOTA homogeneous FedPEFT methods.

AI USE STATEMENT

We used large language models to polish the writing, including grammar correction, style harmonization, and sentence-level revisions. All suggested edits were reviewed by the authors, who take full responsibility for the final content.

REPRODUCIBILITY STATEMENT

The main paper describes the datasets, base models, and evaluation metrics used in our experiments. All datasets are publicly available. The appendix provides the complete experimental setup and implementation details. We will release the code upon acceptance to facilitate reproduction of our results.

REFERENCES

Abdulla Jasem Almansoori, Samuel Horváth, and Martin Takáč. Collaborative and efficient personalization with mixtures of adaptors. *arXiv preprint arXiv:2410.03497*, 2024.

Jieming Bian, Lei Wang, Letian Zhang, and Jie Xu. Fedalt: Federated fine-tuning through adaptive local training with rest-of-world lora. *arXiv preprint arXiv:2503.11880*, 2025.

Jie Cao, Tianwei Lin, Bo Yuan, Rolan Yan, Hongyang He, Wenqiao Zhang, Juncheng Li, Dongping Zhang, Siliang Tang, and Yueting Zhuang. Moa: Heterogeneous mixture of adapters for parameter-efficient fine-tuning of large language models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 21056–21073, 2026.

Tianshi Che, Ji Liu, Yang Zhou, Jiaxiang Ren, Jiwen Zhou, Victor Sheng, Huaiyu Dai, and Dejing Dou. Federated learning of large language models with parameter-efficient prompt tuning and adaptive optimization. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7871–7888, 2023.

Jingxue Chen, Hang Yan, Zhiyuan Liu, Min Zhang, Hu Xiong, and Shui Yu. When federated learning meets privacy-preserving computation. *ACM Computing Surveys*, 56(12):1–36, 2024.

Yae Jee Cho, Luyang Liu, Zheng Xu, Aldi Fahrezi, and Gauri Joshi. Heterogeneous lora for federated fine-tuning of on-device foundation models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 12903–12913, 2024.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53, 2024.

Tianyu Cui, Hongxia Li, Jingya Wang, and Ye Shi. Harmonizing generalization and personalization in federated prompt learning. In *International Conference on Machine Learning*, pp. 9646–9661. PMLR, 2024.

Donghang Duan, Xu Zheng, Lizong Zhang, Chong Mu, and Meng Han. Fedweave: Rethinking the unit of specialization in heterogeneous federated moe-lora. *arXiv preprint arXiv:2607.26618*, 2026.

Pengxin Guo, Shuang Zeng, Yanran Wang, Huijie Fan, Feifei Wang, and Liangqiong Qu. Selective aggregation for low-rank adaptation in federated learning. In *The Thirteenth International Conference on Learning Representations*, 2025.

Tao Guo, Song Guo, and Junxiao Wang. Pfedprompt: Learning personalized prompt for vision-language models in federated learning. In *Proceedings of the ACM Web Conference 2023*, pp. 1364–1374, 2023.

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 *International conference on machine learning*, pp. 2790–2799. PMLR, 2019.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

Weirui Kuang, Bingchen Qian, Zitao Li, Daoyuan Chen, Dawei Gao, Xuchen Pan, Yuexiang Xie, Yaliang Li, Bolin Ding, and Jingren Zhou. Federatedscope-llm: A comprehensive package for fine-tuning large language models in federated learning. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 5260–5271, 2024.

Royson Lee, Minyoung Kim, Fady Rezk, Rui Li, Stylianos I Venieris, and Timothy Hospedales. Fedp^{2}eft: Federated learning to personalize peft for multilingual llms. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 22805–22813, 2026.

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.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 4582–4597, 2021.

Bill Yuchen Lin, Chaoyang He, Zihang Ze, Hulin Wang, Yufen Hua, Christophe Dupuy, Rahul Gupta, Mahdi Soltanolkotabi, Xiang Ren, and Salman Avestimehr. Fednlp: Benchmarking federated learning methods for natural language processing tasks. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pp. 157–175, 2022.

Shichong Liu, Haozhe Jin, Zhiwei Tang, Rui Zhai, Ke Lu, Junyang Yu, and Chenxi Bai. Adapter-guided knowledge transfer for heterogeneous federated learning. *Journal of Systems Architecture*, 159:103338, 2025.

Guodong Long, Tao Shen, Jing Jiang, Michael Blumenstein, et al. Dual-personalizing adapter for federated foundation models. *NeurIPS*, 37:39409–39433, 2024.

Jun Luo, Chen Chen, and Shandong Wu. Mixture of experts made personalized: Federated prompt learning for vision-language models. In *ICLR*, 2025.

Yuning Mao, Lambert Mathias, Rui Hou, Amjad Almahairi, Hao Ma, Jiawei Han, Wen Tau Yih, and Madian Khabsa. Unipelt: A unified framework for parameter-efficient language model tuning. In *60th Annual Meeting of the Association for Computational Linguistics, ACL 2022*, pp. 6253–6264. Association for Computational Linguistics (ACL), 2022.

Danni Peng, Yuan Wang, Huazhu Fu, Jinpeng Jiang, Yong Liu, Rick Siow Mong Goh, and Qingsong Wei. Look back for more: Harnessing historical sequential updates for personalized federated adapter tuning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 19857–19865, 2025.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. Adapterfusion: Non-destructive task composition for transfer learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pp. 487–503, 2021.

Jiaxing Qi, Zhongzhi Luan, Shaohan Huang, Carol Fung, Hailong Yang, and Depei Qian. Fdlora: Personalized federated learning of large language model via dual lora tuning. *arXiv preprint arXiv:2406.07925*, 2024.

Raghav Singhal, Kaustubh Ponkshe, and Praneeth Vepakomma. Fedex-lora: Exact aggregation for federated parameter-efficient fine-tuning of foundation models. In *NeurIPS 2024 Workshop on Fine-Tuning in Modern Machine Learning: Principles and Scalability*, 2024.

Yang Su, Na Yan, and Yansha Deng. Federated llms fine-tuned with adaptive importance-aware lora. In *ICC 2025-IEEE International Conference on Communications*, pp. 6112–6117. IEEE, 2025.

Guangyu Sun, Umar Khalid, Matias Mendieta, Pu Wang, and Chen Chen. Exploring parameter-efficient fine-tuning to enable foundation models in federated learning. In *2024 IEEE International Conference on Big Data (BigData)*, pp. 8015–8024. IEEE, 2024.

Linh Tran, Wei Sun, Stacy Patterson, and Ana Milanova. Privacy-preserving personalized federated prompt learning for multimodal large language models. In *International Conference on Learning Representations*, 2025.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP workshop BlackboxNLP: Analyzing and interpreting neural networks for NLP*, pp. 353–355, 2018.

Lean Wang, Huazuo Gao, Chenggang Zhao, Xu Sun, and Damai Dai. Auxiliary-loss-free load balancing strategy for mixture-of-experts. *arXiv preprint arXiv:2408.15664*, 2024a.

Lei Wang, Jieming Bian, Letian Zhang, and Jie Xu. Adaptive loRA experts allocation and selection for federated fine-tuning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Ziyao Wang, Zheyu Shen, Yexiao He, Guoheng Sun, Hongyi Wang, Lingjuan Lyu, and Ang Li. Flora: Federated fine-tuning large language models with heterogeneous low-rank adaptations. *Advances in Neural Information Processing Systems*, 37:22513–22533, 2024b.

Panlong Wu, Kangshuo Li, Ting Wang, Yanjie Dong, Victor CM Leung, and Fangxin Wang. Fedfmsl: Federated learning of foundation models with sparsely activated lora. *IEEE Transactions on Mobile Computing*, 23(12):15167–15181, 2024.

Chulin Xie, De-An Huang, Wenda Chu, Daguang Xu, Chaowei Xiao, Bo Li, and Anima Anandkumar. Perada: Parameter-efficient federated learning personalization with generalization guarantees. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 23838–23848, 2024.

Mengwei Xu, Dongqi Cai, Yaozong Wu, Xiang Li, and Shangguang Wang. {FwdLLM}: Efficient federated finetuning of large language models with perturbed inferences. In *2024 USENIX Annual Technical Conference (USENIX ATC 24)*, pp. 579–596, 2024.

Fu-En Yang, Chien-Yi Wang, and Yu-Chiang Frank Wang. Efficient model personalization in federated learning via client-specific prompt generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 19159–19168, 2023.

Liping Yi, Han Yu, Gang Wang, Xiaoguang Liu, and Xiaoxiao Li. pfedlora: Model-heterogeneous personalized federated learning with lora tuning. *arXiv preprint arXiv:2310.13283*, 2023.

Yicheng Zhang, Zhen Qin, Zhaomin Wu, Jian Hou, and Shuiguang Deng. Personalized federated fine-tuning for llms via data-driven heterogeneous model architectures. In *Proceedings of the ACM Web Conference 2026*, pp. 5099–5110, 2026.

A Related Work

PEFT and PEFT Composition. Parameter-efficient fine-tuning (PEFT) adapts a frozen pretrained model by optimizing only a small set of additional parameters. Adapter inserts trainable bottleneck transformations into Transformer layers Houlsby et al. (2019), Prefix-tuning augments the attention context with trainable key–value states Li & Liang (2021), and LoRA represents weight updates through trainable low-rank factors Hu et al. (2022). These PEFT types modify different parts of a Transformer and impose different structures on model adaptation.

Beyond using a single PEFT type, prior work has explored composing multiple adaptation modules. AdapterFusion Pfeiffer et al. (2021) combines independently trained adapters for knowledge transfer across tasks. UniPELT Mao et al. (2022) jointly incorporates Adapter, LoRA, and Prefix-tuning and learns their activation through gating. More recently, MoA Cao et al. (2026) constructs a heterogeneous pool of structurally different PEFT experts and studies both soft and sparse routing. These methods establish the value of heterogeneous PEFT composition in centralized fine-tuning. They do not, however, consider federated training, where module selection also determines which local updates are produced, communicated, and available for server aggregation.

Personalized and Heterogeneous FedPEFT. FedPEFT reduces the computation and communication costs by exchanging only lightweight PEFT parameters Che et al. (2023); Sun et al. (2024); Xu et al. (2024). To address non-IID client data, existing methods personalize a predetermined PEFT type through client-specific parameters Yang et al. (2023); Yi et al. (2023); Guo et al. (2023), shared-private parameter decomposition Xie et al. (2024); Cui et al. (2024), dual global-local modules Long et al. (2024); Qi et al. (2024), or selective aggregation Guo et al. (2025). Although these methods differ in how PEFT parameters are shared and aggregated, the underlying PEFT family is generally fixed before federated training.

A separate line of work studies structural or resource heterogeneity within LoRA. HetLoRA Cho et al. (2024) allows clients to use different LoRA ranks and aggregates them through sparsity-aware weighting, while FLoRA Wang et al. (2024b) supports heterogeneous low-rank adapters through stacking-based aggregation. FedP^{2}EFT Lee et al. (2026) further learns client-specific LoRA structures and hyperparameters, including the adapted layers and ranks. These methods address *intra-family* heterogeneity within LoRA. In contrast, Hermes considers *inter-family* heterogeneity across structurally different PEFT types, including LoRA, Adapter, and Prefix-tuning, etc.

MoE-based FedPEFT. Mixture-of-experts designs have also been introduced into federated fine-tuning. FedFMSL Wu et al. (2024) sparsely activates LoRA modules during federated foundation-model training, while pFedMoAP Luo et al. (2025) combines local and non-local prompt experts for personalized federated prompt learning. FLoRAL Almansoori et al. (2024) learns client-specific mixtures over federated low-rank adaptors. More recent methods allocate and select LoRA experts according to client data: FedAMoLE Zhang et al. (2026) assigns different sets of LoRA experts to different clients, FedLEASE Wang et al. (2025) adaptively allocates domain-specific LoRA experts and selects a client-dependent number of them, and FedWeave Duan et al. (2026) coordinates expert and router training at a finer prototype level. These methods demonstrate the effectiveness of federated expert selection, but their expert pools are primarily constructed from a common PEFT family or expert architecture.

Expert balancing is a related issue in sparse routing. Loss-Free Balancing adjusts expert-wise routing biases according to recent loads, avoiding an additional balancing loss Wang et al. (2024a). Hermes adapts this principle to client-specific PEFT usage histories. Its focus is not the balancing rule alone, but the federated coordination of structurally different PEFT modules: client-side routing determines which modules are locally optimized, while the resulting selective updates must be coordinated separately at the server. This selection-aggregation coupling distinguishes Hermes from centralized heterogeneous PEFT mixtures and federated mixtures built from a single PEFT family.

B Algorithm and Implementation Details

B.1 PEFT Approaches

Before introducing Hermes, we briefly review the PEFT modules considered in this work.

Algorithm 1 Hermes

Require: Frozen backbone Θ; PEFT set $\mathcal{M} = \{A, L, P\}$; number of layers L; number of clients n; communication rounds T; local epochs E_p; Top-k value k; learning rate η; aggregation exponent β; bias step size γ_b.

Server initialization. Initialize global PEFT pool $\{\theta^{(l)}_{m,g}\}_{m\in\mathcal{M},\, l=1,\dots,L}$.
Initialize per-client routing bias $b^{(l)}_{m,i} \leftarrow 0$ and historical gradient EMA $s^{(g,l)}_{m,i} \leftarrow 0$.
for each communication round $t = 1, 2, \dots, T$ **do**
  # Client Update (in parallel)
  for each client $i \in \{1, \dots, n\}$ **in parallel do**
    Download global PEFT pool $\{\theta^{(l)}_{m,g}\}_{m\in\mathcal{M}}$ and set $\theta^{(l)}_{m,i} \leftarrow \theta^{(l)}_{m,g}$.
    Reset usage counter $c^{(l)}_{m,i} \leftarrow 0$ for all (m, l).
    for local step $s = 1, \dots, E_p$ **do**
      for each layer $l = 1, \dots, L$ **do**
        Compute feature-aware score $s^{(f,l)}_{m,i}$ from pre-block hidden state via Eq. (4).
        Retrieve historical gradient-aware score $s^{(g,l)}_{m,i}$ (Eq. (5)) and bias $b^{(l)}_{m,i}$.
        Form routing logits $r^{(l)}_{m,i}$ via Eq. (7).
        Select Top-k active set $\mathcal{S}^{(l)}_i$ and obtain sparse-softmax weights $\alpha^{(l)}_{m,i}$ via Eq. (8).
        Evaluate *only* activated PEFT residuals $\Delta^{(l)}_m$, $m \in \mathcal{S}^{(l)}_i$, and compute routed output $h^{(l)}_i$ via Eq. (9).
        Increment usage $c^{(l)}_{m,i} \mathrel{+}= N_b$ for all $m \in \mathcal{S}^{(l)}_i$.
      end for
      Backpropagate loss; update activated PEFT parameters $\theta^{(l)}_{m,i} \leftarrow \theta^{(l)}_{m,i} - \eta \nabla_{\theta^{(l)}_{m,i}} \mathcal{L}_i$ for $m \in \mathcal{S}^{(l)}_i$.
      Update EMA $s^{(g,l)}_{m,i}$ for activated modules (Eq. (5)); update anti-collapse bias $b^{(l)}_{m,i}$ via Eq. (6).
    end for
    Upload activated updates $\{\theta^{(l)}_{m,i},\, c^{(l)}_{m,i}\}_{c^{(l)}_{m,i}>0,\, l=1,\dots,L}$ to server.
  end for
  # Server Aggregation
  for each layer $l = 1, \dots, L$ and each PEFT $m \in \mathcal{M}$ **do**
    Identify uploading client set $\mathcal{C}^{(l)}_m = \{i : c^{(l)}_{m,i} > 0\}$.
    Compute within-client normalized usage $u^{(l)}_{m,i}$ and inverse-frequency weights $\omega^{(l)}_{m,i}$ via Eq. (11).
    Aggregate $\theta^{(l)}_{m,g} \leftarrow \frac{1}{Z^{(l)}_m} \sum_{i\in\mathcal{C}^{(l)}_m} \omega^{(l)}_{m,i} \theta^{(l)}_{m,i}$ via Eq. (10).
  end for
end for

Ensure: Global PEFT pool $\{\theta^{(l)}_{m,g}\}_{m\in\mathcal{M},\, l=1,\dots,L}$.

Adapter. Adapter Houlsby et al. (2019) adds a trainable bottleneck layer after the feedforward network in each Transformer layer of an LLM. Each adapter consists of a down-projection weight $W_{down} \in \mathbb{R}^{r_a \times d}$, an up-projection weight $W_{up} \in \mathbb{R}^{d \times r_a}$, and a non-linearity $\varphi(\cdot)$, such as ReLU. The adapter residual is defined as $\Delta E_{\theta_A}(h) = W_{up} \cdot \varphi(W_{down} \cdot h)$, where $h \in \mathbb{R}^d$.

LoRA. LoRA Hu et al. (2022) only updates the parameters of some layers, *e.g.*, self-attention. Consider a layer in the network, LoRA freezes the LLM weight $W_0 \in \mathbb{R}^{d_1 \times d_2}$ and inserts a trainable update in the form $\Delta E_{\theta_L}(h) = (BA)x$ where $A \in \mathbb{R}^{r \times d_2}$, $B \in \mathbb{R}^{d_1 \times r}$, and $r \leq \min(d_1, d_2)$. To ensure consistency with the pre-trained weight during the initial phase, B is initialized as a zero matrix, while A is initialized with Gaussian noise $\mathcal{N}(0, \sigma^2)$.

Prefix-tuning. Prefix-tuning Li & Liang (2021) optimizes a small set of continuous task-specific vectors, called *prefixes*, which are prepended to the input sequence at each Transformer layer. Prefix-

tuning introduces trainable key-value pairs $(K_p, V_p) \in \mathbb{R}^{l_p \times d}$, where l_p is the prefix length and d is the hidden size. During self-attention, the output is modified as $\Delta E_{\theta_P}(h) = \mathrm{Attn}(Q, [K; K_p], [V; V_p]) - \mathrm{Attn}(Q, K, V)$, $[K; K_p]$ means concatenating the original keys with the *prefixes*.

B.2 Algorithm

Algorithm 1 summarizes the full procedure of Hermes. At each round, clients download the global PEFT pool, perform Top-k sparse routing at every layer using the feature-aware score (Eq. (4)), the historical gradient-aware EMA (Eq. (5)), and the anti-collapse bias (Eq. (6)), and update only the activated PEFT modules locally; inactive modules incur no additional forward or backward cost. The server then aggregates uploaded updates module-wise using the inverse-frequency weighting in Eq. (11)–(10), and broadcasts the refreshed PEFT pool for the next round. A more detailed account of routing-time computation is provided in Appendix B.3, and convergence is analyzed in Appendix D.

B.3 Routing-time Computation and State Updates

Let ν index successive local mini-batches processed by a client, including those in different communication rounds. At the beginning of local step ν, client i stores the historical gradient score $s_{m,i}^{(g,l,\nu)}$ and anti-collapse bias $b_{m,i}^{(l,\nu)}$ for each module–layer pair. These states contain information from previous local steps, not from the current backward pass.

Routing before PEFT execution. At layer l, Hermes first computes the feature-aware score $s_{m,i}^{(f,l,\nu)}$ from the current pre-block hidden states. It then forms

$$r_{m,i}^{(l,\nu)} = \gamma s_{m,i}^{(f,l,\nu)} + (1-\gamma) s_{m,i}^{(g,l,\nu)} + b_{m,i}^{(l,\nu)} \tag{12}$$

and selects

$$\mathcal{S}_i^{(l,\nu)} = \mathrm{TopK}\left(\{r_{m,i}^{(l,\nu)}\}_{m\in\mathcal{M}}, k\right). \tag{13}$$

Only the selected PEFT modules are evaluated in the forward pass. Their outputs are combined using the sparse-softmax weights in Eq. (8).

Gradient EMA after backpropagation. Let $g_{m,i}^{(l,\nu)}$ denote the gradient obtained from the current backward pass for a selected module. With EMA decay $\rho_{\mathrm{ema}} \in [0, 1)$, the historical score is updated as

$$s_{m,i}^{(g,l,\nu+1)} = \begin{cases} \rho_{\mathrm{ema}} s_{m,i}^{(g,l,\nu)} + (1-\rho_{\mathrm{ema}}) \dfrac{\|g_{m,i}^{(l,\nu)}\|_2}{\sqrt{|\theta_m^{(l)}|} + \epsilon_g}, & m \in \mathcal{S}_i^{(l,\nu)}, \\ s_{m,i}^{(g,l,\nu)}, & m \notin \mathcal{S}_i^{(l,\nu)}. \end{cases} \tag{14}$$

Thus, unselected modules retain their previous EMA values. The gradient magnitudes are detached before updating the stored states.

Bias update for all modules. After processing the mini-batch, Hermes updates every routing bias:

$$b_{m,i}^{(l,\nu+1)} = b_{m,i}^{(l,\nu)} + \gamma_b \left(1 - \frac{|\mathcal{M}|}{k} \mathbf{1}\{m \in \mathcal{S}_i^{(l,\nu)}\}\right), \qquad \forall m \in \mathcal{M}. \tag{15}$$

This is equivalent to Eq. (6). In particular, an unselected module receives a positive bias correction even though its PEFT parameters and gradient EMA are unchanged. The bias update is performed outside the gradient graph.

State across communication rounds. The gradient EMAs and routing biases are initialized once and retained across the client's participating rounds. The round-level usage counters $c_{m,i}^{(l)}$ are instead reset at the beginning of each participating round. These counters accumulate usage across all local mini-batches and determine the uploaded module set.

Consequently, sparse routing avoids evaluating unselected PEFT modules at the current step, but does not imply that only k modules are uploaded at the end of a round. The upload set is the union of the module selections made during that round.

C Motivations Analysis

C.1 Hybrid Parameter Space is Less Constrained

We provide a detailed proof for Eq. (2). Recall that each PEFT method $m \in \mathcal{M} = \{A, L, P\}$ defines a feasible parameter space Θ_m for the trainable parameters introduced by the corresponding PEFT block. In the heterogeneous setting, a client maintains multiple PEFT blocks and may activate a subset of them. Strictly speaking, a heterogeneous PEFT configuration consists of both PEFT parameters and activation indicators. Let

$$\mathcal{A}^{\le k} \triangleq \left\{ a \in \{0,1\}^{|\mathcal{M}|} : 1 \le \|a\|_0 \le k \right\}, \tag{16}$$

where $a_m = 1$ means PEFT module m is activated and $a_m = 0$ means it is bypassed. The effective hybrid feasible space is

$$\Theta_{\text{hyb}}^{\text{eff}} \triangleq \Theta_A \times \Theta_L \times \Theta_P \times \mathcal{A}^{\le k}. \tag{17}$$

For notational simplicity, the main text writes only the parameter product $\Theta_{\text{hyb}} = \Theta_A \times \Theta_L \times \Theta_P$ and suppresses the activation indicator.

Bypass convention. The no-op behavior of an inactive PEFT block is implemented by the activation indicator, i.e., the block is bypassed in the forward computation. This does not require every PEFT family to admit a trainable parameter value that exactly produces a zero residual. For PEFT methods with an exact zero-residual parameterization, such as LoRA under zero low-rank updates, the bypass view coincides with the usual parameter-level no-op. For PEFT methods whose internal computation changes attention or sublayer structure, the bypass indicator provides the corresponding effective no-op.

Proposition C.1 (Hybrid feasibility dominates homogeneous feasibility)**.** *For any client i, define the local objective*

$$\mathcal{L}_i(\theta, a) \triangleq \mathbb{E}_{(x,y)\sim\mathcal{D}_i}\big[\ell(f(x;\Theta,\theta,a),y)\big], \qquad (\theta,a) \in \Theta_{\text{hyb}}^{\text{eff}}.$$

Then

$$\inf_{(\theta,a)\in\Theta_{\text{hyb}}^{\text{eff}}} \mathcal{L}_i(\theta,a) \le \min_{m\in\mathcal{M}} \inf_{\theta_m\in\Theta_m} \mathcal{L}_i\big(\iota_m(\theta_m)\big),$$

where $\iota_m(\cdot)$ embeds a homogeneous PEFT parameter into the effective hybrid space by activating only the m-th PEFT module.

Proof. We prove the claim by explicitly constructing, for each homogeneous design, a feasible subset of the effective hybrid space, and then using the monotonicity of the infimum.

Step 1: Define the embedding ι_m and the corresponding homogeneous subset. Fix any $m \in \mathcal{M}$. Let $e_m \in \{0,1\}^{|\mathcal{M}|}$ denote the one-hot activation vector with $(e_m)_m = 1$ and $(e_m)_{m'} = 0$ for all $m' \neq m$. For inactive PEFT blocks, choose arbitrary valid parameters $\bar{\theta}_{m'} \in \Theta_{m'}$; their values do not affect the forward computation because these blocks are bypassed. Define the embedding map $\iota_m : \Theta_m \to \Theta_{\text{hyb}}^{\text{eff}}$ by

$$\iota_m(\theta_m) \triangleq (\theta_A, \theta_L, \theta_P, e_m), \qquad \text{where } \theta_m \text{ is free and } \theta_{m'} = \bar{\theta}_{m'} \text{ for } m' \neq m. \tag{18}$$

Let

$$\widetilde{\Theta}_m \triangleq \{\iota_m(\theta_m) : \theta_m \in \Theta_m\} \subseteq \Theta_{\text{hyb}}^{\text{eff}}$$

denote the embedded feasible set corresponding to the homogeneous choice "use only PEFT m".

Step 2: Establish set inclusion. By construction, every element of $\widetilde{\Theta}_m$ is a valid effective hybrid configuration: it contains valid PEFT parameters and a valid one-hot activation vector. Hence,

$$\widetilde{\Theta}_m \subseteq \Theta_{\text{hyb}}^{\text{eff}}, \qquad \forall m \in \mathcal{M}.$$

Step 3: Use monotonicity of infimum under set inclusion. For any real-valued function ϕ and sets $\mathcal{A} \subseteq \mathcal{B}$,

$$\inf_{x\in\mathcal{B}} \phi(x) \le \inf_{x\in\mathcal{A}} \phi(x).$$

Apply this property with $\phi(\theta, a) = \mathcal{L}_i(\theta, a)$, $\mathcal{A} = \widetilde{\Theta}_m$, and $\mathcal{B} = \Theta_{\text{hyb}}^{\text{eff}}$:

$$\inf_{(\theta,a)\in\Theta_{\text{hyb}}^{\text{eff}}} \mathcal{L}_i(\theta, a) \le \inf_{(\theta,a)\in\widetilde{\Theta}_m} \mathcal{L}_i(\theta, a).$$

Moreover, since $\widetilde{\Theta}_m = \iota_m(\Theta_m)$,

$$\inf_{(\theta,a)\in\widetilde{\Theta}_m} \mathcal{L}_i(\theta, a) = \inf_{\theta_m\in\Theta_m} \mathcal{L}_i\big(\iota_m(\theta_m)\big).$$

Therefore, for every $m \in \mathcal{M}$,

$$\inf_{(\theta,a)\in\Theta_{\text{hyb}}^{\text{eff}}} \mathcal{L}_i(\theta, a) \le \inf_{\theta_m\in\Theta_m} \mathcal{L}_i\big(\iota_m(\theta_m)\big).$$

Step 4: Take the minimum over $m \in \mathcal{M}$. Since the left-hand side does not depend on m, taking the minimum over the right-hand side yields

$$\inf_{(\theta,a)\in\Theta_{\text{hyb}}^{\text{eff}}} \mathcal{L}_i(\theta, a) \le \min_{m\in\mathcal{M}} \inf_{\theta_m\in\Theta_m} \mathcal{L}_i\big(\iota_m(\theta_m)\big),$$

which completes the proof. □

C.2 Why Naively Optimizing Θ_{hyb} Can Be Unstable

This subsection formalizes representation-level interference and explains why directly updating multiple PEFT blocks in Θ_{hyb} can lead to unstable or inefficient optimization. The key message is that Proposition C.1 is a *feasibility* statement: it compares best achievable values under ideal optimization. Practical federated training, however, uses finite local steps, stochastic gradients, non-IID data, and partial participation. In this regime, simultaneously updating multiple PEFT blocks may induce incompatible changes to the same block-level representation.

Notation. Fix a client i and Transformer layer l, and suppress the client and layer indices when clear. Let

$$R_m(h; \theta_m) \triangleq \Delta_m^{(l)}(h; \Theta, \theta_m)$$

denote the block-output residual effect induced by PEFT module $m \in \mathcal{M} = \{A, L, P\}$. Although different PEFT modules are implemented at different internal positions of a Transformer block, their induced effects can be compared in the common block-output representation space. Let

$$g_m \approx \nabla_{\theta_m} \mathcal{L}$$

be the stochastic gradient for PEFT module m. Consider one local update that changes all PEFT modules:

$$\theta_m^+ = \theta_m - \eta g_m, \qquad m \in \mathcal{M}. \tag{19}$$

Proposition C.2 (Representation-level interference under joint PEFT updates)**.** *Assume each residual map $R_m(h; \theta_m)$ is differentiable with respect to θ_m and has locally Lipschitz Jacobian. Let*

$$J_m \triangleq \frac{\partial R_m(h; \theta_m)}{\partial \theta_m}$$

be the parameter Jacobian of PEFT module m at the current point, and define its first-order induced representation update as

$$u_m \triangleq -\eta J_m g_m. \tag{20}$$

Then the block-output change induced by jointly updating all PEFT modules satisfies

$$\delta h_{\text{joint}} = \sum_{m\in\mathcal{M}} u_m + O\left(\eta^2 \sum_{m\in\mathcal{M}} \|g_m\|^2\right). \tag{21}$$

Ignoring higher-order terms, the squared norm of the effective representation update is

$$\left\| \sum_{m\in\mathcal{M}} u_m \right\|^2 = \sum_{m\in\mathcal{M}} \|u_m\|^2 + 2 \sum_{m<m'} \langle u_m, u_{m'} \rangle. \tag{22}$$

Thus, if two PEFT modules induce negatively aligned representation updates, $\langle u_m, u_{m'} \rangle < 0$, their effects can partially cancel in the shared block-output representation.

Proof. We expand the update at the representation level.

Step 1: First-order expansion of each PEFT residual. For PEFT module m, Taylor expansion after the update $\theta_m^+ = \theta_m - \eta g_m$ gives

$$R_m(h;\theta_m^+) = R_m(h;\theta_m) + J_m(\theta_m^+ - \theta_m) + O(\|\theta_m^+ - \theta_m\|^2).$$

Since $\theta_m^+ - \theta_m = -\eta g_m$, we have

$$R_m(h;\theta_m^+) - R_m(h;\theta_m) = -\eta J_m g_m + O(\eta^2\|g_m\|^2) = u_m + O(\eta^2\|g_m\|^2).$$

Step 2: Sum the induced representation changes. When all PEFT modules are updated together, the first-order change in the shared block-output representation is

$$\delta h_{\text{joint}} = \sum_{m\in\mathcal{M}} \left[R_m(h;\theta_m^+) - R_m(h;\theta_m)\right].$$

Substituting the expansion from Step 1 yields

$$\delta h_{\text{joint}} = \sum_{m\in\mathcal{M}} u_m + O\left(\eta^2 \sum_{m\in\mathcal{M}} \|g_m\|^2\right),$$

which proves Eq. (21).

Step 3: Expand the representation-update norm. All vectors u_m lie in the same block-output representation space, so their pairwise inner products are well-defined. Therefore,

$$\left\| \sum_{m\in\mathcal{M}} u_m \right\|^2 = \sum_{m\in\mathcal{M}} \|u_m\|^2 + 2 \sum_{m<m'} \langle u_m, u_{m'}\rangle.$$

This proves Eq. (22).

Step 4: Interpretation. The cross terms arise after PEFT-specific parameter updates are mapped through their Jacobians into the shared block-output representation space. When $\langle u_m, u_{m'}\rangle < 0$, the corresponding PEFT updates push the representation in opposing directions and may partially cancel. This explains why dense joint updating can be inefficient even though the heterogeneous feasible space is larger. □

Discussion and connection to federated training. Importantly, this interference is not claimed to occur in the raw concatenated parameter-gradient norm. For a concatenated gradient vector $g = (g_A, g_L, g_P)$, one has

$$\|g\|^2 = \sum_{m\in\mathcal{M}} \|g_m\|^2,$$

without cross terms between different parameter blocks. The relevant cross terms arise only after parameter updates are mapped into the shared block-output representation space through J_m. In federated training, clients take finite local steps with stochastic gradients under non-IID data and partial participation. The effective representation updates $\{u_m\}$ can vary across clients, layers, and rounds. Therefore, a naive strategy that jointly updates all PEFT blocks in every layer can be sensitive to step size, noise, and client heterogeneity. This motivates selective activation and updating of PEFT blocks, such as sparse routing with Top-k activation and anti-collapse bias.

D Theoretical Analysis

We study the convergence properties of Hermes. We first formalize the setup and assumptions, then establish stability, routing, selection-bias, and aggregation-deviation lemmas, before presenting the main convergence theorem. Throughout this section, the backbone Θ is frozen and the optimization is over PEFT parameters θ.

D.1 SETUP

Global objective. Let the reference global objective be

$$F(\theta) = \sum_{i=1}^{n} \alpha_i f_i(\theta), \qquad f_i(\theta) = \mathbb{E}_{B_i}\big[\ell(B_i; \Theta, \theta)\big], \tag{23}$$

where $\alpha_i > 0$, $\sum_i \alpha_i = 1$, Θ is the frozen backbone, and $\theta = \{\theta_m^{(l)}\}_{l,m}$ collects all PEFT parameters. The expectation is over local mini-batches B_i, since Hermes routes at the mini-batch level. Here, ℓ is the batch task loss evaluated under a reference dense-routing rule chosen before training, so that f_i is a fixed function of θ. This reference is used only to define the objective for analysis; the actual router parameters, gradient histories, and biases may continue to change during training. Their effect on the actual PEFT gradients is included in the routing-error condition below.

We consider the full-participation protocol and a common number K of local SGD steps per round. The symbol K counts mini-batch updates, whereas E_p in the experimental setup counts local epochs.

Block-level PEFT formulation and routed layer output. At Transformer layer l, denote by $h_i^{(l-1)}$ the layer input on client i, and let $F_0^{(l)}(\cdot; \Theta)$ be the frozen Transformer block. For each PEFT module $m \in \mathcal{M} = \{A, L, P\}$, let

$$\Delta_m^{(l)}\big(h_i^{(l-1)}; \Theta, \theta_{m,i}^{(l)}\big)$$

denote its block-output residual effect. The Top-k router selects $\mathcal{S}_i^{(l)} \subseteq \mathcal{M}$ and assigns weights $\alpha_{m,i}^{(l)} \geq 0$ with $\sum_{m \in \mathcal{S}_i^{(l)}} \alpha_{m,i}^{(l)} = 1$. The routed block output is

$$h_i^{(l)} = F_0^{(l)}(h_i^{(l-1)}; \Theta) + \sum_{m \in \mathcal{S}_i^{(l)}} \alpha_{m,i}^{(l)} \Delta_m^{(l)}\big(h_i^{(l-1)}; \Theta, \theta_{m,i}^{(l)}\big). \tag{24}$$

This is a block-level abstraction: PEFT modules may be implemented at different internal positions, but routing and analysis compare their induced effects at the block-output level.

Local SGD and global aggregation. Starting from a fixed initialization $\bar{\theta}^0$, let $\bar{\theta}^t$ denote the global PEFT parameters after communication round t. At the beginning of round t, client i receives $\bar{\theta}^{t-1}$ and initializes

$$\theta_i^{t,0} = \bar{\theta}^{t-1}.$$

Then client i runs K local steps:

$$\theta_i^{t,s+1} = \theta_i^{t,s} - \eta\, \tilde{g}_i^{t,s}, \qquad s = 0, 1, \ldots, K-1, \tag{25}$$

where $\tilde{g}_i^{t,s}$ denotes the possibly biased stochastic gradient induced by sparse routing at iterate $\theta_i^{t,s}$. Unselected module coordinates are treated as zero in the concatenated gradient vector. After local training, clients upload every module activated at least once in the current round, together with its usage count. The server forms $\bar{\theta}^t$ by the parameter aggregation in Eq. (35).

Substituting the local SGD recursion into this parameter-aggregation rule gives the following equivalent expression:

$$\bar{\theta}^t = \bar{\theta}^{t-1} - \eta \sum_{i=1}^{n} \alpha_i \sum_{s=0}^{K-1} \tilde{g}_i^{t,s} + \zeta^t, \tag{26}$$

where ζ^t is the difference between the actual module-wise aggregation and the full-client weighted reference expression, defined explicitly in Eq. (46). This is an algebraic decomposition of parameter aggregation, not an additional server-side optimization step. Lemma D.4 bounds $\mathbb{E}\|\zeta^t\|^2$.

D.2 ASSUMPTIONS

Assumption A1 (Smoothness and lower boundedness). For each client i, the reference objective $f_i(\theta)$ is L_F-smooth:

$$\|\nabla f_i(\theta) - \nabla f_i(\theta')\| \leq L_F \|\theta - \theta'\|. \tag{27}$$

Also, $F(\theta) \geq F^\star$ for a finite lower bound $F^\star$. We use L_F for the smoothness constant and retain L for the number of Transformer layers.

Assumption A2 (Stochastic gradients). Let $\mathcal{F}_{t,s}$ contain the complete training history before the current local mini-batches are sampled, including the local PEFT iterates and the current router parameters, gradient histories, and biases. For the actual sparse gradient $\tilde{g}_i^{t,s}$, define

$$\xi_i^{t,s} = \tilde{g}_i^{t,s} - \mathbb{E}[\tilde{g}_i^{t,s} \mid \mathcal{F}_{t,s}], \qquad \mathbb{E}[\|\xi_i^{t,s}\|^2 \mid \mathcal{F}_{t,s}] \le \sigma^2. \tag{28}$$

By definition, $\mathbb{E}[\xi_i^{t,s} \mid \mathcal{F}_{t,s}] = 0$. At the same local step, the centered noises are conditionally uncorrelated across clients. We also assume the second-moment bound

$$\mathbb{E}\|\tilde{g}_i^{t,s}\|^2 \le G_g^2 \tag{29}$$

for the iterates encountered. These conditions are imposed on the actual sparse gradients, not transferred from an unexecuted dense computation.

Assumption A3 (PEFT residual regularity). Each block-level residual map $\Delta_m^{(l)}(h; \Theta, \theta_{m,i}^{(l)})$ is L_E-Lipschitz in its input and ρ_E-Lipschitz in its parameters. Its parameter Jacobian satisfies

$$\left\| \frac{\partial \Delta_m^{(l)}(h; \Theta, \theta_{m,i}^{(l)})}{\partial \theta_{m,i}^{(l)}} \right\| \le \rho_E. \tag{30}$$

Assumption A4 (Router regularity and Top-k sparsity). The router produces finite logits through a Lipschitz function of its inputs. Before truncation, it forms $w_{\text{dense}}^{(l)} = \mathrm{softmax}(r^{(l)}/\tau)$, where $\tau > 0$. Top-k routing retains the largest k logits and re-normalizes their weights, with $1 \le k \le |\mathcal{M}|$.

Let $\hat{g}_i^{t,s}$ denote the PEFT gradient of a dense computation on the same mini-batch and at the same PEFT parameters and current router states as $\tilde{g}_i^{t,s}$. The dense computation omits Top-k truncation and recomputes its own hidden states and routing scores. We assume the full-network sensitivity condition

$$\|\tilde{g}_i^{t,s} - \hat{g}_i^{t,s}\| \le C_R \sum_{l=1}^{L} \|w_{\text{topk},i}^{(l,t,s)} - w_{\text{dense},i}^{(l,t,s)}\|_1 \tag{31}$$

almost surely for a finite constant C_R along the iterates encountered. The weight pairs on the right are evaluated at the corresponding block inputs of the actual sparse pass. This condition covers the complete forward and backward computations; it is additional to the blockwise regularity in A3. For convergence, we assume the uniform bound in Eq. (45). That bound includes both Top-k truncation and the difference between the current dense-routing gradient and the fixed reference gradient. It is not implied by the bias-drift lemma.

Assumption A5 (Anti-collapse bias with token-equivalent counts). Hermes performs mini-batch-level block routing. Index consecutive local routing decisions by ν, separately from the communication rounds t. Let $N_b^\nu > 0$ be the number of valid non-padding tokens in the current mini-batch. Define

$$n_{m,i}^{(l,\nu)} = N_b^\nu \mathbf{1}\{m \in \mathcal{S}_i^{(l,\nu)}\}. \tag{32}$$

The reference count is

$$\bar{n}^{(l,\nu)} = \frac{N_b^\nu k}{|\mathcal{M}|}. \tag{33}$$

The anti-collapse bias evolves as

$$b_{m,i}^{(l,\nu+1)} = b_{m,i}^{(l,\nu)} + \gamma_b \frac{\bar{n}^{(l,\nu)} - n_{m,i}^{(l,\nu)}}{\bar{n}^{(l,\nu)}}, \tag{34}$$

where $\gamma_b > 0$ is the bias step size. The bias is updated outside the gradient graph for all modules, including unselected modules.

Assumption A6 (Client-normalized inverse-frequency aggregation). For each module-layer pair (m, l), the server aggregates the uploaded PEFT parameters over $\mathcal{C}_m^{(l)} = \{i : c_{m,i}^{(l)} > 0\}$. Suppressing the round index, for a nonempty contributor set it computes:

$$\theta_{m,g}^{(l)} = \frac{1}{Z_m^{(l)}} \sum_{i \in \mathcal{C}_m^{(l)}} \omega_{m,i}^{(l)} \theta_{m,i}^{(l)}, \qquad Z_m^{(l)} = \sum_{i \in \mathcal{C}_m^{(l)}} \omega_{m,i}^{(l)}. \tag{35}$$

Let $c_{m,i}^{(l)}$ be the round-level token-equivalent usage count, and define the within-client normalized usage ratio

$$u_{m,i}^{(l)} = \frac{c_{m,i}^{(l)}}{\sum_{m' \in \mathcal{M}} c_{m',i}^{(l)} + \epsilon_c}. \tag{36}$$

The aggregation weight is

$$\omega_{m,i}^{(l)} = \alpha_i \left(u_{m,i}^{(l)} + \epsilon_c \right)^{-\beta}, \qquad \beta \in [0,1], \tag{37}$$

where α_i is the standard FedAvg client weight and $\epsilon_c > 0$ is used for numerical stability. If no client uploads a module–layer pair, its previous global parameters are retained. For the aggregation-deviation bound, we additionally assume

$$\|\theta_{m,i}^{(l,t,K)} - \bar{\theta}_m^{(l,t-1)}\| \le \eta K G_{m,l} \tag{38}$$

almost surely for finite constants $G_{m,l}$, where $\theta_{m,i}^{(l,t,K)}$ is the corresponding block of the final local iterate $\theta_i^{t,K}$. This follows, for example, from an almost-sure per-step bound on the module gradient. It is separate from the expected second-moment bound in A2.

D.3 Stability of Block-level PEFT Mixtures

Lemma D.1 (Stability of block-level PEFT mixtures)**.** *Suppose*

$$\left\| \Delta_m^{(l)}\big(h_i^{(l-1)}; \Theta, \theta_{m,i}^{(l)}\big) \right\| \le R_E$$

almost surely, and the routed weights satisfy $\sum_{m \in \mathcal{S}_i^{(l)}} \alpha_{m,i}^{(l)} = 1$ *and* $\alpha_{m,i}^{(l)} \ge 0$*. Then the routed output satisfies*

$$\left\| h_i^{(l)} - F_0^{(l)}(h_i^{(l-1)}; \Theta) \right\| \le R_E, \tag{39}$$

and letting $\delta_i^{(l)} \triangleq \partial \ell / \partial h_i^{(l)}$*,*

$$\left\| \nabla_{\theta_{m,i}^{(l)}} \ell \right\| \le \rho_E \left\| \delta_i^{(l)} \right\|. \tag{40}$$

Proof. We prove the two inequalities separately.

(1) Forward stability. From Eq. (24),

$$h_i^{(l)} - F_0^{(l)}(h_i^{(l-1)}; \Theta) = \sum_{m \in \mathcal{S}_i^{(l)}} \alpha_{m,i}^{(l)} \Delta_m^{(l)}\big(h_i^{(l-1)}; \Theta, \theta_{m,i}^{(l)}\big).$$

Taking norms and applying the triangle inequality gives

$$\left\| h_i^{(l)} - F_0^{(l)}(h_i^{(l-1)}; \Theta) \right\| \le \sum_{m \in \mathcal{S}_i^{(l)}} \alpha_{m,i}^{(l)} \left\| \Delta_m^{(l)}\big(h_i^{(l-1)}; \Theta, \theta_{m,i}^{(l)}\big) \right\|.$$

Using the bound by R_E and $\sum_m \alpha_{m,i}^{(l)} = 1$ yields Eq. (39).

(2) Gradient stability. At the current step, the same-layer routing weights depend on the pre-block input and detached historical statistics, not directly on the current-layer PEFT parameters. Module parameters are not shared across layers, and the realized support is treated as fixed during backpropagation. By the chain rule,

$$\nabla_{\theta_{m,i}^{(l)}} \ell = \alpha_{m,i}^{(l)} \left(\frac{\partial \Delta_m^{(l)}(h_i^{(l-1)}; \Theta, \theta_{m,i}^{(l)})}{\partial \theta_{m,i}^{(l)}} \right)^{\top} \delta_i^{(l)}.$$

Taking norms, using $\alpha_{m,i}^{(l)} \le 1$, and applying Assumption A3 gives

$$\left\| \nabla_{\theta_{m,i}^{(l)}} \ell \right\| \le \left\| \frac{\partial \Delta_m^{(l)}(h_i^{(l-1)}; \Theta, \theta_{m,i}^{(l)})}{\partial \theta_{m,i}^{(l)}} \right\| \cdot \|\delta_i^{(l)}\| \le \rho_E \|\delta_i^{(l)}\|,$$

which proves Eq. (40). □

Bridge. Lemma D.1 provides uniform control on PEFT residual effects and PEFT-parameter sensitivities. The full-network gradient comparison in Lemma D.3 additionally uses the sensitivity condition in A4.

D.4 Router Top-k with Anti-collapse Bias

Lemma D.2 (Anti-collapse bias drift)**.** *Under A5, let $\mathcal{H}_\nu$ denote the history before the next local mini-batch is sampled. Then*

$$\mathbb{E}\left[b_{m,i}^{(l,\nu+1)} - b_{m,i}^{(l,\nu)} \mid \mathcal{H}_\nu\right] = \gamma_b \left(1 - \frac{|\mathcal{M}|}{k} \Pr\{m \in \mathcal{S}_i^{(l,\nu)} \mid \mathcal{H}_\nu\}\right). \tag{41}$$

Consequently, a module with conditional selection probability below $k/|\mathcal{M}|$ receives a positive expected correction; a module above this level receives a negative expected correction.

Proof. For each nonempty mini-batch, the normalized token count satisfies

$$\frac{n_{m,i}^{(l,\nu)}}{\bar{n}^{(l,\nu)}} = \frac{|\mathcal{M}|}{k} \mathbf{1}\{m \in \mathcal{S}_i^{(l,\nu)}\}.$$

Substituting this identity into Eq. (34) and taking conditional expectation proves Eq. (41). □

Bridge. Lemma D.2 establishes the direction of the bias feedback. The convergence proof uses the routing-error bound below; a bound on its magnitude does not follow from the feedback direction alone.

D.5 Selection Bias from Top-k

At each layer of the sparse pass, let $w_{\mathrm{dense},i}^{(l,t,s)}$ denote the pre-truncation softmax weights and let $w_{\mathrm{topk},i}^{(l,t,s)}$ denote their Top-k re-normalization. Both are evaluated at the same current block input. Define the discarded mass as

$$\mathrm{tail}_i^{(l,t,s)} = \sum_{m \notin \mathcal{S}_i^{(l,t,s)}} w_{\mathrm{dense},m,i}^{(l,t,s)}. \tag{42}$$

Lemma D.3 (Selection bias bound)**.** *The pre-truncation and Top-k weights satisfy*

$$\|w_{\mathrm{topk},i}^{(l,t,s)} - w_{\mathrm{dense},i}^{(l,t,s)}\|_1 = 2\,\mathrm{tail}_i^{(l,t,s)}. \tag{43}$$

Under A4,

$$\begin{aligned} &\left\|\mathbb{E}[\tilde{g}_i^{t,s} \mid \mathcal{F}_{t,s}] - \nabla f_i(\theta_i^{t,s})\right\| \\ &\quad \le 2C_R\,\mathbb{E}\left[\sum_{l=1}^{L} \mathrm{tail}_i^{(l,t,s)} \,\middle|\, \mathcal{F}_{t,s}\right] + \left\|\mathbb{E}[\hat{g}_i^{t,s} \mid \mathcal{F}_{t,s}] - \nabla f_i(\theta_i^{t,s})\right\|. \end{aligned} \tag{44}$$

Proof. We separate the local weight truncation from the comparison of complete network gradients.

Step 1: Express the block-output difference. At a common input $h_i^{(l-1)}$, dense and sparse block compositions differ by

$$h_{i,\mathrm{topk}}^{(l)} - h_{i,\mathrm{dense}}^{(l)} = \sum_{m \in \mathcal{M}} \left(w_{\mathrm{topk},m}^{(l)} - w_{\mathrm{dense},m}^{(l)}\right) \Delta_m^{(l)}.$$

Under the residual bound in Lemma D.1, its norm is at most $R_E \|w_{\mathrm{topk}}^{(l)} - w_{\mathrm{dense}}^{(l)}\|_1$. This is a single-block comparison at a common input; complete dense and sparse passes can have different intermediate inputs and backpropagated signals.

Step 2: Bound the complete gradient difference. Using conditional Jensen's inequality and A4,

$$\begin{aligned} &\left\|\mathbb{E}[\tilde{g}_i^{t,s} - \hat{g}_i^{t,s} \mid \mathcal{F}_{t,s}]\right\| \\ &\quad \le \mathbb{E}[\|\tilde{g}_i^{t,s} - \hat{g}_i^{t,s}\| \mid \mathcal{F}_{t,s}] \\ &\quad \le C_R\,\mathbb{E}\left[\sum_{l=1}^{L} \|w_{\mathrm{topk},i}^{(l,t,s)} - w_{\mathrm{dense},i}^{(l,t,s)}\|_1 \,\middle|\, \mathcal{F}_{t,s}\right]. \end{aligned}$$

Step 3: Evaluate the routing-weight mismatch. Fix a layer and suppress the indices. Write $p = w_{\text{dense}}$, $q = w_{\text{topk}}$, and $a = \sum_{m \notin \mathcal{S}} p_m$. Because the retained mass is $1 - a > 0$, $q_m = p_m/(1-a)$ on $\mathcal{S}$ and $q_m = 0$ outside $\mathcal{S}$. Therefore,

$$\begin{aligned}\|q - p\|_1 &= \sum_{m \in \mathcal{S}} \left| \frac{p_m}{1-a} - p_m \right| + \sum_{m \notin \mathcal{S}} p_m \\ &= (1-a)\frac{a}{1-a} + a = 2a.\end{aligned}$$

This proves Eq. (43).

Step 4: Relate the current gradient to the reference objective. Add and subtract the conditional mean of $\hat{g}_i^{t,s}$:

$$\begin{aligned}&\mathbb{E}[\tilde{g}_i^{t,s} \mid \mathcal{F}_{t,s}] - \nabla f_i(\theta_i^{t,s}) \\ &\quad = \mathbb{E}[\tilde{g}_i^{t,s} - \hat{g}_i^{t,s} \mid \mathcal{F}_{t,s}] + \mathbb{E}[\hat{g}_i^{t,s} \mid \mathcal{F}_{t,s}] - \nabla f_i(\theta_i^{t,s}).\end{aligned}$$

The triangle inequality and Steps 2–3 prove Eq. (44). □

For convergence, assume that a deterministic constant ϵ_{sel} bounds the right-hand side uniformly over the analyzed iterates:

$$\begin{aligned}&2C_R\,\mathbb{E}\left[\sum_{l=1}^{L} \text{tail}_i^{(l,t,s)} \,\middle|\, \mathcal{F}_{t,s}\right] \\ &\quad + \left\|\mathbb{E}[\hat{g}_i^{t,s} \mid \mathcal{F}_{t,s}] - \nabla f_i(\theta_i^{t,s})\right\| \le \epsilon_{\text{sel}}\end{aligned} \tag{45}$$

almost surely for every client, round, and local step. The first term measures truncation; the second accounts for the difference between the current dense routing computation and the fixed reference objective. The latter need not be zero when the router evolves.

Bridge. Lemma D.3 identifies the two contributions to the routing error. The uniform bound in Eq. (45) is a stated condition for the convergence theorem, not a consequence of the Top-k identity alone.

D.6 Client-normalized Inverse-frequency Aggregation

For a nonempty contributor set in round t, write $\lambda_{m,i}^{(l,t)} = \omega_{m,i}^{(l,t)}/Z_m^{(l,t)}$ for the normalized coefficient in A6. For fixed realized usage counts, the relative weights of two contributing clients satisfy

$$\frac{\lambda_{m,i}^{(l,t)}}{\lambda_{m,j}^{(l,t)}} = \frac{\alpha_i}{\alpha_j}\left(\frac{u_{m,j}^{(l,t)} + \epsilon_c}{u_{m,i}^{(l,t)} + \epsilon_c}\right)^{\beta}.$$

Thus, when $u_{m,i}^{(l,t)} > u_{m,j}^{(l,t)}$ and $\beta > 0$, client i has less relative weight than under $\beta = 0$ over the same contributor set.

To analyze aggregation, use the uploaded parameter differences directly. The module-wise aggregation in A6 gives

$$\bar{\theta}_m^{(l,t)} - \bar{\theta}_m^{(l,t-1)} = \sum_{i \in \mathcal{C}_m^{(l,t)}} \lambda_{m,i}^{(l,t)} \big(\theta_{m,i}^{(l,t,K)} - \bar{\theta}_m^{(l,t-1)}\big).$$

Hence, define its deviation from the full-client weighted reference as

$$\zeta_m^{(l,t)} = \sum_{i \in \mathcal{C}_m^{(l,t)}} \big(\lambda_{m,i}^{(l,t)} - \alpha_i\big)\big(\theta_{m,i}^{(l,t,K)} - \bar{\theta}_m^{(l,t-1)}\big). \tag{46}$$

Clients outside $\mathcal{C}_m^{(l,t)}$ have not updated that module, so its local parameter difference is zero. Substituting

$$\theta_{m,i}^{(l,t,K)} - \bar{\theta}_m^{(l,t-1)} = -\eta \sum_{s=0}^{K-1} \tilde{g}_{m,i}^{(l,t,s)}$$

into the preceding expressions yields Eq. (26). Thus, all gradients in that identity come from local training; the server still performs only parameter aggregation.

Lemma D.4 (Aggregation deviation bound)**.** *For each nonempty contributor set, define*

$$d_m^{(l,t)} = \sum_{i \in \mathcal{C}_m^{(l,t)}} \left| \lambda_{m,i}^{(l,t)} - \alpha_i \right|. \tag{47}$$

Under the local-update bound in A6,

$$\|\zeta_m^{(l,t)}\| \le \eta K G_{m,l} d_m^{(l,t)}. \tag{48}$$

Set $d_m^{(l,t)} = 0$ and $\zeta_m^{(l,t)} = 0$ for an empty contributor set. If the deterministic bound ϵ_{agg} satisfies

$$\sum_{l=1}^{L} \sum_{m \in \mathcal{M}} G_{m,l}^2 \big(d_m^{(l,t)}\big)^2 \le \epsilon_{\mathrm{agg}}^2 \tag{49}$$

almost surely for every round, then

$$\|\zeta^t\|^2 \le \eta^2 K^2 \epsilon_{\mathrm{agg}}^2, \tag{50}$$

where ζ^t concatenates the module–layer deviations.

Proof. Applying the triangle inequality to Eq. (46) and using Eq. (38) gives

$$\begin{aligned} \|\zeta_m^{(l,t)}\| &\le \sum_{i \in \mathcal{C}_m^{(l,t)}} |\lambda_{m,i}^{(l,t)} - \alpha_i| \|\theta_{m,i}^{(l,t,K)} - \bar{\theta}_m^{(l,t-1)}\| \\ &\le \eta K G_{m,l} d_m^{(l,t)}. \end{aligned}$$

Since module–layer pairs occupy disjoint parameter blocks,

$$\begin{aligned} \|\zeta^t\|^2 &= \sum_{l=1}^{L} \sum_{m \in \mathcal{M}} \|\zeta_m^{(l,t)}\|^2 \\ &\le \eta^2 K^2 \sum_{l=1}^{L} \sum_{m \in \mathcal{M}} G_{m,l}^2 \big(d_m^{(l,t)}\big)^2 \le \eta^2 K^2 \epsilon_{\mathrm{agg}}^2. \end{aligned}$$

□

Bridge. Lemma D.4 bounds the actual aggregation deviation, including both module-specific contributor normalization and inverse-frequency reweighting. The bound holds for the realized uploads, even when the weights depend on local training; no zero-mean assumption on ζ^t is required.

D.7 Convergence of Federated Training

We now provide a detailed proof of the convergence bound. Compared with standard FedAvg analysis, our proof additionally handles (i) routing-induced selection bias from Lemma D.3, and (ii) aggregation deviation from Lemma D.4.

Two auxiliary inequalities. We use the following elementary inequalities repeatedly.

(I1) Young's inequality. For any vectors a, b and any $\lambda > 0$,

$$2\langle a, b \rangle \le \lambda \|a\|^2 + \frac{1}{\lambda} \|b\|^2. \tag{51}$$

(I2) Weighted Jensen / Cauchy–Schwarz. For nonnegative weights $\{\alpha_i\}$ with $\sum_i \alpha_i = 1$,

$$\left\| \sum_i \alpha_i v_i \right\|^2 \le \sum_i \alpha_i \|v_i\|^2. \tag{52}$$

Lemma D.5 (Local drift bound)**.** *Assume A2 and $\mathbb{E}\|\tilde{g}_i^{t,s}\|^2 \le G_g^2$ for all iterates encountered. Then for each client i, round t, and $s \in \{0, \dots, K\}$,*

$$\mathbb{E}\left\|\theta_i^{t,s} - \bar{\theta}^{t-1}\right\|^2 \le \eta^2 s^2 G_g^2. \tag{53}$$

Consequently,

$$\sum_{s=0}^{K-1} \mathbb{E}\left\|\theta_i^{t,s} - \bar{\theta}^{t-1}\right\| \le \eta G_g \cdot \frac{K(K-1)}{2}. \tag{54}$$

Proof. From Eq. (25),

$$\theta_i^{t,s} - \theta_i^{t,0} = -\eta \sum_{r=0}^{s-1} \tilde{g}_i^{t,r}.$$

Since $\theta_i^{t,0} = \bar{\theta}^{t-1}$,

$$\theta_i^{t,s} - \bar{\theta}^{t-1} = -\eta \sum_{r=0}^{s-1} \tilde{g}_i^{t,r}.$$

Taking squared norm and using $\| \sum_{r=0}^{s-1} a_r\|^2 \le s \sum_{r=0}^{s-1} \|a_r\|^2$ gives

$$\left\|\theta_i^{t,s} - \bar{\theta}^{t-1}\right\|^2 \le \eta^2 s \sum_{r=0}^{s-1} \left\|\tilde{g}_i^{t,r}\right\|^2.$$

Taking expectation and using $\mathbb{E}\|\tilde{g}_i^{t,r}\|^2 \le G_g^2$ yields

$$\mathbb{E}\left\|\theta_i^{t,s} - \bar{\theta}^{t-1}\right\|^2 \le \eta^2 s^2 G_g^2,$$

which proves Eq. (53). For Eq. (54), Jensen's inequality gives

$$\mathbb{E}\left\|\theta_i^{t,s} - \bar{\theta}^{t-1}\right\| \le \sqrt{\mathbb{E}\left\|\theta_i^{t,s} - \bar{\theta}^{t-1}\right\|^2} \le \eta s G_g.$$

Summing over $s = 0, \dots, K-1$ completes the proof. □

Bridge. Lemma D.5 controls the deviation between local iterates and the global iterate. This is where the local-step factor K enters the convergence rate.

Theorem D.6 (Convergence to a stationary neighborhood)**.** *Under A1–A6 and the uniform error bounds in Eqs. (45) and (49), let K be the common number of local SGD steps per communication round. For $\eta > 0$ such that*

$$L_F \eta K \le c_0 \tag{55}$$

for a sufficiently small numerical constant $c_0 > 0$, after T rounds,

$$\begin{aligned}\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\|\nabla F(\bar{\theta}^{t-1})\|^2 \le \mathcal{O}\left(\frac{F(\bar{\theta}^0) - F^\star}{\eta T K}\right) + \mathcal{O}\left(L_F \eta \sigma^2 \sum_{i=1}^{n} \alpha_i^2\right) \\ + \mathcal{O}\left(L_F^2 \eta^2 K^2 G_g^2\right) + \mathcal{O}\left(\epsilon_{\text{sel}}^2 + \epsilon_{\text{agg}}^2\right).\end{aligned} \tag{56}$$

The constants hidden by $\mathcal{O}$ are numerical. For uniform client weights, $\sum_i \alpha_i^2 = 1/n$.

Proof. We give a detailed derivation and track the routing and aggregation terms.

Step 1: Apply the smoothness inequality to F. By Assumption A1, F is L_F-smooth in θ. Therefore,

$$F(\bar{\theta}^t) \le F(\bar{\theta}^{t-1}) + \left\langle \nabla F(\bar{\theta}^{t-1}), \bar{\theta}^t - \bar{\theta}^{t-1} \right\rangle + \frac{L_F}{2}\left\|\bar{\theta}^t - \bar{\theta}^{t-1}\right\|^2. \tag{57}$$

Define

$$\Delta^t \triangleq \bar{\theta}^t - \bar{\theta}^{t-1}.$$

Then Eq. (57) can be written as

$$F(\bar{\theta}^t) \le F(\bar{\theta}^{t-1}) + \left\langle \nabla F(\bar{\theta}^{t-1}), \Delta^t \right\rangle + \frac{L_F}{2}\|\Delta^t\|^2. \tag{58}$$

Step 2: Substitute the global update decomposition. From Eq. (26),

$$\Delta^t = -\eta \sum_{i=1}^{n} \alpha_i \sum_{s=0}^{K-1} \tilde{g}_i^{t,s} + \zeta^t.$$

Plugging this into Eq. (58) gives

$$F(\bar{\theta}^t) \le F(\bar{\theta}^{t-1}) - \eta \left\langle \nabla F(\bar{\theta}^{t-1}), \sum_{i=1}^{n} \alpha_i \sum_{s=0}^{K-1} \tilde{g}_i^{t,s} \right\rangle + \left\langle \nabla F(\bar{\theta}^{t-1}), \zeta^t \right\rangle + \frac{L_F}{2} \left\| -\eta \sum_{i=1}^{n} \alpha_i \sum_{s=0}^{K-1} \tilde{g}_i^{t,s} + \zeta^t \right\|^2. \tag{59}$$

Step 3: Take expectation and handle the aggregation deviation. Using $\|a+b\|^2 \le 2\|a\|^2 + 2\|b\|^2$ with

$$a = -\eta \sum_i \alpha_i \sum_s \tilde{g}_i^{t,s}, \qquad b = \zeta^t,$$

and taking expectation, we obtain

$$\mathbb{E}[F(\bar{\theta}^t)] \le \mathbb{E}[F(\bar{\theta}^{t-1})] - \eta\, \mathbb{E} \left\langle \nabla F(\bar{\theta}^{t-1}), \sum_{i=1}^{n} \alpha_i \sum_{s=0}^{K-1} \tilde{g}_i^{t,s} \right\rangle + \mathbb{E} \left\langle \nabla F(\bar{\theta}^{t-1}), \zeta^t \right\rangle + L_F \eta^2 \mathbb{E} \left\| \sum_{i=1}^{n} \alpha_i \sum_{s=0}^{K-1} \tilde{g}_i^{t,s} \right\|^2 + L_F \mathbb{E}\|\zeta^t\|^2. \tag{60}$$

The aggregation deviation need not have zero mean. Young's inequality gives

$$\langle \nabla F(\bar{\theta}^{t-1}), \zeta^t \rangle \le \frac{\eta K}{8} \|\nabla F(\bar{\theta}^{t-1})\|^2 + \frac{2}{\eta K} \|\zeta^t\|^2. \tag{61}$$

Consequently,

$$\mathbb{E}[F(\bar{\theta}^t)] \le \mathbb{E}[F(\bar{\theta}^{t-1})] - \eta\, \mathbb{E} \left\langle \nabla F(\bar{\theta}^{t-1}), \sum_i \alpha_i \sum_s \tilde{g}_i^{t,s} \right\rangle + \frac{\eta K}{8} \mathbb{E}\|\nabla F(\bar{\theta}^{t-1})\|^2 + L_F \eta^2 \mathbb{E} \left\| \sum_i \alpha_i \sum_s \tilde{g}_i^{t,s} \right\|^2 + \left(L_F + \frac{2}{\eta K} \right) \mathbb{E}\|\zeta^t\|^2. \tag{62}$$

Step 4: Decompose $\tilde{g}_i^{t,s}$ into true gradient, routing bias, and noise. Conditioned on $\mathcal{F}_{t,s}$, define

$$\delta_i^{t,s} \triangleq \mathbb{E}[\tilde{g}_i^{t,s} \mid \mathcal{F}_{t,s}] - \nabla f_i(\theta_i^{t,s}),$$

which captures the routing-induced selection bias, and define

$$\xi_i^{t,s} \triangleq \tilde{g}_i^{t,s} - \mathbb{E}[\tilde{g}_i^{t,s} \mid \mathcal{F}_{t,s}], \qquad \mathbb{E}[\xi_i^{t,s} \mid \mathcal{F}_{t,s}] = 0.$$

Then

$$\tilde{g}_i^{t,s} = \nabla f_i(\theta_i^{t,s}) + \delta_i^{t,s} + \xi_i^{t,s}. \tag{63}$$

By Lemma D.3,

$$\|\delta_i^{t,s}\| \le \epsilon_{\text{sel}}, \tag{64}$$

where ϵ_{sel} is defined in Eq. (45).

Step 5: Lower-bound the descent inner product. Let

$$\mathcal{T}_1 \triangleq \mathbb{E}\left\langle \nabla F(\bar{\theta}^{t-1}), \sum_{i=1}^{n} \alpha_i \sum_{s=0}^{K-1} \tilde{g}_i^{t,s} \right\rangle.$$

Using Eq. (63), we decompose $\mathcal{T}_1$ into

$$\begin{aligned}\mathcal{T}_1 =& \underbrace{\mathbb{E}\left\langle \nabla F(\bar{\theta}^{t-1}), \sum_i \alpha_i \sum_s \nabla f_i(\theta_i^{t,s}) \right\rangle}_{\mathcal{T}_{1a}} + \underbrace{\mathbb{E}\left\langle \nabla F(\bar{\theta}^{t-1}), \sum_i \alpha_i \sum_s \delta_i^{t,s} \right\rangle}_{\mathcal{T}_{1b}} \\ &+ \underbrace{\mathbb{E}\left\langle \nabla F(\bar{\theta}^{t-1}), \sum_i \alpha_i \sum_s \xi_i^{t,s} \right\rangle}_{\mathcal{T}_{1c}}. \end{aligned} \tag{65}$$

(i) Main gradient term $\mathcal{T}_{1a}$. Add and subtract $\nabla f_i(\bar{\theta}^{t-1})$:

$$\nabla f_i(\theta_i^{t,s}) = \nabla f_i(\bar{\theta}^{t-1}) + \left[\nabla f_i(\theta_i^{t,s}) - \nabla f_i(\bar{\theta}^{t-1})\right].$$

Since

$$\sum_i \alpha_i \nabla f_i(\bar{\theta}^{t-1}) = \nabla F(\bar{\theta}^{t-1}),$$

we have

$$\begin{aligned}\mathcal{T}_{1a} &= K\,\mathbb{E}\left\|\nabla F(\bar{\theta}^{t-1})\right\|^2 \\ &\quad + \sum_{s=0}^{K-1} \mathbb{E}\left\langle \nabla F(\bar{\theta}^{t-1}), \sum_i \alpha_i \left[\nabla f_i(\theta_i^{t,s}) - \nabla f_i(\bar{\theta}^{t-1})\right] \right\rangle. \end{aligned} \tag{66}$$

The second term is controlled by Cauchy–Schwarz, Jensen, and smoothness:

$$\left\| \sum_i \alpha_i \left[\nabla f_i(\theta_i^{t,s}) - \nabla f_i(\bar{\theta}^{t-1})\right] \right\| \le \sum_i \alpha_i L_F \left\|\theta_i^{t,s} - \bar{\theta}^{t-1}\right\|.$$

Therefore,

$$\begin{aligned}&\left| \left\langle \nabla F(\bar{\theta}^{t-1}), \sum_i \alpha_i [\nabla f_i(\theta_i^{t,s}) - \nabla f_i(\bar{\theta}^{t-1})] \right\rangle \right| \\ &\le \left\|\nabla F(\bar{\theta}^{t-1})\right\| \cdot L_F \sum_i \alpha_i \left\|\theta_i^{t,s} - \bar{\theta}^{t-1}\right\|. \end{aligned}$$

Applying Young's inequality with $\lambda = \frac{1}{2}$ gives

$$\begin{aligned}&\left\|\nabla F(\bar{\theta}^{t-1})\right\| \cdot L_F \sum_i \alpha_i \left\|\theta_i^{t,s} - \bar{\theta}^{t-1}\right\| \\ &\le \frac{1}{4}\left\|\nabla F(\bar{\theta}^{t-1})\right\|^2 + L_F^2 \left(\sum_i \alpha_i \left\|\theta_i^{t,s} - \bar{\theta}^{t-1}\right\| \right)^2. \end{aligned}$$

Using

$$\left(\sum_i \alpha_i a_i \right)^2 \le \sum_i \alpha_i a_i^2$$

and Lemma D.5,

$$\sum_i \alpha_i \mathbb{E}\left\|\theta_i^{t,s} - \bar{\theta}^{t-1}\right\|^2 \le \eta^2 s^2 G_g^2.$$

Hence,

$$\mathcal{T}_{1a} \ge \frac{3K}{4}\mathbb{E}\left\|\nabla F(\bar{\theta}^{t-1})\right\|^2 - C_d L_F^2 \eta^2 K^3 G_g^2, \tag{67}$$

for some constant $C_d > 0$, where we used $\sum_{s=0}^{K-1} s^2 = \mathcal{O}(K^3)$.

(ii) Routing-bias term $\mathcal{T}_{1b}$. By Cauchy–Schwarz and Eq. (64),

$$|\mathcal{T}_{1b}| \le \mathbb{E}\left[\left\|\nabla F(\bar{\theta}^{t-1})\right\| \cdot \left\|\sum_i \alpha_i \sum_s \delta_i^{t,s}\right\|\right] \le \mathbb{E}\left[\left\|\nabla F(\bar{\theta}^{t-1})\right\| \cdot K\epsilon_{\text{sel}}\right].$$

Applying Young's inequality,

$$|\mathcal{T}_{1b}| \le \frac{K}{4}\mathbb{E}\left\|\nabla F(\bar{\theta}^{t-1})\right\|^2 + K\epsilon_{\text{sel}}^2. \tag{68}$$

(iii) Stochastic-noise term $\mathcal{T}_{1c}$. Since

$$\mathbb{E}[\xi_i^{t,s} \mid \mathcal{F}_{t,s}] = 0,$$

and $\nabla F(\bar{\theta}^{t-1})$ is measurable with respect to $\mathcal{F}_{t,s}$, the tower property gives:

$$\mathcal{T}_{1c} = 0. \tag{69}$$

Combining Eqs. (67)–(69), we obtain

$$\mathcal{T}_1 \ge \frac{K}{2}\mathbb{E}\left\|\nabla F(\bar{\theta}^{t-1})\right\|^2 - C_d L_F^2 \eta^2 K^3 G_g^2 - K\epsilon_{\text{sel}}^2. \tag{70}$$

Step 6: Bound the quadratic gradient term. We now bound

$$\mathcal{Q}_t \triangleq \mathbb{E}\left\|\sum_i \alpha_i \sum_s \tilde{g}_i^{t,s}\right\|^2.$$

Using Eq. (63) and $\|u+v+w\|^2 \le 3(\|u\|^2 + \|v\|^2 + \|w\|^2)$,

$$\mathcal{Q}_t \le 3\mathcal{Q}_g + 3\mathcal{Q}_\delta + 3\mathcal{Q}_\xi,$$

where

$$\mathcal{Q}_g = \mathbb{E}\left\|\sum_i \alpha_i \sum_s \nabla f_i(\theta_i^{t,s})\right\|^2,$$

$$\mathcal{Q}_\delta = \mathbb{E}\left\|\sum_i \alpha_i \sum_s \delta_i^{t,s}\right\|^2, \qquad \mathcal{Q}_\xi = \mathbb{E}\left\|\sum_i \alpha_i \sum_s \xi_i^{t,s}\right\|^2.$$

(i) Bias part. By Eq. (64),

$$\left\|\sum_i \alpha_i \sum_s \delta_i^{t,s}\right\| \le \sum_i \alpha_i \sum_s \|\delta_i^{t,s}\| \le K\epsilon_{\text{sel}}.$$

Thus,

$$\mathcal{Q}_\delta \le K^2\epsilon_{\text{sel}}^2. \tag{71}$$

(ii) Noise part. By A2, noises at the same step are conditionally uncorrelated across clients. Cross-step terms vanish by their conditional zero means. Hence,

$$\mathcal{Q}_\xi = \mathbb{E}\left\|\sum_i \alpha_i \sum_s \xi_i^{t,s}\right\|^2 \le \sum_i \alpha_i^2 \sum_s \mathbb{E}\|\xi_i^{t,s}\|^2.$$

Using Assumption A2,

$$\mathbb{E}\|\xi_i^{t,s}\|^2 \le \sigma^2,$$

and therefore

$$\mathcal{Q}_\xi \le K\sigma^2 \sum_i \alpha_i^2. \tag{72}$$

For comparable client weights, $\sum_i \alpha_i^2 = \mathcal{O}(1/n)$, giving

$$\mathcal{Q}_\xi \leq \mathcal{O}\left(\frac{K\sigma^2}{n}\right).$$

For non-uniform client weights, the same bound is written as $\mathcal{O}(K\sigma^2 \sum_i \alpha_i^2)$.

(iii) Main gradient part. Let

$$A_s \triangleq \sum_i \alpha_i \nabla f_i(\theta_i^{t,s}).$$

Then

$$\mathcal{Q}_g = \mathbb{E}\left\|\sum_{s=0}^{K-1} A_s\right\|^2 \leq K \sum_{s=0}^{K-1} \mathbb{E}\|A_s\|^2.$$

Add and subtract $\nabla F(\bar{\theta}^{t-1})$:

$$A_s = \nabla F(\bar{\theta}^{t-1}) + \sum_i \alpha_i [\nabla f_i(\theta_i^{t,s}) - \nabla f_i(\bar{\theta}^{t-1})].$$

Thus,

$$\|A_s\|^2 \leq 2\left\|\nabla F(\bar{\theta}^{t-1})\right\|^2 + 2\left\|\sum_i \alpha_i [\nabla f_i(\theta_i^{t,s}) - \nabla f_i(\bar{\theta}^{t-1})]\right\|^2.$$

By weighted Jensen and smoothness,

$$\left\|\sum_i \alpha_i [\nabla f_i(\theta_i^{t,s}) - \nabla f_i(\bar{\theta}^{t-1})]\right\|^2 \leq L_F^2 \sum_i \alpha_i \left\|\theta_i^{t,s} - \bar{\theta}^{t-1}\right\|^2.$$

Applying Lemma D.5,

$$\mathbb{E}\|A_s\|^2 \leq 2\mathbb{E}\left\|\nabla F(\bar{\theta}^{t-1})\right\|^2 + 2L_F^2 \eta^2 s^2 G_g^2.$$

Therefore,

$$\mathcal{Q}_g \leq 2K^2 \mathbb{E}\left\|\nabla F(\bar{\theta}^{t-1})\right\|^2 + C_g L_F^2 \eta^2 K^4 G_g^2, \tag{73}$$

for some constant $C_g > 0$.

Combining Eqs. (71)–(73), we obtain

$$\mathcal{Q}_t \leq C_a K^2 \mathbb{E}\left\|\nabla F(\bar{\theta}^{t-1})\right\|^2 + C_b K \sigma^2 \sum_i \alpha_i^2 + C_c K^2 \epsilon_{\text{sel}}^2 + C_d' L_F^2 \eta^2 K^4 G_g^2, \tag{74}$$

where $C_a, C_b, C_c, C_d' > 0$ are constants. One may take $C_a = 6$, $C_b = 3$, $C_c = 3$, and $C_d' = 6$ using $\sum_{s=0}^{K-1} s^2 \leq K^3$.

Step 7: Bound $\mathbb{E}\|\zeta^t\|^2$ using Lemma D.4. Lemma D.4 directly bounds the difference between the actual parameter aggregation and the full-client weighted reference. Taking expectation in Eq. (50) gives

$$\mathbb{E}\|\zeta^t\|^2 \leq \eta^2 K^2 \epsilon_{\text{agg}}^2. \tag{75}$$

This bound controls both the quadratic aggregation term and the aggregation inner product retained in Step 3.

Step 8: Combine all bounds and telescope. Substitute Eqs. (70), (74), and (75) into Eq. (62). Taking $C_d = 1$ in Eq. (67) and the constants specified after Eq. (74) gives

$$\begin{aligned}\mathbb{E}[F(\bar{\theta}^t)] \leq \mathbb{E}[F(\bar{\theta}^{t-1})] &- \eta K\left(\frac{3}{8} - 6L_F\eta K\right)\mathbb{E}\|\nabla F(\bar{\theta}^{t-1})\|^2 \\ &+ L_F^2 \eta^3 K^3 (1 + 6L_F\eta K) G_g^2 + 3L_F \eta^2 K \sigma^2 \sum_i \alpha_i^2 \\ &+ \eta K(1 + 3L_F\eta K)\epsilon_{\text{sel}}^2 + \eta K(2 + L_F\eta K)\epsilon_{\text{agg}}^2.\end{aligned} \tag{76}$$

For example, $c_0 \le 1/48$ ensures a descent coefficient of at least $\eta K/4$. The remaining coefficients satisfy $1 + 6L_F\eta K \le 2$, $1 + 3L_F\eta K \le 2$, and $2 + L_F\eta K \le 3$. Thus,

$$\begin{aligned}\mathbb{E}[F(\bar{\theta}^t)] \le \mathbb{E}[F(\bar{\theta}^{t-1})] - \frac{\eta K}{4}\mathbb{E}\|\nabla F(\bar{\theta}^{t-1})\|^2 \\ + 2L_F^2\eta^3K^3G_g^2 + 3L_F\eta^2K\sigma^2\sum_i \alpha_i^2 \\ + 2\eta K\epsilon_{\text{sel}}^2 + 3\eta K\epsilon_{\text{agg}}^2.\end{aligned} \tag{77}$$

Summing over $t = 1, \dots, T$, using $F(\bar{\theta}^T) \ge F^\star$, and dividing by $\eta KT/4$ yields

$$\begin{aligned}\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\|\nabla F(\bar{\theta}^{t-1})\|^2 \le \frac{4(F(\bar{\theta}^0) - F^\star)}{\eta TK} + 12L_F\eta\sigma^2\sum_i \alpha_i^2 \\ + 8L_F^2\eta^2K^2G_g^2 + 8\epsilon_{\text{sel}}^2 + 12\epsilon_{\text{agg}}^2.\end{aligned}$$

This proves Eq. (56), including the local-drift term and the step-size dependence of the stochastic-noise term. □

Discussion. Under the stated assumptions, Theorem D.6 bounds the average expected squared gradient norm of the reference objective. For fixed η and K, the initial optimization term decreases with the number of communication rounds. The remaining bound depends on stochastic noise, local-update drift, routing-induced gradient bias ϵ_{sel}, and module-wise aggregation deviation ϵ_{agg}. These two additional terms quantify the effects of adaptive module selection and module-wise parameter aggregation relative to the reference objective and aggregation rule. The effectiveness of the individual components is examined through ablation experiments.

E Additional Experiments

This appendix provides additional empirical results that complement the main experiments in Section 4. These studies focus on six aspects: (i) parameter-budget-controlled motivation, (ii) routing and gradient-signal sanity checks, (iii) hyperparameter sensitivity, (iv) PEFT-pool extensibility and comparison with heterogeneous-rank LoRA methods, (v) scalability under larger data, more clients, and partial participation, and (vi) communication and training-time overhead

E.1 Budget-matched Motivation Study

The motivation study in Fig. 1 is intended to show that no single PEFT type is uniformly suitable and that dense stacking is not automatically stronger. To rule out a simple trainable-parameter-count explanation, we further construct a budget-matched setting where all single-PEFT baselines are aligned to the same trainable budget as Naive-Stack. Specifically, FedLoRA uses rank $r = 32$, FedAdapter uses bottleneck dimension $r_a = 64$, FedPrefix uses prefix length $l_p = 64$, and Naive-Stack uses $(r, r_a, l_p) = (8, 32, 16)$, all corresponding to approximately 1.180M trainable parameters.

Table 5: Budget-matched motivation study on GLUE-S2 with RoBERTa-large. All single-PEFT baselines are matched to the trainable-parameter budget of Naive-Stack.

Method	**SST-2**	**QNLI**	**RTE**	**QQP**
FedLoRA	93.1	90.4	66.9	86.9
FedAdapter	92.6	90.1	69.5	87.3
FedPrefix	91.7	88.9	68.0	87.5
Naive-Stack	92.4	89.8	68.2	86.8

Table 5 shows that the conclusion of the motivation study remains unchanged under matched parameter budgets. Different PEFT types still exhibit complementary task-dependent behavior, and Naive-Stack is not consistently stronger despite using the same trainable budget. This supports the need for controlled heterogeneous PEFT activation rather than dense multi-PEFT stacking.

E.2 Feature-aware Routing Sanity Check

Hermes uses the shared pre-block hidden state to produce feature-aware routing scores. Since LoRA, Adapter, and Prefix-tuning modify different internal computations, we directly test whether this shared block-level input carries useful cross-PEFT routing information.

For each sampled tuple consisting of a client, a layer, and a validation batch, we use the feature-aware score alone to select a Top-2 PEFT pair. We then exhaustively evaluate the three possible PEFT pairs $\{L, A\}$, $\{L, P\}$, and $\{A, P\}$ on the same batch and identify the pair with the lowest validation loss. The feature-aware match rate is the fraction of samples where the feature-aware Top-2 pair matches this best-performing pair. Since there are three candidate pairs, the random baseline is 33.3%.

Table 6: Feature-aware routing sanity check. The feature-aware score is compared against exhaustive validation over all three Top-2 PEFT pairs.

Setting	**Sampled tuples**	**Feature-aware match rate (%)**	**Random (%)**
GLUE-S2, RoBERTa-large	150	61.3	33.3
FLAN, LLaMA-3.2-3B	150	58.7	33.3

As shown in Table 6, the feature-aware signal is substantially above random selection. This does not imply that the router is an oracle, but it confirms that the shared pre-block hidden state provides informative block-level signals for routing over heterogeneous PEFT modules.

E.3 Raw vs. Parameter-normalized Gradient-aware Score

The gradient-aware score is intended as a historical training-dynamics signal. However, raw gradient magnitudes may be affected by the number of parameters in each PEFT module. We therefore compare the raw gradient score with a parameter-normalized variant based on per-parameter RMS gradient:

$$\frac{\|\nabla_{\theta_m}\mathcal{L}\|_2}{\sqrt{|\theta_m|}}.$$

We evaluate both the default PEFT configuration and a deliberately mismatched PEFT-size configuration. The "Default" configuration uses $(r, r_a, l_p) = (8, 32, 16)$, under which the trainable parameter counts of LoRA, Adapter, and Prefix-tuning are of comparable order (≈0.30M, 0.45M, and 0.30M, respectively). The "Mismatch" configuration deliberately enlarges this imbalance by setting $(r, r_a, l_p) = (4, 48, 8)$.

Table 7: Comparison between raw and parameter-normalized gradient-aware scores. "Default" uses $(r, r_a, l_p) = (8, 32, 16)$, while "Mismatch" uses $(4, 48, 8)$ to enlarge PEFT-size imbalance.

Method	**GLUE-S2 Avg.**	**FLAN Avg.**
Default: Hermes-w/o-G	86.6	62.6
Default: Hermes-raw-G	87.8	64.2
Default: Hermes-norm-G	87.9	64.3
Mismatch: Hermes-w/o-G	86.2	62.1
Mismatch: Hermes-raw-G	86.9	63.1
Mismatch: Hermes-norm-G	87.6	63.9

Table 7 shows that raw and normalized gradient-aware scores perform similarly under the default setting, where PEFT sizes are of comparable scale. Under deliberately enlarged PEFT-size mismatch, the normalized version becomes clearly stronger. This supports using parameter-normalized historical gradient statistics in Hermes while confirming that the gradient-aware signal itself remains useful: removing it consistently degrades performance.

E.4 Hyperparameter Sensitivity

We further evaluate the sensitivity of Hermes to three routing and aggregation hyperparameters: the feature–gradient trade-off coefficient γ, the anti-collapse bias strength γ_b, and the aggregation

exponent β. For brevity, γ sensitivity is reported on the harder GLUE-S1 (non-IID) and FLAN settings, and β sensitivity is reported on GLUE-S2 (RoBERTa-large) and FLAN.

Table 8: Sensitivity to the feature–gradient trade-off coefficient γ (higher γ means more weight on the feature-aware signal).

Setting	$\gamma = 0.1$	$\gamma = 0.3$	$\gamma = 0.5$	$\gamma = 0.7$	$\gamma = 0.9$
GLUE-S1, RoBERTa-base Avg.	86.8	87.2	87.5	87.8	87.6
FLAN, LLaMA-3.2-3B Avg.	63.2	63.7	64.1	64.2	64.0

Table 9: Sensitivity to the anti-collapse bias strength γ_b on two representative tasks.

Task	$\gamma_b = 0.1$	$\gamma_b = 0.3$	$\gamma_b = 0.5$	$\gamma_b = 0.7$	$\gamma_b = 0.9$
RTE	72.5	72.9	73.3	73.5	73.0
WebNLG	74.0	75.0	74.5	74.8	74.2

Table 10: Sensitivity to the aggregation exponent β (larger β moderates more strongly the influence of frequently activated modules).

Setting	$\beta = 0.1$	$\beta = 0.3$	$\beta = 0.5$	$\beta = 0.7$	$\beta = 0.9$
GLUE-S2, RoBERTa-large Avg.	87.3	87.5	87.6	87.4	87.2
FLAN, LLaMA-3.2-3B Avg.	57.0	57.3	57.4	57.2	56.9

Tables 8–10 show that Hermes is stable over a reasonable range of hyperparameters. The best or near-best performance is obtained around the default setting, but performance does not collapse when varying the feature–gradient trade-off, the anti-collapse correction strength, or the aggregation exponent.

E.5 Extensibility to Additional PEFT Variants and Larger PEFT Pools

The main experiments focus on LoRA, Adapter, and Prefix-tuning because they naturally fit the block-level residual interface used by Hermes. We do not claim exhaustive coverage of all PEFT methods. As an additional extensibility check, we evaluate Hermes with IA^3 at the implementation level (Table 11). Since IA^3 mainly performs multiplicative modulation, we treat this result as an empirical extension rather than as a core block-level residual instantiation.

Table 11: Extensibility check with IA^3.

Method	**GLUE-S2, RoBERTa-large**				**FLAN, LLaMA-3.2-3B**			
	SST-2	**QNLI**	**RTE**	**QQP**	**WebNLG**	**XSum**	**HellaSwag**	**ARC**
FedIA^3	91.8	88.4	60.5	84.8	70.8	25.9	61.4	78.8
Hermes+IA^3	94.2	92.0	72.0	89.5	74.0	28.1	66.5	83.8

We also enlarge the PEFT pool from three to six experts by introducing two variants for each PEFT type: LoRA-1/LoRA-2, Adapter-1/Adapter-2, and Prefix-1/Prefix-2. This setting tests whether Hermes remains effective beyond the original three-expert pool.

Table 12: Larger PEFT-pool experiment with six experts.

Method	**GLUE-S2, RoBERTa-large**				**FLAN, LLaMA-3.2-3B**			
	SST-2	**QNLI**	**RTE**	**QQP**	**WebNLG**	**XSum**	**HellaSwag**	**ARC**
Hermes-(3 experts)	94.7	92.6	73.5	90.3	75.0	28.8	68.0	85.0
HomoLoRA-(6 experts)	94.3	92.1	72.9	89.8	74.3	28.3	67.1	84.1
Hermes-(6 experts)	94.8	92.8	73.5	90.5	75.2	29.0	68.1	85.1

Table 12 shows that Hermes remains effective when the expert pool is enlarged. It also outperforms the homogeneous LoRA-expert control, suggesting that the gain is not merely from increasing the number of routed experts but from using heterogeneous PEFT modules.

E.6 Comparison with Heterogeneous-rank LoRA Methods

FLoRA and FlexLoRA address a different problem from Hermes: they focus on heterogeneous LoRA ranks or client resource constraints, whereas Hermes studies heterogeneous PEFT types. For completeness, we report a same-rank comparison, where these methods are forced into a setting closer to ours (Table 13). This comparison should be interpreted as an orthogonal-setting check rather than as the primary baseline comparison.

Table 13: Same-rank comparison with heterogeneous-rank LoRA methods.

Method	GLUE-S2, RoBERTa-large				FLAN, LLaMA-3.2-3B			
	SST-2	QNLI	RTE	QQP	WebNLG	XSum	HellaSwag	ARC
FLoRA (same-rank)	92.3	90.1	65.2	85.9	70.2	25.0	61.0	77.6
FlexLoRA (same-rank)	92.2	90.2	65.5	85.8	70.1	25.1	61.3	77.5
FedLoRA	93.1	90.8	66.5	86.8	71.2	25.8	62.0	78.5
Hermes	94.7	92.6	73.5	90.3	75.0	28.8	68.0	85.0

We further test Hermes under a reduced PEFT capacity setting.

Table 14: Hermes under reduced PEFT capacity.

Setting	Adapter bottleneck	LoRA rank	Prefix length	GLUE-S2 Avg.	FLAN Avg.
Hermes (default)	32	8	16	92.6	57.4
Hermes (reduced budget)	16	4	8	91.9	56.6

Table 14 shows that reducing PEFT capacity moderately decreases performance, but Hermes remains competitive. This suggests that the effectiveness of Hermes is not solely due to a large PEFT budget.

E.7 Token-volume Heterogeneity and Aggregation Robustness

In the main experimental setting, all clients use the same number of training samples (300) with similar sequence lengths ($\sim$50 tokens), so token volume is effectively controlled across clients. This subsection simulates token-volume heterogeneity by varying the number of training samples per client (Low=100, Medium=300, High=900) to stress-test the aggregation rule. This check complements the client-normalized inverse-frequency weighting used in Hermes.

Table 15: Aggregation robustness under token-volume heterogeneity on GLUE-S2. Clients are assigned low, medium, or high local sample counts.

Method	SST-2 (Low, 100)	RTE (Medium, 300)	QNLI (Medium, 300)	QQP (High, 900)
Hermes-w/o-A	93.3	72.5	91.8	89.5
Hermes-Sample-count (α_i)	94.1	73.0	92.3	90.0
Hermes-Token-count (α_i)	94.4	73.1	92.5	90.2

Table 15 shows that Hermes remains stable when client token volumes are imbalanced. The aggregation component improves over removing aggregation balancing, and token-count-based client weighting remains competitive in this controlled heterogeneity setting.

E.8 Scalability with Larger Local Data and More Clients

We evaluate Hermes under larger local train/test splits and larger client populations. Following the scalability protocol, we exclude RTE and ARC in this study because they are much smaller than the other tasks.

Tables 16–19 show that Hermes remains the best-performing method when increasing local data sizes and the number of clients.

Table 16: Scalability with larger local train/test splits on GLUE-S2 with RoBERTa-large.

Method	300/200				600/400				1000/500			
	SST	QNLI	QQP	Avg.	SST	QNLI	QQP	Avg.	SST	QNLI	QQP	Avg.
FedDPA	93.4	91.6	88.7	91.2	93.5	92.0	89.2	91.6	93.4	92.3	89.5	91.7
FedLEASE	94.1	91.9	89.4	91.8	94.2	92.3	89.8	92.1	94.3	92.5	90.2	92.3
Hermes	94.8	92.7	90.4	92.6	94.9	93.1	90.9	93.0	95.1	93.4	91.3	93.3

Table 17: Scalability with larger local train/test splits on FLAN with LLaMA-3.2-3B.

Method	300/200				600/400				1000/500			
	WebNLG	XSum	HellaSwag	Avg.	WebNLG	XSum	HellaSwag	Avg.	WebNLG	XSum	HellaSwag	Avg.
FedDPA	72.6	27.1	64.7	54.8	73.5	27.5	65.4	55.5	74.1	27.8	65.9	55.9
FedLEASE	73.6	27.9	66.2	55.9	74.2	28.4	66.9	56.5	74.8	28.7	67.6	57.0
Hermes	75.1	28.9	68.2	57.4	76.0	29.5	69.0	58.2	76.6	30.1	69.7	58.8

Table 18: Scalability with more clients on GLUE-S2 with RoBERTa-large.

Method	$N = 12$			$N = 24$			$N = 48$		
	SST	QNLI	QQP	SST	QNLI	QQP	SST	QNLI	QQP
FedDPA	93.4	91.6	88.7	93.2	91.9	88.6	93.5	91.4	88.8
FedLEASE	94.1	91.9	89.4	94.0	92.3	89.1	94.2	91.8	89.3
Hermes	94.8	92.7	90.4	94.7	93.0	90.2	94.9	92.6	90.4

Table 19: Scalability with more clients on FLAN with LLaMA-3.2-3B.

Method	$N = 12$			$N = 24$			$N = 48$		
	WebNLG	XSum	HellaSwag	WebNLG	XSum	HellaSwag	WebNLG	XSum	HellaSwag
FedDPA	72.6	27.1	64.7	72.8	27.0	64.9	72.4	27.4	64.6
FedLEASE	73.6	27.9	66.2	73.5	28.2	66.0	73.8	27.8	66.2
Hermes	75.1	28.9	68.2	75.0	29.2	68.0	75.2	28.8	68.3

E.9 Partial Participation

We further test partial participation on GLUE-S2 and FLAN. For each communication round, only a subset of clients participates in local training and uploads PEFT updates.

Table 20: Average performance under partial participation. Numbers in column headers indicate the participation rate.

Method	25%	50%	75%	100%
GLUE-S2, RoBERTa-large Avg.				
FedDPA	90.4	90.8	91.1	91.2
FedLEASE	90.9	91.3	91.6	91.8
Hermes	91.7	92.1	92.4	92.6
FLAN, LLaMA-3.2-3B Avg.				
FedDPA	53.7	54.2	54.6	54.8
FedLEASE	54.7	55.2	55.6	55.9
Hermes	56.1	56.7	57.1	57.4

Table 20 shows that Hermes remains robust under partial participation. The performance gap over FedDPA and FedLEASE persists even when only 25% of clients participate per round.

E.10 Communication and Training-time Overhead

The main text reports total per-round communication. Here we clarify the uplink/downlink composition for Hermes. Hermes uploads only activated PEFT modules, but the server broadcasts the full global heterogeneous PEFT pool in each round.

Table 21: Communication breakdown for Hermes. Values are communicated trainable parameters per round in millions (M). For FedUniPELT, only the total per-round communication is shown for reference.

Method	Uplink (M)	Downlink (M)	Total (M)
Hermes	0.9	1.4	2.3
FedUniPELT	1.4	1.4	2.8

Table 21 clarifies the system trade-off. Hermes incurs higher downlink than single-PEFT baselines because the server broadcasts the full PEFT pool, but it remains cheaper than full multi-PEFT synchronization such as FedUniPELT.

We also report training-time overhead. The feature-aware gating network is a shared lightweight linear layer with about 0.003M parameters and 0.006 MFLOPs per forward call, which is less than 1% of each PEFT module in our implementation. The gradient EMA and bias terms maintain only per-layer, per-module scalar states.

Table 22: Training-time overhead.

Time (s)	FedLoRA	FedDPA	FedLEASE	Hermes
Local per-epoch training time	3.65	3.98	3.91	4.34
Extra mechanism time	–	–	3.31 (Clustering)	0.81 (Routing/Stat.)
Global aggregation time	0.047	0.059	0.064	0.074
Total training time	183.24	199.86	203.17	221.48

Table 22 shows that Hermes introduces additional client-side computation compared with single-PEFT baselines, mainly from routing and state maintenance. However, Hermes does not execute all PEFT modules at each step: the router is computed first, and only the selected Top-k modules are evaluated and updated.

E.11 GLUE-S1 under Near-IID Splits

Table 23 reports GLUE-S1 results under Dirichlet(10.0), corresponding to a near-IID partition. Compared with the non-IID setting (Dirichlet(1.0)) reported in Table 1, all methods improve as expected, but the relative ordering remains consistent: Hermes outperforms all baselines across the eight tasks. The performance gap is smaller under near-IID, which is consistent with the intuition that homogeneous PEFT choices become more competitive when client distributions are similar.

Table 23: Federated GLUE-S1 (task-homogeneous) on RoBERTa-base under Dirichlet(10.0) near-IID splits. Metrics are acc except MRPC=$F1$, CoLA=mcc, STS-B=$spearman$.

Method	GLUE-S1, RoBERTa-base, Dirichlet(10.0)							
	SST-2	MRPC	CoLA	RTE	QNLI	STS-B	MNLI	QQP
FedAdapter	93.9	89.7	62.6	70.0	91.5	89.7	86.6	89.0
FedLoRA	94.2	85.1	61.6	68.0	91.4	88.6	85.5	88.0
FedPrefix	93.8	87.6	58.8	59.4	88.4	86.8	83.0	86.5
FedUniPELT	93.3	88.5	60.9	67.1	90.6	88.9	85.0	87.2
FedSA	94.4	87.0	61.8	68.5	91.6	88.9	85.9	88.6
FedDPA	94.1	90.1	63.0	70.5	91.8	90.0	86.9	89.3
FedLEASE	94.6	90.8	63.7	71.2	92.0	90.4	87.2	90.0
Hermes	95.0	91.8	64.6	72.0	92.1	90.9	87.6	90.7

F Broader Impact

Hermes aims to improve federated LLM fine-tuning by reducing the need to centralize client data, which may benefit privacy-preserving adaptation in domains where data ownership or regulation limits

data sharing. However, more efficient personalization of LLMs may also amplify risks associated with deployed language models, including biased outputs, domain-specific misuse, or overfitting to narrow client distributions. Hermes itself does not introduce new data collection or surveillance mechanisms, and its intended use is controlled federated model adaptation with appropriate governance over model deployment and data access.

G LIMITATIONS

To the best of our knowledge, we are the first to study heterogeneous FedPEFT with heterogeneous PEFT mixtures. Despite its effectiveness, Hermes has several limitations. First, although the block-level interface can incorporate additional PEFT methods that preserve the block input and output spaces, our current evaluation covers only LoRA, Adapter, Prefix-tuning, and IA^3. A broader study of PEFT methods with cross-layer or non-standard execution dependencies remains future work. Second, Hermes maintains a heterogeneous PEFT pool on each client and broadcasts the full pool from the server, which trades additional client memory and downlink cost for improved heterogeneous adaptation; this overhead, although smaller than full multi-PEFT synchronization, may still be non-trivial in extremely resource-constrained settings. Third, the convergence analysis establishes framework-level stability under standard non-convex assumptions but does not characterize the optimal trade-off among routing sparsity, aggregation balancing, and personalization, which we leave for future work.