SFT-P: Federated Tuning and Pruning for LLMs

Anonymous ACL submission

Abstract

Deploying large language models (LLMs) on personal devices is appealing because the most useful interactions depend on private, user-specific context, yet on-device inference and adaptation are constrained by memory, latency, and energy budgets. Structural pruning can reduce runtime cost by removing coherent computation units while preserving dense-kernel execution, but most existing pruning pipelines are developed in centralized settings with shared corpora and optimize for broadly averaged capability, which is misaligned with personalized objectives and difficult to transfer to privacy-sensitive, non-IID client data. We present **S**tructural **F**ederated **T**uning and **P**runing (**SFT-P**), a federated framework that learns the pruning decision jointly with training under a round-based FedAvg protocol. SFT-P uses a client-conditioned mask generator with globally shared parameters and a private on-client embedding to produce hard routing masks, enabling client-specific structured pruning under specified budgets; it can optionally co-train lightweight low-rank adapters to improve robustness at higher pruning ratios. Experiments on four heterogeneous client tasks show that federating pruning decisions with training is especially beneficial under aggressive compression, where SFT-P improves the best federated baseline by +8.5 Avg points on LLaMA-7B at 50% pruning.

1 Introduction

Large language models (LLMs) are rapidly becoming personal computing infrastructure: they sit inside email clients to draft replies (Li et al., 2025), inside IDEs to assist coding (Li and Izadi, 2025), and inside note apps to reorganize scattered thoughts (Tsai et al., 2025). The most valuable interactions happen close to the user's context, yet the devices that host this context, phones, laptops, and desktops, operate under tight memory, latency, and energy budgets (Chen et al., 2025). This tension

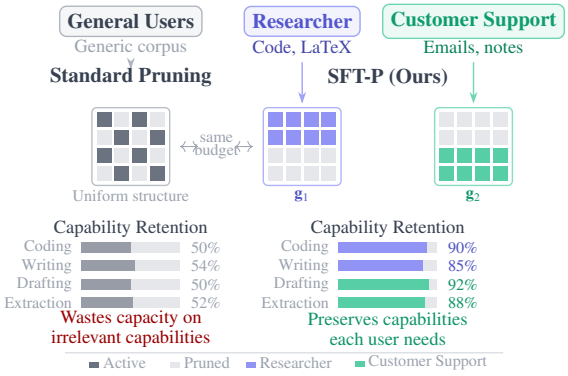

Figure 1: Motivation for personalized structural pruning. Standard pruning uses a generic corpus to produce a uniform structure, yielding mediocre capability retention across all tasks. SFT-P learns client-specific masks ($\mathbf{g}_1$, $\mathbf{g}_2$) under the same pruning budget, preserving capabilities each user needs.

motivates an emerging question: how can we deploy and adapt LLMs on-device without sacrificing the capabilities that each user actually needs?

Structural pruning is a practical step toward this goal (Ma et al., 2023). By removing entire architectural units such as channels, neurons, and attention heads, structural pruning yields real inference speedups on commodity hardware, unlike many unstructured sparsification schemes (Bai et al., 2024; Xu et al., 2024) that depend on specialized kernels. Recent structural pruning methods (Ashkboos et al., 2024; Gao et al., 2024b; Hu et al., 2024) relax architectural dependence along the embedding dimension, allowing different blocks to select different feature subsets and to adopt different widths, which naturally supports depth-dependent budget allocation and improves the accuracy–efficiency tradeoff. As a result, structurally pruned LLMs are attractive for edge deployment (Qu et al., 2025).

However, personal deployment exposes a mismatch between how pruned models are usually produced and what users actually want. Standard pruning pipelines typically rely on centralized recovery data and shared calibration objectives (Zhang et al.,

2024b; Chen et al., 2024), which are often unavailable in privacy-preserving settings and can degrade under non-IID client distributions. Yet user demands are inherently skewed: a researcher may care most about coding assistance and long-form technical writing with consistent citations, while a customer support agent may prioritize fast email drafting with the right tone and reliable extraction of action items from messy notes. As illustrated in Fig. 1, under strict budgets, spending capacity to preserve every capability uniformly is ineffective; the better objective is to preserve the *right capabilities* for each user.

The most direct supervision for such personalization is the user's own data, but this raises two issues. First, per-user data are typically too sparse to reliably steer pruning or recovery. Second, across user devices the data are strongly non-IID (McMahan et al., 2017): one device may hold mostly source code and LaTeX drafts, while another holds email threads and meeting notes. Even users who need several capabilities at once still demand only a small subset of the foundation model's global pretraining distribution, which must allocate capacity across dozens of languages and domains (Conneau et al., 2020); SFT-P reclaims this unused capacity for the user's actual task cluster. Naively aggregating updates can blur distinct requirements and degrade personalization, while centralizing raw data is unacceptable for privacy (McMahan et al., 2017). In our evaluation, we model this heterogeneity by treating distinct capability profiles as separate clients (e.g., different benchmark tasks), which lets us isolate non-IID effects in a controlled setting. In addition, clients often face different device constraints, inducing heterogeneous pruning budgets; jointly learning client-specific structures under a shared global model can therefore lead to budget conflicts during aggregation.

Federated learning (FL) (Kairouz et al., 2021) provides a principled way to leverage private, distributed user data without centralizing raw text, and has recently been explored for adapting LLMs primarily via training or instruction fine-tuning, often through parameter-efficient updates such as low-rank adapters to reduce computation and communication (Zhang et al., 2024a; Qin et al., 2025). However, existing pipelines typically assume a fixed deployed architecture, aggregating only client-side parameter updates while keeping compression and structural choices fixed or centrally determined. In parallel, federated pruning studies investigate how to learn or allocate client-specific architecture under privacy constraints, but have largely focused on non-LLM backbones (Gao et al., 2024a; Huang et al., 2023) or do not address flexible structural pruning for LLMs (Bai et al., 2025). These gaps motivate our central idea: incorporate structural pruning into the federated workflow so that clients collaboratively learn pruned LLM architecture under uniform and heterogeneous pruning budgets.

Motivated by this view, we propose **S**tructural **F**ederated **T**uning and **P**runing (**SFT-P**), a federated framework that integrates (i) round-based optimization and (ii) structured, mask-driven routing for pruning decisions. SFT-P cleanly separates globally shared components from client-private state: the mask generator parameters (and, optionally, lightweight tuning adapters) are synchronized across clients via FedAvg, whereas each client maintains a private conditioning embedding that steers its routed compute pattern from local data without being exposed to the server. This split encourages substantial shared structure via the global generator while allowing small, consistent client-specific deviations through private embeddings. This design supports capability-preserving personalization when clients have different device budgets and non-IID client data, and it also allows optional adapter co-training to recover capacity at higher pruning ratios. Empirically, we find that federating pruning together with training yields stronger personalization–efficiency trade-offs than regular general-purpose pruning methods.

- We integrate structural pruning directly into federated LLM adaptation, removing the dependence on centralized recovery corpora that current pipelines rely on.
- We propose a decoupled mask generation pipeline: a globally shared hypernetwork paired with a private per-client embedding, which jointly enables shared structural learning and client-specific routing without exposing local data.
- We instantiate a residual-safe routed computation interface that supports block-wise feature selection and depth-dependent widths with dense-kernel execution.
- We show that a single shared generator supports mixed pruning budgets across clients (e.g., 20%/50%) without observable conflict,

addressing a practical edge-deployment need that prior work does not.

2 Related Work

Structural pruning in LLMs accelerates inference by removing coherent parameter groups (e.g., heads and channels) so that the compressed model can execute with standard dense kernels on commodity hardware, rather than relying on sparse accelerators. Early LLM-focused work, such as LLM-Pruner (Ma et al., 2023), demonstrates that structured pruning can be paired with lightweight recovery (e.g., LoRA) using a relatively small calibration set (reported as 50K samples) to regain general capability. Subsequent methods increase recovery strength by coupling pruning with post-training or parameter-efficient tuning, often assuming access to a large, centralized corpus: for instance, FLAP (An et al., 2024) relies on broad web corpora for recovery or calibration, such as SlimPajama-derived data (e.g., SlimPajama-6B is a sampled variant with 5,489,000 rows and about 6B tokens). This trend toward large, shared post-training corpora is effective for producing a general-purpose pruned model, but it also explains why such recipes can transfer poorly to federated settings: FL clients typically cannot access a common recovery corpus and instead have small, heterogeneous private data, so the pruning-and-recovery signal becomes client-dependent and non-uniform, making it difficult to maintain a consistent trade-off across capabilities when pruning is aggressive.

Federated Learning (FL) for LLM adaptation has gained momentum alongside parameter-efficient fine-tuning, especially low rank adapters (LoRA), which freeze the backbone and train a small set of injected parameters to reduce both local training cost and communication overhead (Hu et al., 2022). While a foundational line of FL assumes clients collaborate toward a single shared objective, a large body of personalized federated learning (PFL) instead targets clients with divergent objectives and non-IID local distributions (Smith et al., 2017; Kairouz et al., 2021; Tan et al., 2022); our capability-skewed setting falls squarely in this PFL regime rather than the task-aligned one. Recent work applies this parameter-efficient idea to decentralized instruction data: FedIT (Zhang et al., 2024a) aggregates client-side LoRA modules on the server for privacy-preserving instruction tuning, and FedHDS (Qin et al., 2025) improves data efficiency by selecting a representative subset of edge-side data to cut redundant local computation. However, these pipelines primarily optimize a fixed dense architecture, so they improve training and communication efficiency but do not directly solve deployment time constraints, since clients still need to host and run the full backbone for inference. In parallel, federated pruning, long studied on smaller vision backbones, has recently been extended to LLMs: FedSpaLLM (Bai et al., 2025)[1] prunes each client locally on generic C4 calibration data and aggregates the per-client masks into a single global model under one sparsity target; its heterogeneity is therefore system-level (resource-aware layer sampling) rather than data-level. It also targets weight-level (unstructured) sparsity rather than structural pruning. We instead federate **structural** pruning jointly with training under genuinely non-IID, capability-skewed private data, so that each client retains a personalized, capability-preserving structure.

3 Method

3.1 Routing interfaces for residual-safe width decoupling

Transformer blocks are residual: each submodule produces an update that is added back to a running hidden state. Because the add operator requires identical tensor shapes on both branches, many structured pruning designs keep the residual stream width fixed and perform width reduction only inside submodules. A convenient way to describe and implement such designs is to view the model as a fixed residual bus with layer-specific compute lanes, connected by lightweight gather and scatter-add interfaces. We adopt this view as a common solution to avoid prune-together constraints induced by residual shape alignment, while keeping the macro-architecture intact. We treat the bus-and-lane routing as an implementation choice for structured pruning in residual architectures, rather than a new pruning criterion.

System view: a fixed residual bus with variable compute lanes. We consider a fixed residual bus of width d throughout the network, while each submodule executes on a smaller, layer-specific compute lane. Let $X_l \in \mathbb{R}^{B\times T\times d}$ be the residual bus entering layer l (batch size B, sequence length T). For each layer l and submodule $m \in \{\mathrm{attn}, \mathrm{ffn}\}$,

[1]The official repository of FedSpaLLM does not currently contain implementation code.

we define two routing specifications

$$\pi_l^{m,\text{in}} \subseteq \{1,\ldots,d\}, \qquad \pi_l^{m,\text{out}} \subseteq \{1,\ldots,d\},$$

with lane widths $k_l^{m,\text{in}} = |\pi_l^{m,\text{in}}|$ and $k_l^{m,\text{out}} = |\pi_l^{m,\text{out}}|$. We implement two primitives:

$$\text{route}(X_l;\pi) := X_l[:,:,\pi], \tag{1}$$

$$\text{merge}(X_l, U_l;\pi) := X_l + \text{scatter}(U_l;\pi), \tag{2}$$

where $\text{scatter}(U;\pi)$ writes U to indices π along the last axis (zeros elsewhere), and the addition is an indexed add. Here $\text{route}(\cdot)$ returns a $B\times T\times|\pi|$ lane tensor and $\text{merge}(\cdot)$ returns a $B\times T\times d$ residual-bus tensor. In practice, route and merge map to runtime-friendly primitives such as index_select and index_add, and the residual bus always remains d-wide.

Dense-submatrix equivalence. Our route/merge interface can be implemented without unstructured sparsity: routed computation is equivalent to executing smaller dense submatrices (selected rows/columns) plus lightweight gather/scatter on the residual bus. We provide the algebraic equivalence in Appx. A.3.

Generic routed residual update. Under this interface, each submodule m in layer l follows

$$\tilde{X}_l^m = \text{Norm}\Big(\text{route}(X_l;\pi_l^{m,\text{in}})\Big), \tag{3}$$

$$U_l^m = f_l^m\Big(\tilde{X}_l^m;\omega_l^m\Big) \in \mathbb{R}^{B\times T\times k_l^{m,\text{out}}}, \tag{4}$$

$$X_l \leftarrow \text{merge}\Big(X_l, U_l^m;\pi_l^{m,\text{out}}\Big), \tag{5}$$

where f_l^m is instantiated as attention or feed-forward computation using compiled lane-specific weights. Crucially, (5) always returns a tensor in $\mathbb{R}^{B\times T\times d}$, so residual add compatibility is preserved without requiring any cross-layer agreement on which feature channels are used. Particularly, for self-attention, we first form $\tilde{X}_l = \text{Norm}(\text{route}(X_l;\pi_l^{\text{attn,in}}))$, compute $(Q,K,V) = (\tilde{X}_l W_l^Q, \tilde{X}_l W_l^K, \tilde{X}_l W_l^V)$, and merge the routed output:

$$X_l \leftarrow \text{merge}\Big(X_l,\ \text{Attn}(Q,K,V)\,W_l^O;\ \pi_l^{\text{attn,out}}\Big). \tag{6}$$

All projection matrices are compiled into lane-specific dense submatrices (row/column selection) consistent with the route/merge indices; see Appx. A.3. The feed-forward sublayer ffn uses the same route-compute-merge pattern with its MLP projections.

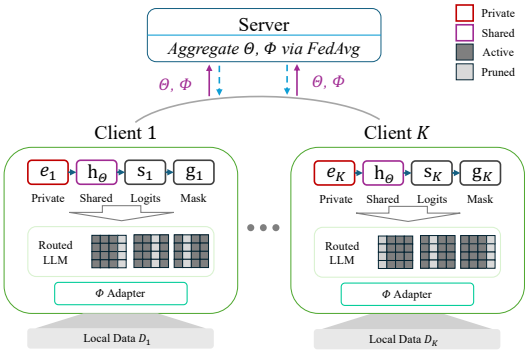

Figure 2: Overview of SFT-P with client sampling ratio $q = 1$. Each client k maintains a private embedding $\mathbf{e}_k$ that conditions a shared mask generator h_Θ to produce client-specific binary masks $\mathbf{g}_k$. These masks route computation through personalized structural patterns in the LLM, while optional adapters Φ provide additional capacity recovery. Only Θ (mask generator) and Φ (adapters) are uploaded to the server for FedAvg aggregation; $\mathbf{e}_k$ and local data $\mathcal{D}_k$ remain private.

3.2 Client-conditioned Mask Generation

The routing interfaces in §3.1 require, for each layer l and submodule $m \in \{\text{attn},\text{ffn}\}$, two index sets $\pi_l^{m,\text{in}}, \pi_l^{m,\text{out}} \subseteq \{1,\ldots,d\}$ that specify which hidden features are read and written. For client k, we represent each index set by a binary mask $\mathbf{g}_{k,l}^{m,\text{in}}, \mathbf{g}_{k,l}^{m,\text{out}} \in \{0,1\}^d$ over the residual bus, where the active index sets $\pi_l^{m,\text{in}}, \pi_l^{m,\text{out}}$ are simply the positions whose mask entries equal 1. The masks are generated by a small conditioning network, so different clients can realize different routing patterns.

Client-conditioned mask generator. Each client k maintains a trainable embedding vector $\mathbf{e}_k \in \mathbb{R}^{d_e}$ that captures persistent local preference. A shared hypernetwork h_Θ maps $\mathbf{e}_k$ (optionally together with a layer identifier) to mask logits:

$$\mathbf{s}_{k,l}^{m,\text{in}} = h_\Theta^{m,\text{in}}(\mathbf{e}_k, l) \in \mathbb{R}^d, \tag{7}$$

$$\mathbf{s}_{k,l}^{m,\text{out}} = h_\Theta^{m,\text{out}}(\mathbf{e}_k, l) \in \mathbb{R}^d. \tag{8}$$

The parameters Θ are globally shared and will be aggregated by the server in FL, while $\mathbf{e}_k$ remains local to client k. This split matches the federated constraint: the generator learns transferable structure across clients, while the embedding provides lightweight client conditioning without exposing raw data.

From continuous logits to hard routing masks. We obtain hard binary masks $\mathbf{g} \in \{0,1\}^d$ from logits $\mathbf{s}$ via a binary straight-through estimator with

the ReinMax surrogate (Liu et al., 2023): the forward pass samples $\mathbf{B} \sim \mathrm{Bernoulli}(\sigma(\mathbf{s}+c))$ (the bias c initializes all gates to 1), while the backward pass uses a ReinMax-adjusted surrogate at temperature τ. Full details and hyperparameters appear in Tab. 5.

We follow a standard round-based federated optimization protocol in which clients collaboratively learn (i) the shared mask generator and (ii) optional lightweight adaptation parameters, while keeping client-specific conditioning variables local. Concretely, the server maintains the global parameters of the mask generator (the hypernetwork) Θ. Each client k maintains a private, trainable conditioning vector $\mathbf{e}_k$ (introduced in the previous subsection), which is never uploaded. Optionally, we also enable parameter-efficient adaptation via low-rank adapters with global parameters Φ, which are aggregated in the same way as Θ.

Client objective. Let $\mathcal{D}_k$ denote client k's local, on-device text corpus, which in practice consists of raw user-generated text already present on the device, i.e., past chat logs with an on-device assistant, local code repositories and Markdown notes, drafted emails, or text extracted from personal documents. SFT-P trains directly on these unmodified token sequences without any centralized labeling. Given current $(\Theta, \mathbf{e}_k)$, the client constructs a discrete structural mask g_k using the soft-to-hard procedure described in §3.2. This mask determines which routed lanes and sliced weight blocks are active during forward computation. The client then minimizes a standard causal language modeling objective with a budget-matching regularizer:

$$\min_{\Theta, \mathbf{e}_k, \Phi} \quad \mathcal{L}_k = \mathcal{L}_k^{\mathrm{CLM}}(\Theta, \mathbf{e}_k, \Phi) + \lambda \mathcal{R}\Big(T(g_k),\, p\,T_{\mathrm{total}}\Big). \tag{9}$$

Here $\mathcal{L}_k^{\mathrm{CLM}}(\Theta, \mathbf{e}_k, \Phi)$ denotes the standard causal language modeling loss on client k's local corpus (next-token prediction), evaluated under the client-specific routed structure induced by $g_k(\Theta, \mathbf{e}_k)$ and the optional adapter parameters Φ. $T(g_k)$ is the number of parameters (or equivalently, the accounted parameter budget) activated by the mask g_k, T_{total} is the total parameter count of the dense backbone, and $p \in (0, 1]$ is the target remaining-parameter ratio (fixed in our experiments, e.g., $p = 0.8$ for 20% pruning). We use a simple scale-invariant penalty that drives the active budget toward pT_{total}:

$$\mathcal{R}(x, y) \;=\; \log\left(\frac{\max(x, y)}{\min(x, y)}\right), \tag{10}$$

which equals 0 when $x = y$ and increases symmetrically when x deviates from y.

Round-based parameter updates. We use a round-based FedAvg protocol (Alg. 1 in Appendix): clients run up to E local steps optimizing Eq. (9) with masks from §3.2, keep $\mathbf{e}_k$ local, and upload only (Θ_k, Φ_k) for server aggregation; repeat for R rounds.

After local steps, the server aggregates by FedAvg: $\Theta^{(r+1)} \leftarrow \sum_{k \in \mathcal{S}_r} (n_k / \sum_j n_j)\, \Theta_k$, and analogously for $\Phi^{(r+1)}$, where n_k is the number of local samples used by client k in this round. This "communicate after a fixed number of local steps" schedule decouples communication frequency from a client's local epoch length, making the communication budget explicit and easy to control.

Optional adapter co-training. When adapters are enabled, Φ is trained jointly with Θ under the same objective (9) and aggregated by FedAvg. Intuitively, when the pruning ratio is mild, the masked backbone may already retain sufficient capacity and adapters can be unnecessary. At higher pruning ratios, adapters provide a small amount of additional trainable capacity that can compensate for the reduced structural flexibility and improve downstream performance, while keeping local computation and communication modest.

Under standard federated assumptions plus a bounded straight-through bias δ_{ST}, the shared parameters (Θ, Φ) reach a stationary point at the usual non-convex rate $\mathcal{O}(1/\sqrt{RE})$, up to an irreducible $\mathcal{O}(\delta_{\mathrm{ST}}^2)$ floor induced by the discrete mask gradient (full statement in Appx. A.13).

4 Experiments

We evaluate SFT-P with two goals. (1) We test whether learning structural pruning within the federated workflow improves the personalization–efficiency trade-off under non-IID client data, compared to applying centralized pruning recipes or naively porting them to FL. (2) We examine whether SFT-P can support heterogeneous client budgets (mixed pruning ratios) while sharing a single global mask generator, without inducing conflicts across clients.

Method	Arc_easy (Acc)	HellaSwag (Acc)	PIQA (Acc)	WinoGrande (Acc)	Avg
20% structural pruning budget (LLaMA-7B)					
FLAP	49.41	64.90	74.81	64.56	63.42
SliceGPT+Finetune	56.90	62.25	74.97	65.82	64.99
LLMPruner+Finetune	56.86	67.40	75.84	64.64	68.19
LLM-Streamline+Finetune	66.71	**70.62**	76.50	<u>69.69</u>	70.88
SliceGPT+Finetune (FL)	66.67	66.24	74.76	67.25	68.73
LLMPruner+Finetune (FL)	54.79	66.31	75.41	60.38	64.22
LLM-Streamline+Finetune (FL)	61.95	65.69	73.94	**69.85**	67.86
SFT-P NOC/NOA	71.17	68.77	77.91	66.30	71.04
SFT-P NOC/+A	71.21	68.08	76.82	63.93	70.01
SFT-P +C/NOA	<u>73.06</u>	69.13	**79.05**	66.46	**71.93**
SFT-P +C/+A	**73.27**	<u>69.50</u>	<u>78.07</u>	66.61	<u>71.86</u>
50% structural pruning budget (LLaMA-7B)					
FLAP	40.32	41.89	62.95	54.06	49.81
SliceGPT+Finetune	28.96	29.61	56.09	50.99	41.41
LLMPruner+Finetune	47.14	**55.26**	69.59	<u>57.85</u>	57.46
LLM-Streamline+Finetune	42.85	49.85	65.72	**62.98**	55.35
SliceGPT+Finetune (FL)	32.32	29.01	57.83	51.14	42.58
LLMPruner+Finetune (FL)	39.27	46.60	63.54	53.91	50.83
LLM-Streamline+Finetune (FL)	30.89	34.05	51.25	55.72	42.98
SFT-P NOC/NOA	54.42	49.93	70.29	53.67	57.08
SFT-P NOC/+A	54.59	49.93	<u>70.57</u>	57.06	58.04
SFT-P +C/NOA	<u>58.12</u>	49.09	**70.78**	56.59	<u>58.65</u>
SFT-P +C/+A	**60.27**	<u>50.67</u>	69.75	56.59	**59.32**

Table 1: **Structured pruning under centralized vs. federated training.** Accuracy (%) on four benchmarks for LLaMA-7B at two pruning budgets (20% and 50%). All baselines without "(FL)" are *centralized* (non-FL) structural pruning baselines reported for reference, while all entries marked "(FL)" and all SFT-P variants are trained under our federated protocol. SFT-P variants ablate client conditioning (C) and adapter co-training (A). Bold/underlined numbers indicate the best/second-best within each pruning-budget block.

4.1 Experimental Setup

Baselines. We compare against representative structural pruning methods, including FLAP (An et al., 2024), SliceGPT+Finetune (Ashkboos et al., 2024), LLM-Pruner+Finetune (Ma et al., 2023), and LLM-Streamline+Finetune (Chen et al., 2024). We implement each baseline by following the authors' released code, and we provide a table to describe the datasets they use in Tab. 4 in Appx. A. For some methods that involve an explicit recovery stage (i.e. post-training), namely SliceGPT+Finetune, LLM-Pruner+Finetune, and LLM-Streamline+Finetune, we also evaluate a federated variant. To obtain a simple and reproducible FL instantiation, we integrate these methods into a FedAvg loop by aggregating once per local epoch, which preserves the baselines' canonical centralized training schedule while introducing periodic synchronization across clients. We emphasize that all "(FL)" baselines are trained on exactly the same per-client local data as SFT-P, namely the training splits of the four target tasks; no shared generic recovery corpus (e.g., Alpaca, SlimPajama) is used. The centralized datasets listed in Tab. 4 apply only to the non-FL reference rows. We report results from a single run for each method due to computational constraints.

Implementation details. We use the budget regularizer in Eq. (9) with λ=16 unless stated otherwise. We evaluate four SFT-P variants by toggling client conditioning (C) and adapter co-training (A): NOC/NOA, NOC/+A, +C/NOA, and +C/+A. Training follows round-based FedAvg with full participation (q=1) over 100 rounds, with at most 100 local update steps per client per round. We report results under 20% and 50% structured pruning budgets and a mixed-budget setting (20/20/50/50). Remaining hyperparameters are in Tab. 5 in Appx. A.

Datasets and federated partitioning. We evaluate on four widely used benchmarks: Arc_easy (Clark et al., 2018), PIQA (Bisk et al., 2020), HellaSwag (Zellers et al., 2019), and WinoGrande (Sakaguchi et al., 2021). We use each task's training split as the client-side fine-tuning corpus, and report multiple-choice accuracy using lm-eval on the standard evaluation split for each task. Although these benchmarks are traditionally framed as supervised, our adaptation objective in Eq. (9) is self-supervised next-token prediction over flattened text sequences, matching the form of training on raw on-device text (e.g., emails or code); the benchmarks serve only as a reproducible, everyday-language stand-in (e.g., WikiHow/ActivityNet for HellaSwag) for private user corpora.

4.2 Performance Analysis

Tab. 1 reinforces our motivation that the compression decision itself should be learned within the federated workflow when client data are non-IID

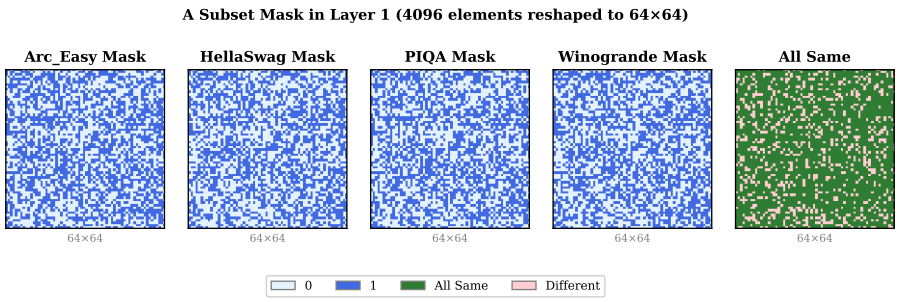

Figure 3: **Cross-client mask agreement in SFT-P.** (SFT-P +C/+A with 50% budget for LLaMA-7B) Visualization of a representative binary routing submask (Layer 1; 4096 dimensions reshaped to 64×64) learned for four clients. The last panel highlights positions where all clients choose the same gate value, showing that the masks are largely aligned.

and capability demands are skewed. Among centralized baselines, LLM-Streamline+Finetune is the strongest overall at 20% pruning (Avg. 70.88), which is consistent with the trend of coupling structural pruning with large, general-purpose post-training corpora: such data-rich recovery effectively preserves broad capabilities in a centralized setting. However, this advantage does not reliably transfer to federated training. When we port prior pipelines to FL (rows marked "(FL)"), performance drops, especially under aggressive pruning (50%), reflecting that their pruning-and-recovery objectives were tuned for abundant shared data and become brittle when recovery supervision is fragmented across small, heterogeneous clients. Notably, SliceGPT+Finetune is the only baseline that benefits from the FL setting at 20% pruning (Avg. 68.73 vs. 64.99 centralized), and it is the best-performing FL baseline in this regime; a plausible explanation is that SliceGPT's pipeline is less dependent on massive centralized recovery data, making it better aligned with the limited-data, per-client supervision available in our federated protocol. In contrast, methods whose effectiveness is largely driven by data-intensive recovery (e.g., LLM-Pruner+Finetune and LLM-Streamline+Finetune) degrade when federated, supporting the interpretation that these designs work best when the recovery signal is sufficiently rich and shared.

Against this backdrop, SFT-P achieves the best overall results across budgets. It consistently outperforms the federated instantiations of prior structural pruning pipelines, and it remains competitive with strong centralized pruning despite operating under stricter privacy and data constraints. The effect is most pronounced at 50% pruning, with an +8.5 Avg-point margin over the strongest federated baseline. Overall, these results align with

Setting	Arc_easy	HellaSwag
+C/NOA (20% all)	73.06	69.13
+C/NOA (mixed)	72.56	69.91
+C/+A (20% all)	73.27	69.50
+C/+A (mixed)	73.78	70.51

Setting	PIQA	WinoGrande
+C/NOA (50% all)	70.78	56.59
+C/NOA (mixed)	69.31	57.06
+C/+A (50% all)	69.75	56.59
+C/+A (mixed)	70.57	56.12

Table 2: **SFT-P under uniform vs. mixed pruning budgets (LLaMA-7B).** Mixed: 20% pruning on Arc_easy/HellaSwag and 50% on PIQA/WinoGrande.

our storyline: rather than relying on a large shared corpus to preserve general-purpose ability, SFT-P federates the pruning decision with training so that clients can learn shared structure where it exists while retaining the flexibility to preserve the right capabilities from private, non-IID data. Additional results are shown in Tab. 3 in Appx. A.

Convergence dynamics and cold-start behavior. Fig. 4 shows that SFT-P converges rapidly despite non-IID clients: per-client validation perplexity drops by an order of magnitude within two rounds (Fig. 4a), and the active-parameter rate of every client reaches the target band within four (Fig. 4b). The fast perplexity drop indicates that the shared generator h_Θ provides a strong transferable initialization while the private $\mathbf{e}_k$ adapts efficiently, so a new client joining the federation incurs only a brief cold-start transient before reaching a comparable operating point.

Mixed-budget Federated Pruning. In practical on-device deployment, clients rarely share a single compression target: device-specific latency and memory constraints naturally induce heterogeneous pruning budgets. To emulate this setting, we evaluate SFT-P under a mixed budget that applies 20% structured pruning on Arc_easy/HellaSwag and 50% on PIQA/WinoGrande (Tab. 2). The

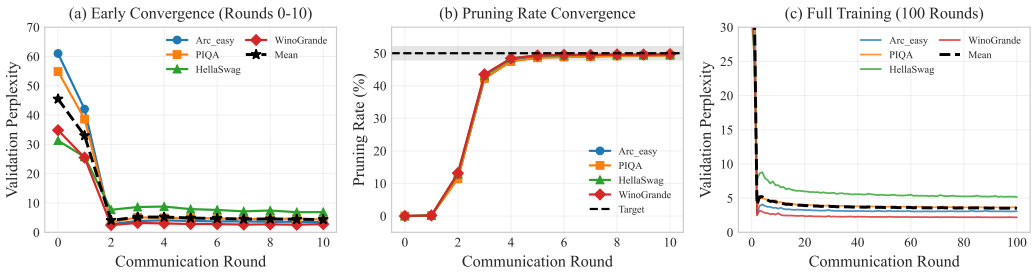

Figure 4: **Training dynamics of SFT-P on LLaMA-7B** across four heterogeneous clients at target pruning rate $p = 0.5$. **(a)** Per-client validation perplexity drops from a mean PPL of 45.5 to below 5.0 within two rounds despite non-IID data. **(b)** The active-parameter rate converges synchronously to the 50% target band ($\pm 2\%$) by round 4, confirming the budget regularizer in Eq. (10) is effective. **(c)** Full 100-round trajectory.

mixed-budget model preserves performance for both budget groups. Mixed-budget training achieves accuracy comparable to the corresponding uniform-budget runs on both lightly pruned (20%) and heavily pruned (50%) clients (Tab. 2), showing no evidence of "budget conflict" between clients with different compression targets. This suggests that a single global mask generator can support heterogeneous budgets, while client conditioning (and optional adapters) absorbs budget-induced heterogeneity without collapsing performance.

4.3 Ablation Study

The ablations in Tab. 1 isolate the contributions of client conditioning (C) and adapter co-training (A), directly testing our core design claims. First, enabling client-conditioned masks (+C) yields the most consistent gains over the unconditioned variants (noC) at both budgets, validating that client-specific routing is critical for non-IID personalization: the shared generator captures transferable structure while private conditioning steers which capacity is preserved per client, aligning with our goal of preserving the right capabilities rather than uniformly retaining average ability. Second, adapter co-training (+A) acts as a robustness mechanism whose benefit is more pronounced under heavier pruning: at 50%, +A improves the best average (60.27 vs. 58.12 on +C/noA), consistent with the intuition that lightweight extra capacity helps compensate when structural flexibility is reduced. Overall, the full model (+C/+A) provides the strongest trade-off by combining personalization via routing with capacity recovery via adapters.

Cross-client mask agreement. Fig. 3 shows substantial overlap among learned masks across clients (about 79% agreement for the illustrated submask and 78% on average across layers), indicating that much of the pruned structure can be learned collaboratively through the global mask generator, reducing per-client optimization burden. Meanwhile, the remaining disagreement is structured rather than noise: each client retains a non-trivial set of task-specific active coordinates, motivating client conditioning as a lightweight mechanism to deviate from the shared structure when local data requires it.

Flexible layer-wise pruning. Fig. 5 in Appx. A.8 illustrates that SFT-P allocates the global pruning budget non-uniformly across depth, with substantial variation in layer-wise mask density for a single client. This is aligned with the general view (e.g., LLM-Streamline) that redundancy is not evenly distributed across layers. Notably, we observe a pronounced density drop around layer $\sim$24 for Arc_easy, suggesting that the task-relevant representation may be concentrated into a narrower subspace at that depth; we leave a mechanistic explanation to future work.

5 Conclusion

We presented SFT-P, a federated framework that jointly learns structural pruning and LLM adaptation under privacy and non-IID constraints, producing client-specific routing masks from a globally shared generator conditioned on private client embeddings, with optional lightweight adapters for capacity recovery. On LLaMA-7B, SFT-P consistently outperforms federated instantiations of prior structural pruning pipelines, supporting our central claim that the compression decision should be learned within the federated workflow rather than applied post hoc. It further supports heterogeneous device budgets: under a mixed-budget setting, clients share a single mask generator without conflicting updates and match uniform-budget accuracy. Finally, the learned masks exhibit substantial cross-client agreement, letting shared structure be amortized while preserving the client-specific deviations that justify personalization.

Limitations

First, our federated evaluation uses four heterogeneous clients (each backed by one public multiple-choice benchmark) to isolate capability-skew and non-IID effects in a controlled way. This is a simplification of production-scale FL, which involves partial participation, intra-task non-IID distributions, and broader system heterogeneity; we view scaling to many-client regimes on 7B–8B LLMs as a natural follow-up that will benefit from continued development of large-scale FL evaluation datasets. We do, however, model device-side system heterogeneity through the mixed-budget setting (Tab. 2), where different clients adopt different pruning targets without observable budget conflict. Second, our main results focus on downstream accuracy under fixed pruning budgets; translating these gains into end-to-end user experience (e.g., latency/energy on diverse hardware and deployment stacks) depends on system-level factors that are outside the scope of this work and often require vendor-specific tooling and careful engineering. Third, we study a specific family of client-conditioned, mask-driven structural pruning under a residual-safe routing interface; while the framework is broadly compatible with other federated optimizers and additional adaptation modules, exploring these combinations (and their stability/efficiency trade-offs) would require substantially expanding the experimental matrix. Finally, as with most federated LLM studies, our experiments do not model all real-world deployment constraints simultaneously (e.g., intermittent connectivity, device availability, and heterogeneous compute), which are important for production systems but are difficult to reproduce faithfully and at scale in academic settings.

References

Yongqi An, Xu Zhao, Tao Yu, Ming Tang, and Jinqiao Wang. 2024. Fluctuation-based adaptive structured pruning for large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 10865–10873.

Saleh Ashkboos, Maximilian L Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. 2024. Slicegpt: Compress large language models by deleting rows and columns. *arXiv preprint arXiv:2401.15024*.

Guangji Bai, Yijiang Li, Zilinghan Li, Liang Zhao, and Kibaek Kim. 2025. Fedspallm: Federated pruning of large language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8361–8373.

Guangji Bai, Yijiang Li, Chen Ling, Kibaek Kim, and Liang Zhao. 2024. Sparsellm: Towards global pruning of pre-trained language models. *Advances in Neural Information Processing Systems*, 37:46203–46225.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*.

Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, page 1175–1191, New York, NY, USA. Association for Computing Machinery.

Daihang Chen, Yonghui Liu, Mingyi Zhou, Yanjie Zhao, Haoyu Wang, Shuai Wang, Xiao Chen, Tegawendé F. Bissyandé, Jacques Klein, and Li Li. 2025. Llm for mobile: An initial roadmap. *ACM Trans. Softw. Eng. Methodol.*, 34(5).

Xiaodong Chen, Yuxuan Hu, Jing Zhang, Yanling Wang, Cuiping Li, and Hong Chen. 2024. Streamlining redundant layers to compress large language models. *arXiv preprint arXiv:2403.19135*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Shangqian Gao, Junyi Li, Zeyu Zhang, Yanfu Zhang, Weidong Cai, and Heng Huang. 2024a. Device-wise federated network pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12342–12352.

Shangqian Gao, Chi-Heng Lin, Ting Hua, Zheng Tang, Yilin Shen, Hongxia Jin, and Yen-Chang Hsu. 2024b. Disp-llm: Dimension-independent structural pruning for large language models. *Advances in Neural Information Processing Systems*, 37:72219–72244.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang,

Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

Yuxuan Hu, Jing Zhang, Zhe Zhao, Chen Zhao, Xiaodong Chen, Cuiping Li, and Hong Chen. 2024. Sp3: Enhancing structured pruning via pca projection. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3150–3170.

Hong Huang, Lan Zhang, Chaoyue Sun, Ruogu Fang, Xiaoyong Yuan, and Dapeng Wu. 2023. Distributed pruning towards tiny neural networks in federated learning. In *2023 IEEE 43rd International Conference on Distributed Computing Systems (ICDCS)*, pages 190–201. IEEE.

Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, and 1 others. 2021. Advances and open problems in federated learning. *Foundations and trends® in machine learning*, 14(1–2):1–210.

Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. 2020. SCAFFOLD: Stochastic controlled averaging for federated learning. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5132–5143. PMLR.

Weijiang Li, Yinmeng Lai, Sandeep Soni, and Koustuv Saha. 2025. Emails by llms: A comparison of language in ai-generated and human-written emails. In *Proceedings of the 17th ACM Web Science Conference 2025*, pages 391–403.

Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. 2020. On the convergence of fedavg on non-iid data. In *International Conference on Learning Representations*.

Ziyou Li and Maliheh Izadi. 2025. Enhancing human-ide interaction in the sdlc using llm-based mediator agents. In *Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering*, pages 1363–1367.

Liyuan Liu, Chengyu Dong, Xiaodong Liu, Bin Yu, and Jianfeng Gao. 2023. Bridging discrete and backpropagation: Straight-through and beyond. *Advances in Neural Information Processing Systems*, 36:12291–12311.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. Llm-pruner: On the structural pruning of large language models. *Advances in neural information processing systems*, 36:21702–21720.

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

H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. 2018. Learning differentially private recurrent language models. In *International Conference on Learning Representations*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *Preprint*, arXiv:1609.07843.

Zhen Qin, Zhaomin Wu, Bingsheng He, and Shuiguang Deng. 2025. Federated data-efficient instruction tuning for large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 15550–15568.

Guanqiao Qu, Qiyuan Chen, Wei Wei, Zheng Lin, Xianhao Chen, and Kaibin Huang. 2025. Mobile edge intelligence for large language models: A contemporary survey. *IEEE Communications Surveys & Tutorials*.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. 2017. Federated multi-task learning. *Advances in neural information processing systems*, 30.

Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R Steeves, Joel Hestness, and Nolan Dey. 2023. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama.

Alysa Ziying Tan, Han Yu, Lizhen Cui, and Qiang Yang. 2022. Towards personalized federated learning. *IEEE transactions on neural networks and learning systems*, 34(12):9587–9603.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca.

Hsin-Ruey Tsai, Shih-Kang Chiu, and Bryan Wang. 2025. Gazenoter: Co-piloted ar note-taking via gaze selection of llm suggestions to match users' intentions. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, pages 1–22.

Peng Xu, Wenqi Shao, Mengzhao Chen, Shitao Tang, Kaipeng Zhang, Peng Gao, Fengwei An, Yu Qiao, and Ping Luo. 2024. Besa: Pruning large language models with blockwise parameter-efficient sparsity allocation. *arXiv preprint arXiv:2402.16880*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Algorithm 1: Federated Training with Client-conditioned Masks

Input: Global rounds R; local steps per round E; client sampling ratio q.
Init : Server initializes shared parameters $\Theta^{(0)}$ (and optional adapters $\Phi^{(0)}$). Each client k initializes private embedding $\mathbf{e}_k^{(0)}$.
for $r = 0, 1, \ldots, R-1$ **do**
  Server samples participating clients $\mathcal{S}_r$ with $|\mathcal{S}_r| \approx qK$;
  Server broadcasts current global parameters $(\Theta^{(r)}, \Phi^{(r)})$ to all $k \in \mathcal{S}_r$ (omit Φ if disabled);
  foreach $k \in \mathcal{S}_r$ **do** `// in parallel`
    Client sets local copies $\Theta_k \leftarrow \Theta^{(r)}$, $\Phi_k \leftarrow \Phi^{(r)}$;
    for $t = 1, 2, \ldots, E$ **do**
      Sample a minibatch from $\mathcal{D}_k$;
      Generate hard masks g_k using §3.2;
      Compute $\mathcal{L}_k$ by Eq. (9);
      Take one optimizer step on $(\Theta_k, \Phi_k, \mathbf{e}_k)$;
    Client uploads (Θ_k, Φ_k) (never uploads $\mathbf{e}_k$);
  Server aggregates $(\Theta^{(r+1)}, \Phi^{(r+1)})$ by FedAvg;

Jianyi Zhang, Saeed Vahidian, Martin Kuo, Chunyuan Li, Ruiyi Zhang, Tong Yu, Guoyin Wang, and Yiran Chen. 2024a. Towards building the federatedgpt: Federated instruction tuning. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6915–6919. IEEE.

Mingyang Zhang, Hao Chen, Chunhua Shen, Zhen Yang, Linlin Ou, Xinyi Yu, and Bohan Zhuang. 2024b. Loraprune: Structured pruning meets low-rank parameter-efficient fine-tuning. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3013–3026.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *The IEEE International Conference on Computer Vision (ICCV)*.

A Appendix

A.1 Federated Training Algorithm

We summarize the round-based federated training procedure of SFT-P in Alg. 1. Each round consists of a server broadcast, E local update steps per participating client (jointly optimizing the shared Θ, optional adapters Φ, and private embedding $\mathbf{e}_k$), and a FedAvg aggregation of (Θ_k, Φ_k); $\mathbf{e}_k$ never leaves the client.

A.2 Additional Results Across Models

To verify that the gains of SFT-P generalize beyond LLaMA-7B, Tab. 3 reports accuracy under the same federated protocol on LLaMA2-7B, Qwen2-7B-Instruct, and LLaMA3.1-8B. Across all backbones, federated instantiations of prior pipelines typically underperform their centralized counterparts, while SFT-P remains competitive and often improves within the federated regime; client conditioning (C) provides the primary gains, and adapter co-training (A) adds robustness at higher pruning ratios.

A.3 Associativity view: equivalent dense submatrix execution.

This bus-and-lane view also admits an equivalent linear-algebra perspective that avoids unstructured sparsity. Let $P(\pi) \in \mathbb{R}^{d\times k}$ be the gather matrix (columns of the identity) so that $\mathrm{route}(X;\pi) = XP(\pi)$. For any dense linear map $W \in \mathbb{R}^{d\times h}$,

$$(XP(\pi))\,(P(\pi)^\top W) \;=\; X\,(P(\pi)P(\pi)^\top)\,W. \tag{11}$$

Rather than materializing the bus mask $P(\pi)P(\pi)^\top$, one can materialize the smaller dense matrix

$$W_\pi := P(\pi)^\top W \in \mathbb{R}^{k\times h},$$

and evaluate on the lane using standard dense GEMMs.

Similarly, output write-back can be expressed directly in terms of the merged update. Suppose a submodule produces an intermediate activation $Z \in \mathbb{R}^{B\times T\times h}$ and an output projection $W^{\mathrm{out}} \in \mathbb{R}^{h\times d}$, so the dense update is $U := ZW^{\mathrm{out}} \in \mathbb{R}^{B\times T\times d}$. If we only intend to write to indices π on the bus, one can compile the lane-specific output matrix

$$W_\pi^{\mathrm{out}} := W^{\mathrm{out}} P(\pi) \in \mathbb{R}^{h\times k}, \tag{12}$$

compute the lane update

$$U_\pi := ZW_\pi^{\mathrm{out}} \in \mathbb{R}^{B\times T\times k}, \tag{13}$$

and then apply $\mathrm{merge}(X, U_\pi; \pi)$. This shows that routed computation can be implemented using smaller dense submatrices (plus lightweight gather/scatter), without introducing unstructured sparsity.

Method	Arc_easy (Acc)	HellaSwag (Acc)	PIQA (Acc)	WinoGrande (Acc)	Avg
20% structured pruning budget (LLaMA2-7B)					
FLAP	40.24	41.97	63.11	53.91	49.81
SliceGPT+Finetune	64.27	62.32	73.07	63.61	65.82
LLM-Streamline+Finetune	68.52	**71.22**	<u>76.38</u>	**67.64**	**70.94**
LLM-Streamline+Finetune (FL)	57.79	65.33	70.78	<u>66.06</u>	64.99
SFT-P +C/NOA	<u>71.76</u>	64.44	**76.93**	63.54	<u>69.17</u>
SFT-P +C/+A	**71.89**	<u>65.34</u>	76.22	62.59	69.01
50% structural pruning budget (LLaMA2-7B)					
FLAP	34.85	39.67	61.64	51.38	46.89
SliceGPT+Finetune	41.71	38.15	61.04	**54.78**	48.92
LLM-Streamline+Finetune	40.66	39.69	65.29	49.25	48.72
LLM-Streamline+Finetune (FL)	26.43	26.42	51.58	50.83	38.82
SFT-P +C/NOA	**52.40**	40.82	**68.61**	50.83	**53.17**
SFT-P +C/+A	<u>52.02</u>	**41.07**	<u>67.08</u>	<u>51.78</u>	<u>52.99</u>
20% structural pruning budget (Qwen2-7B-Instruct)					
SFT-P +C/NOA	74.24	73.93	79.00	68.27	73.86
SFT-P +C/+A	**77.74**	**74.15**	**79.38**	**69.85**	**75.28**
50% structural pruning budget (Qwen2-7B-Instruct)					
SFT-P +C/NOA	**65.61**	55.38	**74.05**	**59.91**	**63.74**
SFT-P +C/+A	64.81	**55.88**	73.50	59.59	63.45
20% structural pruning budget (LLaMA3.1-8B)					
SFT-P +C/NOA	**78.16**	**71.02**	**79.16**	**70.71**	**74.76**
SFT-P +C/+A	74.03	68.00	76.61	65.59	71.06
50% structural pruning budget (LLaMA3.1-8B)					
SFT-P +C/NOA	**60.14**	45.54	68.66	**55.17**	57.38
SFT-P +C/+A	58.96	**45.93**	**70.78**	54.38	**57.51**

Table 3: **Additional results across models and pruning budgets.** We report accuracy (%) on Arc_easy, HellaSwag, PIQA, and WinoGrande under two structured pruning budgets (20% and 50%) for LLaMA2-7B, and additionally include results for Qwen2-7B-Instruct and LLaMA3.1-8B. Methods without "(FL)" are *centralized* (non-federated) structural pruning baselines listed for reference, whereas entries marked "(FL)" and all SFT-P variants are trained with the same round-based FedAvg protocol described in the main paper. Across settings, federated instantiations of prior baselines typically underperform their centralized counterparts, highlighting the difficulty of pruning/recovery under non-IID private data and limited per-client supervision; in contrast, SFT-P remains competitive and often improves within the federated regime, with client conditioning (C) providing the primary gains and adapter co-training (A) offering additional robustness at higher pruning ratios. Bold/underlined numbers indicate the best/second-best within each pruning-budget block.

A.4 Baseline Training Data Sources

Tab. 4 summarizes the centralized training data used by each non-FL pruning baseline in its original implementation, separated into Stage 1 (pruning/calibration) and Stage 2 (recovery/finetuning) where applicable. We reiterate that the "(FL)" variants of these baselines in Tab. 1 use only the per-client local training splits of the four target tasks, identical to SFT-P.

A.5 Training Hyperparameters and Settings

Tab. 5 lists the full configuration used in all SFT-P experiments unless otherwise noted, covering the federated protocol, backbone, mask generator architecture, optimization, and evaluation settings.

A.6 Comparison with Data-Free Heuristic Pruning

To isolate the contribution of our *learned* mask generator, we compare against a fundamental data-free heuristic: L_2-magnitude pruning followed by standard FedAvg fine-tuning in Tab. 6. We compute the L_2 norm of each structural compute lane, permanently prune the target fraction with the lowest magnitude, and then run FedAvg over local client data to fine-tune the surviving weights (LoRA adapters identical to SFT-P).

A.7 Ablation of the Budget Regularizer

We ablate the choice of budget penalty in Eq. (10) against two natural alternatives at a 50% pruning target on the Arc_easy client: (i) Absolute MSE: $\mathcal{R}_{\text{abs}}(x, y) = (x - y)^2$, applied directly to raw parameter counts; (ii) Ratio MSE: $\mathcal{R}_{\text{ratio}}(x, y) = (x/y - 1)^2$, applied to the parameter ratio; (iii) Log-Ratio (ours): $\mathcal{R}_{\log}(x, y) = \log(\max(x, y)/\min(x, y))$.

Two practical points emerge. First, scale invariance is necessary on billion-parameter backbones: any penalty defined in absolute parameter count

Method	Stage 1 dataset	Tokens	Stage 2 dataset	Tokens
FLAP	WikiText-2 (Merity et al., 2016)	2.55M	–	–
SliceGPT+Finetune	WikiText-2	2.55M	Alpaca (Taori et al., 2023)	9.5M
LLMPruner+Finetune	BookCorpus (Zhu et al., 2015)	985M	Alpaca	9.5M
LLM-Streamline+Finetune	SlimPajama-6B (Soboleva et al., 2023)	6.0B	SlimPajama-6B	6.0B

Table 4: **Centralized baseline training data (original implementations).** We summarize the general-purpose datasets used by each non-FL pruning baseline, separating multi-stage pipelines into Stage 1 (pruning/calibration) and Stage 2 (recovery/finetuning).

diverges. Second, scale invariance alone is insufficient: Ratio MSE meets that criterion but still stalls because $\partial \mathcal{R}_{\text{ratio}}/\partial x \to 0$ as $x \to y$. The log-ratio formulation is the simplest closed-form penalty we are aware of that is both scale-invariant and maintains a non-vanishing gradient signal until the target is reached.

A.8 Layer-wise Routing Masks and Sparsity Profile

Fig. 5 visualizes the binary routing masks learned by SFT-P (+C/+A, 50% budget) across all pruned layers for one representative client (Arc_easy). The per-layer mask density on the right shows that SFT-P allocates the global budget non-uniformly across depth: most layers cluster around the 0.5 target, but some layers (e.g., around layer 24) drop substantially below it. This is consistent with the observation in prior depth-pruning work (Chen et al., 2024) that redundancy is not evenly distributed across transformer layers, and suggests SFT-P discovers similar non-uniform allocations *end-to-end* without an explicit per-layer budget schedule.

A.9 Communication Cost Analysis

A natural concern for any federated framework is the per-round communication overhead introduced by the additional shared components. In SFT-P, the server-aggregated payload consists of the mask generator parameters Θ and, when enabled, the LoRA adapter parameters Φ; the client embedding $\mathbf{e}_k$ is private and never communicated. Tab. 8 reports the parameter breakdown for LLaMA-7B under our default configuration.

Two observations are worth highlighting. First, Θ is entirely negligible relative to the backbone ($< 0.5\%$), so SFT-P does not introduce a meaningful communication penalty for typical FL deployments. Second, Θ is not negligible relative to the LoRA payload (roughly $7\times$ larger than Φ in our setup): this is the price paid for federating the structural decision rather than only the parameter update, and it can be reduced further by shrinking the hypernetwork hidden size or factorizing the output projection. We leave a more aggressive payload compression of Θ (e.g., low-rank factorization, quantized transmission) to future work.

A.10 Privacy Considerations

SFT-P follows the standard FedAvg protocol in which clients upload *updated parameters* (Θ_k, Φ_k) rather than raw per-step gradients, so the threat model is that of a (potentially honest-but-curious) server observing parameter deltas across rounds. We discuss two aspects that distinguish SFT-P from prior federated pruning pipelines.

Structural obfuscation via private conditioning. In conventional federated pruning, the shared model itself encodes which structural units are pruned, so the server directly observes each client's sparsity pattern after aggregation. SFT-P decouples this: the server-visible Θ is only a generator that maps a client-specific embedding $\mathbf{e}_k$ to a routing mask g_k. Because $\mathbf{e}_k$ is strictly local and never transmitted, recovering a target client's structural decision from the server's view requires effectively inverting h_Θ without access to its conditioning input. This adds a layer of structural obfuscation beyond the usual "no raw data" guarantee of FedAvg: even the routed compute pattern of a particular client is not directly readable from the aggregation server's state.

Compatibility with standard privacy defenses. SFT-P is algorithm-agnostic with respect to the underlying communication protocol. The uploaded payload (Θ_k, Φ_k) is a standard set of trainable parameters and can be protected with off-the-shelf federated privacy mechanisms, such as Secure Aggregation (Bonawitz et al., 2017) (which prevents the server from inspecting individual client updates) or DP-FedAvg (McMahan et al., 2018) (which provides client-level differential privacy via clipped, noised updates), without any change to the structural pruning methodology. A formal analysis of membership inference and reconstruction

Component	Setting
Federated setup	
Clients / tasks	4 clients: Arc_easy, PIQA, HellaSwag, WinoGrande
FL algorithm	FedAvg, full participation (q=1)
Rounds	100 communication rounds
Local compute budget	max 100 update steps per client per round
Client aggregation	average updates with equal client weight (1 task per client)
Backbone + data	
Backbone	LLaMA-7B
Sequence length	512
Batch size	4
Gradient accumulation	1
Pruning budgets	
Uniform budgets	20% and 50% structured pruning
Mixed budget	20/20/50/50 for (Arc_easy, HellaSwag, PIQA, WinoGrande)
Budget regularization	λ=16 in Eq. (9)
Mask generator + Gating	
Mask generator	Bi-GRU + GeLU + MLP (hidden size=32)
Client conditioning	private client embedding $\mathbf{e}_k$ when C enabled
Binary gating	ReinMax-style straight-through estimator
Gate hyperparameters	bias c=3.0, temperature τ=0.4
Optimization	
Optimizer	AdamW
Learning rate (LoRA, if enabled)	2×10^{-5}
Learning rate (mask generation)	1×10^{-3} for $(\Theta, \mathbf{e}_k)$
Weight decay	0.05 (mask generator)
Gradient clipping	$\|g\|_2 \leq 1.0$
LR schedule	cosine annealing for LoRA; none for mask generator
Adapters (when A enabled)	
LoRA modules	attention projections (`q_proj`, `v_proj`)
LoRA rank / scaling	r=8, α=16
LoRA dropout	0.05
Precision / quantization	
Backbone loading	4-bit quantized backbone for efficiency
Trainable precision	bfloat16 for trainable parameters
Evaluation	
Benchmarks	Arc_easy, HellaSwag, PIQA, WinoGrande (0-shot via lm-eval)
Splits	Arc_easy: test; others: validation (lm-eval default)

Table 5: **Training hyperparameters and experimental settings.** We report the configuration used for all experiments unless stated otherwise.

Method	Arc_e	HSwag	PIQA	WinoG	Avg
20% pruning (LLaMA-7B)					
Magnitude+FedAvg	31.52	37.94	58.81	55.17	45.86
SFT-P +C/+A	**73.27**	**69.50**	**78.07**	**66.61**	**71.86**
50% pruning (LLaMA-7B)					
Magnitude+FedAvg	25.97	28.75	51.96	50.67	39.34
SFT-P +C/+A	**60.27**	**50.67**	**69.75**	**56.59**	**59.32**

Table 6: **SFT-P vs. data-free magnitude pruning + FedAvg on LLaMA-7B.** A static L_2 pruning rule, even when paired with the same federated fine-tuning protocol, collapses well below SFT-P at both budgets, confirming that the gains of SFT-P come from learning structural decisions, not just from the federated tuning loop.

Round	Absolute MSE	Ratio MSE	Log-Ratio (Ours)
0	NaN	0.00%	0.00%
3	–	6.61%	42.16%
5	–	35.16%	48.74%
10	–	44.35%	49.44%
40	–	47.06%	49.83%
100	–	47.53% (stalled)	49.82% (converged)

Table 7: **Achieved pruning rate under three budget penalties (50% target).** Absolute MSE produces exploding gradients on billion-parameter scales and fails to train. Ratio MSE is scale-controlled but its gradient vanishes near the target, leaving the model short of the budget after 100 rounds. The log-ratio design is scale-invariant and produces a stable signal across magnitudes, reaching within 0.2 points of the target by round 10.

attacks specifically targeting the mask generator updates is beyond the scope of this work, and we view it as a promising direction for follow-up study.

A.11 Training-time Resource Usage

A reasonable question is whether SFT-P's structural masks can be exploited to reduce training-time FLOPs and memory on the client, in addition

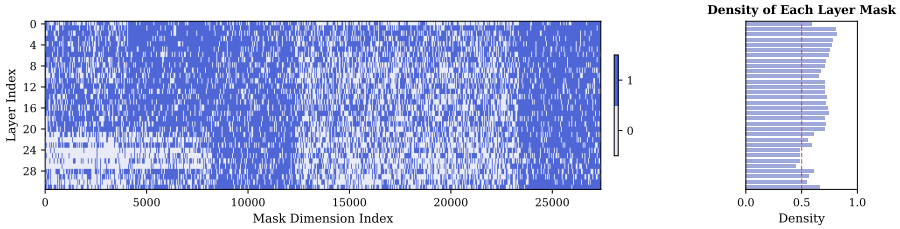

Figure 5: **Layer-wise routing masks and sparsity profile for a single client (Arc_easy).** (SFT-P +C/+A with 50% budget for LLaMA-7B) Left: binary routing masks across all pruned layers (rows) over the hidden dimension (columns), where 1/0 indicates whether a channel is selected by the learned structural gate. Right: per-layer mask density (fraction of active channels), summarizing how the pruning budget is distributed across depth; dashed line marks 0.5 density for reference.

Component	Params	% of backbone
Frozen backbone (not sent)	∼6.738 B	100.00%
Mask generator Θ	∼28 M	0.42%
LoRA adapters Φ	∼4 M	0.06%
Total uploaded	∼32 M	**0.48%**

Table 8: **Per-round communication payload of SFT-P (LLaMA-7B).** Only (Θ, Φ) are uploaded; the backbone is frozen and the client embedding $\mathbf{e}_k$ remains private. Total uploaded parameters amount to less than 0.5% of the frozen backbone size.

to inference-time speedups. We clarify the current scope of our implementation and discuss the trade-offs.

Forward/backward compute is not skipped during training. In our current implementation, the routing masks g_k are applied as multiplicative gates rather than as compile-time index selection during training; the underlying dense GEMMs in the frozen backbone still execute at their full width. This is a deliberate engineering choice: dynamic recompilation of the routed dense submatrices in Appx. A.3 at every optimization step (where g_k changes due to gradient updates and ReinMax sampling) would introduce non-trivial overhead and complicate the implementation. The dense-submatrix equivalence is therefore exploited at deployment time, when the mask is frozen and the routed model can be compiled once.

Memory is minimized by orthogonal mechanisms. Although training FLOPs are not reduced by the mask, SFT-P is explicitly designed so that on-client training remains feasible under tight memory budgets. As detailed in Tab. 5, the backbone is loaded in 4-bit quantization and kept frozen, while only the lightweight trainable components (Θ, $\mathbf{e}_k$, and (optionally) Φ) are stored in bfloat16. Combined with the small absolute size of these trainable components (Tab. 8), this keeps client-side memory dominated by activations rather than optimizer state, which is the regime that matters for edge devices.

Towards mask-aware training acceleration. Skipping computation along inactive lanes *during* training is a natural extension: once g_k stabilizes (as observed empirically in Fig. 4, where the pruning rate converges within a few rounds), the routed model could be partially recompiled to skip masked-out channels in subsequent local steps, trading a small amount of structural flexibility for training-time speedup. We leave a careful study of when and how to switch from soft-gated to hard-routed training to future work.

A.12 Cold-Start Initialization of Client Embeddings

A natural deployment concern is what happens when a new client joins the federation with little or no historical local data. SFT-P handles this scenario gracefully because of the asymmetric size and role of its trainable components. The private client embedding $\mathbf{e}_k \in \mathbb{R}^{d_e}$ is an extremely lightweight vector (less than 0.01% of the trainable payload in Tab. 8) and can be initialized randomly, following standard practice in personalized federated learning. During the cold-start phase, the routing behavior of the new client is dominated by the globally shared mask generator h_Θ, which already encodes transferable structural priors learned across all participating clients. The user therefore experiences a competent generic routed model rather than degenerate behavior on day one. As local data accumulates, $\mathbf{e}_k$ adapts rapidly: the convergence curves in Fig. 4 show that the transition from the generic shared structure to a personalized, budget-compliant one takes only a few local update cycles.

This makes SFT-P amenable to deployments where users frequently onboard with limited prior data.

A.13 Convergence Analysis

We provide a non-convex stationarity statement tailored to Algorithm 1. The argument follows the standard Local-SGD / FedAvg template (Li et al., 2020; Karimireddy et al., 2020), extended to (a) randomness from Bernoulli mask sampling on top of minibatch sampling, and (b) a biased gradient estimator induced by the straight-through / ReinMax surrogate.

Setup. Let $\theta := (\Theta, \Phi)$ collect the server-aggregated parameters; the private embeddings $\{\mathbf{e}_k\}$ are not aggregated and are treated as part of each client's local state. Define the client expected objective

$$F_k(\theta) \;:=\; \mathbb{E}_{(x,y)\sim\mathcal{D}_k,\, g_k\sim p(\cdot\,|\,\theta,\mathbf{e}_k)}\left[\mathcal{L}_k(\theta, \mathbf{e}_k, g_k; x, y)\right],$$

where the expectation is taken jointly over minibatch sampling and Bernoulli mask sampling, and let $F(\theta) := \frac{1}{K}\sum_{k=1}^{K} F_k(\theta)$ be the global objective. Let $\theta_{r,t}$ denote the virtual global average of client iterates at round r, local step t, and η the learning rate.
Assumptions:

(A1) Smoothness. Each F_k is L-smooth in θ.

(A2) Bounded stochastic variance. The stochastic gradients with respect to θ have bounded variance from minibatch sampling (σ^2_{data}) and from Bernoulli mask sampling (σ^2_{mask}).

(A3) Bounded client drift. The non-IID heterogeneity satisfies $\frac{1}{K}\sum_k \|\nabla F_k(\theta) - \nabla F(\theta)\|^2 \le \zeta^2$ for all θ.

(A4) Bounded straight-through bias. The ReinMax surrogate gradient $\tilde{g}$ satisfies $\|\mathbb{E}[\tilde{g}] - \nabla F(\theta)\|^2 \le \delta^2_{\text{ST}}$ for all θ.

Main statement. Under (A1)–(A4), Algorithm 1 with constant learning rate η satisfies

$$\begin{aligned}\frac{1}{RE}\sum_{r=0}^{R-1}\sum_{t=0}^{E-1}&\mathbb{E}\big\|\nabla F(\theta_{r,t})\big\|^2 \\ &\le \mathcal{O}\left(\frac{F_0 - F_{\text{inf}}}{\eta RE}\right) + \mathcal{O}\big(\eta\,(\sigma^2_{\text{data}} + \sigma^2_{\text{mask}})\big) \\ &\quad + \mathcal{O}\big(\eta^2 L^2 E^2 \zeta^2\big) + \mathcal{O}(\delta^2_{\text{ST}}),\end{aligned}$$

where $F_0 := F(\theta_{0,0})$ and $F_{\text{inf}} := \inf_\theta F(\theta)$.

With the standard schedule $\eta = \Theta(1/\sqrt{RE})$, this yields the non-convex stationarity rate $\mathcal{O}(1/\sqrt{RE})$ for the shared parameters, plus an irreducible $\mathcal{O}(\delta^2_{\text{ST}})$ floor from the biased surrogate gradient. We intentionally do not claim global optimality or an $\mathcal{O}(1/RE)$ rate: such guarantees require strong convexity or PL-style conditions that do not hold for LLM objectives.