BUDGET-CONSTRAINED DECOMPOSITION OF LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Adapting large language models (LLMs) is highly resource-demanding: for low-resource research institutions, a full LLM may not even fit into one GPU, and federated fine-tuning still requires every client to run the full model. Existing remedies fall short. First, structured pruning selects one sub-network on a generic calibration corpus and deploys it to every client. Second, federated methods that derive per-client sub-networks bound parameters or memory, not per-token compute. To overcome this, we propose Federated Budgeted Decomposition (FedBUD), which decomposes one pretrained LLM into smaller per-client structural specialists under a hard per-token FLOPs budget. A graph-based reinforcement-learning policy selects each client's blocks, attention heads and feed-forward neurons from its architecture graph and importance statistics. The specialists are then co-trained via federated learning, with LoRA factors aggregated in the coordinates of the source model; an optional shared graph neural network modulates each specialist by its architecture graph, with representation discrepancy bounded by architecture-graph distance for its attention instantiation. FLOPs-matched experiments on six federations show that co-training outperforms independent adaptation across the six datasets, significantly on three, and that sharing structure across clients, selected on validation data, significantly improves both tasks where the selected arm was run at three seeds, whereas the learned allocator does not outperform hand-designed allocation rules. Compute-limited clients can thus co-train LLM specialists without pooling data. Code will be released upon acceptance.

1 INTRODUCTION

Adapting pretrained large language models (LLMs) is a highly resource-demanding task. Federated learning trains a shared model across clients without exchanging their data (McMahan et al., 2017), and low-rank adaptation (LoRA) extends it to LLMs by communicating small adapters instead of full weight matrices (Hu et al., 2022; Zhang et al., 2023; Ye et al., 2024). This combination, however, assumes that every client can run the pretrained model it adapts. For low-resource research institutions, a full LLM may not even fit into the memory of one GPU, and a client with a latency limit per generated token affords only a fraction of the model's per-token compute. Each client's model must therefore be structurally smaller than the source model and selected without pooling data.

Existing remedies fall short. First, structured pruning (Song et al., 2024; Men et al., 2025; Ma et al., 2023; Ashkboos et al., 2024; Muralidharan et al., 2024; Sieberling et al., 2025) selects one sub-network on a calibration corpus that belongs to no client and deploys it to every client, and recent analyses report that the calibration objective determines most of the outcome (Kim et al., 2026; Tang et al., 2026). Second, federated parameter-efficient methods (Zhang et al., 2023; Cho et al., 2024; Bai et al., 2024; Wang et al., 2024) exchange LoRA adapters over a frozen, shared backbone, which leaves the per-token cost of the deployed model unchanged. Third, heterogeneous sub-model federated learning (Diao et al., 2021; Alam et al., 2022; Kim et al., 2023; Su et al., 2024; Li et al., 2021; Wu et al., 2024; Bai et al., 2025; Wu et al., 2025) sizes each sub-model by device capacity or by a parameter count, so the budget that governs deployment is not the budget that is optimized. Fourth, specialists that retain different attention heads break the positional correspondence of their low-rank factors: the same row index names different heads in different clients, so position-wise averaging mixes unrelated computations. Hence, joint training of structurally different LLM specialists under a compute budget is lacking (Fig. 1).

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.

To overcome this, we propose Federated Budgeted Decomposition (FedBUD), a framework that decomposes one pretrained LLM into per-client structural specialists under a hard per-token FLOPs budget. First, prior to any training, a graph-based reinforcement-learning allocator reads the architecture graph and the client's importance statistics and selects the blocks, query heads and feed-forward neurons that client keeps, sharing structure where validation prefers it. Second, we design weight inheritance: the specialist retains the pretrained weights of every component it keeps and recovers with LoRA on its client's data. Third, the specialists are co-trained via federated learning. To co-train specialists that differ in structure, we design aggregation in the coordinate system of the source model, where each original head and neuron is averaged over exactly the clients that retain it; this extends overlap averaging (Diao et al., 2021; Li et al., 2021; Wu et al., 2024) to LoRA factors at head and neuron granularity. Moreover, to condition co-training on each specialist's structure, we design an optional shared graph neural network (GNN) that reads each specialist's architecture graph and modulates its activations; we instantiate it as a graph attention network (GAT) (Veličković et al., 2018) and prove that, for this instantiation, the discrepancy between the representations it produces for two specialists is bounded by a distance between their graphs. Four hypotheses anchor our evaluation:

Hypothesis 1 (co-training). Co-training specialists improves over adapting each one alone.

Hypothesis 2 (alignment). Aligned LoRA averaging improves over position-wise averaging.

Hypothesis 3 (allocation). Learned allocation outperforms random and rule-based structures.

Hypothesis 4 (sharing). Validation-selected shared structure improves over per-client structure.

The main contributions of our work are as follows:

1. *On a methodological level.* We formulate the decomposition of a pretrained LLM into per-client structural specialists under a hard per-token FLOPs budget, solve it with a policy-gradient allocator over an architecture graph, and co-train the specialists by aggregating LoRA factors in the coordinates of the source model.
2. *On a deployment level.* Under matched training and inference FLOPs on six federations, the experiments validate *Hypothesis 1* and *Hypothesis 4*, partially support *Hypothesis 2*.
3. *On a generic level.* We introduce an optional shared GNN, averaged alongside the aligned LoRA factors, that conditions each specialist on its architecture graph, and prove for its GAT instantiation that the representation discrepancy between two specialists is bounded by a distance between their architecture graphs, a property proved for the GAT instantiation in Section 4.

2 Related Work

First, structured pruning removes transformer blocks, coupled parameter groups, or rows and columns of weight matrices, or searches jointly over depth, attention heads and feed-forward width (Song et al., 2024; Men et al., 2025; Ma et al., 2023; Ashkboos et al., 2024; Muralidharan et al., 2024; Sieberling et al., 2025), and each selects one structure for every user. Second, federated learning with heterogeneous sub-models assigns clients nested or rolling width slices, depth prefixes or random layers, derives a sub-network from each client's data, or, for LLMs, builds per-client sub-networks under a sparsity or memory budget (Diao et al., 2021; Alam et al., 2022; Kim et al., 2023; Su et al., 2024; Li et al., 2021; Wu et al., 2024; Bai et al., 2025; Wu et al., 2025); these methods, like DiPaCo (Douillard et al., 2024), average each parameter over the clients that hold it. Averaging the shared parts of sub-models, and serving a sub-model instead of the whole model, are therefore established, and we claim neither. Third, federated adaptation, merging and distillation keep one architecture across participants and exchange adapters, task vectors or outputs (Zhang et al., 2023; Ye et al., 2024; Cho et al., 2024; Bai et al., 2024; Wang et al., 2024; Lee et al., 2026; Yadav et al., 2023; Li et al., 2022; Agarwal et al., 2024). Appendix D discusses each method.

3 Methodology

In this section, we detail FedBUD (Fig. 1B). In the first stage, a policy-gradient allocator selects, for each client and within the budget, which blocks to remove and which to narrow (Sections 3.1

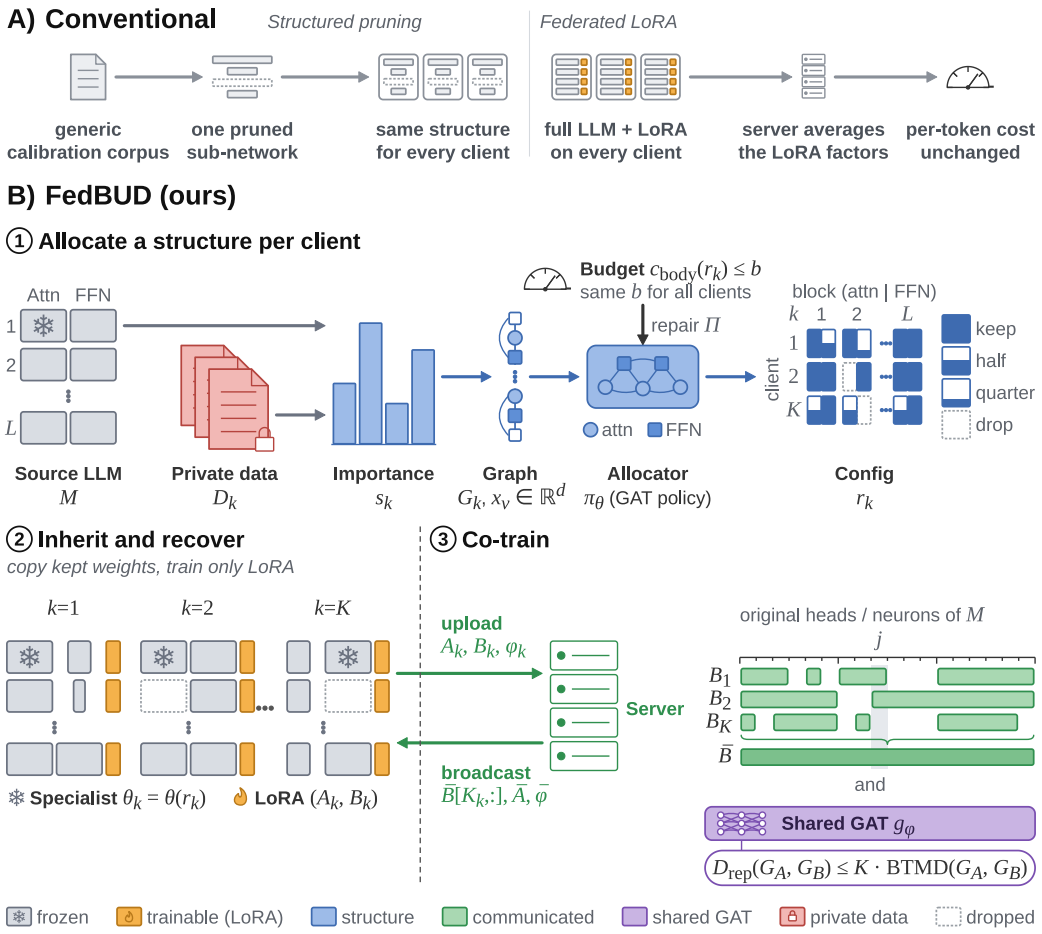


Figure 1: **Conventional pipelines versus FedBUD. A)** Conventional pipelines. Structured pruning selects one sub-network on a generic calibration corpus and deploys it to every client, and federated LoRA keeps the full backbone on every client. **B)** FedBUD. First, we propose a policy-gradient allocator that selects each client's blocks, attention heads and feed-forward neurons under a hard per-token FLOPs budget from that client's importance statistics. Second, we design aggregation of LoRA factors in the coordinates of the source model, where each original head and neuron is averaged over the clients that retain it. Third, an optional shared GNN, averaged alongside the LoRA factors, modulates each specialist according to its architecture graph; reported results omit it. In the figure, $d = 16$ for the node inputs.

and 3.2). In the second stage, the selected sub-network inherits the corresponding pretrained weights and recovers with LoRA on its client's data (Section 3.3). In the third stage, the specialists are co-trained without centralizing data (Section 3.4). Moreover, to handle the structural heterogeneity of the specialists, we propose aggregation in the coordinates of the source model and an optional shared GNN whose stability we prove for its GAT instantiation (Section 4).

Problem statement. Client $k \in \{1, \dots, K\}$ holds a private dataset $\mathcal{D}_k$ with disjoint training, validation and test partitions and n_k training examples, and client distributions differ by label, topic or task. All clients start from the same pretrained decoder-only source model $\mathcal{M}$ with L layers, hidden size d, H query heads in G key-value groups of head dimension d_h, feed-forward width d_{ff} and vocabulary size V. A *pruning configuration* $r = (r_1^{\mathrm{attn}}, \dots, r_L^{\mathrm{attn}}, r_1^{\mathrm{ffn}}, \dots, r_L^{\mathrm{ffn}})$ assigns $r_l^{\mathrm{attn}} \in \{\texttt{keep}, \texttt{half}, \texttt{drop}\}$ to every attention block and $r_l^{\mathrm{ffn}} \in \{\texttt{keep}, \texttt{half}, \texttt{quarter}, \texttt{drop}\}$ to every feed-forward block, and so specifies a sub-network completely. The actions fix the retained fractions $h_l(r) = 1$, $\frac{G}{H}\lceil \frac{H}{2G} \rceil$, 0 of query heads, $g_l(r) = 1, 1, 0$ of key-value groups and $m_l(r) =$

$1, \frac{1}{2}, \frac{1}{4}, 0$ of feed-forward neurons. The per-token cost of a configuration is its forward FLOPs per token at a reference context length C:

$$F(r) \;=\; 2P_{\text{head}} \;+\; \sum_{l=1}^{L} \mathbf{1}[h_l > 0]\Big(2\big(h_l P_l^{qo} + g_l P_l^{kv}\big) + 4\,C\,d\,h_l\Big) \;+\; \sum_{l=1}^{L} 2\,m_l P_l^{\text{mlp}}, \qquad (1)$$

where $P_{\text{head}} = Vd$ is the parameter count of the language-model head, P_l^{qo}, P_l^{kv} and P_l^{mlp} are the parameter counts of the query and output, key and value, and feed-forward projections of layer l, and $4Cdh_l$ counts the attention scores and the weighted sum over values. We set $C = 400$ for every method. With F_{full} the cost of the unpruned model, we report the total cost $c(r) = F(r)/F_{\text{full}}$ and constrain the prunable part:

$$c_{\text{body}}(r) \;=\; \frac{F(r) - 2P_{\text{head}}}{F_{\text{full}} - 2P_{\text{head}}} \;\le\; b, \qquad \mathcal{R}_b = \{\, r : c_{\text{body}}(r) \le b \,\}, \qquad b = 0.6. \qquad (2)$$

The head is excluded from the constraint because no pruning method can reduce it without changing the output space; for Qwen2.5-0.5B it is 26.6% of the per-token cost at $C = 400$, so $b = 0.6$ corresponds to a total cost between 0.68 and 0.71. The goal is, for every client k, a configuration $r_k \in \mathcal{R}_b$ and adapted weights that minimize the client's test loss, obtained without moving any client's data. The federation is simulated, so we make no formal privacy claim. Table 2 collects the symbols.

3.1 Architecture Graph and Importance Statistics

The allocator reads the source model as a graph with $2L + 2$ nodes in computation order, one per block plus the embedding and the head, with edges at distance one and two, so that a node reflects its neighborhood and not its index; only the $2L$ block nodes are prunable. The action set is small and discrete because each action must leave a model that runs without custom kernels. On an attention block, `half` retains $\lceil H/(2G) \rceil$ query heads in every key-value group and all key and value projections, which keeps grouped-query attention consistent; for Qwen2.5-0.5B it retains 8 of 14 heads. On a feed-forward block, `half` and `quarter` retain that fraction of the intermediate neurons. The action `drop` replaces the residual branch of a block by the identity, and a configuration that retains no attention block is inadmissible.

To make two clients receive different sub-networks under the same budget, we design importance scores measured on each client's own data. Let $a_{b,t}^{(l,i)} \in \mathbb{R}^{d_h}$ be the contribution of head i to the input of the output projection of layer l at position t of example b, $M_{b,t} \in \{0, 1\}$ the attention mask with $N = \sum_{b,t} M_{b,t}$, and $W_{l,i}^{o}$ the columns of that projection that read head i. Writing $u_{b,t}^{(l,j)}$ for the activation entering the down projection at neuron j and $W_{l,j}^{\text{down}}$ for its column, the unnormalized scores are

$$\tilde{s}_{l,i}^{\text{attn}} = \Big(\tfrac{1}{N}\textstyle\sum_{b,t} M_{b,t} \|a_{b,t}^{(l,i)}\|_1\Big)\big\|W_{l,i}^{o}\big\|_F, \qquad \tilde{s}_{l,j}^{\text{ffn}} = \Big(\tfrac{1}{N}\textstyle\sum_{b,t} M_{b,t} |u_{b,t}^{(l,j)}|\Big)\big\|W_{l,j}^{\text{down}}\big\|_2, \qquad (3)$$

each divided by its layer total and computed in one forward pass over 32 training examples. Given r, the retained sets $\mathcal{K}_l(r)$ take the highest-scoring components the action allows, per key-value group for attention. Every allocation method we implement shares Equation 3 and this selection rule, so differences between them come from the configuration they select; published pruning methods keep their own criteria. Each node also carries a feature vector $x_v \in \mathbb{R}^{16}$ describing its type, depth, size, measured influence, the budget and a client identifier.

3.2 Allocation Policy and Search

Objective and proxy. The search minimizes, for client k, the *repaired loss*: the validation loss of a sub-network after a short recovery Rec of $T_{\text{rec}} = 100$ LoRA steps on 320 training examples. With $\theta(r)$ the source weights restricted to what r retains and $\mathcal{V}_k$ the 48 validation examples of client k,

$$\ell_k^{\text{prune}}(r) = -\frac{1}{|\mathcal{V}_k|}\sum_{(x,y)\in\mathcal{V}_k} \log p_{\theta(r)}(y \mid x), \qquad \ell_k^{\text{rep}}(r) = -\frac{1}{|\mathcal{V}_k|}\sum_{(x,y)\in\mathcal{V}_k} \log p_{\text{Rec}(\theta(r))}(y \mid x). \qquad (4)$$

Measuring ℓ_k^{rep} takes about one minute per candidate and ℓ_k^{prune} about two seconds, so the search ranks candidates with the pruning loss and measures the repaired loss only for those ranked first.

Proxy correction. The pruning loss is biased in a way that depends on the action, since a short recovery removes more of the loss of a narrowed block than of a deleted one. To handle this bias, we design a learned correction. Let $z_k(r)$ collect $\log \ell_k^{\text{prune}}(r)$, the cost $c(r)$, the number of blocks assigned each action and a client indicator, standardized and extended by an intercept, and let $\mathcal{Q}_k^{\text{ver}}$ be the configurations of client k whose repaired loss has been measured. The corrected predictor is

$$\hat{\ell}_k(r) = \ell_k^{\text{prune}}(r)\, \exp\big(\,\text{clip}(w^\top z_k(r), \rho_{\min}, \rho_{\max})\big), \tag{5}$$

where w is fitted by ridge regression of the log repair ratio on $z_k(r)$ over $\mathcal{Q}_k^{\text{ver}}$ (Equation 5, Appendix G), and $\rho_{\min}, \rho_{\max}$ are the smallest and largest log ratios measured so far, which keeps every prediction inside the measured range. The correction is the identity until $|\mathcal{Q}_k^{\text{ver}}| = 8$ and is refitted after every verification round.

Policy, budget repair and verification. The policy is a two-layer GAT over the architecture graph $\mathcal{G}_k$ with one linear head per node type, and it factorizes over the prunable nodes $\mathcal{N}$:

$$\pi_\theta(r \mid \mathcal{G}_k) = \prod_{v \in \mathcal{N}} \pi_\theta(r_v \mid \mathcal{G}_k), \qquad \pi_\theta(\cdot \mid \mathcal{G}_k)_v = \text{softmax}\left(W_{\kappa(v)} e_v + \log \pi_0\right), \tag{6}$$

where e_v is the node embedding, $W_{\kappa(v)}$ the head for its node type and π_0 a fixed prior, $(0.70, 0.20, 0.10)$ over the attention actions and $(0.35, 0.40, 0.15, 0.10)$ over the feed-forward ones; the head weights start at zero, so an untrained policy samples from π_0. To keep the budget hard, we design a repair operator $\Pi : \mathcal{A}^{2L} \to \mathcal{R}_b$ that maps any configuration into the feasible set by applying the next less expensive action at the node the policy is most willing to give up, and restores an attention block if none survives. In each of 40 iterations the policy draws $J = 8$ configurations per client, repairs and scores them, and minimizes

$$\begin{aligned} R_k(r_i) &= -\hat{\ell}_k\big(\Pi(r_i)\big), \qquad A_i = \frac{R_k(r_i) - \bar{R}}{\sigma_R + \varepsilon}, \\ \mathcal{L}(\theta) &= \frac{1}{J} \sum_{i=1}^{J} \Big(- A_i \, \overline{\log \pi_\theta}\big(\Pi(r_i)\big) - \lambda_H \bar{\mathsf{S}}_i + \lambda_{\text{KL}} \bar{D}_i \Big), \end{aligned} \tag{7}$$

where $\bar{R}$ and σ_R are the group mean and standard deviation of the rewards, $\overline{\log \pi_\theta}$ is the mean log-probability of the executed configuration over prunable nodes, $\bar{\mathsf{S}}_i$ the policy entropy and $\bar{D}_i$ its Kullback-Leibler divergence to π_0, with $\lambda_H = 0.05$ and $\lambda_{\text{KL}} = 0.1$. Equation 7 is a biased policy-gradient estimator, because the log-probability is evaluated at the repaired configuration and prior-drawn samples enter without importance weights; we therefore assess the policy by the subnetworks it proposes and claim no convergence. Every eight iterations and at termination, the greedy configuration and the most promising candidates are verified, and the search returns the verified configuration with the lowest measured repaired loss,

$$r_k^* = \arg\min_{r \in \mathcal{Q}_k^{\text{ver}}} \ell_k^{\text{rep}}(r). \tag{8}$$

3.3 Weight Inheritance and Recovery

To avoid training a small model from a client's limited data, we design weight inheritance: client k receives $\theta_k = \theta(r_k^*)$, the source parameters restricted to what r_k^* retains, and no parameter is reinitialized. The inherited weights are frozen and a LoRA adapter of rank $q = 16$ is attached to each of the seven projections of every retained block, so a projection with frozen weight W computes $W + s\,BA$, and the client minimizes

$$\mathcal{L}_k(A, B) = -\frac{1}{|\mathcal{D}_k|} \sum_{(x,y) \in \mathcal{D}_k} \log p_{\theta_k \oplus (A,B)}(y \mid x) \tag{9}$$

for the number of steps set by the FLOPs-matching rule of Section 5.

3.4 Co-training Across Heterogeneous Structures

All co-training rules train the same specialists for the same steps, seeds and data order over ten rounds, so differences come only from what is exchanged; independent adaptation exchanges nothing.

Aggregation in the coordinates of the source model. Two clients that apply `half` to the same feed-forward block hold factors of the same size, yet row j of each usually names a different neuron, so position-wise averaging, which we keep as the shape-only ablation, combines incomparable quantities. To handle this mismatch, we design aggregation in the coordinates of the source model, where a coordinate names an original head or neuron. Write a projection's update as $\Delta W = s\,BA$ with $B \in \mathbb{R}^{o\times q}$ and $A \in \mathbb{R}^{q\times i}$. For the query, key, value, gate and up projections pruning removes output rows, so the client-specific factor is B_k and the shared factor is A_k; for the output and down projections it removes input columns, so the roles are exchanged. Let $\mathcal{K}_k$ be the original coordinates client k retains, $\mathcal{H}$ the clients that retain the block, $\mathcal{H}(j)$ those that retain coordinate j, and $\iota_k(j)$ the position of coordinate j in client k's sliced matrix. The server computes

$$\bar{B}[j,:] = \frac{\sum_{k\in\mathcal{H}(j)} n_k\, B_k[\iota_k(j),:]}{\sum_{k\in\mathcal{H}(j)} n_k}, \quad \bar{A} = \frac{\sum_{k\in\mathcal{H}} n_k\, A_k}{\sum_{k\in\mathcal{H}} n_k}, \quad B_k \leftarrow \bar{B}[\mathcal{K}_k,:], \;\; A_k \leftarrow \bar{A}, \tag{10}$$

so a head retained by one client keeps that client's rows and no client receives rows for a head it does not have. On identical structures this reduces to federated averaging of LoRA factors (McMahan et al., 2017; Zhang et al., 2023); on heterogeneous structures it applies overlap averaging (Diao et al., 2021) at the granularity of single heads and neurons.

Stacking variant. Equation 10 averages factors, and the product of averaged factors is not the average of the products (Wang et al., 2024), so the aligned update averages factors rather than effective updates. To handle this objection, we also design a variant that averages the updates themselves per coordinate,

$$\overline{\Delta W}[j,:] \;=\; \sum_{k\in\mathcal{H}(j)} n_k\, s_k\, (B_k A_k)[\iota_k(j),:] \;\Big/ \sum_{k\in\mathcal{H}(j)} n_k, \tag{11}$$

after which each client adds its rows of $\overline{\Delta W}$ to its frozen weights and restarts its adapter. We do not evaluate the stacking variant in this paper.

Shared GNN. To condition co-training on each specialist's structure, we design an optional shared encoder following unified learning (Pala & Rekik, 2025b;a), which represents each model as a neuron-level graph; because such graphs do not fit in memory for LLMs, the encoder instead reads the block-level architecture graph of Section 3.1, with one node per attention or feed-forward block. Every specialist holds a copy of a shared GNN with parameters ϕ that reads its architecture graph, and a private head ψ_k that turns the node embeddings into feature-wise linear modulation (Perez et al., 2018) of the activations,

$$z_k = \mathrm{GNN}_\phi(\mathcal{G}_k), \;\; (\gamma_{k,l},\beta_{k,l}) = f_{\psi_k}(z_{k,l}), \;\; h_{k,l} \leftarrow h_{k,l} + \eta\,(\gamma_{k,l}\odot h_{k,l} + \beta_{k,l}), \;\; \phi \leftarrow \frac{\sum_k n_k \phi_k}{\sum_k n_k}, \tag{12}$$

where $h_{k,l}$ is the output of layer l in client k and η is a scalar gate initialized at zero, so co-training starts from aligned LoRA averaging alone. When used, ϕ is averaged alongside the aligned LoRA factors of Equation 10, while ψ_k and η stay local, so the shared encoder's parameters do not depend on any client's structure. Any message-passing GNN can serve as the encoder, since the modulation head only consumes node embeddings; we instantiate it as a two-layer GAT (Veličković et al., 2018), for which Section 4 proves the stability bound. The results of Section 5 use aligned averaging alone, and Appendix H reports aligned co-training with and without the shared GNN. The modulation is assessed by the comparison of Appendix H.

4 Architecture Heterogeneity and GAT Stability

The bound of this section is proved for the GAT instantiation of the shared GNN of Section 3.4. Architectural heterogeneity has been measured by coarse summaries such as layer-type distributions (Pala & Rekik, 2026) and by graph distances such as Graph Edit Distance (Qiu & Miikkulainen, 2023) and the Wasserstein Weisfeiler–Lehman distance (Togninalli et al., 2019). In particular, the Tree Mover's Distance (TMD) (Chuang & Jegelka, 2022) compares computation trees and bounds the output discrepancy of a Graph Isomorphism Network (Xu et al., 2019). However, for a GAT the one-layer discrepancy is controlled by the *largest* disagreement between matched neighbors, not by their sum. Therefore, we introduce the *Bottleneck-TMD* (BTMD), which replaces the summation

over child subtrees with a bottleneck assignment:

$$B_{k+1}(T,S) = \max\left\{ d_{\text{root}}(T,S), \min_{\mu} \max_{R\in\overline{\mathcal{C}}(T)} B_k(R,\mu(R)) \right\}, \tag{13}$$

where T and S are depth-$(k+1)$ rooted computation trees, d_{root} is the Euclidean distance between their root features, $\overline{\mathcal{C}}(T)$ is the multiset of depth-k child subtrees padded with blank trees, and μ ranges over one-to-one matchings between the padded child multisets. At graph level, BTMD keeps the additive assignment across node-rooted computation trees. A GAT with L message-passing layers reads depth-$(L+1)$ computation trees; we state the bound for $L=3$ and write BTMD^4, where the superscript denotes depth and not a power. Let $D_{\text{rep}}(G_A, G_B)$ be the minimum additive matching cost between the final node representations the shared GAT produces for two architecture graphs.

Theorem 4.1 (BTMD controls GAT representation disagreement)**.** *For a fixed three-layer shared GAT with bounded architecture features and bounded node in-degree, there exists a finite constant* K_{arch} *such that*

$$D_{\text{rep}}(G_A, G_B) \;\le\; K_{\text{arch}}\ \text{BTMD}^4(G_A, G_B). \tag{14}$$

The proof is layer-recursive and therefore also yields the corresponding result for a two-layer GAT of the same form, with depth-3 computation trees instead of depth-4 trees.

The constant K_{arch} (written K, with BTMD^4 written BTMD, in Fig. 1) depends on the GAT's weight norms, the feature bound and the in-degree, not on the clients' data, so the bound measures structural distance between the clients' architecture graphs under the node features of Section 3.1. Architectures that are close under BTMD therefore cannot induce arbitrarily different representations under the shared GAT; Appendix A gives the proof and constructs K_{arch}.

5 Experiments

Datasets and setup. The source model is Qwen2.5-0.5B (Qwen Team, 2024). We evaluate on five single-task federations built from CoNLL-2003 (Tjong Kim Sang & De Meulder, 2003), SQuAD v1.1 (Rajpurkar et al., 2016), AG News (Zhang et al., 2015), XSum (Narayan et al., 2018) and SAMSum (Gliwa et al., 2019), each split over three clients, and on a multi-task federation in which four clients hold four different tasks. Clients hold unequal shares and are made non-IID by a Dirichlet draw over a partition key mixed with a fixed fraction of the global distribution. Scores are entity F1 over (type, surface string) pairs for CoNLL, answer F1 for SQuAD, accuracy for AG News, ROUGE-L for XSum and SAMSum, and the test negative log-likelihood (NLL, lower is better) for the multi-task federation, each averaged over clients by test size.

Evaluation and parameter setting. Inference cost is matched by Equation 2. Training cost is matched separately: with $c_k = c(r_k^*)$, a full model fine-tuned on the pooled data performs $E=2$ epochs, and client k performs $E_k = E/c_k$ epochs over its own data, so that $\sum_k c_k E_k n_k = E\sum_k n_k$ and every method receives the same modeled training FLOPs. Model selection uses validation data only. Every experiment is run in two independent series, A and B, each a complete and independent run of all experiments, and differences are never computed across series. In series A, the independent, aligned and selected sharing arms use seeds 42, 1 and 2, reported as mean±sd; single runs are reported as such. Arms trained at the same seeds are compared with a paired two-sided t-test, and co-training and alignment are compared across the six datasets with a paired Wilcoxon signed-rank test; significance ($p < 0.05$) is marked only where ours is better.

Comparison methods. Independent adaptation trains each specialist on its own data only, and isolates what co-training adds. Shape-only averaging replaces Equation 10 by position-wise averaging, and isolates the alignment. Two hand-designed allocation rules, width-only (narrow every block uniformly) and mlp-first (remove feed-forward capacity first), receive the same importance statistics, recovery and co-training as the searched structures, and isolate the learned allocator. A random structure drawn at the same budget and repaired by the same operator isolates the search. A full model fine-tuned on each client's own data at cost 1.0 bounds what the budget costs. Five published structured-pruning families, SLEB, LLM-Pruner, Adapt-Pruner, EvoPress and Minitron (Song et al., 2024; Ma et al., 2023; Pan et al., 2025; Sieberling et al., 2025; Muralidharan et al., 2024), are

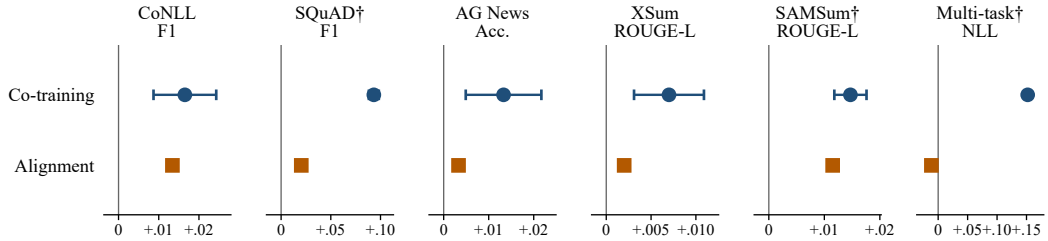


Figure 2: Co-training and alignment gains per task in series A (Table 4 values). Co-training: mean±sd over training seeds 42, 1 and 2 (error bars); alignment: seed 42, no error bar. †: $p < 0.05$ (paired t-test). Multi-task NLL sign reversed.

Table 1: Random structures drawn at the same budget and a full model fine-tuned on each client's data against FedBUD (searched structures), in series B, seed 42 (cost in parentheses). Differences favor the named alternative when positive (reversed for the multi-task NLL). Single runs. Datasets: CoNLL (Tjong Kim Sang & De Meulder, 2003), SQuAD (Rajpurkar et al., 2016), AG News (Zhang et al., 2015), XSum (Narayan et al., 2018), SAMSum (Gliwa et al., 2019).

Dataset	Random	Full model per client ($c = 1.0$)	FedBUD	Random − FedBUD	Full model − FedBUD
CoNLL (Tjong Kim Sang & De Meulder, 2003)	0.8437 (0.701)	0.8701	0.8484 (0.699)	−0.0047	+0.0217
SQuAD (Rajpurkar et al., 2016)	0.6481 (0.678)	0.7991	0.7306 (0.706)	−0.0825	+0.0685
AG News (Zhang et al., 2015)	0.9456 (0.701)	0.9467	0.9456 (0.688)	0.0000	+0.0011
XSum (Narayan et al., 2018)	0.2215 (0.699)	0.2549	0.2335 (0.706)	−0.0120	+0.0214
SAMSum (Gliwa et al., 2019)	0.3731 (0.700)	0.3915	0.3833 (0.693)	−0.0102	+0.0082
Multi-task (NLL↓)	1.0460 (0.698)	0.7616	0.9696 (0.701)	−0.0764	+0.2080

reproduced as their papers define them, each calibrated on its paper's corpus and on each client's data (Appendix C).

Co-training across heterogeneous specialists. We test *Hypothesis 1* at the same cost. The evidence validates it: co-training scores higher than independent adaptation on six of six datasets in the mean over seeds, which is significant across datasets (Wilcoxon signed-rank test, $p = 0.031$), and individually on three of six, SQuAD, SAMSum and the multi-task federation (paired t-test, $p < 0.05$; Fig. 2, Table 4). Clients whose sub-networks differ therefore still gain from each other's updates on shared components, with data kept local.

Coordinate alignment. We test *Hypothesis 2*. The evidence partially supports it: the aligned run scores higher than shape-only averaging on five of six datasets in series A, but it raises the multi-task NLL in both series, and with the shape-only arm run at one seed no effect is established (Tables 4 and 7). Alignment therefore suits federations that share a task; the cause of the reversal is not established.

Learned allocation. We test *Hypothesis 3*. Against random structures the evidence supports it: searched structures score higher on five of six datasets in series B and equal on the sixth, on both datasets where the random structure was redrawn at the searched structure's cost, and on two of three datasets in series A (Tables 1 and 7). Against hand-designed rules *Hypothesis 3* is not supported: the better rule scores higher than the searched structures on four of six datasets in series A (Table 4). A hand-designed rule on the same client importance statistics can thus replace the search.

Sharing structure across clients. Besides giving each client its own structure, FedBUD can share structure across clients: one searched structure reused by every client, or per-client allocations whose heads and neurons are chosen by importance pooled over all clients; the choice is made per task on validation data. We test *Hypothesis 4*. The evidence validates it on both tasks where the selected arm was run at three seeds: the shared structure on CoNLL ($p = 0.024$) and pooled importance on XSum ($p = 0.049$) improve significantly over per-client structures (paired t-test; Table 5, Fig. 3); on SQuAD, validation selects another client's structure, which scores below per-client structures in a

single run (Table 7). On the two tasks where the selected arm was run at three seeds, validation data therefore identify a common structure that benefits the clients; on SQuAD, the selected arm does not.

Cost of the budget. We ask what the budget costs relative to a full model fine-tuned on each client's data. At 0.68 to 0.71 of the per-token FLOPs, the specialists score below the full model on six of six datasets in single runs, and the gap is largest on the multi-task federation, where the same structure must serve four tasks (Table 1).

Published structured pruning. We also ask how co-trained specialists compare with published structured-pruning methods applied as their papers define them. The co-trained searched specialists score above the best published-form arm on six of six datasets, (Tables 6 and 7). The baselines are single runs outside our two series, so the comparison is indicative; it extends *Hypothesis 1* to structures produced by other pruning methods.

6 Discussion

Findings. Under matched FLOPs, the experiments validate *Hypothesis 1* across the six datasets and significantly on three, and *Hypothesis 4* on the two tasks where validation selects sharing; they partially support *Hypothesis 2*, since alignment scores higher than shape-only averaging on five of six datasets but raises the multi-task NLL, and they do not support *Hypothesis 3* against hand-designed rules. The framework is designed for federations in which a per-token compute budget binds every client and client data differ and cannot be pooled. Its analysis covers representation stability rather than convergence.

Limitations. First, the budget is $b = 0.6$ (one check at 0.5, Appendix E), and the search and the partition use one seed each, so search-seed and partition variability are unmeasured and the reported standard deviations cover training-seed variability only. Second, three seeds do not resolve the smaller co-training gains, and single-run comparisons carry no test. Third, structure sharing was evaluated on three tasks at one partition and, on CoNLL and XSum, three seeds. Fourth, coordinate alignment degrades the multi-task federation in both series, and no measured quantity explains why, so the aggregation rule should not be applied to task-heterogeneous federations without validation. Fifth, the published pruning baselines were run once each, outside our two series, so comparisons with them are indicative. Sixth, the federation is simulated and we make no formal privacy claim, and all results use one source model of 0.5B parameters.

7 Conclusion

We proposed FedBUD, a framework that decomposes one pretrained LLM into per-client structural specialists under a hard per-token FLOPs budget, recovers them from inherited weights with LoRA, and co-trains them by aggregation in the coordinates of the source model, optionally with a shared GNN whose representation discrepancy we bound for its GAT instantiation. The experiments validate *Hypothesis 1*, since clients whose sub-networks differ still gain from co-training, and *Hypothesis 4* where validation selects a common structure; they partially support *Hypothesis 2* and do not support *Hypothesis 3* against a hand-designed rule on the clients' importance statistics. Co-training can thus serve federations of institutions that cannot move their text and whose devices impose a per-token budget, without every client running the full model; it does not close the gap to a full model per client, and alignment should be validated before use in task-heterogeneous federations. Future work will measure search-seed and partition variability, test per-client structure on further tasks, and study ensembles of the specialists, which exceed the per-token budget.

Ethics Statement

All datasets are public benchmarks, the federation is simulated, and no personal data were collected.

Reproducibility Statement

Section 5 and Appendix B state every hyperparameter, the FLOPs-matching rule and the statistical protocol; code will be released upon acceptance.

AI Use Statement

In this work, we used generative AI tools to aid and polish the writing of the paper, and to assist in formalizing the theorem of Section 4 and writing its proof in Appendix A. We have reviewed all AI-assisted work. We take responsibility for the final content of this work.

References

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations (ICLR)*, 2024.

Samiul Alam, Luyang Liu, Ming Yan, and Mi Zhang. FedRolex: Model-heterogeneous federated learning with rolling sub-model extraction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Saleh Ashkboos, Maximilian L. Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. SliceGPT: Compress large language models by deleting rows and columns. In *International Conference on Learning Representations (ICLR)*, 2024.

Guangji Bai, Yijiang Li, Zilinghan Li, Liang Zhao, and Kibaek Kim. 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 (NAACL)*, 2025.

Jiamu Bai, Daoyuan Chen, Bingchen Qian, Liuyi Yao, and Yaliang Li. Federated fine-tuning of large language models under heterogeneous tasks and client resources. In *Advances in Neural Information Processing Systems*, volume 37, 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 (EMNLP)*, pp. 12903–12913, 2024.

Ching-Yao Chuang and Stefanie Jegelka. Tree mover's distance: Bridging graph metrics and stability of graph neural networks. In *Advances in Neural Information Processing Systems*, volume 35, pp. 2944–2957, 2022.

Enmao Diao, Jie Ding, and Vahid Tarokh. HeteroFL: Computation and communication efficient federated learning for heterogeneous clients. In *International Conference on Learning Representations (ICLR)*, 2021.

Arthur Douillard, Qixuan Feng, Andrei A. Rusu, Rachita Chhaparia, Yani Donchev, Adhiguna Kuncoro, Marc'Aurelio Ranzato, Arthur Szlam, and Jiajun Shen. DiLoCo: Distributed low-communication training of language models. *arXiv preprint arXiv:2311.08105*, 2023.

Arthur Douillard, Qixuan Feng, Andrei A. Rusu, Adhiguna Kuncoro, Yani Donchev, Rachita Chhaparia, Ionel Gog, Marc'Aurelio Ranzato, Jiajun Shen, and Arthur Szlam. DiPaCo: Distributed path composition. *arXiv preprint arXiv:2403.10616*, 2024.

Tao Fan, Guoqiang Ma, Yuanfeng Song, Lixin Fan, and Qiang Yang. PPC-GPT: Federated task-specific compression of large language models via pruning and chain-of-thought distillation. *arXiv preprint arXiv:2502.15857*, 2025.

Yuan Gao, Zujing Liu, Weizhong Zhang, Bo Du, and Gui-Song Xia. Bypass back-propagation: Optimization-based structural pruning for large language models via policy gradient. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 29356–29377. Association for Computational Linguistics, 2025.

Bogdan Gliwa, Iwona Mochol, Maciej Biesek, and Aleksander Wawer. SAMSum corpus: A human-annotated dialogue dataset for abstractive summarization. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pp. 70–79, 2019.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: On-policy distillation of large language models. In *International Conference on Learning Representations (ICLR)*, 2024.

Pengxin Guo, Yinong Wang, Wei Li, Mengting Liu, Ming Li, Jinkai Zheng, and Liangqiong Qu. Exploring federated pruning for large language models. *arXiv preprint arXiv:2505.13547*, 2025.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *International Conference on Learning Representations (ICLR)*, 2023.

Minjae Kim, Sangyoon Yu, Suhyun Kim, and Soo-Mook Moon. DepthFL: Depthwise federated learning for heterogeneous clients. In *International Conference on Learning Representations (ICLR)*, 2023.

Minkyu Kim, Vincent-Daniel Yun, Youngrae Kim, Suin Cho, Woosang Lim, and Sunwoo Lee. Rethinking layer redundancy: Calibration matters more than search in LLM depth pruning. *arXiv preprint arXiv:2604.24938*, 2026.

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.

Ang Li, Jingwei Sun, Pengcheng Li, Yu Pu, Hai Li, and Yiran Chen. Hermes: An efficient federated learning framework for heterogeneous mobile clients. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking (MobiCom)*, pp. 420–437, 2021. doi: 10.1145/3447993.3483278.

Margaret Li, Suchin Gururangan, Tim Dettmers, Mike Lewis, Tim Althoff, Noah A Smith, and Luke Zettlemoyer. Branch-train-merge: Embarrassingly parallel training of expert language models. *arXiv preprint arXiv:2208.03306*, 2022.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. LLM-Pruner: On the structural pruning of large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

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

Xin Men, Mingyu Xu, Qingyu Zhang, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. ShortGPT: Layers in large language models are more redundant than you expect. In *Findings of the Association for Computational Linguistics: ACL 2025*, 2025.

Alireza Moayedikia and Alicia Troncoso Lora. Capacity confounds and coverage guarantees in adaptive sub-model federated learning. *arXiv preprint arXiv:2608.07157*, 2026.

Saurav Muralidharan, Sharath Turuvekere Sreenivas, Raviraj Joshi, Marcin Chochowski, Mostofa Patwary, Mohammad Shoeybi, Bryan Catanzaro, Jan Kautz, and Pavlo Molchanov. Compact language models via pruning and knowledge distillation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 1797–1807, 2018.

Furkan Pala and Islem Rekik. UnifiedFL: A dynamic unified learning framework for equitable federation. *arXiv preprint arXiv:2510.26350*, 2025a.

Furkan Pala and Islem Rekik. GNN-based unified deep learning. In *Predictive Intelligence in Medicine: 8th International Workshop, PRIME 2025, Held in Conjunction with MICCAI 2025*, volume 16164 of *Lecture Notes in Computer Science*, pp. 34–46. Springer, 2025b. doi: 10.1007/978-3-032-07904-6_4.

Furkan Pala and Islem Rekik. Towards highly heterogeneous federated learning: A graph-based framework for architectural and data heterogeneity. In *MICCAI 2026 Workshop on Distributed, Collaborative and Federated Learning (DeCaF)*, 2026. URL https://openreview.net/forum?id=dakXYmUwP7. OpenReview.

Rui Pan, Shivanshu Shekhar, Boyao Wang, Shizhe Diao, Jipeng Zhang, Xingyuan Pan, Renjie Pi, and Tong Zhang. Adapt-Pruner: Adaptive structural pruning for efficient small language model training. *arXiv preprint arXiv:2502.03460*, 2025.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Xin Qiu and Risto Miikkulainen. Shortest edit path crossover: A theory-driven solution to the permutation problem in evolutionary neural architecture search. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 28422–28447. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/qiu23b.html.

Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, 2016.

Oliver Sieberling, Denis Kuznedelev, Eldar Kurtic, and Dan Alistarh. EvoPress: Accurate dynamic model compression via evolutionary search. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, 2025.

Jiwon Song, Kyungseok Oh, Taesu Kim, Hyungjun Kim, Yulhwa Kim, and Jae-Joon Kim. SLEB: Streamlining LLMs through redundancy verification and elimination of transformer blocks. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.

Shangchao Su, Bin Li, and Xiangyang Xue. FedRA: A random allocation strategy for federated tuning to unleash the power of heterogeneous clients. In *European Conference on Computer Vision (ECCV)*, 2024.

Sainbayar Sukhbaatar, Olga Golovneva, Vasu Sharma, Hu Xu, Xi Victoria Lin, Baptiste Rozière, Jacob Kahn, Daniel Li, Wen-tau Yih, Jason Weston, and Xian Li. Branch-train-mix: Mixing expert llms into a mixture-of-experts llm. *arXiv preprint arXiv:2403.07816*, 2024.

Shengkun Tang, Oliver Sieberling, Eldar Kurtic, Zhiqiang Shen, and Dan Alistarh. DarwinLM: Evolutionary structured pruning of large language models. In *Conference on Language Modeling (COLM)*, 2026.

Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pp. 142–147, 2003.

Matteo Togninalli, Marcello Ghisu, Felipe Llinares-López, Bastian Rieck, and Karsten M. Borgwardt. Wasserstein weisfeiler-lehman graph kernels. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32, pp. 6436–6446. Curran Associates, Inc., 2019.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

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. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Feijie Wu, Xingchen Wang, Yaqing Wang, Tianci Liu, Lu Su, and Jing Gao. FIARSE: Model-heterogeneous federated learning via importance-aware submodel extraction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Yebo Wu, Jingguang Li, Chunlin Tian, Zhijiang Guo, and Li Li. Memory-efficient federated fine-tuning of large language models via layer pruning. *arXiv preprint arXiv:2508.17209*, 2025.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ryGs6iA5Km.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. TIES-Merging: Resolving interference when merging models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Zhiwei Yao, Yang Xu, Hongli Xu, Yunming Liao, and Zuan Xie. Efficient deployment of large language models on resource-constrained devices. *arXiv preprint arXiv:2501.02438*, 2025.

Rui Ye, Wenhao Wang, Jingyi Chai, Dihan Li, Zexi Li, Yinda Xu, Yaxin Du, Yanfeng Wang, and Siheng Chen. OpenFedLLM: Training large language models on decentralized private data via federated learning. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2024.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. Language models are Super Mario: Absorbing abilities from homologous models as a free lunch. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.

Sixing Yu, Arya Mazaheri, and Ali Jannesari. Topology-aware network pruning using multi-stage graph embedding and reinforcement learning. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, volume 162 of *Proceedings of Machine Learning Research*, pp. 25656–25667, 2022.

Jianyi Zhang, Saeed Vahidian, Martin Kuo, Chunyuan Li, Ruiyi Zhang, Tong Yu, Yufan Zhou, Guoyin Wang, and Yiran Chen. Towards building the federated GPT: Federated instruction tuning. *arXiv preprint arXiv:2305.05644*, 2023.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, volume 28, pp. 649–657, 2015.

Yiran Zhao, Guizhen Chen, Kenji Kawaguchi, Lidong Bing, and Wenxuan Zhang. Pruning general large language models into customized expert models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 23377–23391. Association for Computational Linguistics, 2025.

A Proof of Theorem 4.1

A.1 Notation for one GAT layer

For the architecture of Theorem 4.1, fix a layer $\ell \in \{1, 2, 3\}$ and a head $q \in \{1, \dots, H_\ell\}$, where

$$H_1 = 4, \qquad H_2 = 4, \qquad H_3 = 1.$$

For head q, let

$$W_{\ell q} \in \mathbb{R}^{64 \times d_{\ell-1}}$$

be the per-head linear map, and let

$$a_{\ell q}^{\mathrm{tar}},\ a_{\ell q}^{\mathrm{src}} \in \mathbb{R}^{64}$$

be the target and source attention vectors. Define

$$M_{\ell q} := \|W_{\ell q}\|_2, \tag{15}$$

$$A_{\ell q}^{\mathrm{tar}} := \left\|a_{\ell q}^{\mathrm{tar}}\right\|_2, \tag{16}$$

$$A_{\ell q}^{\mathrm{src}} := \left\|a_{\ell q}^{\mathrm{src}}\right\|_2. \tag{17}$$

For a node r, define the transformed message

$$m_r^{(\ell,q)} := W_{\ell q} h_r^{(\ell-1)}.$$

For a target node v and a source $r \in \{v\} \cup \mathcal{N}(v)$, the pre-activation attention score and logit are

$$s_{vr}^{(\ell,q)} := (a_{\ell q}^{\mathrm{tar}})^\top m_v^{(\ell,q)} + (a_{\ell q}^{\mathrm{src}})^\top m_r^{(\ell,q)}, \tag{18}$$

$$e_{vr}^{(\ell,q)} := \rho\Big(s_{vr}^{(\ell,q)}\Big), \tag{19}$$

where ρ is LeakyReLU. The normalized attention coefficient is

$$\pi_{vr}^{(\ell,q)} := \frac{\exp(e_{vr}^{(\ell,q)})}{\sum_{t\in\{v\}\cup\mathcal{N}(v)} \exp(e_{vt}^{(\ell,q)})}, \tag{20}$$

and the one-head output is

$$y_v^{(\ell,q)} := \sum_{r\in\{v\}\cup\mathcal{N}(v)} \pi_{vr}^{(\ell,q)} m_r^{(\ell,q)}. \tag{21}$$

A.2 Equal-cardinality local stability

Take real target nodes $v \in V_A$ and $u \in V_B$ with equally many retained genuine neighbors, and let

$$\mu : \mathcal{N}_A(v) \to \mathcal{N}_B(u)$$

be any bijection. Extend it by setting $\mu(v) = u$ for the self-source. Define

$$D_*(v,u;\mu) := \max\left\{\left\|h_v^{A,(\ell-1)} - h_u^{B,(\ell-1)}\right\|_2, \max_{r\in\mathcal{N}_A(v)} \left\|h_r^{A,(\ell-1)} - h_{\mu(r)}^{B,(\ell-1)}\right\|_2\right\}. \tag{22}$$

Lemma A.1 (One-head GAT stability)**.** *Suppose all representations entering layer ℓ satisfy*

$$\left\|h_r^{(\ell-1)}\right\|_2 \le R_{\ell-1}.$$

Then

$$\left\|y_v^{A,(\ell,q)} - y_u^{B,(\ell,q)}\right\|_2 \le C_{\ell q}\, D_*(v,u;\mu), \tag{23}$$

where

$$\boxed{C_{\ell q} := M_{\ell q} + M_{\ell q}^2 R_{\ell-1}\left(A_{\ell q}^{\mathrm{tar}} + A_{\ell q}^{\mathrm{src}}\right).} \tag{24}$$

Proof. Using the matching μ, add and subtract the mixed term $\pi_{vr}^A m_{\mu(r)}^B$ to obtain

$$\begin{aligned} y_v^A - y_u^B = &\sum_r \pi_{vr}^A \left(m_r^A - m_{\mu(r)}^B\right) \\ &+ \sum_r \left(\pi_{vr}^A - \pi_{u\mu(r)}^B\right) m_{\mu(r)}^B. \end{aligned} \tag{25}$$

All superscripts (ℓ, q) are suppressed inside this proof for readability.

For the first sum, the shared linear map implies

$$\left\|m_r^A - m_{\mu(r)}^B\right\|_2 \le M_{\ell q} \left\|h_r^A - h_{\mu(r)}^B\right\|_2 \le M_{\ell q} D_*.$$

Since the attention coefficients are nonnegative and sum to one,

$$\left\| \sum_r \pi^A_{vr}(m^A_r - m^B_{\mu(r)}) \right\|_2 \le M_{\ell q} D_*. \tag{26}$$

For the attention term, LeakyReLU is 1-Lipschitz. For a genuine neighbor r,

$$\left| e^A_{vr} - e^B_{u\mu(r)} \right| \le M_{\ell q} \left(A^{\mathrm{tar}}_{\ell q} \left\| h^A_v - h^B_u \right\|_2 + A^{\mathrm{src}}_{\ell q} \left\| h^A_r - h^B_{\mu(r)} \right\|_2 \right),$$

while for the self-source,

$$\left| e^A_{vv} - e^B_{uu} \right| \le M_{\ell q} \left(A^{\mathrm{tar}}_{\ell q} + A^{\mathrm{src}}_{\ell q} \right) \left\| h^A_v - h^B_u \right\|_2 .$$

After arranging the two logit vectors according to μ,

$$\left\| \boldsymbol{e}_A - \boldsymbol{e}^{\mu}_B \right\|_\infty \le M_{\ell q} \left(A^{\mathrm{tar}}_{\ell q} + A^{\mathrm{src}}_{\ell q} \right) D_*. \tag{27}$$

Softmax satisfies

$$\left\| \mathrm{softmax}(x) - \mathrm{softmax}(y) \right\|_1 \le \left\| x - y \right\|_\infty . \tag{28}$$

Therefore

$$\left\| \boldsymbol{\pi}_A - \boldsymbol{\pi}^{\mu}_B \right\|_1 \le M_{\ell q} \left(A^{\mathrm{tar}}_{\ell q} + A^{\mathrm{src}}_{\ell q} \right) D_*.$$

Also $\left\| m^B_{\mu(r)} \right\|_2 \le M_{\ell q} R_{\ell-1}$. Hence

$$\begin{aligned} \left\| \sum_r (\pi^A_{vr} - \pi^B_{u\mu(r)}) m^B_{\mu(r)} \right\|_2 &\le M_{\ell q} R_{\ell-1} \left\| \boldsymbol{\pi}_A - \boldsymbol{\pi}^{\mu}_B \right\|_1 \\ &\le M^2_{\ell q} R_{\ell-1} \left(A^{\mathrm{tar}}_{\ell q} + A^{\mathrm{src}}_{\ell q} \right) D_*. \end{aligned} \tag{29}$$

Combining (26) and (29) proves (23). □

Define

$$\boxed{C_\ell := \left(\sum_{q=1}^{H_\ell} C^2_{\ell q} \right)^{1/2}.} \tag{30}$$

For $\ell = 1, 2$, concatenation of the head outputs gives Euclidean norm equal to the square root of the sum of squared per-head norms. The shared bias cancels when comparing the two architectures, and ELU is 1-Lipschitz. Layer three has one head and no output nonlinearity. Hence for any equal-cardinality retained neighbor matching μ,

$$\boxed{\Delta_\ell(v,u) \le C_\ell \max \left\{ \Delta_{\ell-1}(v,u), \max_{r \in \mathcal{N}_A(v)} \Delta_{\ell-1}(r, \mu(r)) \right\},} \tag{31}$$

where

$$\Delta_\ell(a,b) := \left\| h^{A,(\ell)}_a - h^{B,(\ell)}_b \right\|_2 .$$

Equation (31) is the local max-type recurrence that motivates the bottleneck child geometry in (33).

A.3 Metric properties of the recursive bottleneck tree distance

For completeness, we restate the recursive tree distance of Equation 13 (Section 4) used below. At depth one,

$$B_1(T,S) := \left\| \widetilde{x}_{\mathrm{root}(T)} - \widetilde{x}_{\mathrm{root}(S)} \right\|_2 . \tag{32}$$

For depth $k+1$, let $\overline{\mathcal{C}}(T)$ and $\overline{\mathcal{C}}(S)$ denote the child-subtree multisets after symmetric padding to $D_{\max}$ elements with depth-k blank trees. Then

$$B_{k+1}(T,S) := \max \left\{ \left\| \widetilde{x}_{\mathrm{root}(T)} - \widetilde{x}_{\mathrm{root}(S)} \right\|_2 , \min_{\mu} \max_{R \in \overline{\mathcal{C}}(T)} B_k\big(R, \mu(R)\big) \right\}, \tag{33}$$

where μ ranges over bijections between the two padded child-subtree multisets.

We first justify the metric structure of B_k.

Lemma A.2 (Bottleneck assignment is a metric on fixed-cardinality multisets)**.** *Let $(\mathcal{X}, d)$ be a metric space and let $\mathcal{A}, \mathcal{B}$ be multisets of exactly $D_{\max}$ elements from $\mathcal{X}$. Define*

$$d_\infty(\mathcal{A}, \mathcal{B}) := \min_{\mu} \max_{a \in \mathcal{A}} d(a, \mu(a)),$$

where μ ranges over bijections of the two multisets. Then d_∞ is a metric on $D_{\max}$-element multisets.

Proof. Nonnegativity and symmetry are immediate. If $d_\infty(\mathcal{A}, \mathcal{B}) = 0$, some bijection pairs every element at zero d-cost, so the two multisets are identical.

For the triangle inequality, let μ_{AB} and μ_{BC} be optimal bijections for $\mathcal{A} \to \mathcal{B}$ and $\mathcal{B} \to \mathcal{C}$. Their composition $\mu_{AC} := \mu_{BC} \circ \mu_{AB}$ is a bijection from $\mathcal{A}$ to $\mathcal{C}$. For every $a \in \mathcal{A}$,

$$d(a, \mu_{AC}(a)) \le d(a, \mu_{AB}(a)) + d(\mu_{AB}(a), \mu_{AC}(a)).$$

Each term is bounded by the corresponding bottleneck optimum, so

$$\max_a d(a, \mu_{AC}(a)) \le d_\infty(\mathcal{A}, \mathcal{B}) + d_\infty(\mathcal{B}, \mathcal{C}).$$

The optimal $\mathcal{A}$–$\mathcal{C}$ bottleneck cost can only be smaller than the cost of this particular composed bijection, proving the triangle inequality. □

Lemma A.3 (Recursive tree metric)**.** *For every finite depth k, B_k defined by (32)–(33) is a metric on depth-k rooted attributed trees modulo feature-preserving rooted isomorphism and child order.*

Proof. The claim is by induction on k. At depth one, B_1 is Euclidean distance on the augmented root features.

Assume B_k is a metric. By Lemma A.2, the bottleneck assignment of the two $D_{\max}$-element padded child multisets is a metric. The root-feature term is also a metric. Their maximum is again a metric: if d_1, d_2 are metrics, then

$$\max\{d_1(x, z), d_2(x, z)\} \le \max\{d_1(x, y), d_2(x, y)\} + \max\{d_1(y, z), d_2(y, z)\}.$$

Identity of indiscernibles follows because $B_{k+1}(T, S) = 0$ forces both identical root features and zero bottleneck distance between the padded child multisets; the induction hypothesis then gives a feature-preserving rooted isomorphism. Thus B_{k+1} is a metric. □

Lemma A.4 (Depth monotonicity)**.** *Let T^{k+1} and S^{k+1} be depth-$(k+1)$ rooted trees and let T^k, S^k denote their depth-k truncations. Then*

$$B_{k+1}(T^{k+1}, S^{k+1}) \ge B_k(T^k, S^k). \tag{34}$$

Proof. The base case follows directly from (33): B_2 is the maximum of the depth-1 root distance and a nonnegative child term, so $B_2 \ge B_1$.

For the inductive step, assume every depth-k child-pair distance dominates the corresponding depth-$(k-1)$ truncated distance. For any fixed bijection μ of the padded child multisets, its depth-k bottleneck cost

$$F_k(\mu) := \max_R B_k(R, \mu(R))$$

is at least the bottleneck cost of the same matching after truncation,

$$F_{k-1}(\mu) := \max_R B_{k-1}(R^{k-1}, \mu(R)^{k-1}).$$

This holds for every bijection. If $\mu_k^\star$ minimizes F_k, then

$$\min_\mu F_k(\mu) = F_k(\mu_k^\star) \ge F_{k-1}(\mu_k^\star) \ge \min_\mu F_{k-1}(\mu).$$

The minimizing bijection is allowed to change with depth. The root-feature term is unchanged under truncation, so taking the maximum with the nondecreasing child term proves (34). □

Lemma A.5 (Real–blank separation)**.** *For every real depth-k computation tree T,*

$$B_k(T, T_\perp^k) \ge \xi, \tag{35}$$

where $T_\perp^k$ denotes the recursively padded depth-k blank tree.

Proof. The root term in (32) or (33) is

$$\left\|\widetilde{x}_{\mathrm{root}(T)} - 0\right\|_2 = \sqrt{\|x_{\mathrm{root}(T)}\|_2^2 + \xi^2} \geq \xi.$$

Since B_k is the maximum of this root term and a nonnegative child term, the full distance is at least ξ. □

A.4 Deleting bottleneck-unmatched GAT sources

The symmetric padded bottleneck matching at a given target pair can contain four kinds of child pairs: real–real, real–blank, blank–real, and blank–blank. Real–blank or blank–real pairs correspond to genuine neighbor sources that are present in one GAT neighborhood but are not retained in the real–real correspondence used to compare the two actual GAT updates.

For one head, suppose a set $\mathcal{U}(v) \subseteq \mathcal{N}(v)$ of genuine neighbors is deleted. Let $\widehat{y}_v$ be the head output obtained by removing those sources and renormalizing softmax over the retained sources. Define the total deleted attention mass

$$\alpha := \sum_{r \in \mathcal{U}(v)} \pi_{vr}.$$

The retained probabilities become

$$\widehat{\pi}_{vr} = \frac{\pi_{vr}}{1-\alpha}.$$

Because $\|m_r\|_2 \leq M_{\ell q} R_{\ell-1}$, the deleted-message contribution and the softmax-renormalization contribution are each at most $M_{\ell q} R_{\ell-1} \alpha$. Therefore

$$\|y_v - \widehat{y}_v\|_2 \leq 2M_{\ell q} R_{\ell-1} \alpha \leq 2M_{\ell q} R_{\ell-1}. \tag{36}$$

Define

$$E_{\ell q} := 2M_{\ell q} R_{\ell-1}, \tag{37}$$

$$E_\ell := \left(\sum_{q=1}^{H_\ell} E_{\ell q}^2 \right)^{1/2}. \tag{38}$$

After concatenation, bias cancellation in differences, and the 1-Lipschitz ELU, deleting any nonempty set of genuine neighbor sources from one complete layer changes that layer's node representation by at most E_ℓ.

A.5 Node-level induction for arbitrary neighbor counts

We prove

$$\boxed{P(\ell): \qquad \Delta_\ell(a,b) \leq K_\ell \, B_{\ell+1}(T_a^{\ell+1}, T_b^{\ell+1}) \quad \text{for every real node pair } (a,b).} \tag{39}$$

The base case is $\ell = 0$. Since both roots are real,

$$\Delta_0(a,b) = \|x_a - x_b\|_2 = B_1(T_a^1, T_b^1),$$

so $K_0 = 1$.

For the inductive step, assume $P(\ell-1)$ for every real node pair and fix arbitrary targets $v \in V_A$, $u \in V_B$. Let $\mu_\ell^\star$ be a bottleneck-optimal bijection between their two depth-ℓ child multisets after both have been padded to $D_{\max}$ elements. Define

$$Q_\ell^\infty(v,u) := \max_{R \in \overline{\mathcal{C}}(T_v^{\ell+1})} B_\ell\big(R, \mu_\ell^\star(R)\big). \tag{40}$$

By definition,

$$B_{\ell+1}(T_v^{\ell+1}, T_u^{\ell+1}) = \max\left\{ \|x_v - x_u\|_2 , Q_\ell^\infty(v,u) \right\}. \tag{41}$$

Partition the optimal padded child pairs into real–real pairs $\mathcal{R}$, real–blank pairs, blank–real pairs, and blank–blank pairs. Delete from v every genuine neighbor participating in a real–blank pair, and delete from u every genuine neighbor participating in a blank–real pair. Let

$$\widehat{h}_v^{(\ell)}, \qquad \widehat{h}_u^{(\ell)}$$

be the corresponding intermediate complete-layer outputs. After deletion, the two retained genuine neighbor sets have the same cardinality and are paired exactly by $\mathcal{R}$. Hence

$$\begin{aligned}\Delta_\ell(v,u) \le & \left\| h_v^{A,(\ell)} - \widehat{h}_v^{A,(\ell)} \right\|_2 \\ & + \left\| \widehat{h}_v^{A,(\ell)} - \widehat{h}_u^{B,(\ell)} \right\|_2 \\ & + \left\| \widehat{h}_u^{B,(\ell)} - h_u^{B,(\ell)} \right\|_2 . \end{aligned} \tag{42}$$

For the middle term, apply (31) to the retained real–real matching. By the induction hypothesis,

$$\left\| \widehat{h}_v^{A,(\ell)} - \widehat{h}_u^{B,(\ell)} \right\|_2 \le C_\ell K_{\ell-1} \max\Bigg\{ B_\ell(T_v^\ell, T_u^\ell), \max_{(r,s)\in\mathcal{R}} B_\ell(T_r^\ell, T_s^\ell) \Bigg\}. \tag{43}$$

By Lemma A.4,

$$B_\ell(T_v^\ell, T_u^\ell) \le B_{\ell+1}(T_v^{\ell+1}, T_u^{\ell+1}).$$

Every real–real child pair in $\mathcal{R}$ is one of the pairs in the bottleneck-optimal padded assignment, so

$$\max_{(r,s)\in\mathcal{R}} B_\ell(T_r^\ell, T_s^\ell) \le Q_\ell^\infty(v,u) \le B_{\ell+1}(T_v^{\ell+1}, T_u^{\ell+1}).$$

Therefore

$$\left\| \widehat{h}_v^{A,(\ell)} - \widehat{h}_u^{B,(\ell)} \right\|_2 \le C_\ell K_{\ell-1} B_{\ell+1}(T_v^{\ell+1}, T_u^{\ell+1}). \tag{44}$$

Now define indicators

$$I_A := \mathbf{1}\{\text{at least one real–blank child pair occurs}\},$$
$$I_B := \mathbf{1}\{\text{at least one blank–real child pair occurs}\}.$$

The deletion bound gives

$$\left\| h_v^{A,(\ell)} - \widehat{h}_v^{A,(\ell)} \right\|_2 \le E_\ell I_A, \qquad \left\| h_u^{B,(\ell)} - \widehat{h}_u^{B,(\ell)} \right\|_2 \le E_\ell I_B.$$

If $I_A = 1$ or $I_B = 1$, Lemma A.5 implies that the bottleneck child cost $Q_\ell^\infty(v,u)$ is at least ξ. Hence in either case

$$I_A \le \frac{B_{\ell+1}(T_v^{\ell+1}, T_u^{\ell+1})}{\xi}, \qquad I_B \le \frac{B_{\ell+1}(T_v^{\ell+1}, T_u^{\ell+1})}{\xi}.$$

Therefore

$$\left\| h_v^{A,(\ell)} - \widehat{h}_v^{A,(\ell)} \right\|_2 + \left\| \widehat{h}_u^{B,(\ell)} - h_u^{B,(\ell)} \right\|_2 \le \frac{2E_\ell}{\xi} B_{\ell+1}(T_v^{\ell+1}, T_u^{\ell+1}). \tag{45}$$

Combining (42), (44), and (45) gives

$$\boxed{\Delta_\ell(v,u) \le \left(C_\ell K_{\ell-1} + \frac{2E_\ell}{\xi} \right) B_{\ell+1}(T_v^{\ell+1}, T_u^{\ell+1}).} \tag{46}$$

Thus define

$$\boxed{K_0 := 1, \qquad K_\ell := C_\ell K_{\ell-1} + \frac{2E_\ell}{\xi}, \quad \ell = 1,2,3.} \tag{47}$$

The induction closes, so

$$\boxed{\Delta_\ell(v,u) \le K_\ell B_{\ell+1}(T_v^{\ell+1}, T_u^{\ell+1})} \tag{48}$$

for arbitrary real node pairs and arbitrary neighbor-count mismatch.

A.6 Closing the representation-norm bounds

Define

$$R_0 := \max_{v\in V_A \cup V_B} \|x_v\|_2 . \tag{49}$$

For each layer, define

$$\overline{M}_\ell := \left(\sum_{q=1}^{H_\ell} M_{\ell q}^2 \right)^{1/2}. \tag{50}$$

Each attention head output is a convex combination of transformed source vectors, so

$$\left\| y_v^{(\ell,q)} \right\|_2 \le M_{\ell q} R_{\ell-1}.$$

After concatenating the heads, adding the shared layer bias b_ℓ, and using the fact that ELU is 1-Lipschitz with $\mathrm{ELU}(0) = 0$, define the valid upper-bound recursion

$$\boxed{R_\ell := \overline{M}_\ell R_{\ell-1} + \|b_\ell\|_2, \qquad \ell = 1, 2, 3.} \tag{51}$$

Then every node satisfies $\left\| h_v^{(\ell)} \right\|_2 \le R_\ell$, in particular $\|z_v\|_2 \le R_3$. Equations (24), (30), (37), (38), (51), and (47) therefore define all node-level constants explicitly from the fixed GAT parameters and the static feature bound.

A.7 Graph-level additive assignment and lifting to whole architectures

At graph level, let $\widehat{\mathcal{T}}^4_{G_A}$ and $\widehat{\mathcal{T}}^4_{G_B}$ denote the two depth-4 computation-tree multisets after one-sided padding of the smaller multiset with blank trees to equal cardinality. BTMD uses the additive assignment

$$\mathrm{BTMD}^4(G_A, G_B) := \min_{\nu} \sum_{T \in \widehat{\mathcal{T}}^4_{G_A}} B_4\big(T, \nu(T)\big), \tag{52}$$

where ν ranges over bijections between the two equal-cardinality multisets. Because the ground distance B_4 is a metric and the graph-level objective is additive, one-sided padding of the smaller multiset with blank trees has the same useful property as in ordinary TMD: adding extra blanks to both sides cannot reduce the optimal total assignment cost. Indeed, for any real trees T, S,

$$B_4(T, S) \le B_4(T, T^4_\perp) + B_4(T^4_\perp, S)$$

by the triangle inequality for B_4. Thus the graph-level additive construction is a metric on finite depth-4 computation-tree multisets; as a function of the original graphs it is generally a pseudometric.

Let $\nu^\star_{\mathrm{BTMD}}$ be a graph-level BTMD-optimal assignment between the padded depth-4 tree multisets. For every real–real matched pair, (48) at $\ell = 3$ gives

$$\left\| z_v^A - z^B_{\nu^\star_{\mathrm{BTMD}}(v)} \right\|_2 \le K_3 B_4(T^4_v, T^4_{\nu^\star_{\mathrm{BTMD}}(v)}).$$

For a real node matched to a graph-level blank tree, the padded representation cost is at most R_3, while Lemma A.5 gives tree cost at least ξ. Hence

$$\|z_v - 0\|_2 \le \frac{R_3}{\xi} B_4(T^4_v, T^4_\perp).$$

Define

$$\boxed{K_{\mathrm{arch}} := \max\left\{ K_3, \frac{R_3}{\xi} \right\}.} \tag{53}$$

Summing representation costs over the BTMD-optimal graph-level assignment yields

$$C_{\mathrm{rep}}(\nu^\star_{\mathrm{BTMD}}) \le K_{\mathrm{arch}}\ \mathrm{BTMD}^4(G_A, G_B).$$

Finally, D_{rep} is the minimum representation-assignment cost, so it cannot exceed the cost under this particular assignment:

$$D_{\mathrm{rep}}(G_A, G_B) \le C_{\mathrm{rep}}(\nu^\star_{\mathrm{BTMD}}) \le K_{\mathrm{arch}}\ \mathrm{BTMD}^4(G_A, G_B).$$

This proves Theorem 4.1.

B SETTINGS

Symbols. Table 2 lists the symbols that recur across sections, so that none has to be recovered from the equation it first appears in.

Table 2: Symbols used in Sections 3 to 3.4. LoRA as in Hu et al. (2022).

Symbol	Meaning
r, r_k^*	a pruning configuration; the one deployed by client k
$c(r)$, $c_{\mathrm{body}}(r)$, b	total and body per-token cost; the body budget
h_l, g_l, m_l	fractions of layer l retained: query heads, key-value groups, feed-forward width
q	the LoRA (Hu et al., 2022) rank, $q = 16$ (in Appendix A q is instead a head index of the source model)
B_k, A_k, s	the LoRA factors of client k and the adapter scale
$\mathcal{K}_k$, $\iota_k(j)$	coordinates client k retains; the position of coordinate j inside its matrix
$\mathcal{H}$, $\mathcal{H}(j)$	clients retaining a block; clients retaining coordinate j
π_θ, π_0	the RL policy and its fixed prior
Π, $\mathcal{N}$, J	the budget repair operator; the prunable nodes; the group size, $J = 8$
Rec, T_{rec}	the short recovery used inside the search, and its number of steps
ℓ_k^{prune}, ℓ_k^{rep}, $\hat{\ell}_k$	pruning loss, repaired loss, corrected proxy
ϕ, ψ_k, η	shared GNN, private modulation head, and its gate
n_k, $\mathcal{D}_k$, $\mathcal{V}_k$	client k's data share, training data and validation data

Every method shares the optimizer settings, so that comparisons are not confounded by tuning. Adapters are LoRA of rank 16 with $\alpha = 32$ and dropout 0.05 on the seven projections of every kept block, trained with AdamW, a linear schedule with warmup, and a learning rate selected once on validation data and then held fixed. Co-training runs ten rounds; after every aggregation the optimizer state is discarded and rebuilt, so that a client's momentum cannot carry across an exchange. The search uses 40 iterations for the main settings and 30 in the scale study, eight sampled pruning configurations per client per iteration, verification every eight iterations and again at the end, 320 training rows and 48 validation rows for the repair tranche, 100 repair steps per verification, and importance scores measured on 32 training rows of the client. The reference context length for Equation 1 is $C = 400$ tokens. The policy update is biased in two ways: the gradient is taken on the pruning configuration that was executed after projection onto the feasible set rather than on the sampled one, and the pruning configurations drawn from the prior enter with the policy's log-probability and without an importance weight. Kernels on the accelerator are not deterministic, so two runs that share every seed still diverge; we therefore fix the number of repeats of an arm before reading it, report every draw, and say in words whether the arms overlap.

Measures. SQuAD is scored by answer F1 under the standard normalization, AG News by the accuracy of the generated label, XSum and SAMSum by ROUGE-L, and the four-task federation by the test NLL (lower is better).

Partitions. A single-task setting has three clients holding 50%, 30% and 20% of 6000 training, 600 validation and 1500 test examples, with 750 test examples for SAMSum. The Dirichlet concentration is 0.5 and the mixing fraction with the global distribution is 10%. The partition key is the dominant entity type for CoNLL, the class label for AG News, and a TF-IDF topic cluster for SQuAD, XSum and SAMSum. The four-task federation gives each of four clients 2000 training, 200 validation and 500 test examples of one task. Generation metrics use 300 fixed test examples per client.

Matched computation, seeds and selection. A full backbone costs 1.0 per token irrespective of adapter rank, so at E epochs it matches the training FLOPs of the pooled reference and at E/c epochs it matches the number of training tokens. Teacher forward passes and the cost of the search are reported separately and are not folded into the cost of any method. The search is executed once per setting with seed 42, and final training and co-training are repeated over training seeds, which vary data order, adapter initialization and dropout, so the reported dispersion is over training seeds for a single search and excludes the variance of the search itself. Model selection uses validation data exclusively: the verified pruning configuration and the hyperparameters of trained baselines. The

structure choice is selected per task by the mean validation loss over clients, weighted by validation size.

C Baseline Fidelity

Table 3 records, for every published method, the setting its paper prescribes, the setting used here, and the reason for any difference. Architecture and procedure follow each paper; hyperparameters the paper leaves open are selected on validation data.

Table 3: Baseline fidelity: the setting each published method's paper prescribes and the setting used here. Methods: SLEB (Song et al., 2024), LLM-Pruner (Ma et al., 2023), Adapt-Pruner (Pan et al., 2025), EvoPress (Sieberling et al., 2025), Minitron (Muralidharan et al., 2024), FedPruner (Wu et al., 2025), GKD (Agarwal et al., 2024), FLoRA (Wang et al., 2024), LoRA (Hu et al., 2022).

Method	Paper setting	Setting here
SLEB (Song et al., 2024)	WikiText-2, 128 sequences of 2048 tokens	paper corpus, and client data
LLM-Pruner (Ma et al., 2023)	BookCorpus, 10 sequences of 128 tokens	paper corpus, and client data
Adapt-Pruner (Pan et al., 2025)	SlimPajama, 25600 sequences of 64 tokens	paper corpus, and client data
EvoPress (Sieberling et al., 2025)	FineWeb-Edu, 16 sequences of 8192 tokens	paper corpus, and client data
Minitron (Muralidharan et al., 2024)	SlimPajama, 1024 sequences of 4096 tokens	paper corpus, and client data
FedPruner (Wu et al., 2025)	per-round layer selection, dense deployment	as published
GKD (Agarwal et al., 2024) student	on-policy student samples	as published, FLOPs-matched student

D Extended Related Work

D.1 Structured Pruning and Allocation Search

Structured pruning removes whole components of a pretrained model so that the result runs faster without special kernels. LLM-Pruner (Ma et al., 2023) groups coupled structures and ranks them by gradient-based importance, SliceGPT (Ashkboos et al., 2024) deletes rows and columns after a rotation that concentrates the signal, and ShortGPT (Men et al., 2025) reports that whole layers can be dropped by a redundancy score. SLEB (Song et al., 2024) removes transformer blocks one at a time, re-measuring perplexity after each removal, and Adapt-Pruner (Pan et al., 2025) interleaves small pruning steps with recovery training. Minitron (Muralidharan et al., 2024) searches over heads, neurons, embedding channels and depth and distils the result, while EvoPress (Sieberling et al., 2025) runs an evolutionary search over which blocks to remove under a fixed budget. DarwinLM (Tang et al., 2026) makes the same point from the operator side, reporting that how a component is removed and compensated can matter more than which component is chosen.

A second line learns the allocation instead of scoring it. GNN-RL (Yu et al., 2022) represents the network as a graph and searches compression ratios with reinforcement learning (RL) under a resource constraint, and Bypass Back-propagation (Gao et al., 2025) learns layer-wise ratios without gradients through the pruned model. Cus-Prun (Zhao et al., 2025) produces per-task expert models from one source model by removing task-irrelevant neurons. Two studies from 2026 give the reason our controls exist: Kim et al. (2026) find that seven depth-pruning search algorithms land within a few points of each other once the calibration objective is fixed, so the objective matters more than the search, and Moayedikia & Troncoso Lora (2026) find random capacity allocation equal to heterogeneity-aware allocation at a matched budget in sub-model federated learning. This paper differs from all of these in what the structure is for: the model is decomposed once into several sub-networks at the same time, one per client, each chosen from that client's own data under the same per-token budget, and the sub-networks then train together.

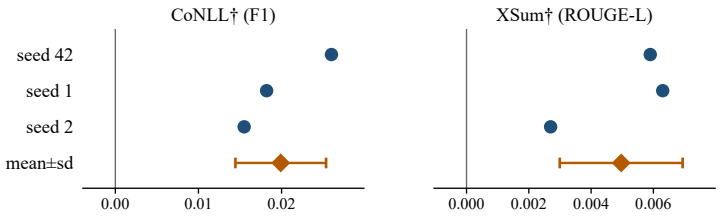


Figure 3: Structure-sharing difference at each training seed (Table 5 values): the selected sharing arm minus per-client structures at seeds 42, 1 and 2, with their mean±sd; †: significant (paired t-test, $p < 0.05$).

D.2 Federated Learning with Heterogeneous Sub-models

Federated learning (McMahan et al., 2017) assumes every client trains the same architecture, which fails when clients cannot run it. HeteroFL (Diao et al., 2021) gives clients nested width slices of one model and averages every parameter over the clients that hold it, FedRolex (Alam et al., 2022) rolls the slice so that all parameters are trained over time, DepthFL (Kim et al., 2023) assigns depth prefixes with early exits, and FedRA (Su et al., 2024) samples a random subset of layers for each client in each round. In these methods a rule fixes the sub-model from the device capacity rather than from the data. Hermes (Li et al., 2021) and FIARSE (Wu et al., 2024) do derive a personalized sub-network from the model and the client's own data, and the server averages only the parameters that the sub-networks share, so averaging over the holders of a parameter is established and we claim no novelty for it. For LLMs, FedSpaLLM (Bai et al., 2025) aggregates over retained entries when clients prune locally, FedPrLLM (Guo et al., 2025) compares pruning and aggregation choices in one-shot federated pruning, FedSpine (Yao et al., 2025) couples pruning with parameter-efficient tuning, and FedPruner (Wu et al., 2025) builds per-client sub-networks with a neural bandit and groups sublayers by centered kernel alignment. PPC-GPT (Fan et al., 2025) prunes with synthetic data on the server side. DiPaCo (Douillard et al., 2024) is the closest precedent for the deployment picture: paths through shared modules are trained on routed data shards, each module is averaged over the paths that use it (the outer step of Douillard et al., 2023), and one path is executed per input, so the whole model is never materialized. What this paper adds is the combination that none of them has: sub-networks that differ in structure, are derived from client data, are held to a hard per-token FLOPs budget rather than a parameter count, and are aggregated across those different structures in the coordinates of the model they came from.

D.3 Federated Adaptation, Merging and Distillation

When the backbone stays fixed, clients can exchange adapters instead of weights. FedIT (Zhang et al., 2023) and OpenFedLLM (Ye et al., 2024) average LoRA (Hu et al., 2022) factors across clients, HetLoRA (Cho et al., 2024) and FlexLoRA (Bai et al., 2024) let clients hold different ranks and reconcile them at the server, and FedP2EFT (Lee et al., 2026) learns where rank should be spent per client. FLoRA (Wang et al., 2024) observes that averaging the factors of a low-rank update does not average the updates themselves and stacks them instead, an objection that applies to factor averaging here as well and that we answer with a stacking variant. Merging offers a different route: task arithmetic (Ilharco et al., 2023), TIES (Yadav et al., 2023) and DARE (Yu et al., 2024) combine models that share an architecture, and Branch-Train-Merge (Li et al., 2022) and Branch-Train-MiX (Sukhbaatar et al., 2024) train experts on data shards and then merge or route between them. Distillation moves capability instead of parameters, either from a larger teacher with on-policy student samples (Agarwal et al., 2024; Gu et al., 2024) or, in federated settings, through public data. All of these keep one architecture across the participants. Here the participants differ by construction, which is what forces aggregation to be defined in the source model's coordinates and what the shared GNN in Section 3.4 is for.

Table 4: Co-training and alignment in series A, at matched FLOPs (cost 0.68–0.71). FedBUD first, then its ablations, then the differences. Independent (FedBUD without co-training) and FedBUD (searched structures co-trained with Equation 10) are mean±sd over training seeds 42, 1 and 2; the heuristic columns replace the learned allocator by a hand-designed rule and are single runs at seed 42; shape-only (FedBUD without alignment) is the mean over its repeated runs of seed 42 (±sd where there are two). FedBUD − Indep. is taken at the same seed, mean±sd over the three seeds; FedBUD − Shape is the seed-42 FedBUD run minus shape-only. Positive means FedBUD is better; for the multi-task NLL the sign is reversed. **Bold**: best mean in the row. †: FedBUD significantly better than independent adaptation (paired two-sided t-test, $p < 0.05$). Datasets: CoNLL (Tjong Kim Sang & De Meulder, 2003), SQuAD (Rajpurkar et al., 2016), AG News (Zhang et al., 2015), XSum (Narayan et al., 2018), SAMSum (Gliwa et al., 2019).

Dataset	Metric	FedBUD	Independent	Heur. mlp	Heur. width	Shape-only	FedBUD − Indep.	FedBUD − Shape
CoNLL (Tjong Kim Sang & De Meulder, 2003)	F1	0.8398±0.0038	0.8233±0.0099	0.8657	**0.8672**	0.8302	+0.0165±0.0078	+0.0134
SQuAD (Rajpurkar et al., 2016)	F1	0.7269±0.0034	0.6336±0.0066	**0.7487**	0.7036	0.7045±0.0003	+0.0933±0.0050†	+0.0203
AG News (Zhang et al., 2015)	Acc.	**0.9448±0.0028**	0.9315±0.0101	0.9367	0.9411	0.9411	+0.0133±0.0084	+0.0033
XSum (Narayan et al., 2018)	ROUGE-L	0.2311±0.0007	0.2241±0.0034	**0.2334**	0.2323	0.2293	+0.0070±0.0039	+0.0020
SAMSum (Gliwa et al., 2019)	ROUGE-L	0.3808±0.0017	0.3660±0.0046	**0.3849**	0.3834	0.3694	+0.0147±0.0029†	+0.0115
Multi-task	NLL↓	0.9697±0.0034	1.1217±0.0012	0.9771	0.9947	**0.9591**	+0.1520±0.0024†	−0.0115

Table 5: Structure sharing at two further training seeds (seeds 1 and 2) in series A. On each task the arm is the one selected by the mean validation loss over clients, weighted by validation size, which selects the same arm as at seed 42 at every seed; scores are F1 on CoNLL and ROUGE-L on XSum, against per-client structures at the same seed. Over seeds 42, 1 and 2 the difference is $+0.0199 \pm 0.0055$ on CoNLL and $+0.0050 \pm 0.0020$ on XSum, both significant (paired two-sided t-test, $p < 0.05$). Datasets: CoNLL (Tjong Kim Sang & De Meulder, 2003), XSum (Narayan et al., 2018).

Task	Selected arm	Seed	FedBUD (selected)	FedBUD (per-client)	Difference
CoNLL (Tjong Kim Sang & De Meulder, 2003)	shared structure	1	0.8542	0.8360	+0.0182
CoNLL	shared structure	2	0.8554	0.8399	+0.0155
XSum (Narayan et al., 2018)	pooled importance	1	0.2366	0.2303	+0.0063
XSum	pooled importance	2	0.2344	0.2317	+0.0027

E Additional Tables

A second budget. At $b = 0.5$ (cost 0.63) on SQuAD in series B, co-training reads $+0.1035$ F1 over independent adaptation (0.6427 against 0.5392; single runs), against $+0.0841$ at $b = 0.6$; lowering the budget from 0.6 to 0.5 costs the co-trained specialists 0.0879 F1.

F Supplementary Settings

Single-run explorations at other scales and in other settings are not reported.

G Repair Corrector

Every allocation method in this paper shares the operator and the repair and differs only in the pruning configuration it produces: the learned allocator, the hand-designed rules, random structures and published pruners.

Table 6: Published structured-pruning methods in their published form (their structure and their own training recipe, without the co-training of FedBUD), each calibrated on its paper's corpus and on each client's data; recovery variants are listed as separate rows. Scores use the metrics of Section 5; the multi-task column is the test NLL (lower is better). The two rows before FedBUD are the federated pruning method FedPruner (Wu et al., 2025) and a GKD student distilled into a sub-network of matched cost; n/a marks a dataset on which the method was not run. All baseline cells are single runs at seed 42, outside our two series, so no difference against them is tested; the last row is FedBUD in series A at seed 42. c is the mean per-token cost over clients and datasets. Datasets: CoNLL (Tjong Kim Sang & De Meulder, 2003), SQuAD (Rajpurkar et al., 2016), AG News (Zhang et al., 2015), XSum (Narayan et al., 2018), SAMSum (Gliwa et al., 2019). Methods: SLEB (Song et al., 2024), LLM-Pruner (Ma et al., 2023), Adapt-Pruner (Pan et al., 2025) with its accelerated variant Adapt-Accel, EvoPress (Sieberling et al., 2025), Minitron (Muralidharan et al., 2024), GKD (Agarwal et al., 2024).

Method	Calibration	c	CoNLL (Tjong Kim Sang & De Meulder, 2003)	SQuAD (Rajpurkar et al., 2016)	AG News (Zhang et al., 2015)	XSum (Narayan et al., 2018)	SAMSum (Gliwa et al., 2019)	Multi-task (NLL↓)
SLEB (Song et al., 2024)	paper corpus	0.694	0.8185	0.4300	0.9345	0.2104	0.3288	1.2191
SLEB	client data	0.694	0.8113	0.4121	0.9289	0.2098	0.3274	1.2504
LLM-Pruner (Ma et al., 2023)	paper corpus	0.692	0.8077	0.3803	0.9389	0.2069	0.3302	1.3834
LLM-Pruner	client data	0.692	0.8006	0.4399	0.9400	0.2136	0.3423	1.3250
LLM-Pruner + recovery	paper corpus	0.692	0.7772	0.3462	0.9344	0.2047	0.3221	1.3188
LLM-Pruner + recovery	client data	0.692	0.7914	0.4148	0.9411	0.2114	0.3378	1.2270
Adapt-Pruner (Pan et al., 2025)	paper corpus	0.699	0.8413	0.6266	0.9367	0.2213	0.3599	1.1504
Adapt-Pruner	client data	0.696–0.698	0.8314	0.6069	0.9356	0.2241	0.3611	1.1982
Adapt-Accel	paper corpus	0.703	0.8159	0.5453	0.9211	0.2145	0.3558	1.2491
Adapt-Accel	client data	0.703	0.8254	0.5972	0.9222	0.2243	0.3633	1.2084
EvoPress (Sieberling et al., 2025)	paper corpus	0.694	0.7958	0.4120	0.9367	0.2106	0.3215	1.3050
EvoPress	client data	0.694	0.8130	0.4291	0.9411	0.2095	0.3363	1.2559
Minitron (Muralidharan et al., 2024)	paper corpus	0.688–0.699	0.7913	0.3923	0.9144	0.2094	0.3264	1.2581
Minitron	client data	0.680–0.699	0.6549	0.4079	0.9133	0.2123	0.3376	1.2812
Minitron + distillation	paper corpus	0.688–0.699	0.6728	0.4013	0.9245	0.2189	0.3393	1.3707
Minitron + distillation	client data	0.680–0.699	0.8033	0.4142	0.9067	0.2214	0.3481	1.4523
FedPruner (Wu et al., 2025)	client data	0.684–0.693	0.7720	0.4060	n/a	0.2021	n/a	n/a
GKD (Agarwal et al., 2024) student	client data	0.688–0.706	0.7604	0.4658	0.9278	0.2191	0.3416	2.3333
FedBUD	client data	0.688–0.706	0.8436	0.7248	0.9444	0.2313	0.3809	0.9706

The corrector is a ridge regression of the logarithm of the ratio between the repaired loss and the pruning loss on features of the pruning configuration r, namely the logarithm of the pruning loss $\ell_k^{\text{prune}}(r)$, the cost, the number of actions of each kind and the client identity, and the prediction is formed as in Equation 5, where $\rho_{\min}$ and $\rho_{\max}$ are the extreme log ratios among the pairs verified so far, so a prediction stays inside the range of repairs that have actually been measured.

H Shared GNN On and Off

The optional shared GNN of Section 3.4, instantiated as a two-layer GAT, is added to aligned co-training at seed 42 in both series (Table 8). The change ranges from -0.0161 to $+0.0095$; with a single run per arm the comparison is descriptive, and the results of Section 5 use aligned averaging alone.

Table 7: Additional comparisons quoted in Section 5, seed 42 unless stated. Differences are signed so that positive favors the first named arm; for the multi-task NLL the sign is reversed. All entries are single comparisons reported without a test; published arms and baseline structures are single runs outside both series, so those differences are indicative.

Comparison	Series	Dataset	Difference
FedBUD − shape-only	B	SAMSum (Gliwa et al., 2019)	+0.0179
FedBUD − shape-only	B	Multi-task	−0.0198
Random − FedBUD, cost-matched	B	SQuAD (Rajpurkar et al., 2016)	−0.0784
Random − FedBUD, cost-matched	B	XSum (Narayan et al., 2018)	−0.0120
Random − FedBUD	A	SQuAD	−0.0661
Random − FedBUD	A	XSum	−0.0063
Random − FedBUD	A	CoNLL (Tjong Kim Sang & De Meulder, 2003)	+0.0099
Better rule − FedBUD	A	CoNLL	+0.0236
Better rule − FedBUD	A	SQuAD	+0.0239
Better rule − FedBUD, seed 43	B	CoNLL	+0.0467
FedBUD shared − per-client	A	CoNLL	+0.0260
FedBUD other client's − per-client	A	SQuAD	−0.0047
FedBUD pooled − per-client	A	SQuAD	+0.0225
FedBUD pooled − per-client	A	XSum	+0.0059
FedBUD − best published arm	A	CoNLL	+0.0023
FedBUD − best published arm	A	SQuAD	+0.0982
FedBUD − best published arm	A	AG News (Zhang et al., 2015)	+0.0033
FedBUD − best published arm	A	XSum	+0.0070
FedBUD − best published arm	A	SAMSum	+0.0176
FedBUD − best published arm	A	Multi-task	+0.1798

Table 8: FedBUD with the shared GNN in its GAT instantiation (+GNN) and without it (FedBUD, the reported configuration), seed 42, in series A and B. Diff. is +GNN minus FedBUD, signed so that positive favors the network; for the multi-task NLL the sign is reversed. Single runs.

		Series A			Series B		
Dataset	Metric	FedBUD	+GNN	Diff.	FedBUD	+GNN	Diff.
CoNLL (Tjong Kim Sang & De Meulder, 2003)	F1	0.8436	0.8388	−0.0048	0.8484	0.8323	−0.0161
SQuAD (Rajpurkar et al., 2016)	F1	0.7248	0.7343	+0.0095	0.7306	0.7322	+0.0016
AG News (Zhang et al., 2015)	Acc.	0.9444	0.9422	−0.0022	0.9456	0.9445	−0.0011
XSum (Narayan et al., 2018)	ROUGE-L	0.2313	0.2293	−0.0020	0.2335	0.2349	+0.0014
SAMSum (Gliwa et al., 2019)	ROUGE-L	0.3809	0.3816	+0.0007	0.3833	0.3847	+0.0014
Multi-task	NLL↓	0.9706	0.9701	+0.0005	0.9696	0.9627	+0.0069