SELECT, THEN TRAIN: EFFICIENT TRANSFORMER ADAPTATION VIA NEURON SELECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

As pretrained transformers continue to scale, reducing both training and inference cost is critical for efficient downstream adaptation. On one hand, parameter-efficient fine-tuning methods reduce training overhead, but they generally do not reduce inference cost. On the other, most pruning methods focus on preserving task-agnostic pretrained behavior rather than post-adaptation performance. We study structured pruning (i.e., pruning neurons, not just weights) for downstream adaptation and show that simple activation statistics measured on a small unlabeled target-task calibration set provide an effective signal for identifying task-relevant neurons. In particular, we observe strong layer-wise complementarity between activation magnitude and activation frequency. Based on this observation, we propose Select, Then Train (STT), a lightweight select–then–train framework that removes under-utilized Feedforward Neural Network (FFN) neurons prior to adaptation, producing compact subnetworks optimized for task-specific performance and efficient inference. Across multiple Vision Transformers and Large Language Models, STT achieves better accuracy–efficiency trade-offs than magnitude-based and weight–activation pruning baselines, yielding average inference speedups of approximately 1.2× on LLMs and 1.7× on Vision Transformers across diverse configurations. These results demonstrate that activation-driven structured pruning is an effective and practical approach for task-specialized transformer adaptation.

1 INTRODUCTION

Pretrained transformers have become the backbone in many computer vision (Dosovitskiy et al., 2021; Siméoni et al., 2025; Radford et al., 2021) and natural language processing (Radford & Narasimhan, 2018; Brown et al., 2020; OpenAI et al., 2024) applications, offering strong zero-shot generalization. Yet to reach optimal performance on a specific downstream task, adaptation by *fine-tuning* remains essential. Full fine-tuning (FFT) updates all parameters and often yields the best performance, but its compute and memory costs can be prohibitive. Further, the resulting model might be much larger and more expensive than necessary to serve in production. Alternatively, parameter-efficient fine-tuning (PEFT) methods, such as LoRA (Hu et al., 2022) and P-Tuning (Liu et al., 2022), are more efficient in fine-tuning by reducing the number of trainable parameters. Yet they leave the underlying model architecture unchanged, so inference remains as expensive as in the original model. In this work, our goal is to make inference faster while achieving the benefits of fine-tuning.

Current pruning methods for large pretrained transformers often aim to increase the sparsity of the weights while preserving strong zero-shot performance across many tasks (Frantar & Alistarh, 2023; Ma et al., 2023; Sun et al., 2024). For example, Wanda (Sun et al., 2024) prunes LLMs based on pretrained weights multipled with activations collected from a *generic* calibration corpus, preserving the model's task agnostic zero-shot capabilities rather than optimizing for a specific downstream task. Such *sparse pruning* methods are not well-suited to our goal of producing lighter-weight, specialized models for two reasons. First, most hardware for accelerating neural networks, including GPUs and FPGAs, do not exploit arbitrary sparsity in the weights for faster inference (Gale et al., 2019). Second, many applications require models dedicated to specific tasks, but these pruning methods do not target a particular task. Instead, we want to identify a subnetwork that is good for *task-specific fine-tuning*.

To fill this gap, we study *structured pruning* for downstream adaptation. In structured pruning, we prune units larger than individual weights, such as entire neurons, that result in acutal inference

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.

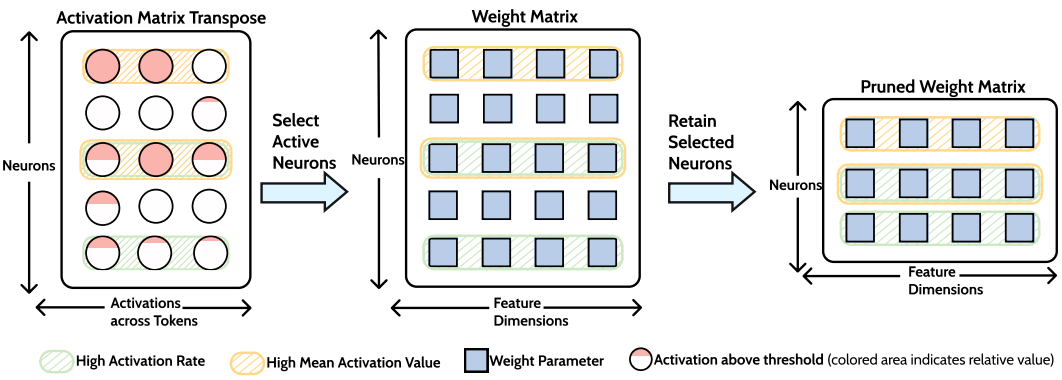


Figure 1: **Select, Then Train (STT).** From unlabeled target-task inputs, we measure how strongly and how often each neuron in a feedforward neural network (indexed by row) is activated. Neurons that are either consistently strong or frequently active are retained for task-specific adaptation.

speedups. We propose **Select, Then Train (STT)**, a simple framework that identifies and removes *under-utilized* feedforward neural network (FFN) neurons under the target data distribution *before adaptation*, producing compact subnetworks optimized for *task-specific performance* and *efficient inference*. Concretely, STT (Figure 1) scores neurons with two complementary token-level statistics on target-task inputs: the *mean activation value*, which measures how strongly a neuron activates on average, and the *activation rate*, which measures how often it is activated as a fraction of tokens (text tokens for LLMs and patch tokens for ViTs). This selection yields a compact, task-specialized subnetwork that reduces both training and *inference costs*.

In practice, STT introduces one simple hyperparamter, the retention ratio k, which specifies the fraction of neurons retained in each FFN block and directly controls training and inference FLOPs, as well as accuracy of the resulting fine-tuned model. STT also requires a *calibration subset*: a small, *unlabeled* set of target-task inputs used only to estimate token-level activation statistics. Neuron rankings are computed from a single forward pass over this subset, and we quantify the cost of this selection stage and show that it contributes only a small overhead compared to the training and inference savings obtained by specializing FFN modules in Section 4.3. We also find the resulting selection to be insensitive to the subset size in a wide range (Appendix A.2).

We evaluate STT through comprehensive experiments across diverse tasks in Section 4 and STT demonstrates compelling empirical performance in Section 4.2. Across all sparsity levels and across both decoder-only Large Language Models (LLMs) and encoder-only Vision Transformers (ViTs), STT consistently improves the accuracy–efficiency trade-off over strong sparse baselines and remains robust in the high-sparsity regime.

Our contributions are:

1. We introduce **STT**, a select-then-train framework that uses a single forward pass on an unlabeled calibration subset of target-task inputs to select and structurally prune FFN neurons, producing a narrower dense subnetwork for downstream adaptation.
2. Across decoder-only LLMs and encoder-only ViTs, STT consistently improves accuracy-efficiency trade-offs over strong sparse baselines, with reliable trends across sparsity levels.
3. We quantify the computational cost of the selection stage and show that its overhead is small relative to the training/inference savings from specializing the FFN layers.

2 Related Work

In this section, we provide an overview of prior research that closely relates to our work.

Pruning. Pruning reduces model size by removing parameters. Early work on CNN compression studied both *weight-based* criteria, such as pruning small-magnitude weights (Han et al., 2016), and

activation-based criteria that remove channels or neurons based on activation statistics (He et al., 2017; Molchanov et al., 2017). For example, APoZ (Hu et al., 2016) identifies persistently inactive neurons using the frequency of ReLU-induced zero outputs. Such exact-zero criteria are less applicable to modern Transformer FFNs with smooth or gated activations such as GELU, QuickGELU, GeGLU, and SwiGLU, where intermediate activations are unlikely to be exactly zero.

Recent pruning methods for pretrained transformers often focus on preserving task-agnostic pretrained behavior or zero-shot performance. SparseGPT (Frantar & Alistarh, 2023) and Wanda (Sun et al., 2024), for example, prune pretrained weights using weight- and activation-dependent criteria evaluated on generic calibration data. Pruning individual weights generally does not reduce computation costs significantly because standard accelerator hardware like GPUs do not exploit that sparsity.

To achieve practical efficiency gains, *structured pruning* methods remove larger architectural units such as FFN neurons, attention heads, or hidden dimensions. However, existing approaches often rely on strong architecture-specific assumptions or rely on gradient information to estimate importance of specific architecture: LLM-Pruner (Ma et al., 2023) traces dependencies between the vocabulary embedding and language modeling head in decoder-only LLMs using gradient estimates; ToaSt (Moon et al., 2026) relies on ViT-specific token structures like CLS-to-patch attention; LoRAPrune (Zhang et al., 2024) uses LoRA gradients during fine-tuning to prune weights and adapters simultaneously; Bonsai (Kolawole et al., 2024) evaluates sampled subnetworks;

STT considers a different adaptation protocol. STT adopts an architecture-minimal criterion operating purely on activation statistics prior to parameter updates. By requiring only target-task calibration activations and FFN intermediate dimensions, structures shared across virtually all transformer variants, STT selects neurons for downstream adaptation across both encoder-only and decoder-only models.

Parameter-efficient fine-tuning (PEFT). PEFT reduces training cost by updating only a small subset of parameters, through adapters (Houlsby et al., 2019; He et al., 2022), prompts or prefixes (Li & Liang, 2021; Lester et al., 2021), low-rank updates (Hu et al., 2022; Dettmers et al., 2023), or selective parameter updates (Ben Zaken et al., 2022; Guo et al., 2020). While PEFT is effective for reducing *training* cost, it typically leaves the model architecture unchanged or even expanded, and thus does not reduce *inference* cost. STT is complementary to PEFT: it shrinks FFN width to improve inference efficiency, and can be combined with PEFT methods when desired (Appendix A.5).

3 METHOD: SELECT, THEN TRAINING

We propose **Select, Then Train (STT)**, a *select–then–train* framework for adapting pretrained transformers to downstream tasks efficiently via structured FFN width reduction.

3.1 STRUCTURED PRUNING FOR FFN MODULES

Denote the model hidden dimension as d, the intermediate dimension of the FFN blocks as d_{ffn}, and the token length as n. Let $X \in \mathbb{R}^{n \times d}$ denote the input to a feed-forward network (FFN) block.

A standard two-layer FFN used in BERT and ViT (Devlin et al., 2019; Dosovitskiy et al., 2021) is defined as

$$F_{\text{ffn}}(X) = \sigma(XW_1)W_2, \tag{1}$$

where $W_1 \in \mathbb{R}^{d \times d_{\text{ffn}}}$ and $W_2 \in \mathbb{R}^{d_{\text{ffn}} \times d}$, and $\sigma(\cdot)$ is a non-linear activation function. Each index $j \in \{1, \ldots, d_{\text{ffn}}\}$ corresponds to one intermediate FFN neuron.

The goal of FFN structured pruning is to reduce the intermediate dimension from d_{ffn} to t_{ffn} by removing entire intermediate neurons. This can be expressed by selecting a subset of neuron indices $I_\ell \subseteq \{1, \ldots, d_{\text{ffn}}\}$ with $|I| = t_{\text{ffn}}$.

For a standard FFN, pruning corresponds to constructing

$$W_1' = W_1[:, I], \qquad W_2' = W_2[I, :], \tag{2}$$

where $W[:, I]$ (resp., $W[I, :]$) denotes the submatrix formed by retaining all rows and the columns (resp., rows) indexed by I.

Many modern LLMs employ gated variants of FFNs (Shazeer, 2020). In this case, the FFN block is defined as

$$F_{\text{ffn}}(X) = \big(\sigma(XW_g) \odot (XW_u)\big)W_d, \tag{3}$$

where $W_g, W_u \in \mathbb{R}^{d \times d_{\text{ffn}}}$ are the gate and up projections, respectively, $W_d \in \mathbb{R}^{d_{\text{ffn}} \times d}$ is the down projection, and $\odot$ denotes element-wise multiplication. The intermediate dimension d_{ffn} is shared across all projections.

For a gated FFN, the same index set I is applied consistently across all projections:

$$W_g' = W_g[:, I], \quad W_u' = W_u[:, I], \quad W_d' = W_d[I, :]. \tag{4}$$

In particular, the down-projection layer W_d is not independently pruned; its input dimension is reduced solely to align with the selected intermediate neurons, while its output dimension remains unchanged.

This structured pruning reduces the FFN intermediate dimension from d_{ffn} to t_{ffn} while preserving the model hidden dimension d. STT performs this structured pruning independently in every FFN block ℓ.

3.2 SELECTION

Algorithm 1 Select, Then Train (STT)

1: **Input:** pretrained parameters θ, calibration set $\mathcal{S}$, retention ratio k, threshold δ
2: **for** each FFN block ℓ **do**
3: Collect token-level activations $A_\ell \in \mathbb{R}^{N \times d_{\text{ffn}}}$ from the first FFN projection for standard FFNs or the gate projection for gated FFNs using one forward pass on $\mathcal{S}$
4: $M_\ell \leftarrow \frac{1}{N}\sum_{i=1}^{N} A_\ell[i,:]$ {mean activation value}
5: $R_\ell \leftarrow \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}(A_\ell[i,:] > \delta)$ {activation rate}
6: $I_\ell \leftarrow \text{TopK}(M_\ell, k) \cup \text{TopK}(R_\ell, k)$
7: Apply I_ℓ to prune the FFN intermediate dimension: retain selected columns in W_1 for standard FFNs, retain selected columns in W_g and W_u for gated FFNs, and retain selected rows in the output/down projection
8: **end for**
9: Freeze parameters except the retained FFN subnetworks and the task head; fine-tune to obtain θ'
10: **return** θ'

To guide structured pruning, STT (Algorithm 1) relies on activations measured under the target task distribution. Given an unlabeled calibration set $\mathcal{S}$ sampled from the downstream task, we collect token-level activations from a single forward pass of the pretrained model. For standard FFNs, we use the output of the first FFN projection, We flatten activations across all calibration examples and token or image-patch positions, such that $A_\ell \in \mathbb{R}^{N \times d_{\text{ffn}}}$, where N is the total number of token or image-patch activations and i indexes one such activation. For gated FFNs, which are used in our evaluated Qwen2.5 decoder-only LLMs, we collect activations from the gate projection,

$$A_\ell = XW_g,$$

Thus, column j of A_ℓ corresponds directly to the intermediate neuron j pruned from the associated FFN projections in Section 3.1.[1]. We perform neuron selection using two complementary activation-only statistics computed from A_ℓ.

For each intermediate neuron j, we define

$$M_{\ell,j} = \frac{1}{N}\sum_{i=1}^{N} A_{\ell,ij}, \tag{5}$$

$$R_{\ell,j} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}(A_{\ell,ij} > \delta)\,, \tag{6}$$

[1]Appendix A.8 ablates gate-projection versus product activations for gated FFNs, and Appendix A.12 compares Top-K union with alternative combination rules.

where $M_{\ell,j}$ is the *mean activation value* and $R_{\ell,j}$ is the *activation rate*, i.e., the fraction of tokens for which the neuron is activated above a threshold δ. We fix $\delta = 0.01$ in all experiments following prior work (Dohare et al., 2024; 2023) and Appendix A.7 further shows that the selected subnetworks are stable across a range of activation thresholds. Let $M_\ell, R_\ell \in \mathbb{R}^{d_{\text{ffn}}}$ denote the vectors of $(M_{\ell,j})_j$ and $(R_{\ell,j})_j$, respectively. Given a retention ratio $k \in (0, 1]$, we retain $t_{\text{ffn}} = \lceil k\, d_{\text{ffn}} \rceil$ neurons per metric in each FFN block. Let $\text{TopK}(v, k)$ return the indices of the t_{ffn} largest entries of v. STT forms the retained neuron index set by taking the union

$$I_\ell = \text{TopK}(M_\ell, k) \;\cup\; \text{TopK}(R_\ell, k). \tag{7}$$

We then perform structured pruning as defined in Section 3.1. Because the two TopK sets may overlap, the realized retained width satisfies $|I_\ell| \in [t_{\text{ffn}}, 2t_{\text{ffn}}]$.[2]

4 Experiments

We conduct a comprehensive evaluation of the proposed STT framework to answer the following research questions:

1. **RQ1:** Can STT select a trainable subnetwork that, after fine-tuning, matches full fine-tuning? For the same reduction in parameters, which selection signal yields better downstream adaptation: activation-based scores, or baselines like weight magnitude or weight×activation?
2. **RQ2:** Is neuron selection sufficiently lightweight, and do the efficiency gains it enables justify its computational cost?
3. **RQ3**: Given that we find activation-scores work better, does combining mean activation value with activation rate outperform either statistic alone?

We describe the experimental setup in Section 4.1 and answer these RQs in Sections 4.2–4.4.

4.1 Experimental Setup

Models. We evaluate STT on both decoder-only and encoder-only transformer architectures. For decoder-only transformers, we use the Qwen2.5-Base family (Yang et al., 2024) at 1.5B, 3B, and 7B parameter scales. We also evaluate STT on BERT family models (Devlin et al., 2019; Warner et al., 2025) and report results in Appendix A.4. For encoder-only transformers, we use four widely adopted ViTs (Dosovitskiy et al., 2021) at different scales and patch configurations: CLIP ViT-L/14 and CLIP ViT-B/32 (Radford et al., 2021), and SigLIP-ViT-B/16 and SigLIP-ViT-L/16 (Zhai et al., 2023).

Downstream Tasks. For LLMs, we use three widely adopted question-answering benchmarks that probe different aspects of language understanding. **BoolQ** (Clark et al., 2019) is a yes/no reading comprehension task where models must perform entailment inference over a provided passage. **ARC** (Clark et al., 2018) consists of science multiple-choice questions requiring factual knowledge and reasoning, while **OBQA** (Mihaylov et al., 2018) requires multi-hop reasoning that combines a provided science fact with unstated common knowledge to answer multiple-choice questions.

For ViTs, we select tasks on which zero-shot transfer is weak, indicating that pretrained representations require task-specific adaptation for reliable deployment. We use **EuroSAT** (Helber et al., 2018), a 10-class land-use classification task over Sentinel-2 satellite imagery; **GTSRB** (Stallkamp et al., 2011), a 43-class traffic-sign recognition benchmark; and **CLEVR-Count** (Johnson et al., 2017), a synthetic object-counting task. These datasets cover specialized visual domains or compositional reasoning settings that are poorly handled by zero-shot CLIP: CLIP ViT-B/32 achieves 49.4%, 32.2%, and 17.8%, respectively (Radford et al., 2021).

Comparison protocol. All baseline methods are evaluated under a matched structured-pruning budget. For each FFN block ℓ, after STT forms its retained index set I_ℓ, each sparse baseline is constrained to retain exactly $|I_\ell|$ neurons in the same block. Thus, sparse methods compared at the same operating point have the same realized per-layer FFN width and inference parameter count; they differ only in the neuron-selection criterion. All sparse methods use the same fine-tuning and evaluation protocol for each model–task pair. Implementation details are provided in Appendix A.2.

[2] All parameter counts and baseline comparisons use this realized per-layer width.

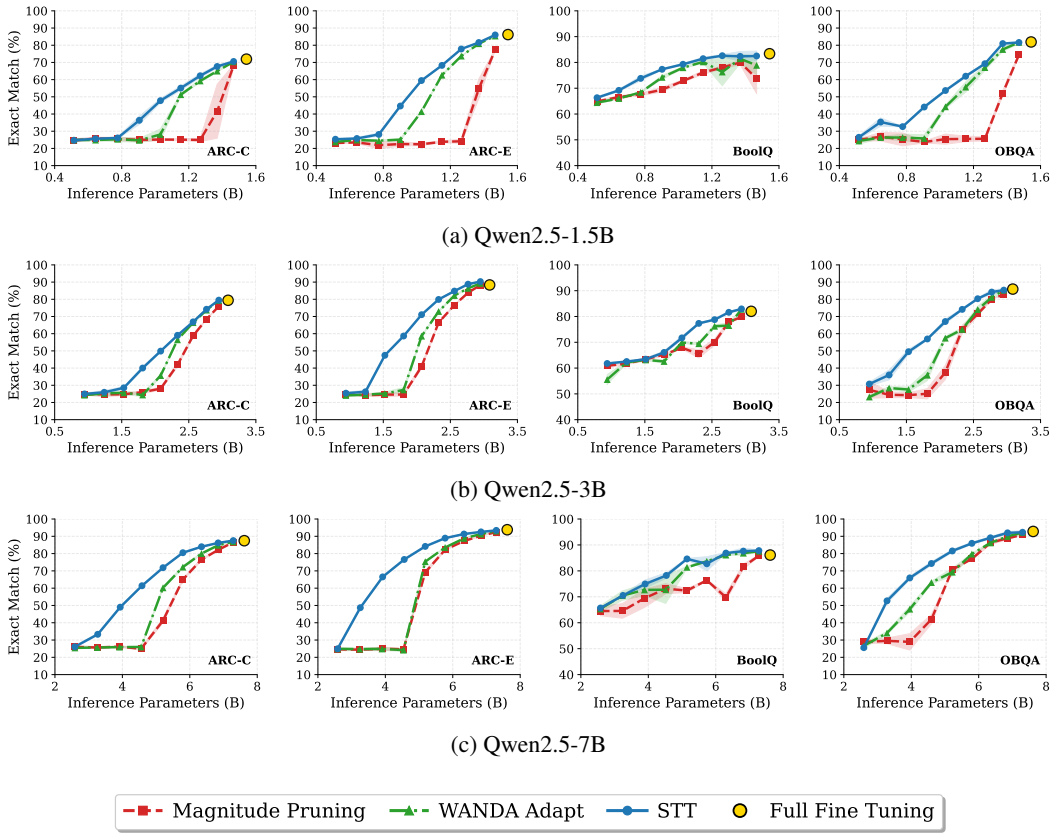


Figure 2: **Main results on LLMs.** STT shows improved performance-efficiency trade-off curves over baseline methods. The x-axis shows inference parameter count after pruning (in billions), and the y-axis shows performance. Each curve sweeps retention ratio $k \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$. The yellow marker denotes full fine-tuning. Shaded regions denote 95% confidence intervals across 5 random seeds.

4.2 Main Results

We answer **RQ1** by comparing STT against three baselines: **Magnitude Pruning** ranks hidden units j by the column-wise ℓ_1 norm of the first FFN projection matrix, $s_j = \|W_{[:,j]}\|_1$; **Wanda Adapt** is a structured adaptation of WANDA (Sun et al., 2024) that scores neurons using a weight-activation criterion from calibration data; and **Full Fine-Tuning** updates all backbone parameters without pruning, serving as an upper-bound performance reference. For completeness, we also considered a "train, then select" baseline that prunes using the same approach as STT after FFT. It did not perform well, likely because of pruning neurons without further training is too destructive (Appendix A.10).

Decoder-Only Transformers. As shown in Figure 2, across Qwen2.5-1.5B, 3B, and 7B, STT provides a stronger accuracy–efficiency trade-off than the sparse baselines. On ARC-C and ARC-E, STT recovers performance more smoothly as the retained FFN width increases. To show STT generalizes beyond Qwen-family models, we evaluate STT on Gemma3-1B (Team et al., 2025) and show STT again achieves the overall strongest performance-efficiency trade-off (Appendix A.11).

Encoder-Only Transformers. As shown in Figure 3, STT improves the performance-efficiency trade-off across the evaluated ViT architectures and datasets. The advantage is most visible in the high sparsity regime, where the baseline selectors often suffer larger performance degradation.

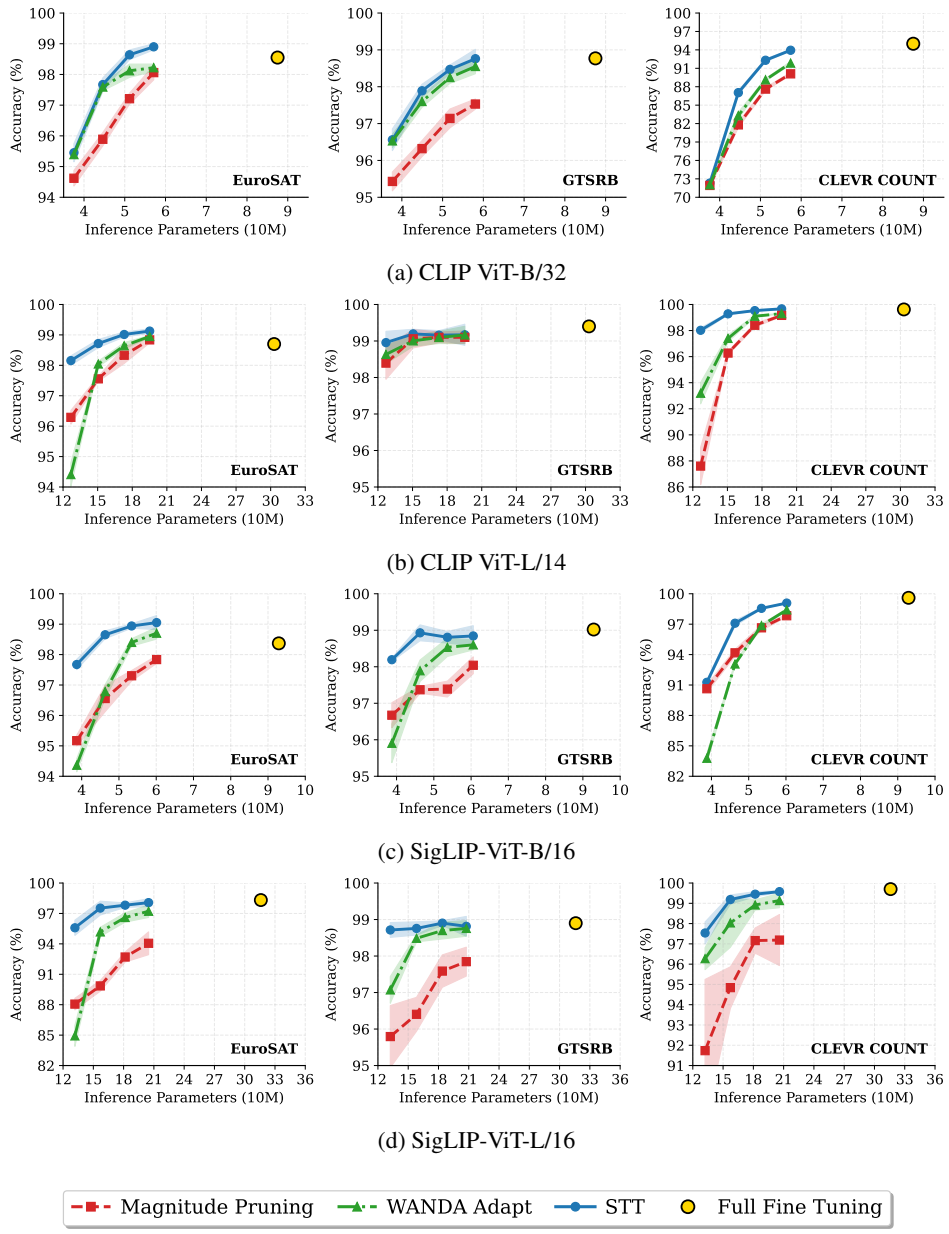
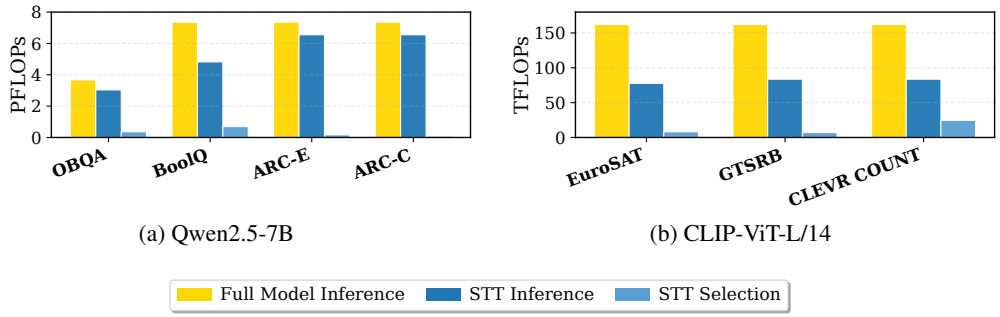


Figure 3: **Main results on ViTs.** STT achieves improved performance–efficiency trade-off curves over baseline methods. The x-axis shows inference parameter count after pruning (in units of 10M), and the y-axis reports accuracy (%). Each curve sweeps retention ratio $k \in \{0.1, 0.2, 0.3, 0.4\}$. The yellow marker denotes full fine-tuning. Shaded regions denote 95% confidence intervals across 5 random seeds.

4.3 Is Selection Cheap and Worth it?

STT identifies task-specialized FFN subnetworks using a one-time selection step on a small unlabeled calibration subset: 1% of the training split for LLMs and 0.25% for ViTs. Here we answer **RQ2**: is this overhead worthwhile? For each task, we choose the sparsest setting that remains competitive with full fine-tuning, following the main trade-off curves in Section 4.2. To compare tasks with different dataset sizes, Figure 4 reports FLOPs normalized per 1,000 inference samples and separates full-model inference, STT inference, and STT selection cost. The selection overhead is quickly

amortized: for Qwen2.5-7B, it accounts for only 1.1%–9.4% of the compute required for 1,000 inference samples, while for CLIP ViT-L/14 it remains within 4.3%–15.2%.



(a) Qwen2.5-7B (b) CLIP-ViT-L/14

Figure 4: **Compute cost analysis of STT.** We report FLOPs (PFLOPs for LLMs; TFLOPs for ViTs) for 1,000 inference samples, with the *one-time* STT selection overhead shown separately. Bars partition full-model inference, STT pruned-subnetwork inference, and selection cost. We use the same retention ratio k as in the main trade-off curves, choosing the sparsest operating point that remains competitive with full fine-tuning.

Table 1: Representative inference throughput results for full models and STT-pruned subnetworks. Throughput is measured in samples/sec at the sparsest operating point that remains competitive with full fine-tuning. Results are reported as mean $\pm$ std over 5 seeds.

Model	Task	Full Model	STT	Speedup
Qwen2.5-7B	ARC-C	82.5 ± 0.4	93.5 ± 0.6	$1.13\times$
	ARC-E	190.6 ± 1.2	231.9 ± 1.2	$1.22\times$
	BoolQ	64.0 ± 0.4	87.6 ± 0.3	$1.37\times$
	OBQA	133.86 ± 1.5	154.18 ± 0.2	$1.15\times$
CLIP ViT-L/14	CLEVR Count	75.26 ± 1.20	200.47 ± 0.97	$2.66\times$
	EuroSAT	64.31 ± 1.08	115.87 ± 0.87	$1.80\times$
	GTSRB	74.62 ± 1.18	199.75 ± 0.71	$2.68\times$

Wall-clock throughput gains. Table 1 reports representative large-model throughput results, while Appendix A.3 provides the complete model–task breakdown. Across all throughput measurements reported in Table 1 and Appendix A.3, STT achieves average speedups of $1.22\times$ on LLMs and $2.41\times$ on ViTs.

4.4 Does Activation Rate Help Beyond Activation Value?

To isolate the contribution of each activation signal and answer **RQ3**, we compare STT against two single-factor selectors under matched sparsity: (1) **Mean Activation Value-only**, which selects $I_\ell^M = \mathrm{TopK}(M_\ell, k)$, and (2) **Activation Rate-only**, which selects $I_\ell^R = \mathrm{TopK}(R_\ell, k)$. This setup ensures that any observed performance gap is driven by the quality of the selected neurons rather than the quantity of parameters.

Neither mean activation value nor activation rate is uniformly dominant across models, tasks, and sparsity levels as shown in Figure 5. For instance, on Qwen2.5-3B (ARC-E) at $k = 0.3$, Mean Activation Value-only outperforms Activation Rate-only (43.16% vs. 33.47%), whereas on CLIP ViT-L/14 (CLEVR-Count) at k=0.1k=0.1 k=0.1 the reverse holds, with Activation Rate-only leading by roughly 8 percentage points. By taking the union of both neuron sets, STT avoids committing to a single task-dependent selector and is competitive across settings—reaching 47.43% on the same Qwen2.5-3B (ARC-E) configuration, a 4.27-point gain over the stronger single-factor alternative. These results suggest that the two statistics capture complementary neuron subsets; Appendix A.9 further supports this by quantifying the layer-wise divergence between the selected neuron sets.

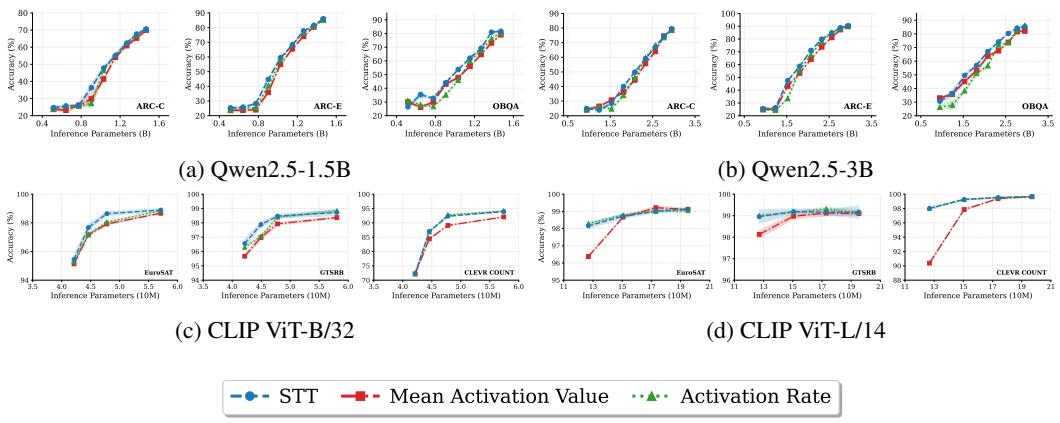


Figure 5: **Ablation on activation statistics.** We compare STT, which combines mean activation value and activation rate, with single-factor selectors using only one statistic. All settings follow the main experimental protocol. Shaded regions denote 95% confidence intervals across 5 random seeds.

5 Discussion

Our empirical results show that pruning strategies designed for downstream adaptation can achieve favorable accuracy–efficiency trade-offs. STT targets a different setting from many existing pruning pipelines: instead of preserving task-agnostic zero-shot behavior, it identifies task-relevant FFN subnetworks that remain trainable after pruning. Because our goal is to evaluate the neuron-selection signal itself, we use a controlled comparison protocol that fixes the calibration data, FFN-neuron pruning granularity, realized per-layer width, and fine-tuning procedure. By controlling these factors, our experiments isolate whether target-task activation statistics provide a better selection criterion for structured FFN pruning. Under this setting, STT improves the accuracy–efficiency trade-off across both encoder-only and decoder-only transformers.

A key observation is that activation rate provides a useful signal beyond mean activation value. Activation rate captures how frequently a neuron is engaged under the target-task distribution, which is not always reflected by its average activation value. By taking the union of both selectors, STT avoids relying on a single task-dependent criterion and yields more stable performance across diverse configurations.

From a practical perspective, STT is lightweight and deployment-friendly. The selection step requires only a single forward pass over a small unlabeled calibration subset, and the resulting structured FFN pruning reduces dense computation rather than relying on unstructured sparsity. This is important because unstructured sparsity often fails to translate theoretical FLOP reductions into wall-clock speedups under standard dense kernels. In contrast, STT reduces FFN width directly and yields consistent throughput gains in our measurements.

6 Conclusion

We presented STT, a select–then–train framework for efficient downstream adaptation of pretrained transformers. STT uses unlabeled target-task calibration inputs to select FFN neurons according to two activation-only statistics: mean activation value and activation rate. The selected neurons define a compact structured subnetwork that is fine-tuned for the downstream task. Across language and vision models, STT achieves improved accuracy–efficiency trade-offs compared with weight magnitude and weight–activation pruning baselines. It often approaches full fine-tuning performance while reducing inference parameters and improving wall-clock throughput. Ablation studies further show that mean activation value and activation rate provide complementary selection signals, explaining the robustness of the union-based criterion.

REFERENCES

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 1–9, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-short.1. URL https://aclanthology.org/2022.acl-short.1/.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2924–2936, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1300. URL https://aclanthology.org/N19-1300/.

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

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=OUIFPHEgJU.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pp. 4171–4186, 2019.

Shibhansh Dohare, J Fernando Hernandez-Garcia, Parash Rahman, A Rupam Mahmood, and Richard S Sutton. Maintaining plasticity in deep continual learning. *arXiv preprint arXiv:2306.13812*, 2023.

Shibhansh Dohare, J. F. Hernandez-Garcia, Parash Rahman, A. Rupam Mahmood, and Richard S. Sutton. Loss of plasticity in deep continual learning. *Nature*, 2024.

William B. Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the International Workshop on Paraphrasing*, 2005.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International conference on machine learning*, pp. 10323–10337. PMLR, 2023.

Trevor Gale, Erich Elsen, and Sara Hooker. The state of sparsity in deep neural networks, 2019. URL https://arxiv.org/abs/1902.09574.

Demi Guo, Alexander M. Rush, and Yoon Kim. Parameter-efficient transfer learning with diff pruning. In *Annual Meeting of the Association for Computational Linguistics*, 2020. URL https://api.semanticscholar.org/CorpusID:229152766.

Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In *ICLR*, 2016.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. Towards a unified view of parameter-efficient transfer learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=0RDcd5Axok.

Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks, 2017. URL https://arxiv.org/abs/1707.06168.

Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Introducing eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. In *IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium*, pp. 204–207. IEEE, 2018.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2790–2799. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/houlsby19a.html.

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*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Hengyuan Hu, Rui Peng, Yu-Wing Tai, and Chi-Keung Tang. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *arXiv preprint arXiv:1607.03250*, 2016.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *CVPR*, 2017.

Steven Kolawole, Lucio Dery, Jean-François Kagy, Virginia Smith, Graham Neubig, and Ameet Talwalkar. Everybody prune now: Structured pruning of llms with only forward passes. *arXiv preprint arXiv:2402.05406*, 2024.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 3045–3059, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.243. URL https://aclanthology.org/2021.emnlp-main.243/.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 4582–4597, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.353. URL https://aclanthology.org/2021.acl-long.353/.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 61–68, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-short.8. URL https://aclanthology.org/2022.acl-short.8/.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

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

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*, 2018.

Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference, 2017. URL https://arxiv.org/abs/1611.06440.

Hyunchan Moon, Cheonjun Park, and Steven L Waslander. Toast: Token channel selection and structured pruning for efficient vit. *arXiv preprint arXiv:2602.15720*, 2026.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren

Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Alec Radford and Karthik Narasimhan. Improving language understanding by generative pre-training. 2018. URL https://api.semanticscholar.org/CorpusID:49313245.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*, pp. 8748–8763, 2021.

Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

Oriane Siméoni, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, Francisco Massa, Daniel Haziza, Luca Wehrstedt, Jianyuan Wang, Timothée Darcet, Théo Moutakanni, Leonel Sentana, Claire Roberts, Andrea Vedaldi, Jamie Tolan, John Brandt, Camille Couprie, Julien Mairal, Hervé Jégou, Patrick Labatut, and Piotr Bojanowski. Dinov3, 2025. URL https://arxiv.org/abs/2508.10104.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP*, pp. 1631–1642, 2013.

Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In *IEEE International Joint Conference on Neural Networks*, pp. 1453–1460, 2011.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. A simple and effective pruning approach for large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=PxoFut3dWW.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, Gaël Liu, Francesco Visin, Kathleen Kenealy, Lucas Beyer, Xiaohai Zhai, Anton Tsitsulin, Robert Busa-Fekete, Alex Feng, Noveen Sachdeva, Benjamin Coleman, Yi Gao, Basil Mustafa, Iain Barr, Emilio Parisotto, David Tian, Matan Eyal, Colin Cherry, Jan-Thorsten Peter, Danila Sinopalnikov, Surya Bhupatiraju, Rishabh Agarwal, Mehran Kazemi, Dan Malkin, Ravin Kumar, David Vilar, Idan Brusilovsky, Jiaming Luo, Andreas Steiner, Abe Friesen, Abhanshu Sharma, Abheesht Sharma, Adi Mayrav Gilady, Adrian Goedeckemeyer, Alaa Saade, Alex Feng, Alexander Kolesnikov, Alexei Bendebury, Alvin Abdagic, Amit Vadi, András György, André Susano Pinto, Anil Das, Ankur Bapna, Antoine Miech, Antoine Yang, Antonia Paterson, Ashish Shenoy, Ayan Chakrabarti, Bilal Piot, Bo Wu, Bobak Shahriari, Bryce Petrini, Charlie Chen, Charline Le Lan, Christopher A. Choquette-Choo, CJ Carey, Cormac Brick, Daniel Deutsch, Danielle Eisenbud, Dee Cattle, Derek Cheng, Dimitris Paparas, Divyashree Shivakumar Sreepathihalli, Doug Reid, Dustin Tran, Dustin Zelle, Eric Noland, Erwin Huizenga, Eugene Kharitonov, Frederick Liu, Gagik Amirkhanyan, Glenn Cameron, Hadi Hashemi, Hanna Klimczak-Plucińska, Harman Singh, Harsh Mehta, Harshal Tushar Lehri, Hussein Hazimeh, Ian Ballantyne, Idan Szpektor, Ivan Nardini, Jean Pouget-Abadie, Jetha Chan, Joe Stanton, John Wieting, Jonathan Lai, Jordi Orbay, Joseph Fernandez, Josh Newlan, Ju yeong Ji, Jyotinder Singh, Kat Black, Kathy Yu, Kevin Hui, Kiran Vodrahalli, Klaus Greff, Linhai Qiu, Marcella Valentine, Marina Coelho, Marvin Ritter, Matt Hoffman, Matthew Watson, Mayank Chaturvedi, Michael Moynihan, Min Ma, Nabila Babar, Natasha Noy, Nathan Byrd, Nick Roy, Nikola Momchev, Nilay Chauhan, Noveen Sachdeva, Oskar Bunyan, Pankil Botarda, Paul Caron, Paul Kishan Rubenstein, Phil Culliton,

Philipp Schmid, Pier Giuseppe Sessa, Pingmei Xu, Piotr Stanczyk, Pouya Tafti, Rakesh Shivanna, Renjie Wu, Renke Pan, Reza Rokni, Rob Willoughby, Rohith Vallu, Ryan Mullins, Sammy Jerome, Sara Smoot, Sertan Girgin, Shariq Iqbal, Shashir Reddy, Shruti Sheth, Siim Põder, Sijal Bhatnagar, Sindhu Raghuram Panyam, Sivan Eiger, Susan Zhang, Tianqi Liu, Trevor Yacovone, Tyler Liechty, Uday Kalra, Utku Evci, Vedant Misra, Vincent Roseberry, Vlad Feinberg, Vlad Kolesnikov, Woohyun Han, Woosuk Kwon, Xi Chen, Yinlam Chow, Yuvein Zhu, Zichuan Wei, Zoltan Egyed, Victor Cotruta, Minh Giang, Phoebe Kirk, Anand Rao, Kat Black, Nabila Babar, Jessica Lo, Erica Moreira, Luiz Gustavo Martins, Omar Sanseviero, Lucas Gonzalez, Zach Gleicher, Tris Warkentin, Vahab Mirrokni, Evan Senter, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, Yossi Matias, D. Sculley, Slav Petrov, Noah Fiedel, Noam Shazeer, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Jean-Baptiste Alayrac, Rohan Anil, Dmitry, Lepikhin, Sebastian Borgeaud, Olivier Bachem, Armand Joulin, Alek Andreev, Cassidy Hardin, Robert Dadashi, and Léonard Hussenot. Gemma 3 technical report, 2025. URL https://arxiv.org/abs/2503.19786.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of ICLR*, 2019.

Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, et al. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2526–2547, 2025.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *arXiv e-prints*, pp. arXiv–2412, 2024.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 11941–11952. IEEE Computer Society, 2023.

Mingyang Zhang, Hao Chen, Chunhua Shen, Zhen Yang, Linlin Ou, Xinyi Yu, and Bohan Zhuang. LoRAPrune: Structured pruning meets low-rank parameter-efficient fine-tuning. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 3013–3026, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.178. URL https://aclanthology.org/2024.findings-acl.178/.

A Appendix

A.1 Limitations

Our study focuses on structured pruning of FFN layers in transformer architectures. While FFNs account for a substantial fraction of parameters and inference cost, we do not explore pruning of attention modules or joint sparsification across multiple architectural components. Extending STT to jointly optimize sparsity across FFN and attention layers may further improve efficiency gains.

Our evaluation is conducted on representative classification and reasoning benchmarks for both language and vision models. While these tasks cover a range of downstream adaptation scenarios, we do not evaluate STT on long-context generation, instruction-following with open-ended outputs, or continual learning settings. Additional evaluation on these scenarios would further clarify the generality of our approach.

Finally, our reported inference speedups are measured under specific hardware and implementation settings. Although structured FFN pruning provides consistent acceleration in our experiments, absolute speedup values may vary across hardware platforms and backends.

A.2 HYPER-PARAMETER

We tune all hyperparameters on a held-out validation set. When a task provides an official validation split, we use it as-is; otherwise, we randomly sample 20% of the training data as a validation set and train on the remaining 80%. The random split is fixed across models for comparability, and the same search protocol and ranges are applied to all methods. Learning rates are selected via a grid over $\{1e{-}5,\ 5e{-}5,\ 1e{-}4,\ 5e{-}4\}$ using a single reference seed on the validation set; we then freeze the chosen configuration.For STT training and evaluation, each run is trained for a fixed number of epochs, and the *final* model (last epoch) is evaluated. For optimization, we use AdamW (Loshchilov & Hutter, 2019) with weight decay values of 0.1 and 0.01 for our LLM and ViT configurations, respectively.

We report the task performance (e.g., accuracy or Exact Match) computed on the test set *once per run*, and summarize results as performance $\pm$ standard error over the five seeds.Regarding the neuron selection ratio k, we choose optimal values based on task-specific performance saturation. For vision tasks, we find that performance consistently achieves over 99% of the maximal accuracy within the range $k \in [0.2, 0.4]$. Specifically, for CLIP-ViT-B/32 and SIGLIP-B/16, the optimal ratio is typically $k = 0.40$ (e.g., on ClevrCount) or $k = 0.30$ (on EuroSAT). Larger models such as CLIP-ViT-L/14 and SIGLIP-L/16 exhibit even higher redundancy, consistently favoring an aggressive $k = 0.20$ across all vision benchmarks (ClevrCount, EuroSAT, and GTSRB). For LLMs, the optimal k require more conservative retention, with optimal k values falling between 0.80 and 0.90.

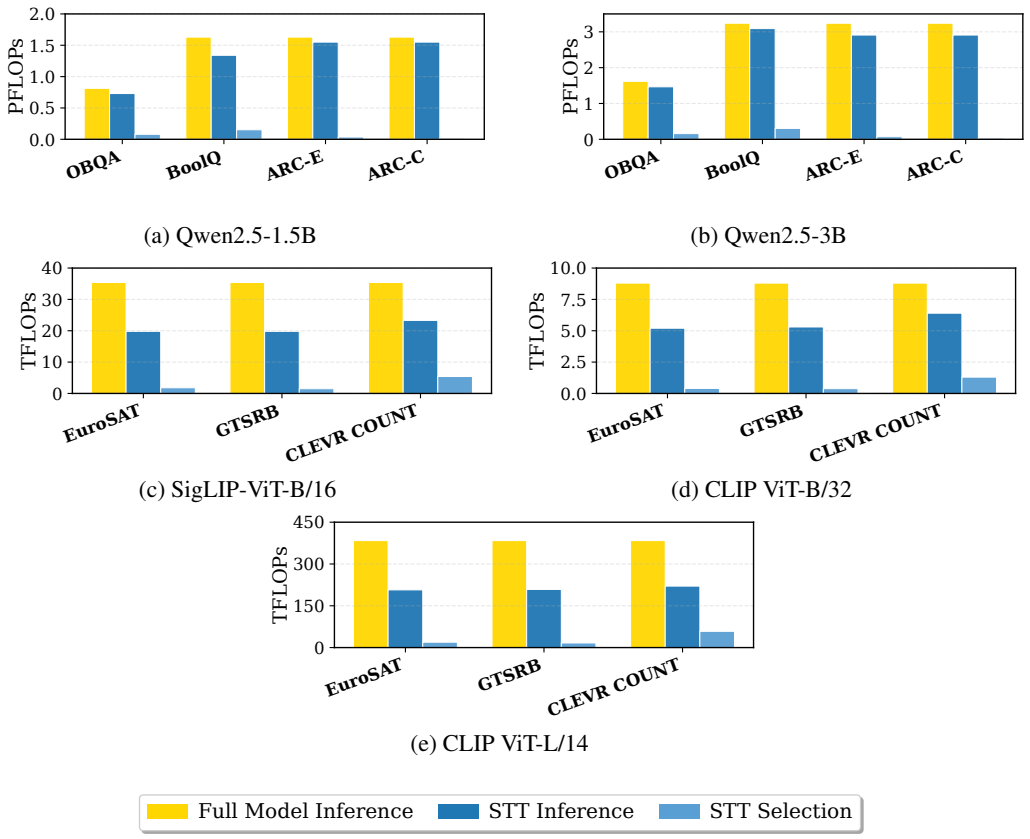
A.3 FLOPS AND THROUGHPUT

We provide all relevant details for FLOPs counting and throughput measurements in this section. All experiments are conducted on NVIDIA L40S GPUs (ViTs) and H100 GPUs (LLMs). We measure computational efficiency using FLOPs computed via PyTorch's FlopCounterMode(Paszke et al., 2019). For vision models, we compute per-sample FLOPs by averaging across multiple batch samples. For language models, we assume the maximum sequence length to ensure consistent FLOP accounting across variable-length inputs (2048 tokens for Qwen2.5 models). Inference FLOPs count forward-only operations across evaluation sets. Selection overhead is measured as the base model's forward FLOPs on the calibration set.

Table 2: **Inference throughput (samples/sec) for full models vs. STT-pruned subnetworks and the resulting speedup.**. For each task/model, we use the same retention ratio k as in the main trade-off curves, selecting the largest pruning level that remains competitive with full fine-tuning (mean $\pm$ std over 5 seeds).

Model	Task	Full Model	STT	Speedup
Qwen2.5-1.5B	ARC-C	268.9 ± 0.52	282.0 ± 0.66	$1.05\times$
	ARC-E	616.6 ± 1.47	652.3 ± 2.07	$1.06\times$
	BoolQ	208.9 ± 1.16	267.2 ± 1.00	$1.32\times$
	OBQA	390.52 ± 3.4	413.0 ± 0.48	$1.06\times$
Qwen2.5-3B	ARC-C	152.4 ± 0.73	161.2 ± 0.34	$1.06\times$
	ARC-E	352.3 ± 1.55	389.4 ± 1.77	$1.11\times$
	BoolQ	118.1 ± 0.94	125.8 ± 0.65	$1.06\times$
	OBQA	241.78 ± 2.1	265.34 ± 2.4	$1.10\times$

In Section 4.3, we report the normalized selection cost as a percentage of the compute required for a single batch of 1,000 inference samples for selected models (Qwen2.5-7B and CLIP ViT-L/14). The remaining model results are presented in this section. We measure inference throughput using wall-clock time measurements via Python's time module, with CUDA synchronization (torch.cuda.synchronize) to ensure accurate GPU timing. Throughput is computed by averaging across multiple forward passes (200 iterations after 20 warmup iterations) on real evaluation data, reporting samples processed per second. For throughput measurements, we apply dimension padding to align pruned linear layers to multiples of 128, optimizing CUDA kernel performance (particularly for Tensor Core alignment). We note that this padding doesn't affect effective parameter counts and effective FLOPs.



(a) Qwen2.5-1.5B (b) Qwen2.5-3B

(c) SigLIP-ViT-B/16 (d) CLIP ViT-B/32

(e) CLIP ViT-L/14

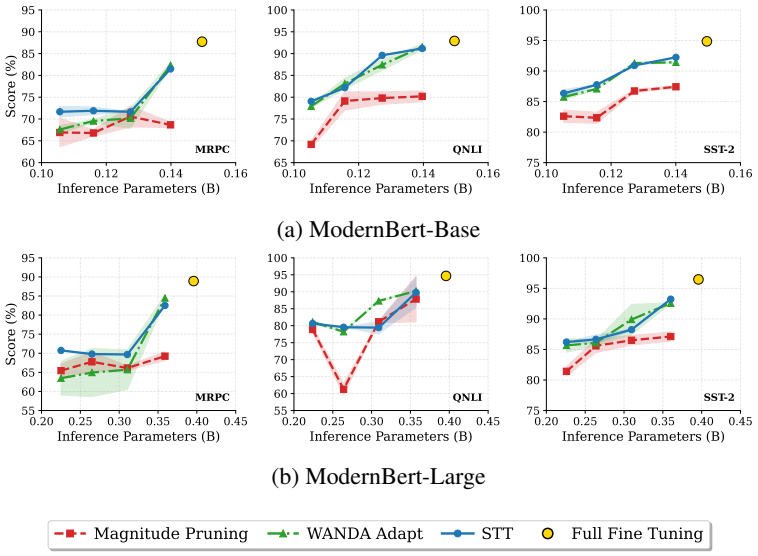
Figure 6: **Compute cost analysis of STT.** We report FLOPs (PFLOPs for LLMs; TFLOPs for ViTs) for 1,000 inference samples, with the one-time STT selection overhead shown separately. Bars partition full-model inference, STT pruned-subnetwork inference, and selection cost. We use the same retention ratio k as in the main trade-off curves, choosing the largest pruning level that remains competitive with full fine-tuning.

Table 3: **Inference throughput (samples/sec) for full models vs. STT-pruned subnetworks and the resulting speedup.**. For each task/model, we use the same retention ratio k as in the main trade-off curves, selecting the largest pruning level that remains competitive with full fine-tuning (mean $\pm$ std over 5 seeds).

Model	Task	Full Model	STT	Speedup
CLIP ViT-B/32	CLEVR Count	1214.4 ± 8.22	1785.1 ± 17.50	$1.47\times$
	EuroSAT	1063.5 ± 6.57	1655.7 ± 9.98	$1.56\times$
	GTSRB	1200.2 ± 6.82	1755.3 ± 13.97	$1.46\times$
SigLIP ViT-B/16	CLEVR Count	335.5 ± 5.29	465.4 ± 2.89	$1.39\times$
	EuroSAT	288.7 ± 4.79	461.8 ± 1.83	$1.60\times$
	GTSRB	343.5 ± 5.56	540.3 ± 2.50	$1.57\times$
SigLIP ViT-L/16	CLEVR Count	67.0 ± 0.34	101.8 ± 0.84	$1.52\times$
	EuroSAT	66.5 ± 0.26	101.1 ± 0.84	$1.52\times$
	GTSRB	66.6 ± 0.29	100.5 ± 0.73	$1.51\times$

A.4 BERT

We evaluate STT against full fine-tuning (FFT) on three representative GLUE classification tasks—MRPC (Dolan & Brockett, 2005), SST-2 (Socher et al., 2013), and QNLI (Wang et al., 2019)—using Bert and ModernBert family models (Warner et al., 2025; Devlin et al., 2019). MRPC is a sentence-pair paraphrase detection task with roughly 3.7k examples. SST-2 is single-sentence



(a) ModernBert-Base

(b) ModernBert-Large

Figure 7: **Results on ModernBert-family models.** The x-axis shows inference parameter count after pruning (in units of 1B), and the y-axis reports accuracy (%). Each curve sweeps retention ratio $k \in \{0.2, 0.4, 0.6, 0.8\}$. The yellow marker denotes full fine-tuning. Shaded regions denote 95% confidence intervals across 5 random seeds.

sentiment classification with about 67k examples. QNLI is a sentence-pair entailment task converted from SQuAD with about 105k examples.

All experiments use the same optimizer, batch size, and number of epochs. We did grid search across $\{1e-5, 5e-5, 1e-4, 5e-4\}$ and swept retention rate k values across $\{0.2, 0.4, 0.6, 0.8\}$.

Our evaluation demonstrates that STT generally outperforms Magnitude Pruning and WANDA Adapt across different BERT family models, with performance advantages that are particularly pronounced on challenging tasks and at lower retention rates. Compared to full fine-tuning (FFT), STT achieves competitive or superior accuracy while using significantly fewer parameters.

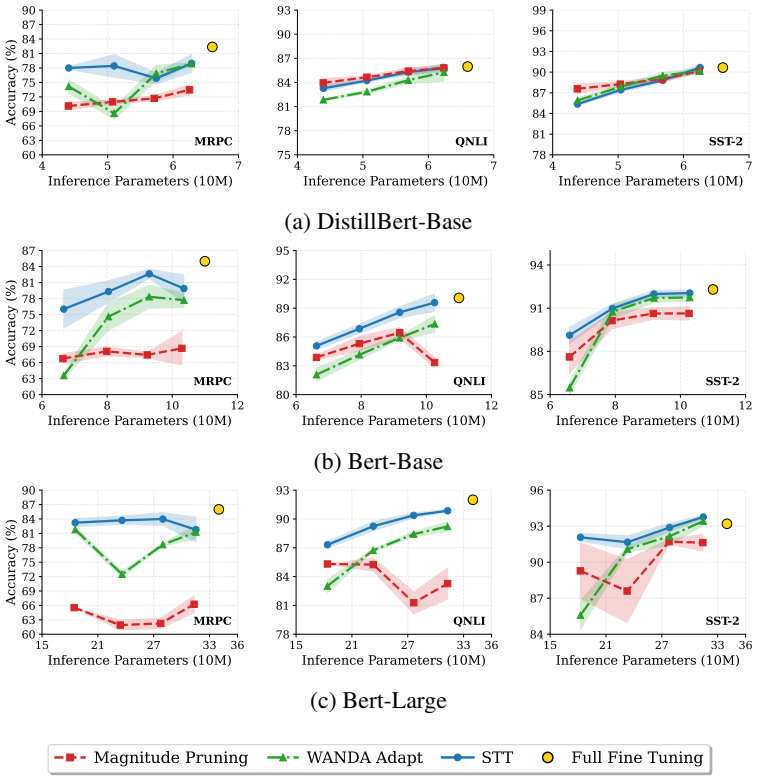


Figure 8: **Results on BERT-family models.** STT often achieves improved performance–efficiency trade-off curves over baseline methods on BERT-family models. The x-axis shows inference parameter count after pruning (in units of 10M), and the y-axis reports accuracy (%). Each curve sweeps retention ratio $k \in \{0.2, 0.4, 0.6, 0.8\}$. The yellow marker denotes full fine-tuning. Shaded regions denote 95% confidence intervals across 5 random seeds.

A.5 LoRA

As mentioned in Section 1, our framework is compatible with common PEFT methods, such as LoRA. STT-LoRA combines neuron selection with LoRA, achieving comparable performance to standard LoRA while significantly improving inference efficiency.

We set $r = 16$ and $\alpha = 32$ for LLMs and $r = 32$ and $\alpha = 64$ for ViT models for both LoRA (Hu et al., 2022) and STT-LoRA experiments. All experiments use the same optimizer, batch size, and number of epochs. We did grid search across $\{1e-5, 5e-5, 1e-4, 5e-4\}$ and swept retention rate k values across $\{0.1, 0.2, 0.3, 0.4\}$ for ViTs.

Our evaluation demonstrates that STT-LoRA achieves competitive performance compared to other sparsification or PEFT methods across CLIP and SigLIP ViTs, as shown in Figure 9. For CLIP models, we observe that the performance of STT-LoRA often matches or exceeds LoRA's performance while using much smaller inference parameter counts. Notably, on CLIP-ViT-Large, STT-LoRA achieves better performance than STT on EuroSat dataset under extreme low retention ratio $k = 0.1$. We also observed that at high sparsity (retention ratio $k = 0.1$), STT-LoRA shows a performance gap compared to STT and STT-LoRA often matches STT performance at moderate sparsity levels (retention ratio $k = 0.3$).

We did grid search across $\{1e-5, 5e-5, 1e-4, 5e-4\}$ and swept retention rate k values across $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$ for LLMs. STT often outperforms state-of-the-art pruning methods (Magnitude Pruning and WANDA Adapt). We observed that STT-LoRA achieves comparable or superior accuracy in many configurations compared to STT. However, compared to STT, STT-LoRA shows less stability and exhibits more performance dips. Despite this trade-

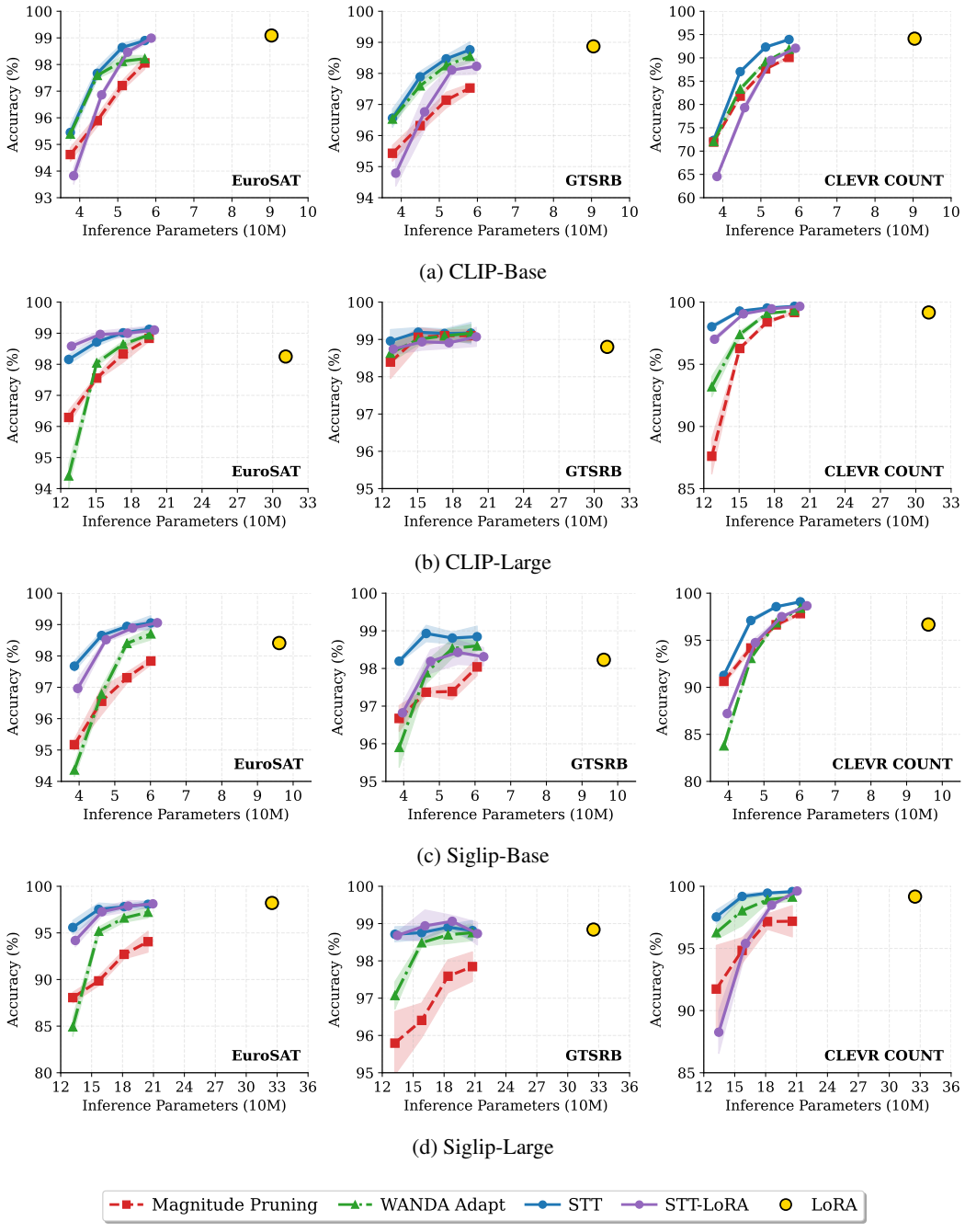


Figure 9: **STT-LoRA results on ViTs.** STT-LoRA achieves competitive performance–efficiency trade-off curves compared with STT. The x-axis shows inference parameter count after pruning (in units of 10M), and the y-axis shows task performance. Each curve sweeps retention ratio $k \in \{0.1, 0.2, 0.3, 0.4\}$. Shaded regions denote 95% confidence intervals across 5 random seeds.

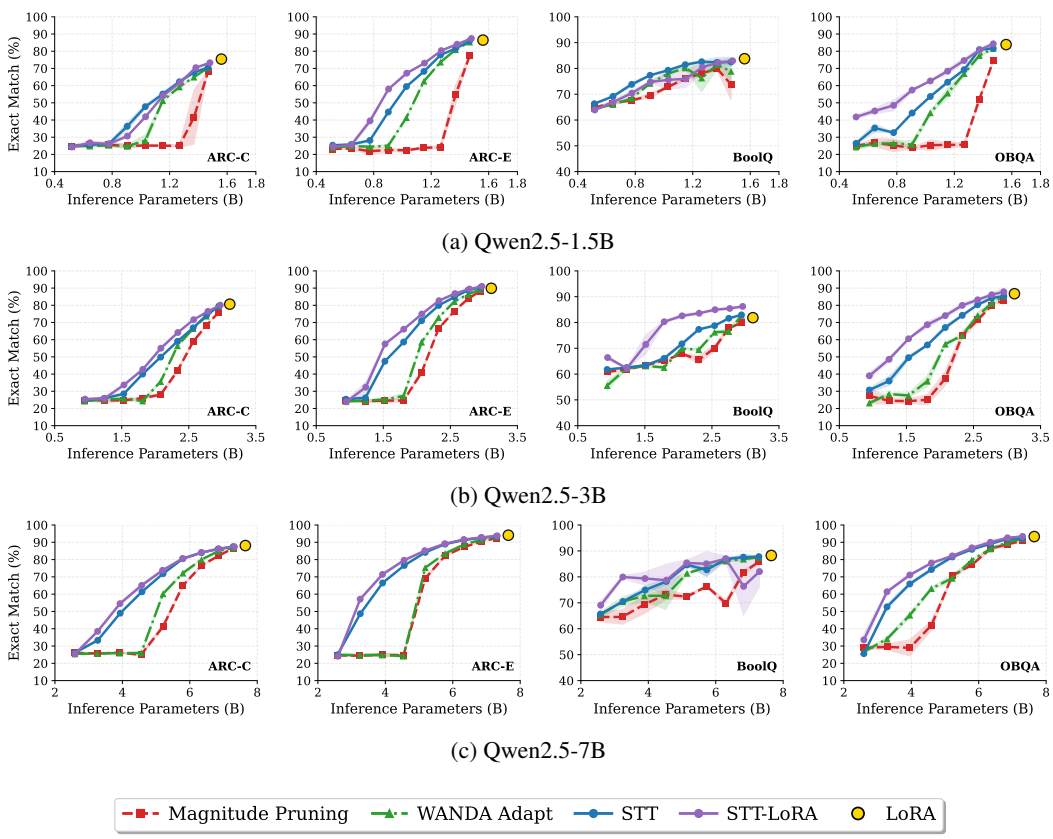


Figure 10: **STT-LoRA results on LLMs.** STT-LoRA achieves competitive performance–efficiency trade-off curves compared with STT. The x-axis shows inference parameter count after pruning (in billions), and the y-axis shows task performance (see axis labels for the metric). Each curve sweeps retention ratio $k \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$. Shaded regions denote 95% confidence intervals across 5 random seeds.

off, STT-LoRA consistently outperforms Magnitude Pruning and WANDA Adapt across most evaluated configurations, achieving superior accuracy in most cases, respectively. Moreover, STT-LoRA demonstrates fewer performance dips compared to Magnitude Pruning and WANDA Adapt, indicating stronger robustness across different sparsity levels.

A.6 Selection Regarding the Size of Calibration Set

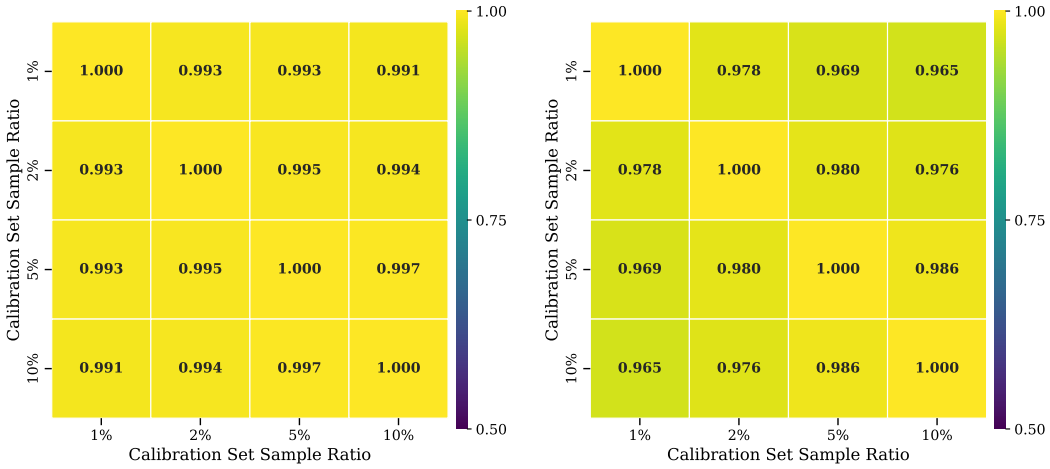
We assessed whether STT is robust to different size of calibration set or not. We sampled 1%,2%,5%,10% of the training set to create different unlabeled calibration set and for each ratio r and MLP layer l, we record the selected unit set and compute the per-layer Jaccard overlap

$$J\left(A_\ell^{(r_i)}, A_\ell^{(r_j)}\right) = \frac{\left|A_\ell^{(r_i)} \cap A_\ell^{(r_j)}\right|}{\left|A_\ell^{(r_i)} \cup A_\ell^{(r_j)}\right|},$$

and then take average across layers to obtain a single similarity score as

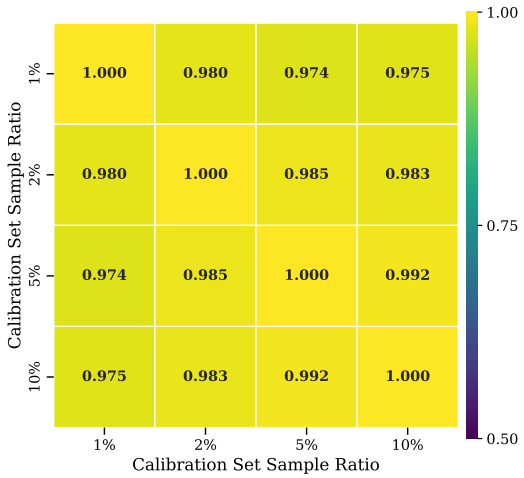
$$\mathbf{S} = \left[S(r_i, r_j)\right]_{r_i, r_j \in \{1\%, 2\%, 5\%, 10\%\}}.$$

and the heatmap visualized the matrix. Values near 1 indicate the MLP subnetwork selected under different calibration set are nearly identical.



(a) CLEVR-Count (CLIP-B/32)

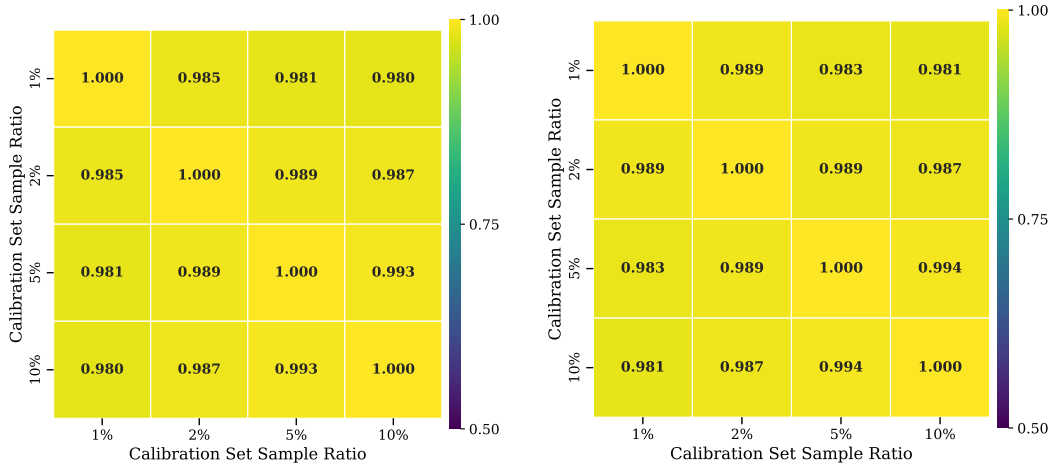
(b) EuroSAT (CLIP-B/32)



(c) GTSRB (CLIP-B/32)

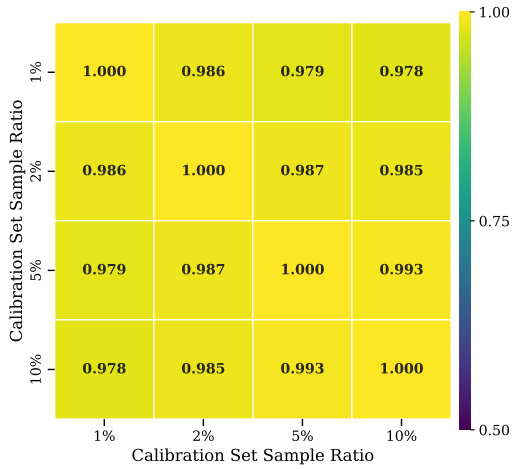
Figure 11: Sub-network stability (Jaccard) across calibration-set sizes $\{1\%, 2\%, 5\%, 10\%\}$ on CLIP.

As shown in Fig. 11, , the subnetworks selected on the CLIP-B/32 backbone for all three tasks are very stable. All pairwise values exceed 0.95 using retention ratio $k = 0.3$. Figure 12 shows the same analysis on LLMs (Qwen2.5-1.5B) with calibration ratios $r \in \{1\%, 2\%, 5\%, 10\%\}$. For **ARC-E** and **ARC-C** we use a relatively large retention ratio, $k = 0.9$, and the layer-averaged Jaccard overlaps between subnetworks are uniformly high (≥ 0.98), indicating essentially identical selections across calibration sizes.

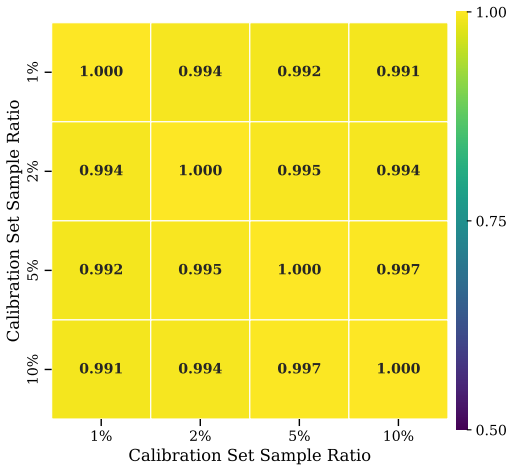


(a) ARC-C (Qwen2.5-1.5B)

(b) ARC-E (Qwen2.5-1.5B)



(c) BoolQ (Qwen2.5-1.5B)



(d) OBQA (Qwen2.5-1.5B)

Figure 12: Sub-network stability (Jaccard) across calibration ratios $\{1\%, 2\%, 5\%, 10\%\}$ on Qwen2.5-7B decoder benchmarks.

We further fine-tune the subnetworks selected under different calibration-set sizes on Qwen2.5-1.5B and CLIP-ViT-Base-Patch32. The resulting downstream performance remains highly similar across these settings, indicating that STT is robust not only in terms of selected-neuron overlap, but also in end-task accuracy after adaptation as shown in Figure 13.

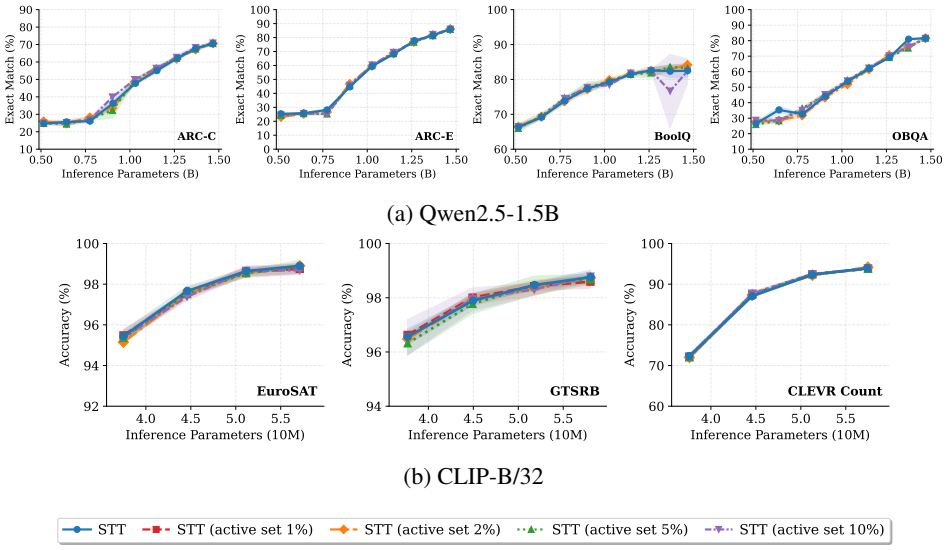
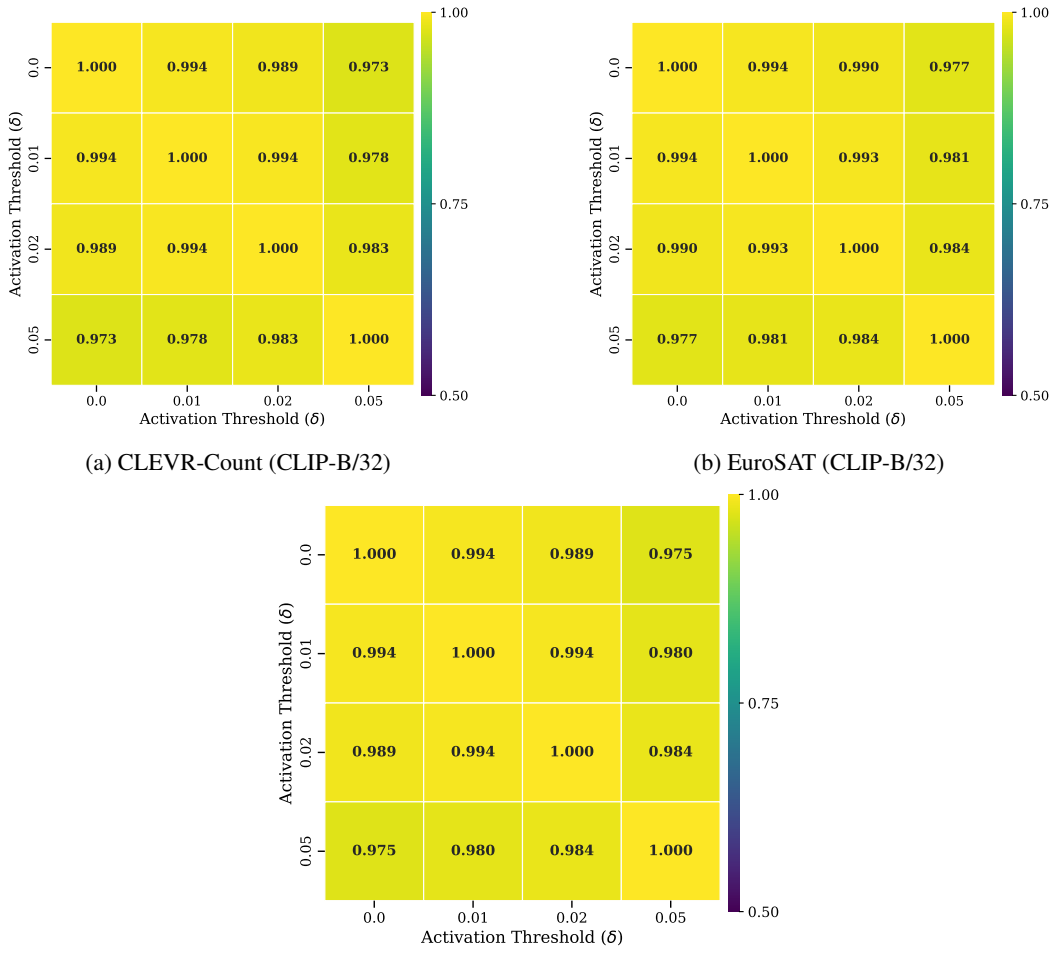


Figure 13: Robustness of STT to calibration-set size. We fine-tune the subnetworks selected using different calibration-set sizes on Qwen2.5-1.5B and CLIP-ViT-Base-Patch32. The downstream performance remains highly similar across settings, showing that STT is robust not only in selected-neuron overlap, but also in end-task accuracy after adaptation.

A.7 Selection Regarding the Activation Threshold



(a) CLEVR-Count (CLIP-B/32)

(b) EuroSAT (CLIP-B/32)

(c) GTSRB (CLIP-B/32)

Figure 14: Sub-network stability (Jaccard) across activation thresholds $\delta \in \{0, 0.01, 0.02, 0.05\}$ on CLIP-B/32. The calibration-set size is fixed to 0.25% and the retained budget is matched across thresholds.

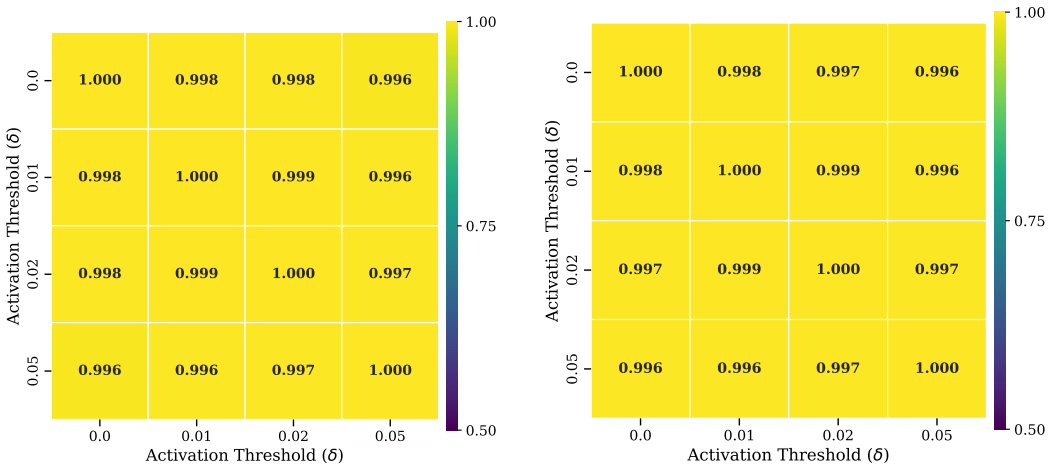
We assessed whether STT is robust to the choice of activation threshold δ. We fix the calibration-set size to 1% for LLMs and 0.25% for ViTs, and vary the activation threshold over $\delta \in \{0, 0.01, 0.02, 0.05\}$. For each threshold δ and MLP layer ℓ, we record the selected unit set and compute the per-layer Jaccard overlap

$$J\left(A_\ell^{(\delta)}, A_\ell^{(\delta)}\right) = \frac{\left|A_\ell^{(\delta)} \cap A_\ell^{(\delta)}\right|}{\left|A_\ell^{(\delta)} \cup A_\ell^{(\delta)}\right|}.$$

We then average across layers to obtain a single similarity score,

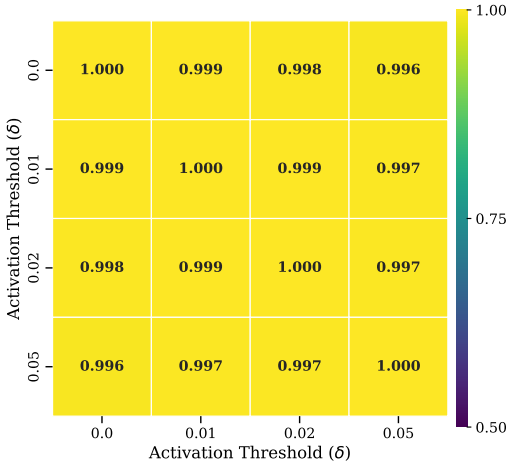
$$\mathbf{S} = \left[S(\delta, \delta)\right]_{\delta, \delta \in \{0,\, 0.01,\, 0.02,\, 0.05\}}.$$

The heatmaps visualize this matrix. Values near 1 indicate that the MLP subnetworks selected under different activation thresholds are nearly identical.

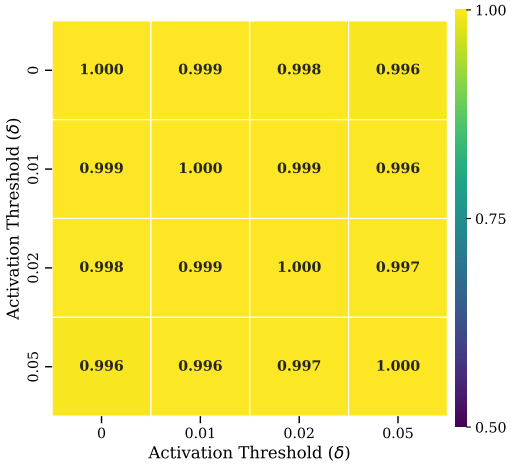


(a) ARC-C (Qwen2.5-1.5B)

(b) ARC-E (Qwen2.5-1.5B)



(c) BoolQ (Qwen2.5-1.5B)

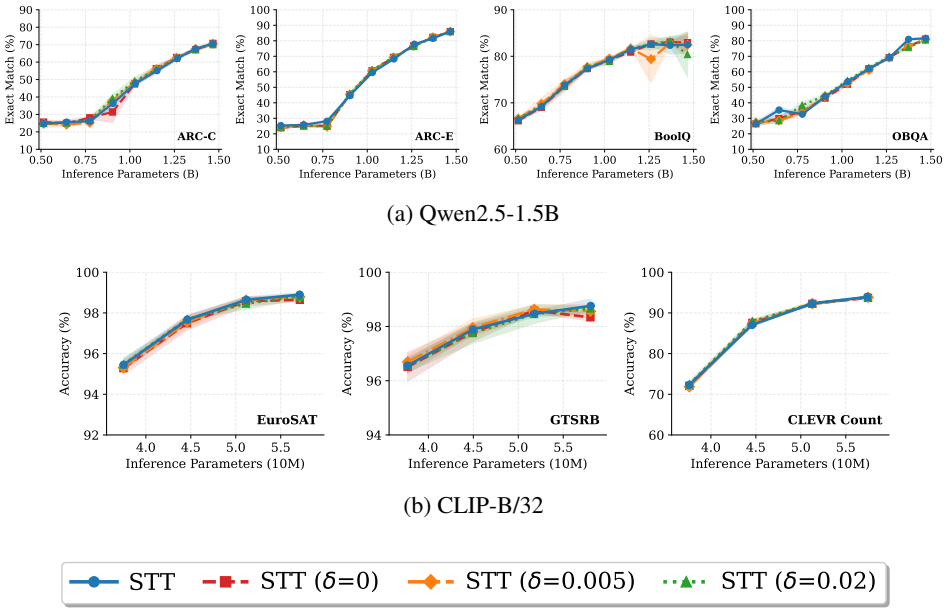


(d) OBQA (Qwen2.5-1.5B)

Figure 15: Sub-network stability (Jaccard) across activation thresholds $\delta \in \{0, 0.01, 0.02, 0.05\}$ on Qwen2.5-1.5B. The calibration-set size is fixed to 1% and the size of retained subnetwork is matched across thresholds.

As shown in Figure 14 and Figure 15, the subnetworks selected on CLIP-ViT-Base-Patch32 and Qwen2.5-1.5B remain highly stable across activation thresholds.

We further fine-tune the subnetworks selected under different activation thresholds on Qwen2.5-1.5B and CLIP-ViT-Base-Patch32. The resulting downstream performance remains highly similar across



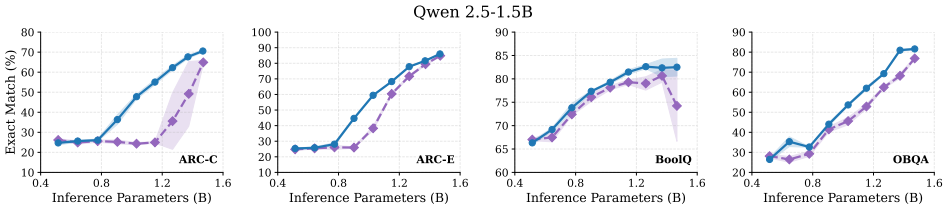
(a) Qwen2.5-1.5B

(b) CLIP-B/32

Figure 16: Robustness of STT to the activation threshold δ. We fine-tune the subnetworks selected using different activation thresholds on Qwen2.5-1.5B and CLIP-ViT-Base-Patch32. The downstream performance remains highly similar across settings, showing that STT is robust not only in selected-neuron overlap, but also in end-task accuracy after adaptation.

these settings, indicating that STT is robust not only in terms of selected-neuron overlap, but also in end-task accuracy after adaptation, as shown in Figure 16.

A.8 Ablation on Activation Collection in Gated FFNs



(a) Qwen2.5-1.5B

For gated FFNs, the FFN output is computed as

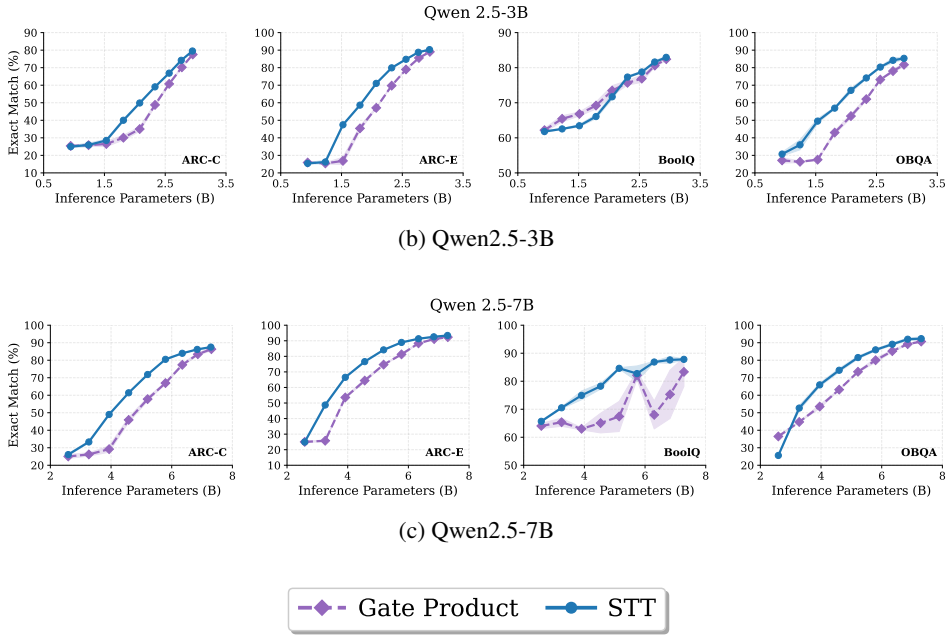
$$F_{\text{ffn}}(X) = \big(\delta(XW_g) \odot (XW_u)\big)W_d, \tag{8}$$

where W_g and W_u are the gate and up projections, respectively. In the main STT implementation, for gated FFNs we collect A_ℓ from the gate branch $\delta(XW_g)$. This choice follows the intuition that the gate branch directly controls which intermediate neurons are activated under the target task distribution.

To examine whether this design choice affects the selected subnetwork, we compare STT with a gate-product variant that instead collects activations from

$$A_\ell^{\text{prod}} = \delta(XW_g) \odot (XW_u). \tag{9}$$

All other components of STT are kept unchanged. In particular, both variants compute the same two activation-only statistics, $M_{\ell,j}$ and $R_{\ell,j}$, and use the same union-based selection rule in Eq. 7. For a fair comparison, we match the retained-neuron budget layer by layer: we first obtain the number



(b) Qwen2.5-3B

(c) Qwen2.5-7B

Figure 17: Ablation on where activations are collected in gated FFNs. The default STT selector collects A_ℓ from the gate branch $\delta(XW_g)$, while the gate-product variant uses $A_\ell^{\text{prod}} = \delta(XW_g) \odot (XW_u)$. Both variants use the same activation statistics and are evaluated under matched layer-wise retained-neuron budgets. Across model scales and downstream tasks, the default gate-branch selector consistently yields a more favorable accuracy-efficiency trade-off than the gate-product variant, suggesting that the gate branch provides a cleaner and more reliable signal for neuron selection.

of retained neurons selected by the default STT selector in each FFN block, and then apply the gate-product selector under the same per-layer budget.

As shown in Figure 17, collecting activations from the post-gating product does not improve STT. Instead, the default gate-branch selector achieves a consistently stronger accuracy–efficiency trade-off across Qwen2.5 model scales and downstream tasks.

A.9 Selection Divergence Between Mean Activation Value and Activation Rate

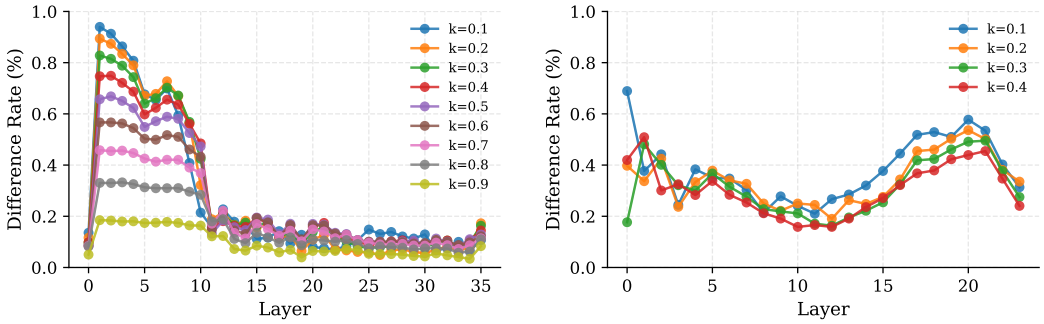
To further analyze this complementarity, we quantify the selection divergence between the two activation-based heuristics across transformer layers. This analysis complements the matched-sparsity ablation in the main text by examining whether the two heuristics select similar or distinct neuron subsets.

We define the **Difference Rate** (DiffRate_ℓ) as the Jaccard distance between the neuron index sets I_ℓ^M and I_ℓ^R selected by mean activation value and activation rate:

$$\text{DiffRate}_\ell = 1 - \frac{|I_\ell^M \cap I_\ell^R|}{|I_\ell^M \cup I_\ell^R|}. \tag{10}$$

A value of 0 indicates identical selections, while a value of 1 indicates completely disjoint neuron sets.

Figure 18 shows the layer-wise Difference Rate for representative LLM and ViT models. We observe substantial divergence between the two selection criteria across Transformer layers. For Qwen2.5-3B on ARC-E, the Difference Rate exceeds 80% in early blocks under aggressive pruning and gradually decreases in deeper layers. For CLIP ViT-L/14 on CLEVR-Count, the two criteria also differ substantially, with Difference Rates remaining above 50% across a range of pruning ratios.



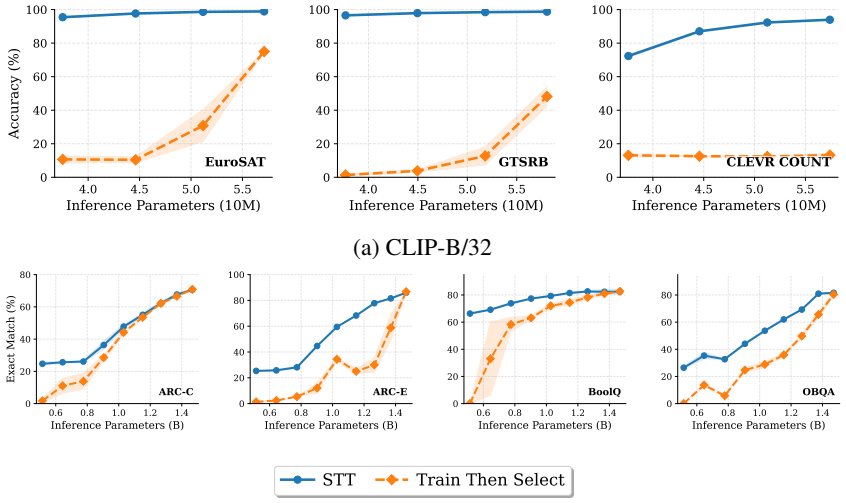
(a) ARC-E (Qwen2.5-3B) (b) CLEVR-Count (CLIP ViT-L/14)

Figure 18: **Selection divergence analysis.** We report the Difference Rate between neuron sets selected by mean activation value and activation rate. Larger divergence indicates that the two statistics capture more distinct neuron subsets.

These results indicate that mean activation value and activation rate often select substantially different neuron subsets at multiple depths of the network. Together with the matched-sparsity ablation results in Figure 5, this provides further evidence that the two statistics are complementary, which helps explain the performance gains of STT.

A.10 Train, then Select

We also compared STT with another baseline method "train, then select" that prunes using the same approach as STT after performing full fine tuning under a matched structured-pruning budget.



(a) CLIP-B/32

Figure 19: Comparison between STT and using the same approach a STT but after full fine tuning under the same pruning budget. Across model scales and downstream tasks, STT outperforms doing full fine tuning first follow up by STT.

As illustrated in Figure 19, STT consistently outperforms 'train, then select" across both vision and language modalities. We attribute this to the fact that the weight updates during FFT become highly co-adapted to the specific dense architecture; pruning these neurons post-hoc, without additional recovery training, causes catastrophic interference with the learned task-specific representations. In contrast, STT identifies a good sub-network *prior* to adaptation, allowing the subsequent training phase to specialize the weights specifically for the sparse structure.

A.11 Generalization to Gemma-3

To evaluate whether STT generalizes beyond the Qwen model family, we additionally included Gemma-3-1B for evaluation. We use the same matched-width evaluation protocol as in Section 4.2, with five random seeds for each method and a fixed learning rate of 5×10^{-5}.

Figure 20 reports the complete sweep over 36 task-budget settings. STT achieves the highest mean accuracy in 30 of 36 settings and outperforms WANDA-Adapt and magnitude pruning in 31 of 36 and 35 of 36 pairwise comparisons, respectively. These results show that the accuracy-efficiency gains of STT extend beyond the Qwen model family.

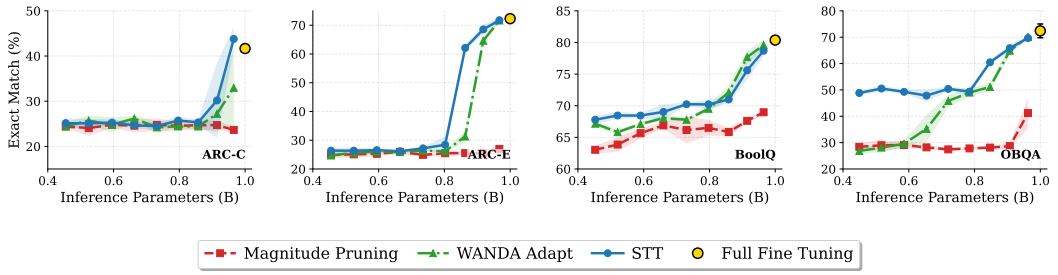
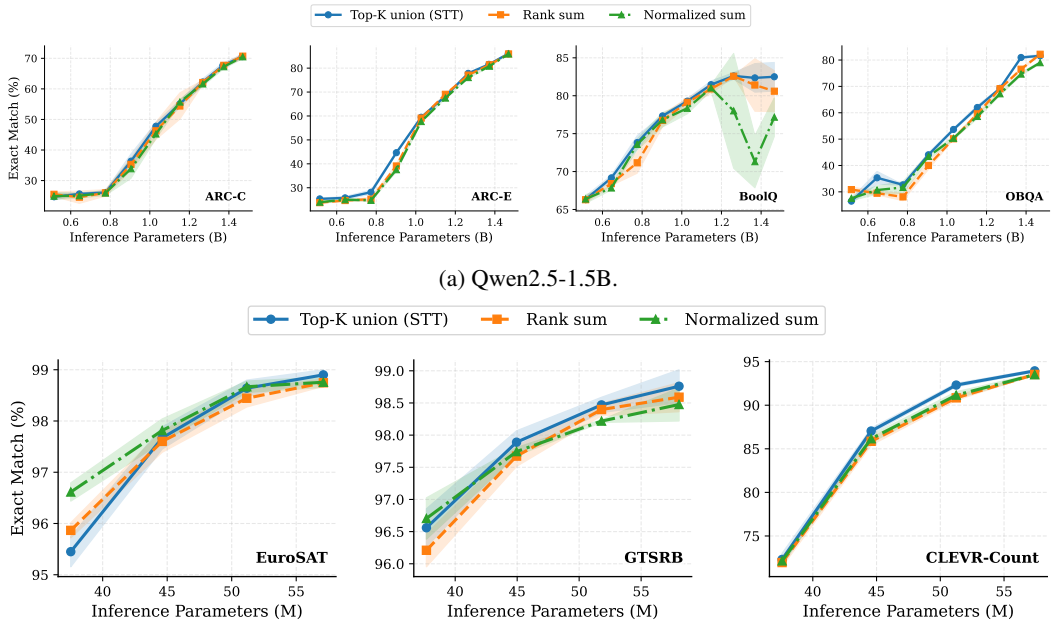


Figure 20: Performance-efficiency trade-offs on Gemma-3-1B across ARC-C, ARC-E, BoolQ, and OBQA. All structured methods are compared at matched realized per-layer FFN widths. Shaded regions denote 95% confidence intervals over five random seeds.

A.12 Alternative Combination Rules



(a) Qwen2.5-1.5B.

(b) CLIP ViT-B/32.

Figure 21: Comparison of Top-K union, rank-sum, and normalized-sum aggregation across language and vision tasks. All methods are compared at matched realized per-layer FFN widths.

STT combines mean activation and activation rate by taking the union of their Top-K sets. We compare this choice with two simple alternatives: rank-sum aggregation and normalized-sum aggregation. All methods use the same activation statistics and are compared at matched realized per-layer FFN

widths. We reuse the hyperparameters from the main experiments and report results over five random seeds.

Figures 21a and 21b report 48 matched-width task-budget settings: 36 across four LLM tasks and 12 across three vision tasks. No aggregation rule dominates universally. Top-K union achieves the highest mean accuracy in 26 of 48 settings and the highest task-level average on five of seven tasks. We therefore use Top-K union as a simple, normalization-free default rather than claiming that it is universally optimal.

A.13 Use of Large Language Models

We used AI Assistants such as ChatGPT and Grammarly for spell-checking, fixing minor grammatical mistakes, and polishing the writing.