# UNDERSTANDING SPARSE FINE-TUNING: LESSONS AND INSIGHTS FROM A UNIFIED EMPIRICAL STUDY

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Sparse fine-tuning updates a small subset of task-critical neurons and has been reported to outperform full fine-tuning (FFT) while resisting catastrophic forgetting. Existing methods, however, differ in the selection criterion, module, schedule, and neuron-update ratio, and why sparse fine-tuning works remains poorly understood. We select eight representative sparse fine-tuning methods and conduct a unified empirical study on mathematical reasoning and code generation, using FFT as a reference, establishing a performance leaderboard of sparse fine-tuning methods. We further ablate the key design choices underlying sparse fine-tuning to provide a comprehensive analysis of this paradigm. We characterise the role of the sparse masked update: it acts as a *capacity allocator* that concentrates learning on task-critical neurons, and as a *regulariser* that pins the remaining parameters to their pre-trained values. This dual role mitigates the trade-off between task performance and anti-forgetting in FFT: sparse methods improve over FFT by 2.1 points on average while retaining 96–109% of the base model's general capability across all tested families, scales, and tasks. We further find that: (1) The advantage of sparse fine-tuning over FFT grows with the capability of the base model; (2) Task-relevant neurons reside in both the MHA and the FFN of the backbone, so the selection location has only a marginal effect; (3) The optimal neuron-update ratio is method- and task-dependent, making adaptive neuron selection essential; (4) Dynamic mask regeneration trades general capability for a modest task gain, at a recurring compute and memory cost. Our analysis clarifies the design choices of sparse fine-tuning and identifies future research directions.

## 1 INTRODUCTION

Full fine-tuning (FFT) remains the standard approach for adapting large language models (LLMs) to downstream tasks (Zhang et al., 2026b), yet it incurs substantial computational and storage costs and is prone to catastrophic forgetting (Luo et al., 2025). Parameter-efficient fine-tuning (PEFT) reduces this cost by freezing the backbone and training only a small number of additional parameters (Xu et al., 2026); however, its adaptation capacity is then bounded by the size and placement of those parameters rather than by the model itself (Liu et al., 2026). These limitations have spurred a rapidly growing literature on sparse fine-tuning (Sung et al., 2021). Unlike PEFT, sparse fine-tuning introduces no new parameters: it locates a small subset of task-critical neurons already present in the pre-trained LLM (Dai et al., 2022) and updates only those. Concentrating the learning budget on these neurons is intended to suffice for task adaptation, while leaving the remaining parameters unchanged limits unnecessary drift from pre-trained knowledge (Wu et al., 2026b). Prior studies frequently report that this outperforms FFT on the target task, at a substantially smaller update cost and with stronger retention of general capability (Zhang et al., 2025; Yamaguchi et al., 2026).

To date, a large number of sparse fine-tuning methods have been proposed; however, the selection criteria adopted by different studies differ markedly, and a systematic, paradigm-level organisation and analysis of these methods is still lacking. As summarised in Table 1, existing methods can be distinguished along three dimensions: the criterion used to judge neuronal importance, the schedules of mask construction, and the network modules over which selection is performed. This comparison reveals substantial divergence at the method level: the eight representative methods differ not only in the criterion used to identify task-relevant neurons, but also in the scope of selected modules and the moment at which the mask is built, precluding a direct, like-for-like comparison of their reported



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.

Table 1: Summary of the eight representative sparse fine-tuning methods studied.

| Method | Criterion | Schedule | Location |
|---|---|---|---|
| $S^2$FT (Yang et al., 2024d) | **Low activation**: smallest L2 norm of channel activations on a calibration set. | Static | MHA+FFN |
| NeFT (Xu et al., 2025) | **Weight drift**: lowest cosine similarity of weights before and after a short FFT. | Static | Only FFN |
| NCFT (Leng & Xiong, 2025) | **High gradient & activation**: top k% neurons by activation times loss gradient (Taylor attribution). | Static | Only FFN |
| NMKE (Liu et al., 2025) | **Perturbation & entropy**: log-prob gain under activation amplification; ratio from attribution entropy. | Dynamic (per batch) | Only FFN (4-8 layers) |
| SSU (Yamaguchi et al., 2026) | **Wanda**: weight magnitude times activation norm per column. | Static | MHA+FFN |
| NeuroAda (Zhang et al., 2025) | **High weight magnitude**: largest incoming weights per neuron get a zero-init bypass; task-agnostic. | Static | MHA+FFN |
| GMT (Li et al., 2025) | **High gradient**: mean absolute loss gradient over batches; values above a preset quantile are updated. | Dynamic (per step) | MHA+FFN |
| NGFT (Wu et al., 2026b) | **High activation & low variation**: positive activations with low coefficient of variation. | Static | Only FFN |

performance. Several basic questions therefore remain open: **(1)** Does sparse fine-tuning deliver consistent gains across parameter scales, model families, and generations, and does it outperform FFT and PEFT? **(2)** How large a fraction of neurons should be updated? Is the optimal neuron-update ratio universal across methods and tasks, or does it vary from one to another? **(3)** Are task-relevant neurons concentrated in particular modules (the MHA or the FFN), or widely distributed across the whole network? **(4)** Masks can be induced once before fine-tuning or regenerated in real time during training: does dynamic regeneration genuinely help, and at what cost? **(5)** Although existing works report impressive accuracies, the mechanism behind the effectiveness of sparse fine-tuning remains insufficiently explored. The conclusions of related studies even contradict one another: $S^2$FT focuses on neurons with low activations (Yang et al., 2024d), whereas NCFT and NGFT focus on those with high activations (Leng & Xiong, 2025; Wu et al., 2026b), yet all three improve over FFT; how can this contradiction be explained? To answer these questions, we conduct a unified study on commonly used mathematical reasoning and code generation tasks. Our main findings on the performance of sparse fine-tuning are as follows:

- **Sparse fine-tuning applies to LLMs of different scales, families, and generations, consistently outperforming FFT and PEFT methods.** Across both tasks, sparse fine-tuning outperforms FFT by 2.1 points on average while retaining 96–109% of the base model's general capability, thereby combining effective adaptation with strong resistance to forgetting (Section 4.1, 4.2). PEFT offers similar retention but reaches a lower performance ceiling: on Llama3.1-8B math, the strongest PEFT method gains only 1.1 points over FFT, versus 2.6 for NGFT (Section 4.6).
- **Different neuron-update ratios lead to different effects, and adaptively adjusting this ratio is the key ingredient missing from most existing studies.** Ablating the neuron-update ratio across all sparse fine-tuning methods, we find that this ratio significantly affects performance and that the optimal ratio differs from one method to another. Most methods fix the ratio via one-off ablation, a local rather than global optimum (Section 4.3).
- **The position of the located neurons is not clearly correlated with the fine-tuning result.** Whether the selected neurons lie in the MHA, the FFN, or both (MHA+FFN), sparse fine-tuning performance shows no significant fluctuation (within about 0.2 points), suggesting task-relevant neurons are spread across the model rather than concentrated in any single region (Section 4.4).
- **Dynamic mask regeneration offers modest task gains while risking general capability and adding recurring overhead.** In the controlled schedule crossover (Section 4.5), with the criterion fixed, forcing the originally static methods to regenerate their mask raises task scores by at most only 0.4 but costs up to 0.7 general capability points; freezing the mask of the dynamic NMKE and GMT slightly lowers their task scores yet improves their general capability by 1.1 and 0.5 points on mathematics.

- **The intrinsic mechanism of sparse fine-tuning stems from the two complementary roles of the sparse mask: a capacity allocator and a regulariser.** The mask concentrates the learning budget on a few critical parameter dimensions, so updating only a small fraction of neurons attains task-performance gains on par with FFT; meanwhile, the mask pins all remaining parameters at their pre-trained values, mechanically safeguarding the model's general capability. The true value of different sparse fine-tuning methods lies in confining the adaptation process to the set of neurons that exert a substantial influence on the model's behaviour (Section 4.7).

## 2 Related Work

Neuron-level interpretability studies suggest that modifying a small subset of neurons can effectively steer and improve LLM behaviour (Yang et al., 2024b; Li & Kim, 2026). Motivated by these findings, sparse fine-tuning updates only neurons selected by importance criteria, achieving performance comparable to or better than FFT (Yang et al., 2024d; Wu et al., 2026b). Existing analyses, however, either survey neuron attribution for model alignment (Zhang et al., 2026a) or examine communication savings in distributed sparse fine-tuning (Sung et al., 2021). Compared with FFT (Zhang et al., 2026b) and PEFT (Liu et al., 2026), which have already been systematically studied, sparse fine-tuning still lacks systematic comparisons across tasks, model families, and parameter scales, and its underlying mechanisms remain poorly understood. This work addresses these gaps.

## 3 Methodology

Modern LLMs are decoder-only Transformers (Vaswani et al., 2023), each a stack of $L$ blocks composed of a multi-head attention (MHA) sub-block and a feed-forward network (FFN) sub-block (Zhao et al., 2026), with normalisation and residual connections. Let $Z_{\ell-1} \in \mathbb{R}^{n\times d}$ denote the hidden states entering block $\ell \in [L]$, where $n$ is the sequence length and $d$ the hidden size, one token per row, and let $\mathrm{Norm}(\cdot)$ denote row-wise normalisation. The block computes

$$Z'_\ell = Z_{\ell-1} + \mathrm{MHA}\big(\mathrm{Norm}(Z_{\ell-1})\big), \qquad Z_\ell = Z'_\ell + \mathrm{FFN}\big(\mathrm{Norm}(Z'_\ell)\big), \tag{1}$$

**Multi-head attention.** The MHA sub-block models dependencies across token positions, allowing each token to aggregate context from preceding tokens. For an input $X \in \mathbb{R}^{n\times d}$, it combines $H$ heads of dimension $d_h = d/H$:

$$\mathrm{MHA}(X) = \sum_{h=1}^{H} \mathrm{softmax}\left(\frac{(XW_h^Q)(XW_h^K)^\top}{\sqrt{d_h}} + M\right)(XW_h^V)W_h^O, \tag{2}$$

where $W_h^Q, W_h^K, W_h^V \in \mathbb{R}^{d\times d_h}$, $W_h^O \in \mathbb{R}^{d_h\times d}$, and $M$ is the causal mask. Following Zhang et al. (2026a), we define an MHA neuron $n_{\ell,j}$ as attention channel $j$: column $j$ of $W^Q$, $W^K$, and $W^V$, together with row $j$ of $W^O$. Each head therefore comprises $d_h$ consecutive neurons.

**Feed-forward network.** The FFN sub-block acts as a position-wise feature transformer, operating independently on each token. Current LLMs adopt the gated variant:

$$\mathrm{FFN}(X') = \sigma(X'W_{\mathrm{gate}}) \odot (X'W_{\mathrm{up}})W_{\mathrm{down}}, \tag{3}$$

where $X' \in \mathbb{R}^{n\times d}$ is the input to the FFN, $W_{\mathrm{gate}}, W_{\mathrm{up}} \in \mathbb{R}^{d\times d_{\mathrm{ff}}}$, $W_{\mathrm{down}} \in \mathbb{R}^{d_{\mathrm{ff}}\times d}$, $\sigma$ is an element-wise non-linearity (e.g., SiLU), and $\odot$ denotes the element-wise product. In the FFN of block $\ell$, we define the $j$-th neuron $n_{\ell,j}$, $j \in [d_{\mathrm{ff}}]$; its parameters are column $j$ of $W_{\mathrm{gate}}$ and $W_{\mathrm{up}}$, together with row $j$ of $W_{\mathrm{down}}$ (Wu et al., 2026b).

**Binary mask induction and constrained updates.** At step $t$, sparse fine-tuning selects a set $\mathcal{S}_t$ of task-relevant neurons and updates only their associated parameters:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta\left(\hat{\mathbf{g}}_t \odot \mathbf{m}_t\right), \qquad m_{\ell,j}^{(t)} = \mathbb{I}[n_{\ell,j} \in \mathcal{S}_t], \tag{4}$$

where $\mathbb{I}[\cdot]$ is the indicator function, $\hat{\mathbf{g}}_t$ is the (possibly preconditioned) gradient, and $\odot$ denotes element-wise multiplication. Each neuron indicator $m_{\ell,j}^{(t)}$ is assigned to every parameter coordinate associated with neuron $n_{\ell,j}$, including its row and column slices, thereby inducing the parameter-level mask $\mathbf{m}_t \in \{0,1\}^p$. Masked coordinates receive no task-gradient update and, with consistent masking of weight decay and optimiser states, remain at their pre-trained values (Xu et al., 2025).

**Mask construction schedules: static and dynamic.** The methods in Table 1 differ in when and from how much data they estimate neuronal importance, and consequently in how the gradient mask $\mathbf{m}_t$ of Equation 4 is generated. Algorithm 1 formalises both procedures. A static method runs its selection rule $g$ once, before fine-tuning, on a relatively large selection set $\mathcal{D}_{\text{sel}}$, typically comprising hundreds or thousands of examples. It selects the neurons within the neuron-update ratio $\rho$, and freezes the resulting mask for every subsequent update, incurring the selection cost only once. A dynamic method instead recomputes importance during fine-tuning from the current mini-batch $\mathcal{D}_t$. At each refresh step $t \in \mathcal{T}_{\text{sel}}$, it replaces $\mathcal{S}_{t-1}$ with $\mathcal{S}_t = g(\boldsymbol{\theta}_t, \mathcal{D}_t; \rho)$ and induces a new mask $\mathbf{m}_t$. Dynamic selection can track changes in both the model and the training data, but its selection cost recurs throughout training (Liu et al., 2025; Li et al., 2025).

# 4 Experiments

## 4.1 Experiments on Mathematical Reasoning

**Setup.** We evaluate the eight sparse methods on three models from two families: Qwen2.5-Math-1.5B and Qwen2.5-Math-7B (Yang et al., 2024c), together with Llama3.1-8B (Grattafiori et al., 2024), with FFT and the un-fine-tuned base model as references. Following (Wu et al., 2026a), for the mathematical reasoning task, all methods are trained on NuminaMath-CoT (LI et al., 2024), from which we sample 100k instances balanced across its categories. Following (Zhan et al., 2025), for the mathematical domain, we report four widely recognised benchmarks: GSM8K (Cobbe et al., 2021), MATH500 (Hendrycks et al., 2021), Minerva (Lewkowycz et al., 2022), and OlympiadBench (He et al., 2024). For the general domain, we adopt four benchmarks that are commonly used to profile a model's general capabilities: MMLU-Pro (Wang et al., 2024), HellaSwag (Zellers et al., 2019), ARC (Clark et al., 2018), and TyDiQA (Clark et al., 2020). Detailed descriptions of all benchmarks are provided in Appendix A. The hyperparameters shared across methods (learning rate, batch size, and decay parameters) are kept identical, and each method's own unique hyperparameters are fixed to the values reported as optimal in its original paper. Details are provided in Appendix A.3.

**Main Result.** As shown in Table 2, FFT buys task performance at the price of the base model's general capability: on Llama3.1-8B, FFT gains only $+4.1$ MATH points while its GEN drops by nearly 9. Sparse fine-tuning breaks this trade-off: in 21 of the 24 method–model settings it matches or exceeds the MATH gain of FFT, while retaining 96%–109% of the base general capability, exceeding the base outright on Llama3.1-8B. Moreover, *the stronger the base model's capability, the more pronounced the advantage of sparse fine-tuning.* Ordered by base MATH score, the average margin of the eight methods over FFT grows monotonically: $+0.8$ points on Qwen2.5-Math-1.5B, $+1.1$ on Llama3.1-8B, and $+3.1$ on Qwen2.5-Math-7B.

**Performance and stability.** A method's rank stability is closely tied to its neuron selection mechanism, and averaging each method's rank across the three settings makes the differences clear. NGFT is the most stable of all (2.0 on MATH, 2.7 on GEN), a direct benefit of its dual-indicator criterion and adaptive neuron-update ratio. The remaining multi-criterion methods each strike this trade-off differently. NCFT (2.7/4.0) fuses gradient and activation and wins the best MATH score at 1.5B (50.2) but fluctuates across scales. NMKE (2.7/6.7) finds task-critical neurons reliably by activation perturbation, yet updates many neurons involved in semantic understanding, weakening its resistance to forgetting. SSU, built on the two-factor Wanda criterion, attains the highest MATH score of this study on Qwen2.5-Math-7B (56.7), but on Llama3.1-8B it is the only sparse method whose MATH falls below FFT. NeFT (4.7/1.7), locating neurons by weight drift, preserves general capability best but is also unstable on downstream tasks.

By contrast, *methods that rely on a single criterion find it far harder to have it both ways.* $S^2$FT, which deliberately selects the lowest-activation channels, adapts most conservatively: bottom-two MATH on both Qwen panels and only upper-middle on Llama3.1-8B (average 6.3) against the best GEN at 1.5B and the second-best at 7B (average 3.3), confirming that perturbing quiet neurons disturbs the pre-trained distribution least. NeuroAda, which selects by weight magnitude alone with no task-specific criterion involved, averages 6.7 on MATH and 4.3 on GEN, showing little downstream specificity. GMT, re-selecting the highest-gradient parameters at every step, stays mid-pack throughout, without any distinguishing strength.

Table 2: Main results on Qwen2.5-Math-1.5B, Qwen2.5-Math-7B, and Llama3.1-8B. MATH and GEN are the averages over the four mathematical reasoning and the four general capability columns, respectively. For each model, the top-three results among all sparse fine-tuning methods in the MATH and GEN columns are shaded in blue, with darker shades indicating better ranks.

| | | Mathematical reasoning | | | | General capability | | | | Average | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Model** | **Method** | GSM8K | MATH500 | Minerva | Olympiad | MMLU (P) | HellaSwag | ARC | TyDiQA | Math | Gen |
| Qwen2.5-Math 1.5B | Base | 55.0 | 43.6 | 5.1 | 25.2 | 27.6 | 49.8 | 40.7 | 33.0 | 32.2 | 37.8 |
| | FFT | 81.2 | 60.2 | 22.1 | 28.4 | 25.9 | 46.0 | 34.8 | 25.5 | 48.0 | 33.1 |
| | $S^2$FT | 81.0 | 62.0 | 20.6 | 26.8 | 28.3 | 49.6 | 41.0 | 31.6 | 47.6 | 37.6 |
| | SSU | 82.3 | 64.6 | 21.7 | 29.2 | 28.5 | 49.4 | 40.3 | 29.2 | 49.4 | 36.9 |
| | NCFT | 82.3 | 64.4 | 23.9 | 30.2 | 29.9 | 48.5 | 39.7 | 30.2 | 50.2 | 37.1 |
| | NeFT | 81.7 | 63.2 | 20.6 | 29.3 | 31.1 | 49.1 | 39.8 | 29.8 | 48.7 | 37.5 |
| | NGFT | 82.9 | 63.6 | 23.5 | 30.4 | 31.2 | 49.0 | 39.5 | 30.2 | 50.1 | 37.5 |
| | NeuroAda | 80.7 | 59.8 | 20.6 | 26.4 | 28.2 | 48.3 | 39.8 | 30.4 | 46.9 | 36.7 |
| | GMT | 81.6 | 60.4 | 21.9 | 28.7 | 28.5 | 48.2 | 39.5 | 30.1 | 48.2 | 36.6 |
| | NMKE | 81.7 | 62.8 | 22.8 | 31.0 | 28.9 | 49.7 | 40.1 | 26.5 | 49.6 | 36.3 |
| Qwen2.5-Math 7B | Base | 69.4 | 60.4 | 16.9 | 27.7 | 39.4 | 65.3 | 50.4 | 49.8 | 43.6 | 51.2 |
| | FFT | 85.8 | 62.0 | 28.7 | 32.0 | 38.5 | 62.8 | 46.5 | 20.9 | 52.1 | 42.2 |
| | $S^2$FT | 89.2 | 68.4 | 18.4 | 34.7 | 40.2 | 65.2 | 51.1 | 48.6 | 52.7 | 51.3 |
| | SSU | 88.7 | 72.0 | 29.8 | 36.4 | 40.3 | 64.9 | 49.0 | 42.9 | 56.7 | 49.3 |
| | NCFT | 88.6 | 66.8 | 29.4 | 36.3 | 42.2 | 65.7 | 51.0 | 38.1 | 55.3 | 49.3 |
| | NeFT | 88.1 | 69.2 | 28.7 | 37.5 | 42.6 | 66.4 | 52.7 | 46.3 | 55.9 | 52.0 |
| | NGFT | 89.2 | 68.0 | 29.6 | 37.3 | 42.0 | 65.4 | 50.2 | 46.1 | 56.0 | 50.9 |
| | NeuroAda | 87.9 | 65.8 | 27.8 | 34.8 | 41.8 | 65.3 | 50.7 | 45.2 | 54.1 | 50.8 |
| | GMT | 88.3 | 66.8 | 28.4 | 34.6 | 41.3 | 65.1 | 51.7 | 44.6 | 54.5 | 50.7 |
| | NMKE | 89.2 | 70.6 | 30.1 | 36.6 | 42.2 | 65.6 | 51.1 | 38.7 | 56.6 | 49.4 |
| Llama3.1-8B | Base | 83.1 | 39.8 | 16.0 | 13.3 | 44.6 | 79.2 | 55.4 | 34.1 | 38.1 | 53.3 |
| | FFT | 84.2 | 47.6 | 19.2 | 17.6 | 24.3 | 69.3 | 40.7 | 43.3 | 42.2 | 44.4 |
| | $S^2$FT | 83.2 | 49.0 | 21.7 | 18.8 | 43.6 | 79.1 | 52.8 | 45.7 | 43.2 | 55.3 |
| | SSU | 84.4 | 49.0 | 15.4 | 19.4 | 40.5 | 79.6 | 52.6 | 47.9 | 42.1 | 55.1 |
| | NCFT | 85.7 | 50.0 | 22.1 | 18.7 | 44.1 | 78.8 | 49.3 | 60.1 | 44.1 | 58.1 |
| | NeFT | 84.2 | 49.8 | 19.6 | 18.5 | 44.6 | 79.3 | 50.7 | 56.0 | 43.0 | 57.7 |
| | NGFT | 84.6 | 52.4 | 23.2 | 18.9 | 43.0 | 78.6 | 50.2 | 57.6 | 44.8 | 57.4 |
| | NeuroAda | 84.7 | 48.2 | 20.5 | 18.7 | 43.7 | 78.4 | 50.4 | 57.2 | 43.0 | 57.4 |
| | GMT | 84.2 | 48.6 | 20.7 | 18.4 | 44.3 | 77.8 | 50.8 | 56.7 | 43.0 | 57.4 |
| | NMKE | 85.6 | 47.8 | 20.6 | 19.7 | 44.2 | 78.6 | 50.6 | 53.5 | 43.4 | 56.7 |

### 4.2 EXPERIMENTS ON CODE GENERATION

**Setup.** To test whether the conclusions of Section 4.1 transfer beyond mathematical reasoning, we repeat the unified protocol on code generation with two models: the Qwen2.5-Coder-3B (Hui et al., 2024) and the Llama3.1-8B. All methods are trained on OpenCodeInstruct-2 (Ahmad et al., 2025), from which we sample 100k instances. Except for the task and its benchmarks, all settings follow Section 4.1. For the code domain, we report three benchmarks: HumanEval+ (Liu et al., 2023), MBPP+ (Liu et al., 2023), and LiveCodeBench v6 (Jain et al., 2025). Detailed descriptions are likewise provided in Appendix A.1.

**Main Results.** FFT behaves much as it did on mathematics, gaining code performance at a general capability price. The sparse fine-tuning methods again alleviate this trade-off: they outperform FFT by 2.7 points in CODE and 10.9 in GEN. Two trends at the top of the ranking likewise persist. NGFT remains strong on both models, leading both columns on Qwen2.5-Coder-3B (CODE 50.8, GEN 54.6); NMKE is again the runner-up, ranking second in CODE on Qwen2.5-Coder-3B (50.1). **Their strong performance plausibly benefits from multi-faceted neuron localisation combined with adaptive neuron selection.** Finally, similar to Section 4.1, the advantage of sparse fine-tuning again tracks base-model capability: despite its smaller size, the code-specialised Qwen2.5-Coder-3B has a stronger base CODE score than Llama3.1-8B (41.7 vs. 40.3), and its average margin over FFT is correspondingly larger (+4.4 vs. +1.0 points).

**Additional findings from the code generation task.** The behaviour of most other methods also carries over, with one exception. $S^2$FT achieves the best CODE score on Llama3.1-8B (44.7) but performs unremarkably on the code-specialised Qwen-Coder. Similarly, it ranks in the bottom two on the Qwen2.5-Math models but fourth on Llama3.1-8B. *We conjecture that $S^2$FT benefits general-purpose models with weak target-domain capability, whose low-activation neurons retain unused capacity, but offers less to domain-specialised models where that capacity has been consumed.* Fi-

Table 3: Code-generation results on Qwen2.5-Coder-3B and Llama3.1-8B. CODE and GEN are the averages over the three code generation and the four general capability columns, respectively. For each model, the top-three results among all sparse fine-tuning methods in the CODE and GEN columns are shaded in blue, with darker shades indicating better ranks.

| | | Code generation | | | General capability | | | | Average | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Model** | **Method** | HumanEval+ | MBPP+ | LiveCodeBench v6 | MMLU (P) | HellaSwag | ARC | TyDiQA | Code | Gen |
| Qwen2.5-Coder 3B | Base | 47.0 | 61.9 | 16.1 | 34.5 | 70.2 | 46.5 | 64.2 | 41.7 | 53.9 |
| | FFT | 56.1 | 57.4 | 17.0 | 21.9 | 64.6 | 45.3 | 48.4 | 43.5 | 45.1 |
| | $S^2$FT | 50.6 | 65.3 | 21.7 | 34.4 | 70.3 | 46.0 | 64.1 | 45.9 | 53.7 |
| | SSU | 63.2 | 66.6 | 17.9 | 34.9 | 70.1 | 47.4 | 61.9 | 49.2 | 53.6 |
| | NCFT | 60.9 | 67.9 | 21.1 | 35.3 | 71.0 | 48.5 | 61.7 | 50.0 | 54.1 |
| | NeFT | 50.6 | 62.7 | 19.5 | 33.7 | 69.8 | 44.8 | 61.0 | 44.3 | 52.3 |
| | NGFT | 65.2 | 67.2 | 20.0 | 34.9 | 71.0 | 48.8 | 63.6 | 50.8 | 54.6 |
| | NeuroAda | 58.1 | 61.5 | 17.8 | 34.5 | 69.2 | 46.8 | 62.0 | 45.8 | 53.1 |
| | GMT | 57.6 | 64.8 | 18.4 | 34.7 | 70.2 | 45.9 | 61.4 | 46.9 | 53.1 |
| | NMKE | 59.1 | 69.3 | 21.8 | 35.8 | 70.6 | 47.2 | 61.3 | 50.1 | 53.7 |
| Llama3.1-8B | Base | 54.7 | 57.5 | 8.6 | 44.6 | 79.2 | 55.4 | 34.1 | 40.3 | 53.3 |
| | FFT | 53.5 | 59.8 | 11.6 | 31.1 | 72.2 | 41.0 | 30.3 | 41.6 | 43.7 |
| | $S^2$FT | 56.7 | 63.2 | 14.2 | 44.7 | 79.1 | 55.6 | 41.0 | 44.7 | 55.1 |
| | SSU | 60.9 | 60.8 | 9.2 | 44.6 | 79.3 | 55.2 | 43.0 | 43.6 | 55.5 |
| | NCFT | 54.8 | 61.1 | 8.7 | 43.8 | 78.5 | 54.4 | 58.1 | 41.5 | 58.7 |
| | NeFT | 54.2 | 61.9 | 11.4 | 44.1 | 79.2 | 54.6 | 49.3 | 42.5 | 56.8 |
| | NGFT | 56.1 | 61.5 | 10.6 | 44.3 | 78.5 | 55.2 | 52.7 | 42.7 | 57.7 |
| | NeuroAda | 54.3 | 60.1 | 7.2 | 43.7 | 78.4 | 55.3 | 50.5 | 40.5 | 57.0 |
| | GMT | 56.3 | 60.5 | 8.6 | 44.1 | 78.6 | 54.9 | 51.6 | 41.8 | 57.3 |
| | NMKE | 56.7 | 62.7 | 10.1 | 44.0 | 78.7 | 55.2 | 55.9 | 43.2 | 58.5 |

nally, on LiveCodeBench v6, Llama3.1-8B has the lowest base score (8.6), and every sparse method except $S^2$FT trails FFT. *This suggests that sparse fine-tuning partly depends on base-model capability and offers limited gains when it is low, in which case FFT is the safer choice.*

### 4.3 ABLATION ON THE NEURON-UPDATE RATIO

This section examines how the neuron-update ratio $\rho$ affects sparse fine-tuning. Most methods fix the ratio: $S^2$FT, NCFT, and NeFT tune it via ablation and keep it constant during training, whereas NMKE and NGFT determine it adaptively. Whether $\rho$ materially affects performance, or whether fixed-ratio choices generalise across settings, remains open. To isolate its effect, we sweep $\rho \in \{1, 3, 5, 7, 10, 15, 20, 30\}\%$ on Llama3.1-8B for the three fixed-ratio methods, with FFT ($\rho = 100\%$) as reference and the adaptive methods (NGFT and NMKE) marked at their self-selected points. Figure 1 reports task accuracy and general capability for mathematics and code.

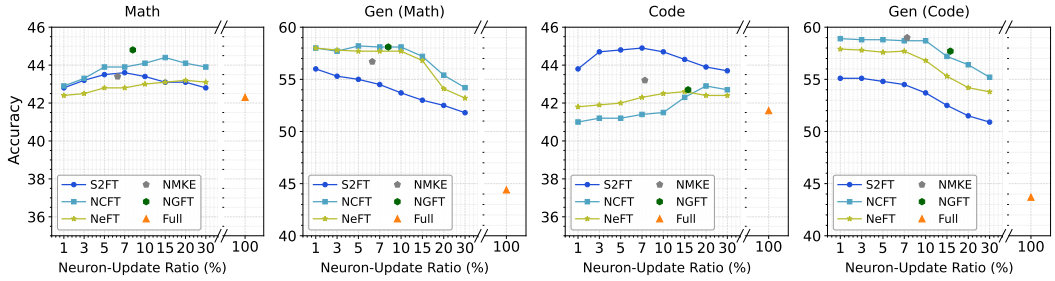


Figure 1: Effect of the neuron-update ratio $\rho$ on Llama3.1-8B. The three fixed-mask methods ($S^2$FT, NCFT, NeFT) sweep $\rho$ from 1% to 30%; FFT ($\rho = 100\%$) is the isolated point on the broken axis. NMKE and NGFT set $\rho$ adaptively and are marked at their chosen operating points.

**The optimal neuron-update ratio varies across methods and tasks, making adaptive neuron selection essential.** As shown in Figure 1, the task-accuracy peak depends on both the importance criterion and the task itself. On mathematics, $S^2$FT and NeFT peak at around 10% whereas NCFT peaks at around 15%; on code, the peaks of the same methods scatter between 7% ($S^2$FT) and 20% (NCFT), so a ratio tuned for a single method or task is at best locally optimal and fails to generalise.

By contrast, the adaptive methods adjust the neuron-update ratio automatically according to the task: NGFT's self-selected ratio shifts from $8.2\%$ on mathematics to $15.8\%$ on code, tracking the task-dependent peaks, and its mathematics score ($44.8$) exceeds every static sweep point; NMKE likewise adjusts from $6.3\%$ to $7.4\%$ and stays near the Pareto frontier on both tasks. In contrast to the local optimality of fixed ratios, adaptivity stays close to the task-dependent optimum on both tasks without any manual tuning, avoiding the per-method, per-task search that fixed ratios require; it is therefore an efficient design choice worth advocating. At the same time, the ratio is tightly coupled with catastrophic forgetting: for every method, general capability declines steadily as $\rho$ increases. NeFT, for instance, loses $4.8$ points on mathematics and $4.1$ on code, showing that updating more neurons than necessary directly harms the base model. The ratio should therefore be treated as a two-sided trade-off, for which a smaller update is generally preferable.

### 4.4 Ablation on the Neuron-Update Location

As Table 1 shows, some sparse fine-tuning methods focus their updates exclusively on the FFN, while others extend selection to both MHA and FFN. Whether this choice materially affects performance, however, has not been isolated from the importance criterion. A preliminary comparison in Tables 2 and 3 suggests that *location may have a weak effect*: on Qwen2.5-Math-7B, the top MHA+FFN method (SSU, 56.7 Math) edges out the top FFN-only method (NGFT, 56.0), yet on general capability the order reverses (50.9 vs. 49.3). On Llama3.1-8B code, the same swap occurs. Because each method also differs in its importance criterion, this comparison is confounded. Moreover, although recent work has begun exploring the MHA module (Zhou et al., 2025), its role in sparse fine-tuning remains unclear. To isolate the effect of location, we take NGFT and NCFT, both originally configured for FFN-only selection, and create two additional variants per method: one selecting neurons only within MHA, and one selecting across both MHA and FFN. This yields three location settings with the importance criterion and all other hyperparameters held fixed.

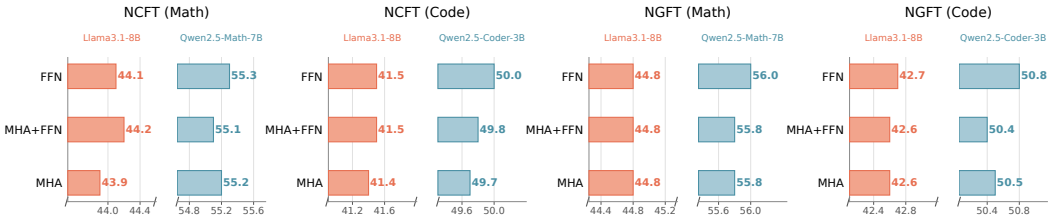


Figure 2: Ablation on selection location for NCFT and NGFT across mathematical reasoning and code generation tasks. Each panel shows horizontal bars of task scores for Llama3.1 and Qwen2.5 under three configurations: FFN-only, MHA+FFN, and MHA-only.

**Neuron position does not strongly correlate with fine-tuning effectiveness.** As shown in Figure 2, the gaps among the three configurations are small: performance fluctuates by about $0.2$ points, and every configuration outperforms FFT. Although a slight ordering of FFN $>$ MHA+FFN $>$ MHA emerges, the contribution of the MHA should not be overlooked, as the MHA-only configuration remains close to the other two. This suggests that task-relevant neurons are spread throughout the network rather than concentrated in a single module, and the selection location has only a subtle influence on sparse fine-tuning performance, far smaller than that of different selection criteria.

### 4.5 Static vs. Real-Time Gradient Masking

Among the eight sparse fine-tuning methods compared, GMT and NMKE dynamically regenerate their masks: GMT updates its mask using averaged gradient magnitude, whereas NMKE re-selects neurons based on activation perturbation and attribution entropy. Tables 2 and 3 show markedly different outcomes. NMKE consistently ranks among the top three across all settings, while GMT remains in the middle and never enters the leading group. Since the two methods use substantially different selection criteria, *it therefore remains unclear whether real-time mask regeneration itself is truly beneficial.* To isolate the mask schedule, we keep each method's selection criterion fixed while swapping its schedule: static methods refresh masks during training according to $\mathcal{T}_{\text{sel}}$ (Algorithm 1),

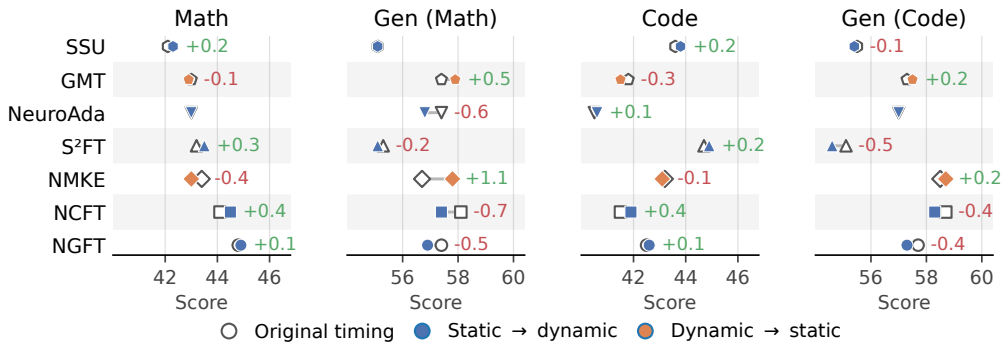


Figure 3: Crossover effect on Llama3.1-8B: each dumbbell pairs a method's original mask schedule (open marker, Tables 2 and 3) with the swapped schedule (filled marker, Tables 4 and 5); blue marks static→dynamic and orange dynamic→static conversions, with the signed change at the right.

whereas dynamic methods use a frozen mask. NeFT is excluded because its before-and-after criterion cannot be evaluated online. The swap changes only $\mathcal{T}_{\text{sel}}$ and the calibration statistics supplied to $g$. Implementation details for each method are given in Appendix B.

**Mask regeneration enlarges cumulative coverage, trading general capability for a marginal task gain.** As shown in Figure 3, mask regeneration improves the five originally static methods by $0.2$ task points on average (at most $0.4$), while fixing the masks of the two dynamic methods reduces task performance by $0.1$–$0.4$ points. This consistent task benefit is therefore modest. General capability shows a stronger trend in the opposite direction: regeneration costs the static methods up to $0.7$ points, whereas fixed masks improve GMT by $0.5$/$0.2$ points and NMKE by $1.1$/$0.2$ points on mathematics/code. The two trends share a common explanation: although regeneration preserves the per-step neuron-update ratio, each refresh expands the union of updated neurons, leaving fewer parameters at their pre-trained values (Figure 14). It therefore increases effective capacity, yielding small task gains, but weakens regularisation and harms general capability; a fixed mask has the reverse effect. Overall, real-time regeneration trades general capability for marginal task gains while incurring recurring selection overhead.

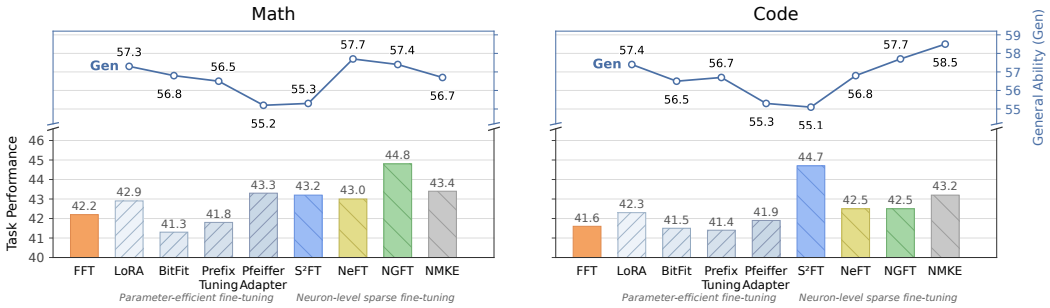


Figure 4: Task performance (bars) and general capability (line) of PEFT and sparse fine-tuning on Llama3.1-8B across mathematical reasoning and code generation tasks. Sparse fine-tuning attains the highest task scores while retaining general ability on par with PEFT.

### 4.6 Comparison with Parameter-Efficient Fine-Tuning

**Setup.** We compare against four representative PEFT methods, each from a distinct paradigm: **Prefix Tuning** (Li & Liang, 2021) (prompt-based), **Pfeiffer** (Pfeiffer et al., 2021) (adapter-based), **BitFit** (Zaken et al., 2022) (direct selective tuning), and **LoRA** (Hu et al., 2022) (efficient selective tuning). Experiments are conducted on Llama3.1-8B using the same mathematical reasoning and code generation benchmarks as in Section 4. Detailed method descriptions and hyperparameter settings are provided in Appendix A.2.

**Sparse fine-tuning retains general capability on par with PEFT while pushing the task-performance ceiling further.** As shown in Figure 4, all four PEFT methods freeze the backbone and train only extra modules or bias terms, which provides strong protection for the model's general capability: after fine-tuning Llama3.1-8B, their GEN scores remain stable in the 55.2–57.4 band, 11–14 points above FFT and roughly on par with the sparse fine-tuning methods. However, they also exhibit a capability ceiling: the improvements of all four methods over FFT are limited, and even the strongest performer, the Pfeiffer Adapter, exceeds FFT by only 1.1 points on the mathematical task. Moreover, some PEFT methods even underperform FFT: on mathematics, BitFit (41.3) and Prefix Tuning (41.8) both fall below FFT (42.2), indicating that this indirect parameterisation limits the extent to which new behaviour can be injected into a frozen model. In contrast, sparse fine-tuning methods push this ceiling further: NGFT reaches 44.8 on mathematics, 2.6 points above FFT, while S$^2$FT reaches 44.7 on code, 3.1 points above FFT. This shows that, unlike PEFT methods that add or modify auxiliary parameters, sparse fine-tuning, which directly updates pre-trained weights, can break through the model's capability ceiling more thoroughly.

### 4.7 WHY SPARSE FINE-TUNING WORKS

Neuron attribution studies have shown that LLMs harbour sparsely distributed, causally important neurons: editing these neurons selectively modifies the targeted behaviour without degrading other factual knowledge (Dai et al., 2022; Christ et al., 2025). Drawing on both prior work and our experimental results, we hypothesise that the sparse mask of sparse fine-tuning plays two complementary roles that trade off well: (1) **Capacity allocator.** The sparse mask concentrates the learning budget on the few parameter dimensions that matter, so updating only a small fraction of neurons, free from interference by irrelevant ones, attains a task gain comparable to updating all of them. In Figure 5, the data points of sparse fine-tuning cover the same or a higher range on the task axis as the FFT stars. (2) **Regulariser.** The sparse mask pins all remaining parameters at their pre-trained values, mechanically guaranteeing that the model's general capability does not degrade severely; in Figure 5, the data points of sparse fine-tuning all lie close to the $\Delta$Gen=0 baseline, whereas the FFT stars lie far from it. Experiments with randomly sampled masks (Appendix D) further support this view: even a random mask at the same sparsity yields task performance close to FFT, indicating that these two roles are inherent properties of the masked-update constraint itself. This does not mean, however, that which neurons are selected is irrelevant: at the same sparsity, random masks still fall short of most criterion-guided methods, suggesting that a substantial part of the benefit comes from constraining the adaptation to a small set of behaviourally influential neurons. In summary, the effectiveness of sparse fine-tuning arises from the joint effect of the sparse mask and the neuron selection criterion.

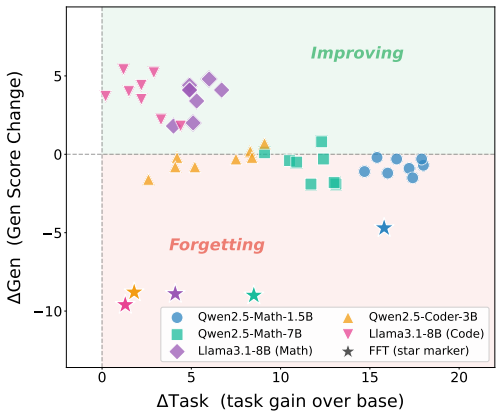


Figure 5: Task gain ($x$-axis) versus general capability change ($y$-axis). The $\Delta$Task=0 and $\Delta$Gen=0 lines indicate no change in task performance and general capability.

## 5 CONCLUSION

This work presents a systematic and unifying empirical study of sparse fine-tuning. We build a unified evaluation framework spanning diverse tasks, model families, and parameter scales for reproducible, like-for-like comparisons, and release a performance leaderboard of sparse fine-tuning methods. Drawing on experimental results, we analyse the strengths and limitations of different sparse fine-tuning methods, and conduct ablation studies along three key dimensions: the neuron-update ratio, the neuron-update location, and the mask construction strategy. Beyond these empirical findings, we further investigate why sparse fine-tuning works at the mechanism level and offer new insights into its underlying causes. We hope that this study provides practical guidance for selecting and applying sparse fine-tuning methods, and inspires future research on this paradigm.

### AI USE STATEMENT

In this work, we used generative AI tools to assist with grammar and language polishing of the manuscript, and with debugging the experimental code. We have reviewed all AI-assisted work and take full responsibility for the content of this work, including text and artefacts produced with the aid of generative AI.

### REPRODUCIBILITY STATEMENT

All sparse fine-tuning methods are implemented with the official code publicly released by their original authors on GitHub, using the default hyperparameters reported in the original papers and repositories (Appendix A.3). For methods whose released code involves only the neuron-selection step, we keep their selection procedure unchanged and conduct all fine-tuning experiments in a unified LLaMA-Factory framework to ensure like-for-like comparisons. All training and evaluation datasets used in this work are publicly available; benchmark details are given in Appendix A.1.

## REFERENCES

Wasi Uddin Ahmad, Aleksander Ficek, Mehrzad Samadi, Jocelyn Huang, Vahid Noroozi, Somshubra Majumdar, and Boris Ginsburg. Opencodeinstruct: A large-scale instruction tuning dataset for code llms, 2025. URL https://arxiv.org/abs/2504.04030.

Bryan R Christ, Zachary Gottesman, Jonathan Kropko, and Thomas Hartvigsen. Math neurosurgery: Isolating language models' math reasoning abilities using only forward passes. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 24803–24840, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1209. URL https://aclanthology.org/2025.acl-long.1209/.

Jonathan H. Clark, Eunsol Choi, Michael Collins, Dan Garrette, Tom Kwiatkowski, Vitaly Nikolaev, and Jennimaria Palomaki. TyDi QA: A benchmark for information-seeking question answering in typologically diverse languages. *Transactions of the Association for Computational Linguistics*, 8:454–470, 2020. doi: 10.1162/tacl_a_00317. URL https://aclanthology.org/2020.tacl-1.30/.

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, 2018. URL https://arxiv.org/abs/1803.05457.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168, 2021. URL https://arxiv.org/abs/2110.14168.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pretrained transformers. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pp. 8493–8502. Association for Computational Linguistics, 2022. doi: 10.18653/V1/2022.ACL-LONG.581. URL https://doi.org/10.18653/v1/2022.acl-long.581.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. Olympiadbench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume*

*1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pp. 3828–3850. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.211. URL https://doi.org/10.18653/v1/2024.acl-long.211.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In Joaquin Vanschoren and Sai-Kit Yeung (eds.), *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/be83ab3ecd0db773eb2dc1b0a17836a1-Abstract-round2.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 *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Kai Dang, An Yang, Rui Men, Fei Huang, Xingzhang Ren, Xuancheng Ren, Jingren Zhou, and Junyang Lin. Qwen2.5-coder technical report. *CoRR*, abs/2409.12186, 2024. doi: 10.48550/ARXIV.2409.12186. URL https://doi.org/10.48550/arXiv.2409.12186.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=chfJJYC3iL.

Yongqi Leng and Deyi Xiong. Towards understanding multi-task learning (generalization) of llms via detecting and exploring task-specific neurons. In Owen Rambow, Leo Wanner, Marianna Apidianaki, Hend Al-Khalifa, Barbara Di Eugenio, and Steven Schockaert (eds.), *Proceedings of the 31st International Conference on Computational Linguistics, COLING 2025, Abu Dhabi, UAE, January 19-24, 2025*, pp. 2969–2987. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.coling-main.200/.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay V. Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/18abbeef8cfe9203fdf9053c9c4fe191-Abstract-Conference.html.

Haoling Li, Xin Zhang, Xiao Liu, Yeyun Gong, Yifan Wang, Qi Chen, and Peng Cheng. Enhancing large language model performance with gradient-based parameter selection. In Toby Walsh, Julie Shah, and Zico Kolter (eds.), *Thirty-Ninth AAAI Conference on Artificial Intelligence, Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence, Fifteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2025, Philadelphia, PA, USA, February 25 - March 4, 2025*, pp. 24431–24439. AAAI Press, 2025. doi: 10.1609/AAAI.V39I23.34621. URL https://doi.org/10.1609/aaai.v39i23.34621.

Jia LI, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. Numinamath. [https://huggingface.co/AI-MO/NuminaMath-CoT](https://github.com/project-numina/aimo-progress-prize/blob/main/report/numina_dataset.pdf), 2024.

Jianwei Li and Jung-Eun Kim. Superficial safety alignment hypothesis. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 145127–145156,

2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/eab780aff8d47494e203f96af54f9425-Paper-Conference.pdf.

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, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pp. 4582–4597. Association for Computational Linguistics, 2021. doi: 10.18653/V1/2021.ACL-LONG.353. URL https://doi.org/10.18653/v1/2021.acl-long.353.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/43e9d647ccd3e4b7b5baab53f0368686-Abstract-Conference.html.

Jinzhe Liu, Junshu Sun, Shufan Shen, Chenxue Yang, and Shuhui Wang. Edit less, achieve more: Dynamic sparse neuron masking for lifelong knowledge editing in llms. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/d1bfaf1bbefb344df643997689b46613-Abstract-Conference.html.

YongKang Liu, Xingle Xu, Ercong Nie, Zijing Wang, Shi Feng, Daling Wang, Qian Li, and Hinrich Schuetze. Look within or beyond? a theoretical comparison between parameter-efficient and full fine-tuning. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 47800–47820, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.2208. URL https://aclanthology.org/2026.acl-long.2208/.

Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *IEEE Transactions on Audio, Speech and Language Processing*, 2025. URL https://ieeexplore.ieee.org/document/11151751/citations#citations.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. AdapterFusion: Non-destructive task composition for transfer learning. In Paola Merlo, Jorg Tiedemann, and Reut Tsarfaty (eds.), *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pp. 487–503, Online, April 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.eacl-main.39. URL https://aclanthology.org/2021.eacl-main.39/.

Yi-Lin Sung, Varun Nair, and Colin Raffel. Training neural networks with fixed sparse masks. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 24193–24205. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/cb2653f548f8709598e8b5156738cc51-Paper.pdf.

Qwen Team. Qwen3 technical report. *CoRR*, abs/2505.09388, 2025. doi: 10.48550/ARXIV.2505.09388. URL https://doi.org/10.48550/arXiv.2505.09388.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, et al. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288, 2023. doi: 10.48550/ARXIV.2307.09288. URL https://doi.org/10.48550/arXiv.2307.09288.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. URL https://arxiv.org/abs/1706.03762.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/ad236edc564f3e3156e1b2feafb99a24-Abstract-Datasets_and_Benchmarks_Track.html.

Taiqiang Wu, Runming Yang, Jiayi Li, Pengfei Hu, Yik-Chung Wu, Ngai Wong, and Yujiu Yang. Shadow-ft: Tuning instruct model via training on paired base model, 2025. URL https://arxiv.org/abs/2505.12716.

Yongliang Wu, Yizhou Zhou, Ziheng Zhou, Yingzhe Peng, Xinyu Ye, Xinting Hu, Wenbo Zhu, Lu Qi, Ming-Hsuan Yang, and xu yang. On the generalization of sft: A reinforcement learning perspective with reward rectification. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 27550–27571, 2026a. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/2e53c02ea028cbf603f4b6b47fef3d97-Paper-Conference.pdf.

Zeyu Wu, Junchao Wu, Shudong Liu, Runzhe Zhan, Xin Chen, Shu Yang, Yichao Du, Longyue Wang, Weihua Luo, Jinsong Su, and Derek F. Wong. Neuron-guided fine-tuning: Unlocking efficient alignment mechanisms for large language models, 2026b. URL https://arxiv.org/abs/2609.05913.

Haoyun Xu, Runzhe Zhan, Yingpeng Ma, Derek F. Wong, and Lidia S. Chao. Let's focus on neuron: Neuron-level supervised fine-tuning for large language model. In Owen Rambow, Leo Wanner, Marianna Apidianaki, Hend Al-Khalifa, Barbara Di Eugenio, and Steven Schockaert (eds.), *Proceedings of the 31st International Conference on Computational Linguistics, COLING 2025, Abu Dhabi, UAE, January 19-24, 2025*, pp. 9393–9406. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.coling-main.630/.

Lingling Xu, Haoran Xie, S. Joe Qin, Xiaohui Tao, and Fu Lee Wang. Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment. *IEEE Trans. Pattern Anal. Mach. Intell.*, 48(6):6107–6126, 2026. doi: 10.1109/TPAMI.2026.3657354. URL https://doi.org/10.1109/TPAMI.2026.3657354.

Atsuki Yamaguchi, Terufumi Morishita, Aline Villavicencio, and Nikolaos Aletras. Mitigating catastrophic forgetting in target language adaptation of llms via source-shielded updates. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2026, San Diego, California, United States, July 2-7, 2026*, pp. 18949–18975. Association for Computational Linguistics, 2026. doi: 10.18653/V1/2026.ACL-LONG.865. URL https://doi.org/10.18653/v1/2026.acl-long.865.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, et al. Qwen2 technical report. *CoRR*, abs/2407.10671, 2024a. doi: 10.48550/ARXIV.2407.10671. URL https://doi.org/10.48550/arXiv.2407.10671.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, et al. Qwen2.5 technical report. *CoRR*, abs/2412.15115, 2024b. doi: 10.48550/ARXIV.2412.15115. URL https://doi.org/10.48550/arXiv.2412.15115.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. Qwen2.5-math technical report: Toward mathematical expert

model via self-improvement. *CoRR*, abs/2409.12122, 2024c. doi: 10.48550/ARXIV.2409.12122. URL https://doi.org/10.48550/arXiv.2409.12122.

Xinyu Yang, Jixuan Leng, Geyang Guo, Jiawei Zhao, Ryumei Nakada, Linjun Zhang, Huaxiu Yao, and Beidi Chen. $S^2$ft: Efficient, scalable and generalizable LLM fine-tuning by structured sparsity. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024d. URL http://papers.nips.cc/paper_files/paper/2024/hash/6e3b9fb0c0c56cf6e1ee61e6a068fca4-Abstract-Conference.html.

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), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pp. 1–9. Association for Computational Linguistics, 2022. doi: 10.18653/V1/2022.ACL-SHORT.1. URL https://doi.org/10.18653/v1/2022.acl-short.1.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In Anna Korhonen, David R. Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pp. 4791–4800. Association for Computational Linguistics, 2019. doi: 10.18653/V1/P19-1472. URL https://doi.org/10.18653/v1/p19-1472.

Runzhe Zhan, Yafu Li, Zhi Wang, Xiaoye Qu, Dongrui Liu, Jing Shao, Derek F. Wong, and Yu Cheng. Exgrpo: Learning to reason from experience. *CoRR*, abs/2510.02245, 2025. doi: 10.48550/ARXIV.2510.02245. URL https://doi.org/10.48550/arXiv.2510.02245.

Hengyuan Zhang, Zhihao Zhang, Mingyang Wang, Zunhai Su, Yiwei Wang, Qianli Wang, Shuzhou Yuan, Ercong Nie, Xufeng Duan, et al. Locate, steer, and improve: A practical survey of actionable mechanistic interpretability in large language models. *Computer Science Review*, 62:101011, 2026a. ISSN 1574-0137. doi: https://doi.org/10.1016/j.cosrev.2026.101011. URL https://www.sciencedirect.com/science/article/pii/S157401372600119X.

Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Guoyin Wang, and Fei Wu. Instruction tuning for large language models: A survey. *ACM Comput. Surv.*, 58(7):169:1–169:36, 2026b. doi: 10.1145/3777411. URL https://doi.org/10.1145/3777411.

Zhi Zhang, Yixian Shen, Congfeng Cao, and Ekaterina Shutova. Neuroada: Activating each neuron's potential for parameter-efficient fine-tuning. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, EMNLP 2025, Suzhou, China, November 4-9, 2025*, pp. 10949–10966. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.EMNLP-MAIN.555. URL https://doi.org/10.18653/v1/2025.emnlp-main.555.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Zican Dong, Yupeng Hou, Beichen Zhang, Yingqian Min, Junjie Zhang, Peiyu Liu, Xiaolei Wang, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Yiwen Hu, Jian-Yun Nie, and Ji-Rong Wen. A survey of large language models. *Frontiers Comput. Sci.*, 20(12): 2012627, 2026. doi: 10.1007/S11704-026-60308-3. URL https://doi.org/10.1007/s11704-026-60308-3.

Zhenhong Zhou, Haiyang Yu, Xinghua Zhang, Rongwu Xu, Fei Huang, Kun Wang, Yang Liu, Junfeng Fang, and Yongbin Li. On the role of attention heads in large language model safety. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 84042–84071, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/d0bcff6425bbf850ec87d5327a965db9-Paper-Conference.pdf.

# A BASELINES

## A.1 BENCHMARK DESCRIPTIONS

**Mathematical reasoning.** We adopt four widely recognised and authoritative benchmarks spanning grade-school arithmetic to Olympiad-level problems. **GSM8K** (Cobbe et al., 2021) contains $8.5$K grade-school word problems requiring 2–8 steps of arithmetic reasoning, scored by exact match on the final answer. **MATH500** (Hendrycks et al., 2021) is a held-out $500$-problem subset of the competition-level MATH benchmark and a de facto standard for evaluating mathematical reasoning. **Minerva** (Lewkowycz et al., 2022) comprises $272$ undergraduate-level STEM problems curated from MIT OpenCourseWare. **OlympiadBench** (He et al., 2024) offers $8{,}476$ bilingual problems from top-tier competitions.

**General capability.** We use four authoritative benchmarks that are standard probes of general capability in the fine-tuning literature. **MMLU-Pro** (Wang et al., 2024) is an enhanced successor of MMLU with over $12{,}000$ questions across $14$ domains and ten answer options, designed to be more discriminative and robust. **HellaSwag** (Zellers et al., 2019) contains roughly $70$K adversarially filtered multiple-choice questions on commonsense inference about everyday scenarios. **ARC** (Clark et al., 2018) is the largest public-domain set of grade-school science questions from human standardised tests; we report its Challenge split. **TyDiQA** (Clark et al., 2020) provides information-seeking questions written natively in $11$ languages, testing multilingual reading comprehension.

**Code generation.** We adopt three authoritative benchmarks that jointly test functional correctness and contamination-free coding ability. **HumanEval+** (Liu et al., 2023) extends HumanEval, the classic benchmark of $164$ hand-written Python function-synthesis problems, with an $80\times$ larger unit-test suite to expose functionally incorrect code that passes the original tests. **MBPP+** (Liu et al., 2023) applies the same augmentation, with a $35\times$ larger test suite over sanitised crowd-sourced introductory Python problems. **LiveCodeBench v6** (Jain et al., 2025) continuously collects newly released problems from LeetCode, AtCoder, and Codeforces contests, making it largely immune to training-data contamination.

## A.2 PEFT DESCRIPTIONS AND HYPERPARAMETERS.

**Prompt-based: Prefix Tuning.** Prefix Tuning (Li & Liang, 2021) keeps the pre-trained backbone frozen and learns a sequence of continuous task-specific prefix vectors. Rather than being restricted to the input embedding layer, the prefix is made available to the attention mechanism at every Transformer layer, where it acts as a sequence of virtual tokens through the corresponding key and value states. This allows subsequent tokens to attend to task-specific information without modifying the pre-trained model parameters.

**Adapter-based: Pfeiffer.** The Pfeiffer configuration (Pfeiffer et al., 2021) inserts a single bottleneck Adapter after the FFN sub-layer of each Transformer block. Each Adapter consists of a down-projection, a non-linear activation, an up-projection, and a residual connection. Its bottleneck dimension is substantially smaller than the model hidden dimension. Using one Adapter per layer reduces the number of inserted modules and associated parameters relative to the Houlsby configuration while retaining competitive performance.

**Direct selective tuning: BitFit.** BitFit (Zaken et al., 2022) freezes the weight matrices of the pre-trained backbone and updates only its bias parameters, together with the task-specific output head. The trainable parameters include biases in the attention projections, feed-forward layers, and LayerNorm modules, whereas the LayerNorm scaling parameters remain frozen.

**Efficient selective tuning: LoRA.** LoRA (Hu et al., 2022) freezes the pre-trained weights and learns low-rank additive updates for selected weight matrices. In the original experiments, LoRA is commonly applied to the Query and Value projection matrices in the multi-head attention sub-layer. For deployment with a fixed adapter, the learned updates can be merged into the corresponding pre-trained weights, thereby introducing no additional inference latency.

**Algorithm 1** Neuron-level sparse fine-tuning with static or dynamic mask induction

**Require:** pre-trained parameters $\boldsymbol{\theta}_0$, selection mode $q \in \{\text{static}, \text{dynamic}\}$, selection set $\mathcal{D}_{\text{sel}}$ (static only), training mini-batches $\{\mathcal{D}_t\}_{t=0}^{T-1}$, neuron-update ratio $\rho$, selection rule $g$, optimiser Opt, refresh schedule $\mathcal{T}_{\text{sel}}$ with $0 \in \mathcal{T}_{\text{sel}}$ (dynamic only)
**Ensure:** fine-tuned parameters $\boldsymbol{\theta}_T$
**if** $q = \text{static}$ **then**
  $\mathcal{S} \leftarrow g(\boldsymbol{\theta}_0, \mathcal{D}_{\text{sel}}; \rho)$
**end if**
**for** $t = 0, 1, \ldots, T-1$ **do**
  **if** $q = \text{dynamic}$ and $t \in \mathcal{T}_{\text{sel}}$ **then**
    $\mathcal{S} \leftarrow g(\boldsymbol{\theta}_t, \mathcal{D}_t; \rho)$
  **end if**
  $m_{\ell,j} \leftarrow \mathbf{1}\big[n_{\ell,j} \in \mathcal{S}\big]$ for all $(\ell, j)$, then expand to $\mathbf{m}_t$
  $\mathbf{g}_t \leftarrow \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_t; \mathcal{D}_t)$
  $\boldsymbol{\theta}_{t+1} \leftarrow \text{Opt}\left(\boldsymbol{\theta}_t, \mathbf{g}_t \odot \mathbf{m}_t\right)$
**end for**

All methods are trained with a learning rate of $2 \times 10^{-4}$, a batch size of 32, and 5 epochs. For Prefix Tuning, the prefix number is 100, applied to the full backbone. For the Pfeiffer adapter, the adapter scale factor is 1 and the bottleneck dimension is 16. BitFit has no method-specific hyperparameters. For LoRA, the bottleneck (rank) is 16.

### A.3 Experimental setup of Sparse Fine-Tuning.

All experiments are conducted on a server equipped with 4 NVIDIA H100 GPUs. Unless otherwise specified, all methods share the same training configuration: a learning rate of $2 \times 10^{-5}$, a global batch size of 32, and 2 training epochs. For neuron selection, each method adopts the default hyperparameters reported in its original paper and open-source code, as detailed below.

- **S$^2$FT**: updates the $\rho = 3\%$ of channels with the smallest activation norms.
- **NeFT**: updates the $\rho = 10\%$ of FFN neurons with the largest weight drift relative to a short full-fine-tuned reference.
- **NCFT**: updates the top $\rho = 10\%$ of FFN neurons by gradient–activation attribution.
- **NMKE**: the neuron-update ratio is determined adaptively from attribution entropy.
- **SSU**: freezes the top $50\%$ of columns by the Wanda score and updates the remainder.
- **NeuroAda**: adapts the top-20 highest-magnitude incoming connections of each neuron.
- **GMT**: updates the top $10\%$ of parameters by gradient magnitude at every step.
- **NGFT**: the neuron-update ratio is determined adaptively from activation statistics.

## B Implementation Details of Static–Dynamic Mask Crossover

The crossover in Section 4.5 never modifies the selection rule $g$ or the update rule of Equation 4; it changes only (i) the refresh schedule $\mathcal{T}_{\text{sel}}$ and (ii) whether the statistics consumed by $g$ are collected once or repeatedly. All conversions use the same held-out calibration set $\mathcal{D}_{\text{cal}}$ (disjoint from the training mini-batches $\mathcal{D}_t$), so that selection is not driven by optimisation noise.

**Static → Dynamic.** For each originally static method, at every refresh step $t \in \mathcal{T}_{\text{sel}}$ (every $K$ optimisation steps) we re-evaluate $g$ on the current parameters $\boldsymbol{\theta}_t$ and re-induce $\mathbf{m}_t$:

1. **S$^2$FT.** Re-run forward passes over $\mathcal{D}_{\text{cal}}$ under $\boldsymbol{\theta}_t$ and recompute the channel activation norms $\|A\|_2$; the channels with the smallest norms are re-selected.
2. **NCFT.** Recompute the first-order Taylor score $|(\partial L/\partial \omega) \cdot \omega|$ from calibration gradients evaluated at $\boldsymbol{\theta}_t$, averaged over $N$ consecutive calibration batches to reduce variance, and re-apply the top-$10\%$ threshold.

3. **SSU.** Recompute the Wanda product $|\theta_{ij}| \, \|X_j\|_2$ with the current weights and fresh calibration activations; the columns entering the top-$k\%$ are frozen, and the previously frozen columns are released.
4. **NeuroAda.** Re-rank the input magnitudes $|w_j|$ at $\boldsymbol{\theta}_t$; a new zero-initialised bypass is attached to each newly selected input, while bypasses on deselected inputs are first merged into the backbone weights and then removed.
5. **NGFT.** Re-estimate the activation mean $\mu$ and standard deviation $\sigma$ on a rolling calibration window under $\boldsymbol{\theta}_t$, re-apply the filter $\{\text{activation} > 0, \;\; R = \sigma/(\mu + \epsilon) < \tau\}$, and re-determine the neuron-update ratio $\rho_t$; unlike the other conversions, $\rho_t$ is allowed to vary across refreshes because adaptivity is part of the criterion itself.

NeFT has no dynamic variant: its score requires a comparison against a short full-fine-tuned reference model and is therefore only defined after training, not at any intermediate $\boldsymbol{\theta}_t$.

**Dynamic → Static.** For each originally dynamic method, the online estimation loop is replaced by a one-shot evaluation at $t = 0$, with $\mathcal{T}_{\text{sel}} = \{0\}$:

1. **NMKE.** The amplification probes are run once over $\mathcal{D}_{\text{cal}}$ at $\boldsymbol{\theta}_0$: each neuron is temporarily amplified and scored by the resulting log-probability gain, and a single neuron-update ratio $\rho$ is fixed from the attribution entropy measured on that set. The selected set is then frozen for the entire run, with no per-batch probing.
2. **GMT.** Gradient magnitudes $|\nabla_{\theta_{ij}} \mathcal{L}|$ are averaged over $N$ calibration batches at $\boldsymbol{\theta}_0$, the top-$\kappa$ quantile is applied exactly once, and the resulting mask is frozen; the per-step requantisation is removed.

Tables 4 and 5 report the per-benchmark results underlying Figure 3 (Section 4.5), in which each method is evaluated with the swapped mask-regeneration schedule on the code generation and the mathematical reasoning task with Llama3.1-8B, respectively.

Table 4: Crossover results on the code generation task with Llama3.1-8B (summarised in Figure 3).

| | | **Code generation** | | | **General capability** | | | | **Average** | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Model** | **Method** | HumanEval+ | MBPP+ | LiveCodeBench v6 | MMLU (P) | HellaSwag | ARC | TyDiQA | Code | Gen |
| Llama3.1-8B | $S^2$FT | 57.0 | 63.4 | 14.2 | 44.3 | 79.1 | 55.4 | 39.4 | 44.9 | 54.6 |
| | SSU | 61.2 | 60.9 | 9.2 | 44.6 | 79.3 | 55.3 | 42.4 | 43.8 | 55.4 |
| | NCFT | 55.2 | 61.3 | 9.1 | 43.8 | 78.4 | 54.2 | 56.7 | 41.9 | 58.3 |
| | NGFT | 56.2 | 61.5 | 10.2 | 44.3 | 78.5 | 54.5 | 51.8 | 42.6 | 57.3 |
| | NeuroAda | 54.4 | 60.2 | 7.2 | 43.7 | 78.4 | 55.6 | 50.1 | 40.6 | 57.0 |
| | GMT | 55.9 | 60.5 | 8.2 | 44.1 | 79.2 | 54.8 | 51.8 | 41.5 | 57.5 |
| | NMKE | 56.6 | 62.7 | 10.1 | 44.2 | 79.3 | 55.8 | 55.4 | 43.1 | 58.7 |

Table 5: Crossover results on the mathematical reasoning task with Llama3.1-8B (summarised in Figure 3).

| | | **Mathematical reasoning** | | | | **General capability** | | | | **Average** | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Model** | **Method** | GSM8K | MATH500 | Minerva | Olympiad | MMLU (P) | HellaSwag | ARC | TyDiQA | Math | Gen |
| Llama3.1-8B | $S^2$FT | 84.2 | 49.0 | 21.9 | 18.8 | 44.0 | 79.6 | 51.8 | 45.0 | 43.5 | 55.1 |
| | SSU | 84.6 | 49.2 | 15.9 | 19.5 | 40.1 | 79.4 | 53.2 | 47.8 | 42.3 | 55.1 |
| | NCFT | 85.6 | 51.2 | 22.3 | 18.9 | 43.6 | 78.0 | 48.6 | 59.3 | 44.5 | 57.4 |
| | NGFT | 85.0 | 52.4 | 23.5 | 18.8 | 42.6 | 78.1 | 49.6 | 57.3 | 44.9 | 56.9 |
| | NeuroAda | 84.8 | 48.2 | 20.3 | 18.7 | 43.0 | 77.8 | 49.5 | 57.0 | 43.0 | 56.8 |
| | GMT | 84.3 | 48.4 | 20.8 | 18.1 | 44.3 | 78.5 | 50.8 | 58.1 | 42.9 | 57.9 |
| | NMKE | 85.3 | 47.6 | 20.2 | 19.0 | 44.3 | 78.5 | 50.9 | 57.5 | 43.0 | 57.8 |

## C Performance on Recent Advanced LLMs

Existing fine-tuning studies, including those on sparse fine-tuning, are predominantly conducted on well-established model families such as Llama2/Llama3 (Touvron et al., 2023; Grattafiori et al.,

2024) and Qwen2/Qwen2.5 (Yang et al., 2024a;b), whose architectures and training recipes have remained broadly stable. As pretraining and post-training techniques continue to advance, however, a new generation of more capable models has emerged, exemplified by Qwen3 (Team, 2025). Wu et al. (2025) report that, on recent instruct models, some PEFT methods yield only marginal improvements over the base checkpoint and may even degrade performance. This section therefore investigates whether sparse fine-tuning still retains its advantages on these newer pre-trained models. We conduct training on Qwen3 (Team, 2025), with all experimental settings identical to those of the main experiments.

Table 6 reports the per-benchmark results on Qwen3-8B, from which two conclusions emerge.

**(1)** The advantages of sparse fine-tuning persist on this new generation of models. All sparse methods outperform FFT on MATH by 2.7–3.8 points, while their GEN scores remain within 1.6–3.4 points of the base model; by contrast, FFT loses 9.4 points of general capability. Sparse fine-tuning therefore still delivers stable downstream gains on stronger new-generation models, together with strong resistance to forgetting, answering question (1) raised in Section 1.

**(2)** The same trend extends to individual methods. On Qwen3-8B, $S^2$FT ties with NGFT for the best MATH score and attains the best GEN score, corroborating the observation in Section 4.2 that $S^2$FT tends to be more effective on general-purpose models. Given the narrow spread among the sparse methods (about 1.1 points), this change in ranking further supports the view of Section 4.7 that the effectiveness of an importance criterion depends on the base model rather than being universal, so the choice of method should be made with the target model in mind.

Table 6: Main results on Qwen3-8B. MATH and GEN are the averages over the four mathematical reasoning and the four general capability columns.

| | | **Mathematical reasoning** | | | | **General capability** | | | | **Average** | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Model** | **Method** | GSM8K | MATH500 | Minerva | Olympiad | MMLU (P) | HellaSwag | ARC | TyDiQA | Math | Gen |
| Qwen3-8B | Base | 87.3 | 57.0 | 26.1 | 28.1 | 62.9 | 74.9 | 56.2 | 67.4 | 49.6 | 65.4 |
| | FFT | 87.3 | 64.0 | 29.6 | 31.7 | 49.7 | 72.2 | 50.5 | 51.7 | 53.2 | 56.0 |
| | $S^2$FT | 89.8 | 68.8 | 34.2 | 35.3 | 57.9 | 76.3 | 58.4 | 62.5 | 57.0 | 63.8 |
| | SSU | 89.5 | 68.4 | 33.1 | 36.2 | 57.3 | 76.7 | 56.5 | 59.5 | 56.8 | 62.5 |
| | NCFT | 88.8 | 69.6 | 32.4 | 35.6 | 56.7 | 76.4 | 55.2 | 64.7 | 56.6 | 63.3 |
| | NeFT | 90.1 | 68.4 | 33.1 | 35.0 | 56.8 | 76.8 | 57.5 | 58.3 | 56.7 | 62.4 |
| | NGFT | 89.7 | 68.6 | 33.5 | 36.1 | 57.4 | 76.8 | 57.5 | 61.1 | 57.0 | 63.2 |
| | NeuroAda | 88.5 | 68.0 | 32.2 | 34.8 | 56.4 | 76.7 | 56.8 | 59.8 | 55.9 | 62.4 |
| | GMT | 89.4 | 68.4 | 31.9 | 35.3 | 56.9 | 76.4 | 56.4 | 58.2 | 56.3 | 62.0 |
| | NMKE | 90.4 | 68.2 | 32.7 | 35.5 | 56.8 | 76.6 | 55.2 | 61.3 | 56.7 | 62.5 |

## D CONTROL EXPERIMENTS WITH RANDOM MASKS

This section tests the influence of random masks on sparse fine-tuning, thereby probing the underlying mechanism and role of the mask. We randomly generate masks that update 10% of the neurons on Llama3.1-8B, using three different seeds (24, 48, and 128), and fine-tune the model under each; all remaining conditions are kept identical to the main experiments. Table 7 reports the results, from which two conclusions emerge.

**(1)** The masked-update constraint alone delivers most of the performance gain: the three random draws attain MATH scores of 42.0–42.9, on par with or exceeding FFT (42.2) and 3.9–4.8 points above the base model (38.1), while their GEN scores (57.1–57.2) are on par with the criterion-guided sparse methods and far above FFT (44.4). This corroborates the hypothesis of Section 4.7: capacity allocation and regularisation are inherent properties of the masked-update constraint itself.

**(2)** The selection criterion still plays an essential role: the criterion-guided methods attain MATH scores of 42.1–44.8; apart from SSU, all of them outperform every random draw. This confirms that confining the parameter update to neurons with a decisive influence on model behaviour yields a further gain on top of the constraint itself.

Table 7: Random-mask control results on Llama3.1-8B. Random$_k$ denotes the $k$-th independent random draw.

| | | Mathematical reasoning | | | | General capability | | | | Average | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Model** | **Method** | GSM8K | MATH500 | Minerva | Olympiad | MMLU (P) | HellaSwag | ARC | TyDiQA | Math | Gen |
| Llama3.1-8B | Base | 83.1 | 39.8 | 16.0 | 13.3 | 44.6 | 79.2 | 55.4 | 34.1 | 38.1 | 53.3 |
| | FFT | 84.2 | 47.6 | 19.2 | 17.6 | 24.3 | 69.3 | 40.7 | 43.3 | 42.2 | 44.4 |
| | Random$_1$ | 85.1 | 48.3 | 20.3 | 17.9 | 43.7 | 78.6 | 50.4 | 56.2 | 42.9 | 57.2 |
| | Random$_2$ | 84.2 | 47.4 | 19.6 | 16.8 | 43.1 | 78.1 | 50.8 | 56.9 | 42.0 | 57.2 |
| | Random$_3$ | 83.9 | 48.1 | 19.6 | 17.1 | 44.2 | 77.6 | 50.5 | 56.2 | 42.1 | 57.1 |

# E Results Detail

In this appendix, we provide the detailed results of all ablation experiments summarised in the main text. Table 8 reports the performance of S$^2$FT, NCFT, and NeFT at different neuron-update ratios, complementing the summary in Section 4.3. Tables 10 and 11 report the per-benchmark results of the selection-location ablation, complementing Figure 2 in Section 4.4; Tables 12 and 13 report the per-benchmark results of the PEFT baselines on the mathematical reasoning and code generation tasks, respectively, complementing the comparison in Section 4.6. Finally, Table 14 reports the neuron-update ratio of each sparse fine-tuning method under the static and dynamic mask schedules, complementing the crossover analysis in Section 4.5.

Table 8: Per-benchmark results of S$^2$FT, NCFT and NeFT at different neuron-update ratios on the math task with Llama3.1-8B; M$_k$ denotes $\rho = k\%$ for method M (summarised in Figure 1).

| | | Mathematical reasoning | | | | General capability | | | | Average | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Model** | **Method** | GSM8K | MATH500 | Minerva | Olympiad | MMLU (P) | HellaSwag | ARC | TyDiQA | Math | Gen |
| Llama3.1-8B | S$^2$FT$_1$ | 82.9 | 48.5 | 21.2 | 18.7 | 43.8 | 79.1 | 54.3 | 46.9 | 42.8 | 56.0 |
| | S$^2$FT$_3$ | 83.2 | 49.0 | 21.7 | 18.8 | 43.6 | 79.1 | 52.8 | 45.7 | 43.2 | 55.3 |
| | S$^2$FT$_5$ | 83.7 | 49.4 | 21.8 | 18.9 | 43.3 | 78.8 | 52.5 | 45.4 | 43.5 | 55.0 |
| | S$^2$FT$_7$ | 83.7 | 49.3 | 22.2 | 19.1 | 43.1 | 78.2 | 51.6 | 45.2 | 43.6 | 54.5 |
| | S$^2$FT$_{10}$ | 83.5 | 49.0 | 22.1 | 18.9 | 42.4 | 77.4 | 50.8 | 44.0 | 43.4 | 53.7 |
| | S$^2$FT$_{15}$ | 83.2 | 48.8 | 21.7 | 18.6 | 41.8 | 76.6 | 50.4 | 43.2 | 43.1 | 53.0 |
| | S$^2$FT$_{20}$ | 83.1 | 48.7 | 21.7 | 18.8 | 41.2 | 76.4 | 49.7 | 42.7 | 43.1 | 52.5 |
| | S$^2$FT$_{30}$ | 83.0 | 48.6 | 21.1 | 18.3 | 40.8 | 75.3 | 49.2 | 41.9 | 42.8 | 51.8 |
| | NCFT$_1$ | 84.7 | 48.7 | 21.1 | 17.2 | 44.2 | 78.7 | 49.1 | 59.9 | 42.9 | 58.0 |
| | NCFT$_3$ | 85.3 | 49.0 | 21.4 | 17.6 | 43.8 | 78.4 | 48.8 | 59.7 | 43.3 | 57.7 |
| | NCFT$_5$ | 85.6 | 49.6 | 21.9 | 18.3 | 44.3 | 78.8 | 49.2 | 60.4 | 43.9 | 58.2 |
| | NCFT$_7$ | 85.5 | 49.8 | 21.8 | 18.4 | 44.2 | 78.9 | 49.1 | 60.3 | 43.9 | 58.1 |
| | NCFT$_{10}$ | 85.7 | 50.0 | 22.1 | 18.7 | 44.1 | 78.8 | 49.3 | 60.1 | 44.1 | 58.1 |
| | NCFT$_{15}$ | 85.9 | 50.2 | 22.6 | 19.0 | 42.9 | 77.8 | 48.6 | 59.4 | 44.4 | 57.2 |
| | NCFT$_{20}$ | 85.4 | 50.2 | 22.0 | 18.9 | 41.4 | 76.3 | 46.5 | 57.3 | 44.1 | 55.4 |
| | NCFT$_{30}$ | 85.5 | 49.6 | 22.2 | 18.4 | 40.6 | 74.1 | 44.8 | 57.2 | 43.9 | 54.2 |
| | NeFT$_1$ | 83.3 | 49.1 | 19.2 | 18.1 | 45.1 | 79.4 | 51.2 | 56.1 | 42.4 | 58.0 |
| | NeFT$_3$ | 83.8 | 49.2 | 19.0 | 18.1 | 44.9 | 79.5 | 50.5 | 56.1 | 42.5 | 57.8 |
| | NeFT$_5$ | 84.1 | 49.4 | 19.6 | 18.2 | 44.7 | 79.4 | 50.7 | 56.1 | 42.5 | 57.7 |
| | NeFT$_7$ | 84.0 | 49.7 | 19.3 | 18.3 | 44.6 | 79.2 | 50.8 | 56.0 | 42.8 | 57.7 |
| | NeFT$_{10}$ | 84.2 | 49.8 | 19.6 | 18.5 | 44.6 | 79.3 | 50.7 | 56.0 | 43.0 | 57.7 |
| | NeFT$_{15}$ | 84.1 | 50.0 | 19.7 | 18.5 | 44.0 | 78.2 | 49.7 | 55.1 | 43.1 | 56.8 |
| | NeFT$_{20}$ | 84.2 | 50.1 | 20.0 | 18.6 | 41.9 | 75.4 | 46.8 | 52.1 | 43.2 | 54.1 |
| | NeFT$_{30}$ | 84.4 | 49.9 | 19.6 | 18.6 | 39.9 | 74.2 | 46.1 | 52.5 | 43.1 | 53.2 |

Table 9: Per-benchmark results of S$^2$FT, NCFT and NeFT at different neuron-update ratios on the code task with Llama3.1-8B; $M_k$ denotes $\rho = k\%$ for method M (summarised in Figure 1).

| | | Code generation | | | General capability | | | | Average | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Model** | **Method** | HumanEval+ | MBPP+ | LiveCodeBench v6 | MMLU (P) | HellaSwag | ARC | TyDiQA | Code | Gen |
| Llama3.1-8B | $S^2FT_1$ | 56.5 | 62.6 | 12.3 | 44.9 | 79.1 | 55.3 | 41.1 | 43.8 | 55.1 |
| | $S^2FT_3$ | 56.7 | 63.2 | 14.2 | 44.7 | 79.1 | 55.6 | 41.0 | 44.7 | 55.1 |
| | $S^2FT_5$ | 56.8 | 63.5 | 14.1 | 44.3 | 78.8 | 55.4 | 40.7 | 44.8 | 54.8 |
| | $S^2FT_7$ | 57.0 | 63.5 | 14.2 | 44.1 | 78.3 | 55.1 | 40.5 | 44.9 | 54.5 |
| | $S^2FT_{10}$ | 57.1 | 63.1 | 13.9 | 43.7 | 77.5 | 54.1 | 39.5 | 44.7 | 53.7 |
| | $S^2FT_{15}$ | 56.8 | 62.9 | 13.2 | 42.8 | 76.2 | 52.8 | 38.2 | 44.3 | 52.5 |
| | $S^2FT_{20}$ | 56.2 | 62.5 | 13.0 | 41.5 | 75.3 | 52.0 | 37.2 | 43.9 | 51.5 |
| | $S^2FT_{30}$ | 56.1 | 62.3 | 12.7 | 40.9 | 74.8 | 51.2 | 36.7 | 43.7 | 50.9 |
| | $NCFT_1$ | 54.5 | 60.8 | 7.8 | 44.0 | 78.8 | 54.7 | 58.1 | 41.0 | 58.9 |
| | $NCFT_3$ | 54.9 | 60.6 | 8.2 | 44.2 | 78.7 | 54.5 | 58.6 | 41.2 | 58.8 |
| | $NCFT_5$ | 54.7 | 60.9 | 8.1 | 43.8 | 78.6 | 54.5 | 58.3 | 41.2 | 58.8 |
| | $NCFT_7$ | 54.7 | 61.0 | 8.6 | 43.6 | 78.7 | 54.4 | 58.1 | 41.4 | 58.7 |
| | $NCFT_{10}$ | 54.8 | 61.1 | 8.7 | 43.8 | 78.5 | 54.4 | 58.1 | 41.5 | 58.7 |
| | $NCFT_{15}$ | 55.9 | 61.9 | 9.2 | 42.7 | 76.3 | 52.5 | 57.3 | 42.3 | 57.2 |
| | $NCFT_{20}$ | 56.3 | 62.7 | 9.8 | 41.1 | 75.8 | 51.9 | 56.8 | 42.9 | 56.4 |
| | $NCFT_{30}$ | 55.8 | 63.1 | 9.3 | 40.1 | 74.5 | 50.6 | 56.6 | 42.7 | 55.2 |
| | $NeFT_1$ | 53.8 | 61.0 | 10.6 | 44.7 | 79.8 | 55.7 | 51.4 | 41.8 | 57.9 |
| | $NeFT_3$ | 53.6 | 61.1 | 11.0 | 44.3 | 79.6 | 55.6 | 51.7 | 41.9 | 57.8 |
| | $NeFT_5$ | 53.9 | 61.2 | 10.9 | 44.5 | 79.8 | 55.6 | 50.5 | 42.0 | 57.6 |
| | $NeFT_7$ | 53.9 | 62.0 | 11.0 | 44.1 | 79.3 | 55.8 | 51.6 | 42.3 | 57.7 |
| | $NeFT_{10}$ | 54.2 | 61.9 | 11.4 | 44.1 | 79.2 | 54.6 | 49.3 | 42.5 | 56.8 |
| | $NeFT_{15}$ | 54.6 | 61.9 | 11.3 | 42.3 | 77.1 | 53.0 | 48.8 | 42.6 | 55.3 |
| | $NeFT_{20}$ | 54.5 | 61.4 | 11.3 | 41.3 | 76.1 | 51.7 | 47.7 | 42.4 | 54.2 |
| | $NeFT_{30}$ | 54.6 | 61.5 | 11.1 | 41.1 | 75.8 | 51.4 | 46.9 | 42.4 | 53.8 |

Table 10: Results of the selection-location ablation on the math task (summarised in Figure 2).

| | | Mathematical reasoning | | | | General capability | | | | Average | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Model** | **Method** | GSM8K | MATH500 | Minerva | Olympiad | MMLU (P) | HellaSwag | ARC | TyDiQA | Math | Gen |
| Qwen2.5-Math 7B | $NCFT_{FFN}$ | 88.6 | 66.8 | 29.4 | 36.3 | 42.2 | 65.7 | 51.0 | 38.1 | 55.3 | 49.3 |
| | $NCFT_{MHA+FFN}$ | 88.4 | 66.7 | 29.2 | 36.0 | 42.5 | 65.1 | 51.7 | 37.8 | 55.1 | 49.3 |
| | $NCFT_{MHA}$ | 88.5 | 66.7 | 29.3 | 36.1 | 41.9 | 66.2 | 51.2 | 38.4 | 55.2 | 49.4 |
| | $NGFT_{FFN}$ | 89.2 | 68.0 | 29.6 | 37.3 | 42.0 | 65.4 | 50.2 | 46.1 | 56.0 | 50.9 |
| | $NGFT_{MHA+FFN}$ | 89.0 | 67.9 | 29.3 | 37.0 | 42.6 | 65.9 | 49.8 | 46.7 | 55.8 | 51.3 |
| | $NGFT_{MHA}$ | 89.1 | 67.7 | 29.2 | 37.2 | 42.4 | 64.7 | 51.0 | 45.8 | 55.8 | 51.0 |
| Llama3.1-8B | $NCFT_{FFN}$ | 85.7 | 50.0 | 22.1 | 18.7 | 44.1 | 78.8 | 49.3 | 60.1 | 44.1 | 58.1 |
| | $NCFT_{MHA+FFN}$ | 85.9 | 50.3 | 21.8 | 18.7 | 43.5 | 79.4 | 49.0 | 60.8 | 44.2 | 58.2 |
| | $NCFT_{MHA}$ | 85.7 | 49.7 | 22.0 | 18.2 | 44.8 | 78.3 | 49.9 | 59.4 | 43.9 | 58.1 |
| | $NGFT_{FFN}$ | 84.6 | 52.4 | 23.2 | 18.9 | 43.0 | 78.6 | 50.2 | 57.6 | 44.8 | 57.4 |
| | $NGFT_{MHA+FFN}$ | 84.5 | 52.4 | 23.4 | 18.9 | 43.5 | 77.8 | 50.1 | 58.0 | 44.8 | 57.3 |
| | $NGFT_{MHA}$ | 84.9 | 52.3 | 23.2 | 18.7 | 42.6 | 78.1 | 50.5 | 57.0 | 44.8 | 57.1 |

Table 11: Results of the selection-location ablation on the code task (summarised in Figure 2).

| | | Code generation | | | General capability | | | | Average | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Model** | **Method** | HumanEval+ | MBPP+ | LiveCodeBench v6 | MMLU (P) | HellaSwag | ARC | TyDiQA | Code | Gen |
| Qwen2.5-Coder 3B | $\text{NCFT}_{\text{FFN}}$ | 60.9 | 67.9 | 21.1 | 35.3 | 71.0 | 48.5 | 61.7 | 50.0 | 54.1 |
| | $\text{NCFT}_{\text{MHA+FFN}}$ | 60.9 | 67.8 | 20.6 | 35.5 | 70.8 | 48.5 | 61.5 | 49.8 | 54.1 |
| | $\text{NCFT}_{\text{MHA}}$ | 60.7 | 67.6 | 20.7 | 35.5 | 71.1 | 48.7 | 61.7 | 49.7 | 54.3 |
| | $\text{NGFT}_{\text{FFN}}$ | 65.2 | 67.2 | 20.0 | 34.9 | 71.0 | 48.8 | 63.6 | 50.8 | 54.6 |
| | $\text{NGFT}_{\text{MHA+FFN}}$ | 64.7 | 66.7 | 19.8 | 34.9 | 71.1 | 48.6 | 62.8 | 50.4 | 54.4 |
| | $\text{NGFT}_{\text{MHA}}$ | 64.9 | 66.8 | 19.9 | 35.1 | 71.3 | 49.3 | 63.5 | 50.5 | 54.8 |
| Llama3.1-8B | $\text{NCFT}_{\text{FFN}}$ | 54.8 | 61.1 | 8.7 | 43.8 | 78.5 | 54.4 | 58.1 | 41.5 | 58.7 |
| | $\text{NCFT}_{\text{MHA+FFN}}$ | 55.0 | 60.7 | 8.9 | 43.9 | 78.9 | 54.2 | 58.3 | 41.5 | 58.8 |
| | $\text{NCFT}_{\text{MHA}}$ | 54.8 | 61.0 | 8.5 | 44.0 | 78.3 | 53.6 | 58.2 | 41.4 | 58.5 |
| | $\text{NGFT}_{\text{FFN}}$ | 56.1 | 61.5 | 10.6 | 44.3 | 78.5 | 55.2 | 52.7 | 42.7 | 57.7 |
| | $\text{NGFT}_{\text{MHA+FFN}}$ | 55.8 | 61.0 | 10.5 | 44.3 | 78.5 | 54.8 | 52.9 | 42.4 | 57.6 |
| | $\text{NGFT}_{\text{MHA}}$ | 55.7 | 61.4 | 10.2 | 43.6 | 78.1 | 55.1 | 52.5 | 42.4 | 57.3 |

Table 12: Results of four PEFT methods on the mathematical reasoning task with Llama3.1-8B (summarised in Figure 4).

| | | Mathematical reasoning | | | | General capability | | | | Average | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Model** | **Method** | GSM8K | MATH500 | Minerva | Olympiad | MMLU (P) | HellaSwag | ARC | TyDiQA | Math | Gen |
| Llama3.1-8B | LoRA | 84.8 | 48.4 | 19.9 | 18.4 | 43.7 | 77.6 | 50.5 | 57.4 | 42.9 | 57.3 |
| | BitFit | 83.4 | 46.7 | 18.5 | 16.6 | 43.3 | 77.3 | 50.1 | 56.5 | 41.3 | 56.8 |
| | Prefix Tuning | 85.1 | 48.8 | 20.3 | 19.0 | 43.0 | 76.5 | 49.7 | 56.8 | 41.8 | 56.5 |
| | Pfeiffer Adapter | 85.0 | 48.7 | 20.5 | 18.6 | 40.9 | 76.1 | 47.5 | 56.3 | 43.3 | 55.2 |

Table 13: Results of four PEFT methods on the code generation task with Llama3.1-8B (summarised in Figure 4).

| | | Code generation | | | General capability | | | | Average | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Model** | **Method** | HumanEval+ | MBPP+ | LiveCodeBench v6 | MMLU (P) | HellaSwag | ARC | TyDiQA | Code | Gen |
| Llama3.1-8B | LoRA | 55.6 | 62.0 | 9.3 | 43.7 | 76.8 | 56.5 | 52.6 | 42.3 | 57.4 |
| | BitFit | 54.2 | 61.3 | 9.0 | 43.2 | 75.8 | 54.3 | 52.7 | 41.5 | 56.5 |
| | Prefix Tuning | 54.1 | 61.6 | 8.5 | 43.5 | 76.1 | 56.6 | 51.6 | 41.4 | 56.7 |
| | Pfeiffer Adapter | 55.1 | 61.7 | 8.9 | 41.1 | 75.1 | 53.6 | 51.4 | 41.9 | 55.3 |

Table 14: Neuron-update ratios of each method with static and dynamic sparse masks on Llama3.1-8B. All ratios are statistics obtained by comparing the model parameters before and after fine-tuning. Dynamic ratios thus reflect the union of the masks induced at all refresh steps and can exceed the per-refresh budget; SSU and NeuroAda fix the proportion by construction and remain constant.

| | | Math | | Code | |
|---|---|---|---|---|---|
| **Model** | **Method** | Static | Dynamic | Static | Dynamic |
| Llama3.1-8B | $\text{S}^2\text{FT}$ | 3 | 6.5 | 3 | 5.1 |
| | SSU | 50 | 50 | 50 | 50 |
| | NCFT | 10 | 11.2 | 10 | 13.6 |
| | NGFT | 8.2 | 10.3 | 15.8 | 16.2 |
| | NeuroAda | 0.3 | 0.3 | 0.3 | 0.3 |
| | GMT | 10 | 13.8 | 10 | 14.5 |
| | NMKE | 6.1 | 6.3 | 6.9 | 7.4 |