arXiv:2603.21606v3 [cs.LG] 25 Mar 2026

MSFT: Addressing Dataset Mixtures Overfitting Heterogeneously in Multi-task SFT

Woosung Koh[♠⋆], **Jeyoung Jeon**[◇⋆], **Youngjin Song**[◇], **Yujin Cheon**, **Soowon Oh**[♠♡], **Jaehyeong Choi**[◇], **Se-Young Yun**[♠†]
[♠]KAIST AI [◇]Yonsei University [♡]Samsung Electronics
{reiss.koh, yunseyoung}@kaist.ac.kr
[⋆]Equal contribution [†]Corresponding author

Abstract

Current language model training commonly applies multi-task Supervised Fine-Tuning (SFT) using a homogeneous compute budget across all sub-datasets. This approach is fundamentally sub-optimal: heterogeneous learning dynamics cause faster-learning tasks to overfit early while slower ones remain under-fitted. To address this, we introduce MSFT, an iterative, overfitting-aware search algorithm for multi-task data mixtures. MSFT trains the model on an active mixture, identifies and excludes the earliest overfitting sub-dataset, and reverts to that specific optimal checkpoint before continuing. Extensive evaluations demonstrate that MSFT *consistently* outperforms **4** baselines across **10** benchmarks and **6** base models. Further analysis confirms MSFT maintains robust gains across diverse dataset sizes, task granularities, and is insensitive to its single new hyperparameter (compute budget). Notably, at low compute budget, MSFT can improve performance *while* lowering training FLOPs. Ultimately, MSFT establishes a *practical* overfitting-aware algorithm for multi-task SFT that maximizes the potential of models across diverse data mixtures.

Code

1 Introduction

Since the introduction of transformers (Vaswani et al., 2017) and scaling laws (Kaplan et al., 2020), *general* foundation models trained on *diverse* data have overtaken specialized models (Maslej et al., 2025). These foundation models undertake a multi-task Supervised Fine-tuning (SFT) stage, where diverse sub-datasets are commonly randomly mixed together (Adler et al., 2024; Hui et al., 2024; Grattafiori et al., 2024); primarily to avoid forgetting from sequential training (Wang et al., 2025; Luo et al., 2025). Within this paradigm, practitioners follow a well-known approach, identifying the pre-overfitting optimal training compute (epoch) given a fixed data size (Vapnik, 1991). This optimal compute level is determined empirically by allocating a large amount of compute while saving intermediate checkpoints in memory, then identifying the checkpoint with the best generalization benchmark scores (Prechelt, 1998; Hu & Lei, 2022).

Within this framework, frontier open-weight models inherently assume that the *global* optimal compute budget aligns with the optimal compute of each underlying sub-dataset. Consider Tab. 1, where `Magistral` (Rastogi et al., 2025), `OLMo` (Groeneveld et al., 2024; Walsh et al., 2025; Olmo et al., 2025), `DeepSeek` (Liu et al., 2024; Guo et al., 2025), and `Qwen` (Qwen et al., 2025; Yang et al., 2025) family of models identify the final compute-level homogeneously (i.e., same compute for all sub-datasets).

We hypothesize that this *de facto* approach is sub-optimal as each sub-dataset embody distinct distributions that lead to different learning and generalization dynamics. Nemotron (Nvidia et al., 2024) demonstrated that their code sub-dataset required less compute than every other sub-dataset. Nevertheless, their compute allocation remains coarse, which we term as "Multi-stage Homogenous" in Tab. 1.

Method	Type	Epochs
`Magistral` (Rastogi et al., 2025)	Homogenous	2
`OLMo` (Groeneveld et al., 2024)	Homogenous	3
`OLMo 2` (Walsh et al., 2025)	Homogenous	2
`OLMo 3` (Olmo et al., 2025)	Homogenous	2
`DeepSeek-V3` (Liu et al., 2024)	Homogenous	2
`DeepSeek-R1` (Guo et al., 2025)	Homogenous	2
`Qwen2.5` (Qwen et al., 2025)	Homogenous	2
`Qwen3` (Yang et al., 2025)	Homogenous	2
`Nemotron-4` (Nvidia et al., 2024)	Multi-stage Homogenous	1 (Code) + 3 (General)
MSFT (**ours**)	Heterogeneous	Dynamic

Table 1: **Status quo.** Frontier open-weight models continue to employ homogeneous SFT, where all sub-datasets are trained on the *same* amount of compute.

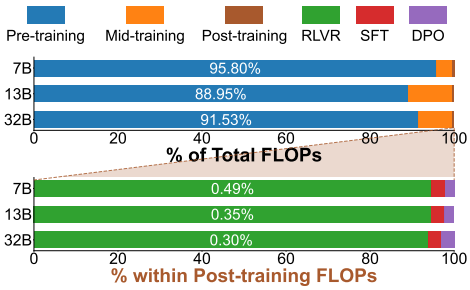


Figure 1: **SFT is compute-light.** Using `OLMo 2` as an example, SFT is relatively compute-light, and therefore additional compute usage at this stage is negligible.

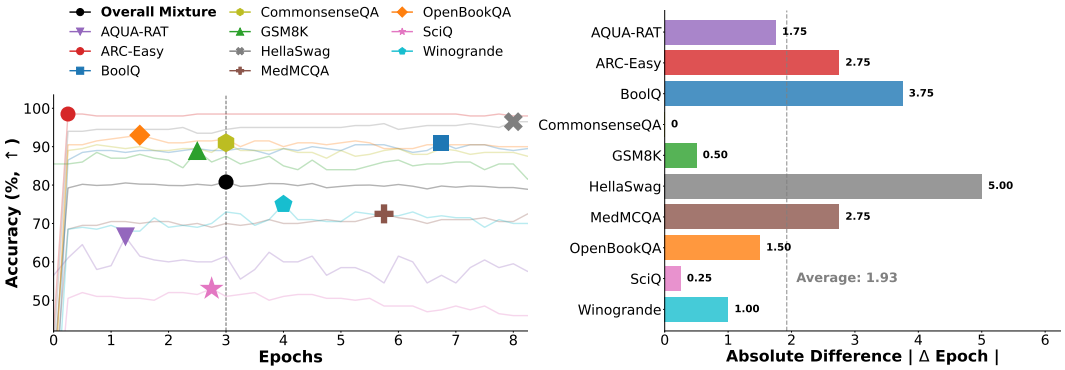
Although empirically searching for the optimal compute per sub-dataset incurs additional costs, we argue these increases are negligible since SFT is one of the computationally lightest training stage. Consider Fig. 1, where we visualize the proportion of training compute allocated to the SFT stage considering the end-to-end training pipeline. We detail how this was derived based on open-source information in Appendix A. We observe that the SFT stage takes approximately 0.01% of total training compute. Moreover, consistent performance gains with additional compute usage has been an influential philosophy guiding modern training (Chen et al., 2025; Tan et al., 2025; Koh et al., 2026).

Contribution. Given this backdrop, we first empirically demonstrate that dataset mixtures composed of sub-datasets overfit heterogenously, confirming our hypothesis that the *status quo* is sub-optimal (§ **2**, Fig. 2). In response, we propose MSFT (m representing multi-task mixture), an overfitting search algorithm for multi-task SFT (§ **3**). Prior to introducing our approach, we discuss the limitations of a naïve approach (§ **3.1**). Then, we introduce our search method which dynamically excludes sub-datasets by iteratively rolling back to the checkpoint where a sub-dataset over-fitted the quickest (§ **3.2**, Alg. 1). Finally, we empirically demonstrate that MSFT is useful for practitioners, including extensive further analyses (§ **4**):

- MSFT's average performance across **10** benchmarks outperform **4** baselines (and **2** ablative baselines) across **6** base models (§ **4.2**, Tab. 2, 3).
  - We observe that performance gains are not from disproportionate gains on a few outlier tasks, as seen by a decrease in standard deviation across benchmarks (Fig. 4).
- MSFT performance gains are robust across diverse dataset sizes (9K, 18K, 27K) and task counts (5, 10, 15) (§ **4.4**, Fig. 5).
- Reducing MSFT's only hyperparameter, compute budget C does not lead to performance degradation; with low C enabling FLOPs savings against SFT *while* improving performance (§ **4.4**, Fig. 6).
- We demonstrate that MSFT works on diverse levels of task granularity by experimenting MSFT on a single dataset with sub-categories (§ **4.4**, Fig. 7).
- We decompose the performance difference of SFT and MSFT through the lense of overfitting avoidance and catastrophic forgetting; and also show that MSFT commonly achieves a lower train loss (§ **4.4**, Fig. 8, 9).

2 Motivation: Dataset Mixtures Overfit Heterogeneously

Multi-task SFT suffers from a fundamental misalignment between the diverse learning dynamics of individual tasks and the rigid nature of standard training paradigms. To formalize this, consider SFT of Language Models (LMs) parameterized by θ on a multi-task



(a) Test set training curves across sub-tasks with annotation at peak performance.

(b) Absolute peak epoch difference of overall mixture and individual sub-datasets.

Figure 2: **Heterogeneous learning dynamics.** Multi-task SFT on `Qwen3 8B` demonstrates that underlying sub-datasets overfitting dynamics vary greatly. This observation is consistent across all other models; visualized in Appendix B.

dataset mixture $\mathcal{D} = \bigcup_{i=1}^{N} \mathcal{D}_i$, which consists of N distinct tasks. We measure training progress using a continuous compute variable c, generalizing training epochs into finer-grained units (e.g., fractional epochs). For any given task i, there exists an optimal compute c_i^*, defined as the stopping point where the model achieves maximum generalization on the task's held-out test set:

$$c_i^* = \underset{c}{\operatorname{argmax}} \operatorname{Metric}(\theta_c; \mathcal{D}_i^{\text{test}}) \tag{1}$$

Under the standard *homogeneous* training paradigm, this inherent diversity in optimal stopping points is ignored. The model is trained on the dataset mixture $\mathcal{D}$ for a fixed global compute budget c_{global}. This imposes a rigid constraint where every task i is forced to adhere to the exact same training compute, meaning $c_i := c_{\text{global}}, \forall i \in \{1, \ldots, N\}$.

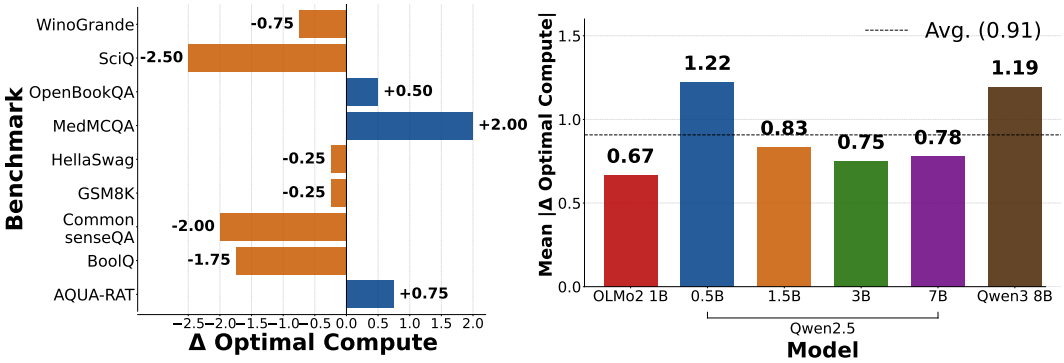
Consequently, enforcing a single global compute budget inevitably produces sub-optimal outcomes across the mixture due to *heterogeneous learning dynamics*. Because distinct tasks differ significantly in data distribution and complexity, their convergence rates and optimal compute levels vary widely ($c_i^* \neq c_j^*$). Empirically, individual sub-datasets reach peak generalization performance at substantially different compute levels (see Fig. 2). Thus, applying c_{global} creates an inherent optimization conflict: rapidly converging tasks begin to overfit when $c_{\text{global}} > c_i^*$, while slower-learning tasks remain under-fitted when $c_{\text{global}} < c_i^*$.

3 MSFT: Heterogeneous Early-stopping for Multi-task Data Mixtures

3.1 Limitation of a Naïve Solution

A straightforward solution to heterogenous overfitting (as visualized in Fig. 2) is leveraging the optimal compute found for each sub-dataset in Fig. 2a and exclude these sub-datasets at these points during a new training run. We name this method single roll-out search SFT (SRO SFT), and embodies two stages: **(i)** single roll-out search (Fig. 2a), and **(ii)** train from scratch with heterogeneous exclusion. For instance, in the example in Fig. 2a, in stage **(ii)**, AQUA-RAT would be excluded in epoch 1.25, while SciQ would be excluded in epoch 2.75. Pseudocode is available in Appendix C.

However, the key limitation of SRO search is that the optimal compute found during the search stage is an approximation after the first sub-dataset is excluded. Formally, let the model parameter update at step t be driven by the aggregate gradient of the active dataset mixture. In the search stage **(i)**, the exclusion set is empty ($\mathcal{E} = \emptyset$), so the update is a



(a) Δc_j^* (Δ Optimal Compute) for individual benchmarks on `Qwen3 8B`.

(b) Mean absolute shift in optimal compute across various model architectures and scales.

Figure 3: **Divergence of optimal compute upon dataset exclusion.** Excluding a small fraction of the training mixture alters the optimization trajectory, shifting optimal stopping points for remaining tasks. (a) Δ optimal compute varies across individual sub-tasks. (b) This divergence is consistent across model families and scales, averaging an absolute shift of 0.91 epochs. Detailed decomposition across other models available in Appendix D

summation over all tasks i in $\mathcal{D}$:

$$\Delta\theta_t \propto \sum_{\mathcal{D}_i \in \mathcal{D}} w_i \nabla \mathcal{L}(\theta_t; \mathcal{D}_i), \tag{2}$$

where w_i is the weight of the sub-dataset i. Consequently, the optimal compute budget c_i^* for any specific task i is conditional on the gradient interactions from the complete mixture.

However, in the SRO training stage **(ii)**, once a sub-dataset $\mathcal{D}_{\text{exclude}}$ is added to the exclusion set $\mathcal{E}$, the update rule shifts to:

$$\Delta\theta'_t \propto \sum_{\mathcal{D}_i \in \mathcal{D} \setminus \mathcal{E}} w_i \nabla \mathcal{L}(\theta'_t; \mathcal{D}_i) \tag{3}$$

The removal of $\nabla\mathcal{L}(\cdot; \mathcal{D}_{\text{exclude}})$ causes the optimization trajectory to diverge ($\theta'_t \neq \theta_t$). Crucially, this drift exacerbates as $|\mathcal{E}|$ increases: as more tasks are dropped over time, the active gradient sum deviates further from the original search dynamics, rendering the pre-computed c_i^* increasingly inaccurate for late-stage tasks.

Empirical Analysis. We empirically validate whether the parameter divergence $\theta'_t \neq \theta_t$ (Eq. 2, 3) translates into shifted optimal compute. We construct an equal-weighted mixture of $N = 10$ sub-datasets, each containing $|\mathcal{D}_i| = 1800$ samples. We train a model on the full mixture $\mathcal{D}$ until the first sub-dataset, which we denote as $\mathcal{D}_k$, overfits. At this exact checkpoint, we bifurcate the training process into two branches: one continues training on the full mixture $\mathcal{D}$, while the other continues on the reduced mixture $\mathcal{D} \setminus \{\mathcal{D}_k\}$. For each of the 9 remaining tasks ($j \neq k$), we compare the optimal compute achieved on the full mixture (c_j^*) against the optimal compute on the reduced mixture ($c_j'^*$). We report the shift, defined as $\Delta c_j^* := c_j'^* - c_j^*$, in Fig. 3. The results clearly demonstrate that excluding even a small fraction of the training data (1/10) significantly alters the optimal stopping points for the remaining tasks, confirming our hypothesis that $c_j'^* \neq c_j^*$.

3.2 Iterative Overfitting-Aware Search

In response to this limitation, we propose MSFT, a training algorithm that ensures that the search and train phase is aligned. MSFT follows an iterative **roll-out** and **roll-back** search algorithm described below and conceptualized in Alg. 1.

Initialization. First, the algorithm initializes the exclusion set $\mathcal{E}$ that keeps track of the excluded sub-datasets, and the parameter $\hat{\theta}$ is set to the base model θ_0 (line 1). The algorithm loops as long as there is at least one active sub-dataset (line 2).

Roll-out. For every active sub-dataset $\mathcal{D} \setminus \mathcal{E}$ the model $\hat{\theta}$ is trained by a pre-determined compute budget C hyperparameter (line 3). C is analogous to epochs in the literature, however, we call it compute budget (e.g., 1/4 of an epoch) as we aim to record more granular levels of compute as we observe granular overfitting behavior in our preliminary analysis in Fig. 2 and Appendix B. For each active sub-dataset, the optimal compute is recorded (line 4). The sub-dataset that over-fitted earliest is expected to be excluded $\mathcal{D}_{\text{exclude}}$ (line 5). In the rare case that no sub-dataset $\mathcal{D}_i$ over-fitted within the compute budget C, the algorithm continues without rolling back.

Roll-back. The earliest over-fitted dataset $\mathcal{D}_{\text{exclude}}$ will no longer be included in the active set (line 9), and the model is reverted to the point at which it overfit (line 10).

Algorithm 1: MSFT

Input : Dataset mixture $\mathcal{D}$, base model θ_0, compute budget C

1 $\mathcal{E} \leftarrow \emptyset$; $\hat{\theta} \leftarrow \theta_0$; // Initialization
2 **while** $\mathcal{D} \setminus \mathcal{E} \neq \emptyset$ **do**
  /* **Roll-out**: Search for per-sub-dataset peaks */
3 $\theta,\ \{\text{acc}(\mathcal{D}_i, c)\}_{i,c} \leftarrow \text{SFT-ROLL-OUT}(\hat{\theta},\ \mathcal{D} \setminus \mathcal{E},\ C)$;
4 $c_i^* \leftarrow \arg\max_c\ \text{acc}(\mathcal{D}_i, c) \quad \forall \mathcal{D}_i \notin \mathcal{E}$; // Optimal compute per sub-dataset
5 $c_{\min}, \mathcal{D}_{\text{exclude}} \leftarrow \arg\min_{\mathcal{D}_i \notin \mathcal{E}}\ c_i^*$;
6 **if** $c_{\min} = C$ **then**
    /* No overfitting: update model and continue */
7 $\hat{\theta} \leftarrow \theta(C)$;
8 **else**
    /* **Roll-back**: Revert to the checkpoint where the sub-dataset overfit */
9 $\mathcal{E} \leftarrow \mathcal{E} \cup \{\mathcal{D}_{\text{exclude}}\}$;
10 $\hat{\theta} \leftarrow \theta(c_{\min})$; // Revert to checkpoint at $c_{\min}$
11 **end**
12 **end**

4 Empirical Study

4.1 Experiment Set-up

Base Models. For a broad range of model sizes and families, we employ `OLMo 2 1B` (Walsh et al., 2025), `Qwen2.5 0.5, 1.5, 3, 7B` (Qwen et al., 2025), and `Qwen3 8B` (Yang et al., 2025).

Baselines. We compare our approach with four baselines: **[1]** standard SFT (Rastogi et al., 2025; Groeneveld et al., 2024; Walsh et al., 2025; Olmo et al., 2025; Liu et al., 2024; Guo et al., 2025; Qwen et al., 2025; Yang et al., 2025; Nvidia et al., 2024), the *de facto* norm , **[2]** continual SFT (Scialom et al., 2022) which trains each of the sub-datasets sequentially, allowing each of them to arrive at the optimal early-stopping point, **[3]** DynamixSFT (Shin et al., 2025) which optimizes dataset mixture ratios using multi-armed bandits with 1-step roll-out, and **[4]** Instance-dependant Early Stopping (IES; Yuan et al. (2025)) which computes second-order derivatives for each instance, and leverages a threshold hyperparameter for exclusion.

Training and Evaluation Setting. For fair comparison, all overlapping training configurations are equalized across methods. Overlapping hyperparameters were optimized for standard SFT. We use $N = 10$ sub-datasets: CommonsenseQA (Talmor et al., 2019), OpenBookQA (Mihaylov et al., 2018), AQUA-RAT (Ling et al., 2017), GSM8K (Cobbe et al.,

Model:	OLMo 2		Qwen2.5								Qwen3			
Size:	1B		0.5B		1.5B		3B		7B		8B		Average	
	Acc.	Ep.	Acc.	Ep.	Acc.	Ep.	Acc.	Ep.	Acc.	Ep.	Acc.	Ep.	Acc.	Ep.
Science and Knowledge														
Base	32.4	—	26.1	—	54.6	—	12.1	—	4.0	—	24.6	—	25.6	—
SFT	47.9	9.75	37.5	0.50	65.8	3.00	71.8	5.00	74.5	2.00	77.9	3.00	62.5	3.88
Continual SFT	48.5	1.90	24.6	1.95	66.6	2.08	71.4	1.80	72.9	1.40	77.5	1.15	$60.2_{-2.3}$	1.71
DynamixSFT	47.9	5.75	39.5	0.50	65.6	2.75	71.5	3.00	74.5	7.25	75.2	5.00	$62.4_{-0.1}$	4.04
IES	47.6	10.00	39.5	0.50	65.4	4.00	71.9	3.50	74.4	3.00	78.1	2.25	$\underline{62.8}_{+0.3}$	3.88
MSFT (**ours**)	50.4	9.75	39.2	0.25	65.4	4.75	72.9	5.50	73.6	1.50	78.0	3.00	$\mathbf{63.2}_{+0.7}$	4.12
Commonsense and Language														
Base	9.9	—	22.2	—	42.5	—	8.1	—	8.4	—	19.0	—	18.4	—
SFT	50.9	9.75	32.9	0.50	73.0	3.00	81.6	5.00	84.2	2.00	86.9	3.00	68.2	3.88
Continual SFT	48.6	1.90	19.0	1.95	71.1	2.08	80.2	1.80	86.1	1.40	86.0	1.15	$65.2_{-3.0}$	1.71
DynamixSFT	49.0	5.75	39.9	0.50	72.6	2.75	83.0	3.00	84.6	7.25	84.9	5.00	$69.0_{+0.8}$	4.04
IES	51.0	10.00	38.8	0.50	72.6	4.00	82.4	3.50	85.5	3.00	86.1	2.25	$\underline{69.4}_{+1.2}$	3.88
MSFT (**ours**)	53.8	9.75	42.5	0.25	72.8	4.75	80.6	5.50	86.5	1.50	87.6	3.00	$\mathbf{70.6}_{+2.4}$	4.12
Mathematic and Quantitative														
Base	19.5	—	26.2	—	42.8	—	58.0	—	68.0	—	71.0	—	47.6	—
SFT	20.2	9.75	24.2	0.50	43.0	3.00	59.5	5.00	66.5	2.00	74.5	3.00	$\underline{48.0}$	3.88
Continual SFT	18.5	1.90	23.8	1.95	45.0	2.08	60.0	1.80	67.0	1.40	72.5	1.15	$47.8_{-0.2}$	1.71
DynamixSFT	20.8	5.75	25.0	0.50	43.2	2.75	58.2	3.00	65.8	7.25	74.2	5.00	$47.9_{-0.1}$	4.04
IES	21.5	10.00	25.5	0.50	43.0	4.00	60.2	3.50	65.2	3.00	72.5	2.25	$\underline{48.0}_{-0.0}$	3.88
MSFT (**ours**)	23.2	9.75	23.5	0.25	48.8	4.75	64.2	5.50	70.0	1.50	76.0	3.00	$\mathbf{51.0}_{+3.0}$	4.12
Average Accuracy Across 10 Benchmarks														
Base	20.8	—	24.6	—	47.4	—	19.7	—	18.6	—	31.6	—	27.1	—
SFT	43.6	9.75	33.0	0.50	$\underline{64.1}$	3.00	73.2	5.00	76.8	2.00	$\underline{80.8}$	3.00	61.9	3.88
Continual SFT	42.6	1.90	22.2	1.95	$\underline{64.1}$	2.08	72.6	1.80	$\underline{77.0}$	1.40	79.9	1.15	$59.7_{-2.2}$	1.71
DynamixSFT	42.9	5.75	$\underline{36.8}$	0.50	64.0	2.75	73.4	3.00	76.8	7.25	78.9	5.00	$62.1_{+0.2}$	4.04
IES	$\underline{43.8}$	10.00	36.4	0.50	63.8	4.00	$\underline{73.8}$	3.50	77.0	3.00	80.2	2.25	$\underline{62.5}_{+0.6}$	3.88
MSFT (**ours**)	**46.3**	9.75	**37.4**	0.25	**65.0**	4.75	**74.2**	5.50	**78.0**	1.50	**81.4**	3.00	$\mathbf{63.7}_{+1.8}$	4.12

Table 2: **Main results.** Comparison of six methodologies across six underlying models (OLMo 2, Qwen2.5, and Qwen3), evaluating performance across three major task categories. We report both accuracy (Acc.) and the epoch (Ep.) at which the best accuracy was achieved. Continual SFT's Ep. is the average across benchmarks making values not in intervals of 1/4 epochs like others. The best scores are **bolded**, and second best underlined.

2021), SciQ (Welbl et al., 2017), ARC-Easy (Clark et al., 2018), HellaSwag (Zellers et al., 2019), Winogrande (Sakaguchi et al., 2020), BoolQ (Clark et al., 2019), and MedMCQA (Pal et al., 2022). All methods are greedy decoding evaluated 5-shot (Brown et al., 2020) on the test set in intervals of 1/4 epochs, with the best performing checkpoint being reported. Further training details can be found in Appendix E.

4.2 Main Results

Overall Performance and Robustness. As detailed in Tab. 2, MSFT consistently outperforms all baseline methodologies across the six evaluated models (OLMo 2, Qwen2.5, Qwen3), achieving the highest average accuracy. While advanced baselines like DynamixSFT and IES yield marginal gains, and Continual SFT suffers from catastrophic forgetting (-2.2%), MSFT remains uniquely robust. It is the only approach to exhibit consistent improvements across all three major domains: Science & Knowledge (+0.7%), Commonsense & Language (+2.4%), and Mathematical & Quantitative reasoning (+3.0%).

Consistency and Outlier Analysis. Beyond aggregate accuracy, MSFT demonstrates superior systematic stability. As illustrated in Fig. 4 [left], it generally maintains the lowest standard deviation across benchmarks, confirming that the average improvements stem from uniformly distributed gains rather than skewed outlier performances. Furthermore, Fig. 4 [right] shows that MSFT achieves 1st place on individual benchmarks 26 times across all model configurations, doubling the frequency of the next best baseline (IES, 13 times).

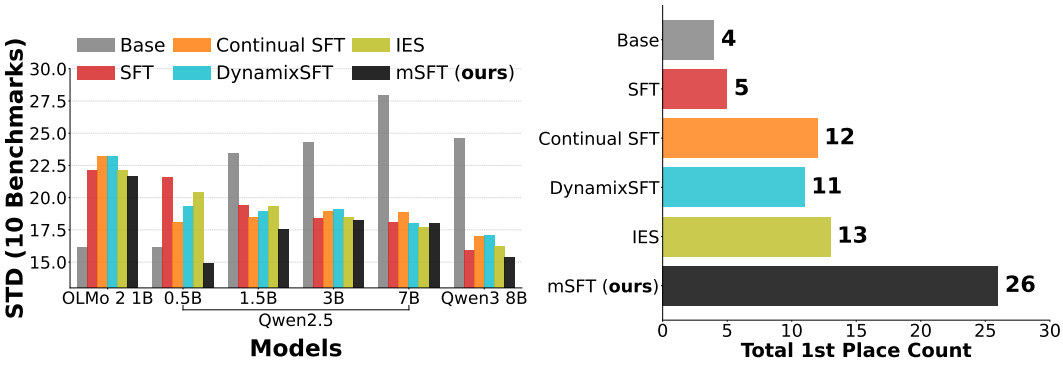


Figure 4: **Further details of main results. [left]** MSFT achieves the lowest levels of standard deviation across benchmarks (STD), indicating performance gains are not due to large outliers. **[right]** Across models, MSFT achieves 1st place the most. The 1st place count does not add up to $60 = 6 \cdot 10$ (models $\cdot$ benchmarks) as there are cases where 1st place is tied.

This affirms that MSFT reliably elevates both the performance floor and ceiling across a diverse suite of tasks.

4.3 Ablation Study

Set-up. We examine two naïve alternative heterogeneous early-stopping algorithms, that serve as ablation studies: **[4]** Single roll-out searched SFT (SRO SFT), and **[5]** Soft SRO SFT. SRO SFT is introduced in § 3.1, and Soft SRO SFT is the soft version, which aims to replicate SRO SFT via mixture ratios rather than hard exclusions, reducing catastrophic forgetting. SRO SFT and Soft SRO SFT are introduced with pseudo-codes in Appendix C.

Result. As observed in Tab. 3, MSFT's average performance is superior to both SRO SFT and Soft SRO SFT. This verifies that the naïve approach of using approximate optimal compute c_i^* through single roll-out search introduced in § 3.1 is sub-optimal.

	Average	
	Acc.	**Ep.**
SFT	61.9	3.88
SRO SFT	<u>63.4</u>	3.75
Soft SRO SFT	62.1	3.79
MSFT (**ours**)	**63.7**	4.12

Table 3: **Ablation study results.** Comparison of our proposed method (MSFT) against two naïve alternative heterogeneous early-stopping algorithms averaged across six underlying models.

4.4 Further Analysis

To rigorously evaluate the practical utility of MSFT, we conduct additional analyses using `Qwen2.5 3B`. We primarily benchmark against standard SFT, the most widely adopted paradigm, and IES, which emerged as the strongest baseline in § 4.2.

(I) MSFT Gains are Robust Across Dataset Scales. We find that the performance gains of MSFT remain robust across varying dataset sizes and task counts ($N \in \{5, 10, 15\}$) indicating that MSFT is valuable across a wide range of real-world scenarios. Across all three configurations, MSFT consistently outperforms SFT, yielding an average improvement of +5.4% (see Fig. 5).

(II) MSFT is Insensitive to Compute Budget C, with Simultaneous FLOPs Savings and Performance Gains. We demonstrate that under restricted compute budget, MSFT improves downstream performance while simultaneously reducing FLOPs. When $C = 1$, we observe a +3.4% performance gain alongside an average compute reduction of 120.3 PFLOPs (see Fig. 6). This efficiency is achieved because MSFT introduces no additional roll-out overhead compared to SFT, while dynamically excluding sub-datasets during training to

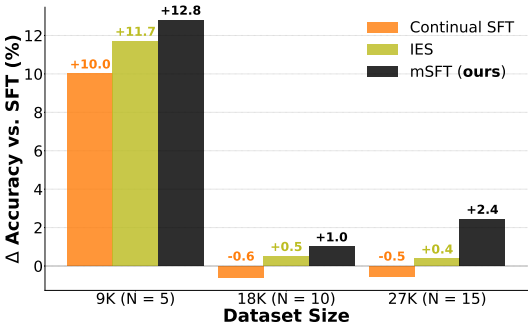


Figure 5: **Robustness across varying dataset sizes.** Δ Accuracy of Continual SFT, IES, and MSFT relative to SFT. MSFT consistently achieves the highest performance gains across different total dataset sizes and tasks (N), avoiding the degradation seen in Continual SFT at larger scales.

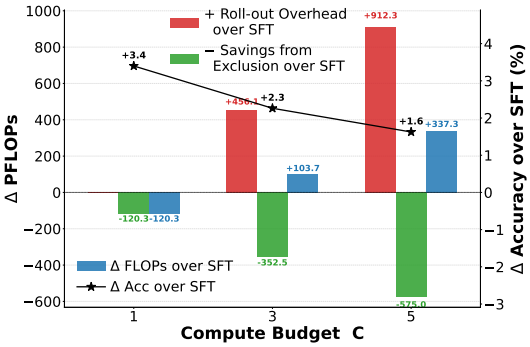


Figure 6: **Accuracy and FLOPs across compute budget.** Accuracy gains and FLOPs decomposition of MSFT across different compute budgets (C). At $C = 1$, MSFT achieves accuracy gain while strictly reducing net compute due to zero roll-out overhead.

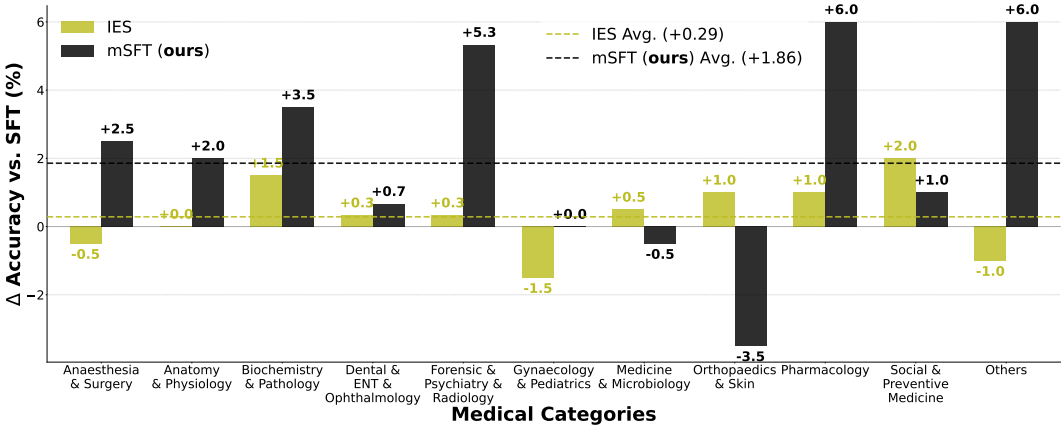


Figure 7: **Performance on further granular decompositions.** Evaluating MSFT across MedMCQA sub-categories using `Qwen2.5 3B` demonstrates an average accuracy improvement of +1.86% over the SFT baseline, outperforming IES (+0.29%).

save compute. Notably, these performance gains do not degrade as the budget C decreases. Refer to Appendix F for details on how FLOPs are measured across all methods.

(III) MSFT Remains Effective on Granular Decompositions. We further investigate whether MSFT remains effective at a highly granular level by applying it to the 21 predefined sub-categories of the MedMCQA dataset (Pal et al., 2022). As shown in Fig. 7 (grouped into 11 broad categories for legibility), MSFT yields an average accuracy improvement of +1.86% over SFT, outperforming IES (+0.29%). We observe particularly pronounced gains in specialized domains such as Pharmacology (+6.0%) and Forensic, Psychiatry & Radiology (+5.3%). Despite topic-specific variance, MSFT consistently improves performance across most sub-categories, validating its efficacy on fine-grained task distributions.

(IV) Decomposing Overfitting Prevention and Catastrophic Forgetting. To better understand the trade-off between preventing overfitting and the risk of catastrophic forgetting, we decompose MSFT's performance gains relative to SFT (Fig. 8). Specifically, we quantify the effect of dataset exclusion as:

$$\text{Forgetting (or Transfer)} := \text{Metric}(c_{\text{final}}) - \text{Metric}(c_{\min}), \tag{4}$$

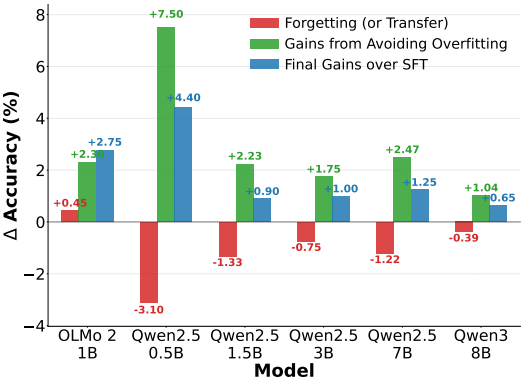


Figure 8: **Decomposition of performance gains.** MSFT's accuracy improvement over SFT is decomposed into overfitting prevention benefits and dataset exclusion effects. Minor catastrophic forgetting from hard exclusion is outweighed by gains from mitigating heterogeneous overfitting.

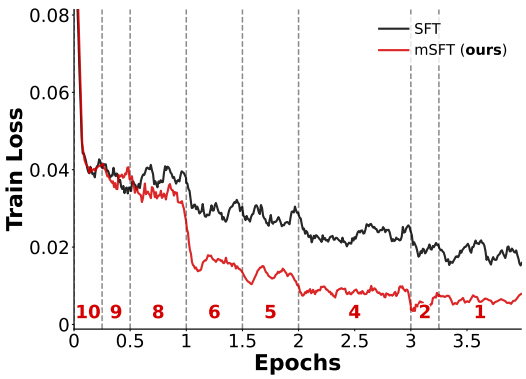


Figure 9: **Training loss curve comparison at 8B.** Smoothed with moving average with sliding window 10. Dashed vertical lines denote roll-back where a sub-dataset is excluded. Numerical annotation at the bottom indicate the number of remaining sub-datasets at each interval.

where c_{final} denotes the globally optimal checkpoint and $c_{\min}$ represents the peak performance checkpoint identified during the roll-out search (Alg. 1, line 5).

A negative Eq. 4 indicates forgetting from hard exclusions, which is the most common empirical outcome. Conversely, a positive value, as occasionally observed, suggests that continued training on the remaining mixture induces positive transfer. By subtracting Eq. 4 from the overall performance gain over standard SFT, we isolate the benefit of overfitting prevention. Ultimately, our analysis reveals that while hard exclusion incurs minor forgetting penalties on average, the performance gains achieved by mitigating heterogeneous overfitting outweigh these losses, driving the overall superiority of MSFT.

(V) MSFT Commonly Embodies Lower Training Loss. As seen in Fig. 9 (and Appendix G), MSFT commonly achieves a consistently lower training loss than standard SFT. With base model `Qwen3 8B`, the curve occasionally exhibits sharp, step-wise loss descents immediately after overfitted sub-datasets are excluded. We hypothesize this reflects a relief from gradient conflict. In SFT, simultaneous updates can cause progress on some tasks to actively disrupt others. Furthermore, once a fast-learning dataset passes its optimal compute point, it likely introduces noisy, over-specialized gradients. By dynamically filtering out these post-peak datasets, MSFT unburdens the optimizer, enabling the model to reallocate its capacity and more efficiently minimize the loss of the remaining, slower-learning tasks.

5 Discussion

Additional Related Work. Numerous works explore which datasets to include in the SFT stage (Dong et al., 2024; Li et al., 2024), and the optimal mixture ratios (Xiao et al., 2024; Zhu et al., 2025; Shi et al., 2025; Wang et al., 2026; Li et al., 2025). Another line of research addresses task imbalance through continuous loss-reweighting or gradient manipulation, primarily studied in computer vision, reinforcement learning, and early LM multi-tasking (Chen et al., 2018; Yu et al., 2020; Liu et al., 2021; 2023; Gong et al., 2024). While Gong et al. (2024) dynamically adjust task weights to balance convergence rates, they require continuous gradient-level interventions during the forward-backward pass and introduce multiple sensitive hyperparameters (e.g., history windows, warm-up steps, temperature parameter). In contrast, MSFT operates strictly at the data-scheduling level and hard exclusions, entirely avoiding this per-step computational overhead.

Efficient Disk Management. An operational limitation of MSFT is the additional storage overhead incurred by saving intermediate checkpoints during the roll-out phase. To mitigate this, we introduce a dynamic checkpoint pruning algorithm in Appendix H that actively discards redundant model states. Empirically, this strategy results in average storage footprint by approximately 4.44× SFT (see Appendix I). Because disk space is rarely the primary bottleneck in large-scale LM training, especially given the negligible cost of storage relative to compute, we consider this an acceptable trade-off. Nevertheless, future work could further optimize this process to reduce disk overhead entirely.

References

Bo Adler, Niket Agarwal, Ashwath Aithal, Dong H Anh, Pallab Bhattacharya, Annika Brundyn, Jared Casper, Bryan Catanzaro, Sharon Clay, Jonathan Cohen, et al. Nemotron-4 340b technical report. *arXiv preprint arXiv:2406.11704*, 2024.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared 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 M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher 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, Virtual, December 2020. Curran Associates, Inc. URL https://proceedings.neurips.cc/paper_files/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International conference on machine learning*, pp. 794–803. PMLR, 2018.

Zhengyu Chen, Siqi Wang, Teng Xiao, Yudong Wang, Shiqi Chen, Xunliang Cai, Junxian He, and Jingang Wang. Revisiting scaling laws for language models: The role of data quality and training strategies. 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. 23881–23899, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1163. URL https://aclanthology.org/2025.acl-long.1163/.

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, 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, 2021. URL https://arxiv.org/abs/2110.14168.

Guanting Dong, Hongyi Yuan, Keming Lu, Chengpeng Li, Mingfeng Xue, Dayiheng Liu, Wei Wang, Zheng Yuan, Chang Zhou, and Jingren Zhou. How abilities in large language models are affected by supervised fine-tuning data composition. 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)*, pp. 177–198, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.12. URL https://aclanthology.org/2024.acl-long.12/.

Zi Gong, Hang Yu, Cong Liao, Bingchang Liu, Chaoyu Chen, and Jianguo Li. Coba: Convergence balancer for multitask finetuning of large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 8063–8077, 2024.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Dirk Groeneveld, Iz Beltagy, Evan Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, Shane Arora, David Atkinson, Russell Authur, Khyathi Chandu, Arman Cohan, Jennifer Dumas, Yanai Elazar, Yuling Gu, Jack Hessel, Tushar Khot, William Merrill, Jacob Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew Peters, Valentina Pyatkin, Abhilasha Ravichander, Dustin Schwenk, Saurabh Shah, William Smith, Emma Strubell, Nishant Subramani, Mitchell Wortsman, Pradeep Dasigi, Nathan Lambert, Kyle Richardson, Luke Zettlemoyer, Jesse Dodge, Kyle Lo, Luca Soldaini, Noah Smith, and Hannaneh Hajishirzi. OLMo: Accelerating the science of language models. 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)*, pp. 15789–15809, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.841. URL https://aclanthology.org/2024.acl-long.841/.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Ting Hu and Yunwen Lei. Early stopping for iterative regularization with general loss functions. *Journal of Machine Learning Research*, 23(339):1–36, 2022. URL http://jmlr.org/papers/v23/21-0983.html.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, et al. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*, 2024.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020. URL https://arxiv.org/abs/2001.08361.

Woosung Koh, Sungjun Han, Segyu Lee, Se-Young Yun, and Jamin Shin. Generative visual code mobile world models, 2026. URL https://arxiv.org/abs/2602.01576.

Ming Li, Yong Zhang, Zhitao Li, Jiuhai Chen, Lichang Chen, Ning Cheng, Jianzong Wang, Tianyi Zhou, and Jing Xiao. From quantity to quality: Boosting LLM performance with self-guided data selection for instruction tuning. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 7602–7635, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.421. URL https://aclanthology.org/2024.naacl-long.421/.

Yuan Li, Zhengzhong Liu, and Eric Xing. Data mixing optimization for supervised fine-tuning of large language models. *arXiv preprint arXiv:2508.11953*, 2025.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In Regina Barzilay and Min-Yen Kan (eds.), *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 158–167, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1015. URL https://aclanthology.org/P17-1015/.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. Conflict-averse gradient descent for multi-task learning. *Advances in neural information processing systems*, 34: 18878–18890, 2021.

Bo Liu, Yihao Feng, Peter Stone, and Qiang Liu. Famo: Fast adaptive multitask optimization. *Advances in Neural Information Processing Systems*, 36:57226–57243, 2023.

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*, 33:3776–3786, 2025. doi: 10.1109/TASLPRO.2025.3606231.

Nestor Maslej, Loredana Fattorini, Raymond Perrault, Yolanda Gil, Vanessa Parli, Njenga Kariuki, Emily Capstick, Anka Reuel, Erik Brynjolfsson, John Etchemendy, et al. Artificial intelligence index report 2025. *arXiv preprint arXiv:2504.07139*, 2025.

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 Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2381–2391, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1260. URL https://aclanthology.org/D18-1260/.

Nvidia, :, Bo Adler, Niket Agarwal, Ashwath Aithal, Dong H. Anh, Pallab Bhattacharya, Annika Brundyn, Jared Casper, Bryan Catanzaro, Sharon Clay, Jonathan Cohen, Sirshak Das, Ayush Dattagupta, Olivier Delalleau, Leon Derczynski, Yi Dong, Daniel Egert, Ellie Evans, Aleksander Ficek, Denys Fridman, Shaona Ghosh, Boris Ginsburg, Igor Gitman, Tomasz Grzegorzek, Robert Hero, Jining Huang, Vibhu Jawa, Joseph Jennings, Aastha Jhunjhunwala, John Kamalu, Sadaf Khan, Oleksii Kuchaiev, Patrick LeGresley, Hui Li, Jiwei Liu, Zihan Liu, Eileen Long, Ameya Sunil Mahabaleshwarkar, Somshubra Majumdar, James Maki, Miguel Martinez, Maer Rodrigues de Melo, Ivan Moshkov, Deepak Narayanan, Sean Narenthiran, Jesus Navarro, Phong Nguyen, Osvald Nitski, Vahid Noroozi, Guruprasad Nutheti, Christopher Parisien, Jupinder Parmar, Mostofa Patwary, Krzysztof Pawelec, Wei Ping, Shrimai Prabhumoye, Rajarshi Roy, Trisha Saar, Vasanth Rao Naik Sabavat, Sanjeev Satheesh, Jane Polak Scowcroft, Jason Sewall, Pavel Shamis, Gerald Shen, Mohammad Shoeybi, Dave Sizer, Misha Smelyanskiy, Felipe Soares, Makesh Narsimhan Sreedhar, Dan Su, Sandeep Subramanian, Shengyang Sun, Shubham Toshniwal, Hao Wang, Zhilin Wang, Jiaxuan You, Jiaqi Zeng, Jimmy Zhang, Jing Zhang, Vivienne Zhang, Yian Zhang, and Chen Zhu. Nemotron-4 340b technical report, 2024. URL https://arxiv.org/abs/2406.11704.

Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, et al. Olmo 3. *arXiv preprint arXiv:2512.13961*, 2025.

Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. Medmcqa: A large-scale multi-subject multi-choice dataset for medical domain question answering. In Gerardo Flores, George H Chen, Tom Pollard, Joyce C Ho, and Tristan Naumann (eds.), *Proceedings of the Conference on Health, Inference, and Learning*, volume 174 of *Proceedings of Machine Learning Research*, pp. 248–260. PMLR, 07–08 Apr 2022. URL https://proceedings.mlr.press/v174/pal22a.html.

Lutz Prechelt. Automatic early stopping using cross validation: quantifying the criteria. *Neural Networks*, 11(4):761–767, 1998. ISSN 0893-6080. doi: https://doi.org/10.1016/S0893-6080(98)00010-0. URL https://www.sciencedirect.com/science/article/pii/S0893608098000100.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Abhinav Rastogi, Albert Q Jiang, Andy Lo, Gabrielle Berrada, Guillaume Lample, Jason Rute, Joep Barmentlo, Karmesh Yadav, Kartik Khandelwal, Khyathi Raghavi Chandu, et al. Magistral. *arXiv preprint arXiv:2506.10910*, 2025.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 8732–8740, 2020.

Thomas Scialom, Tuhin Chakrabarty, and Smaranda Muresan. Fine-tuned language models are continual learners. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 6107–6122, 2022.

Kai Shi, Jun Yang, Ni Yang, Binqiang Pan, Qingsong Xie, Chao Zhang, Zhenyu Yang, Tianhuang Su, and Haonan Lu. Damo: Data mixing optimizer in fine-tuning multimodal llms for mobile phone agents, 2025. URL https://arxiv.org/abs/2510.19336.

Haebin Shin, Lei Ji, Xiao Liu, Zhiwei Yu, Qi Chen, and Yeyun Gong. Dynamixsft: Dynamic mixture optimization of instruction tuning collections. *arXiv preprint arXiv:2508.12116*, 2025.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. 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. 4149–4158, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1421. URL https://aclanthology.org/N19-1421/.

Zelin Tan, Hejia Geng, Xiaohang Yu, Mulei Zhang, Guancheng Wan, Yifan Zhou, Qiang He, Xiangyuan Xue, Heng Zhou, Yutao Fan, Zhongzhi Li, Zaibin Zhang, Guibin Zhang, Chen Zhang, Zhenfei Yin, Philip Torr, and Lei Bai. Scaling behaviors of llm reinforcement learning post-training: An empirical study in mathematical reasoning, 2025. URL https://arxiv.org/abs/2509.25300.

V. Vapnik. Principles of risk minimization for learning theory. In J. Moody, S. Hanson, and R.P. Lippmann (eds.), *Advances in Neural Information Processing Systems*, volume 4. Morgan-Kaufmann, 1991. URL https://proceedings.neurips.cc/paper_files/paper/1991/file/ff4d5fbbafdf976cfdc032e3bde78de5-Paper.pdf.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Evan Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Oyvind Tafjord, Taira Anderson, David Atkinson, Faeze Brahman, Christopher Clark, Pradeep Dasigi, Nouha Dziri, Allyson Ettinger, Michal Guerquin, David Heineman, Hamish Ivison, Pang Wei Koh, Jiacheng Liu, Saumya Malik, William Merrill, Lester James Validad Miranda, Jacob Morrison, Tyler Murray, Crystal Nam, Jake Poznanski, Valentina Pyatkin, Aman Rangapur, Michael Schmitz, Sam Skjonsberg, David Wadden, Christopher Wilhelm, Michael Wilson, Luke Zettlemoyer, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. 2 OLMo 2 furious (COLM's version). In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=2ezugTT9kU.

Boxin Wang, Chankyu Lee, Nayeon Lee, Sheng-Chieh Lin, Wenliang Dai, Yang Chen, Yangyi Chen, Zhuolin Yang, Zihan Liu, Mohammad Shoeybi, et al. Nemotron-cascade: Scaling cascaded reinforcement learning for general-purpose reasoning models. *arXiv preprint arXiv:2512.13607*, 2025.

Weixuan Wang, Minghao Wu, Barry Haddow, and Alexandra Birch. HBO: Hierarchical balancing optimization for fine-tuning large language models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=JnhahbMvRE.

Johannes Welbl, Nelson F. Liu, and Matt Gardner. Crowdsourcing multiple choice science questions. In Leon Derczynski, Wei Xu, Alan Ritter, and Tim Baldwin (eds.), *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pp. 94–106, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-4413. URL https://aclanthology.org/W17-4413/.

Yuxin Xiao, Shujian Zhang, Wenxuan Zhou, Marzyeh Ghassemi, and Sanqiang Zhao. Sft-mix: Elevating language model instruction tuning with mixup recipe. *arXiv preprint arXiv:2410.05248*, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *Advances in neural information processing systems*, 33:5824–5836, 2020.

Suqin Yuan, Runqi Lin, Lei Feng, Bo Han, and Tongliang Liu. Instance-dependent early stopping. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=P42DbV2nuV.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In Anna Korhonen, David Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472/.

Tong Zhu, Daize Dong, Xiaoye Qu, Jiacheng Ruan, Wenliang Chen, and Yu Cheng. Dynamic data mixing maximizes instruction tuning for mixture-of-experts. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 1663–1677, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.80. URL https://aclanthology.org/2025.naacl-long.80/.

A Computation of FLOPs Proportion

The `OLMo 2` technical paper (Walsh et al., 2025) reports total FLOPs computed via the standard formula from Kaplan et al. (2020). We adopt the same formula and extend it to each training stage to compute proportional contributions. We use the reported parameter size ($|\theta| \in \{7\text{B}, 13\text{B}, 32\text{B}\}$).

Pre-training and mid-training. Pre-training token counts are taken from Walsh et al. (2025) §2.3. Mid-training tokens follow from the model souping procedure (§4.5): `7B` performs three annealing runs of 50B tokens each (150B total); `13B` performs three 100B runs plus one 300B run (600B total); `32B` is derived by subtracting pre-training from the overall base (pre- + mid-training) total (6.60T − 6.06T = 0.54T).

SFT. Data is from `allenai/tulu-3-sft-olmo-2-mixture` (7B, 13B; $n_{\text{sft}} = 939{,}334$) and `allenai/tulu-3-sft-olmo-2-mixture-0225` (32B; $n_{\text{sft}} = 866{,}138$). Per `docs/tulu3.md`, maximum sequence length is 4,096 tokens and training runs for 2 epochs:

$$\text{FLOPs}_{\text{SFT}} = 6\,|\theta| \times n_{\text{sft}} \times \bar{l}_{\text{SFT}} \times 2,$$

where n_{sft} is the number of samples, and $\bar{l}_{\text{SFT}}$ is the average token length per sample, capped at 4,096 and computed by streaming the full dataset with the `OLMo 2` tokenizer.

DPO. Pair counts are from `allenai/olmo-2-1124-7b-preference-mix` (366,700 pairs, 7B), `allenai/olmo-2-1124-13b-preference-mix` (377,700 pairs, 13B), and `allenai/olmo-2-0325-32b-preference-mix` (377,900 pairs, 32B). Per `docs/tulu3.md`, training uses 1 epoch and maximum sequence length is 2,048 tokens. Each pair is processed as two separate forward–backward passes:

$$\text{FLOPs}_{\text{DPO}} = 6\,|\theta| \times n_{\text{pairs}} \times 2\bar{l}_{\text{DPO}},$$

where $\bar{l}_{\text{DPO}}$ is the average token length across all chosen and rejected sequences pooled together, capped at 2,048.

RLVR. The 7B and 13B models use PPO; the 32B model uses GRPO. All sizes use 10M total episodes. For PPO (7B, 13B), rollouts are collected in batches of 32, giving $n_{\text{grad}} = 10\text{M}/32 = 312{,}500$ gradient update steps. For GRPO (32B), 16 completions are sampled per prompt, giving $n_{\text{grad}} = 10\text{M}/16 = 625{,}000$ gradient update steps. Prompt and response are each capped at 2,048 tokens. FLOPs split into forward-only (RLVR-roll) and forward–backward (RLVR-grad) passes:

$$\text{FLOPs}_{\text{RLVR-roll}} = 2\,|\Theta| \times 10\text{M} \times 4096 \times 2,$$

$$\text{FLOPs}_{\text{RLVR-grad}} = 6\,|\Theta| \times n_{\text{grad}} \times 4096 \times \begin{cases} 2 & \text{PPO}, \\ 1 & \text{GRPO}, \end{cases}$$

where the factor of 2 in $\text{FLOPs}_{\text{RLVR-roll}}$ covers policy rollout and the frozen reference model (one forward pass each per episode), and the factor of 2 in the PPO $\text{FLOPs}_{\text{RLVR-grad}}$ term covers the policy and value model gradients. $\text{FLOPs}_{\text{RLVR}} = \text{FLOPs}_{\text{RLVR-roll}} + \text{FLOPs}_{\text{RLVR-grad}}$.

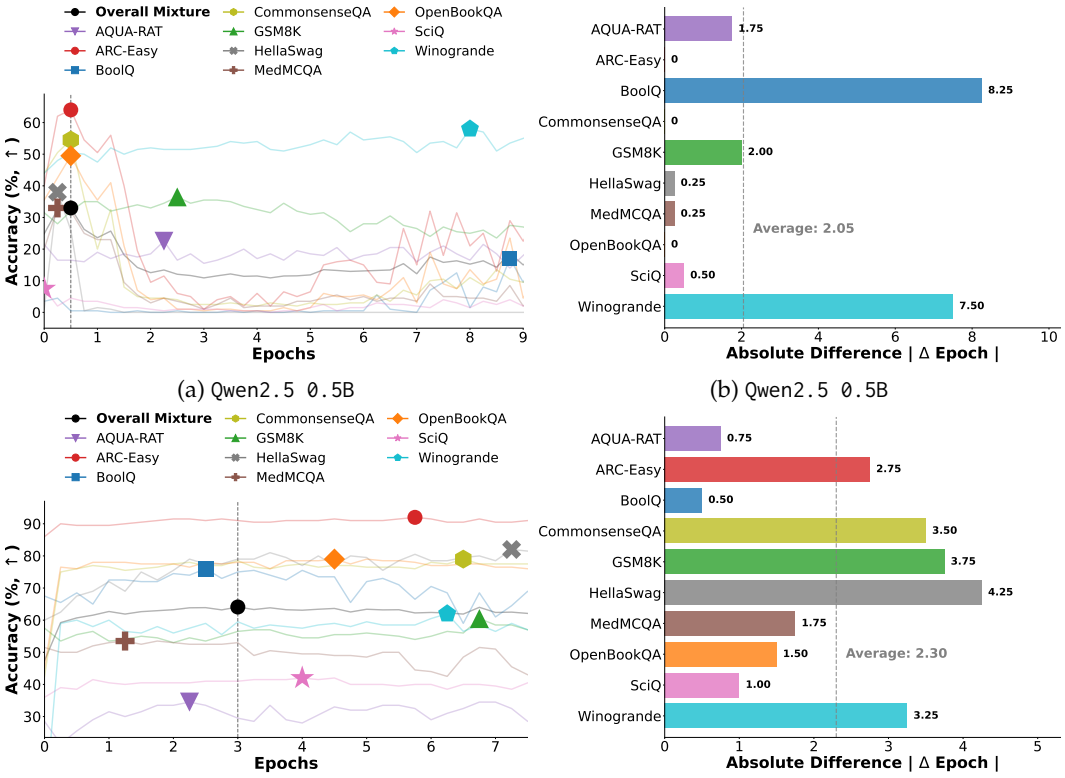
Results. Tab. 4 reports the resulting FLOPs per stage.

	Source			FLOPs		
Stage	7B	13B	32B	7B	13B	32B
Pre-training	paper §2.3			1.64×10^{23}	3.90×10^{23}	1.16×10^{24}
Mid-training	paper §4.5			6.30×10^{21}	4.68×10^{22}	1.04×10^{23}
SFT	HF dataset			2.85×10^{19}	5.29×10^{19}	1.20×10^{20}
DPO	HF dataset			1.94×10^{19}	3.70×10^{19}	1.26×10^{20}
RLVR-grad	`tulu3.md` script			7.12×10^{19}	1.32×10^{20}	3.26×10^{20}
RLVR-roll	`tulu3.md` script			7.60×10^{20}	1.41×10^{21}	3.47×10^{21}
Post total	—			8.79×10^{20}	1.63×10^{21}	4.05×10^{21}
Post / Total	—			0.517%	0.374%	0.319%
SFT / Post	—			3.24%	3.24%	2.97%

Table 4: **`OLMo 2` training FLOPs by stage.** "Post" denotes the sum of SFT, DPO, RLVR-grad, and RLVR-roll. Post/Total is the ratio of total post-training FLOPs to total training FLOPs. SFT/Post is the fraction of post-training compute spent on SFT.

B Additional Figures for Heterogeneous Overfitting

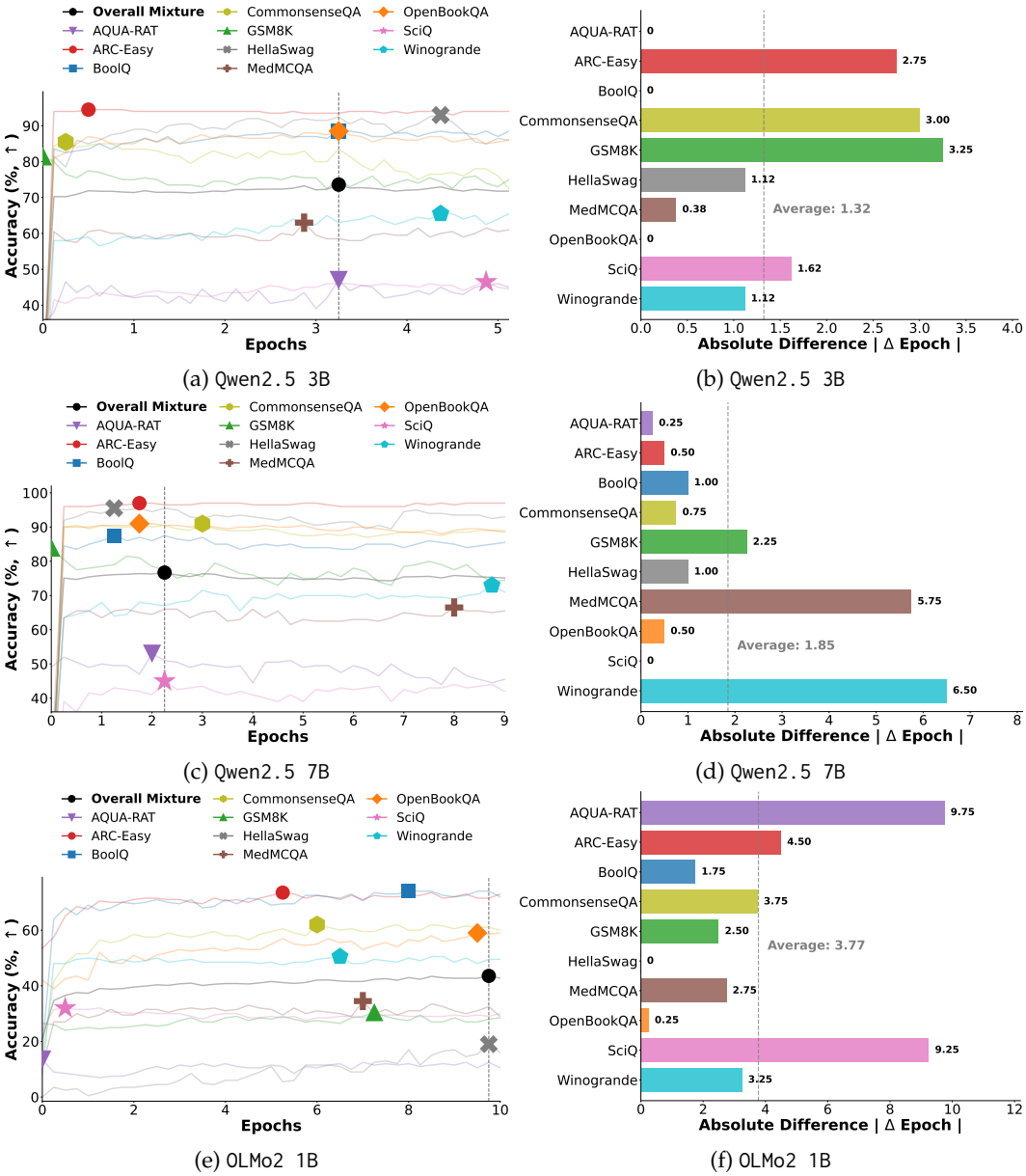
Fig. 10 and 11 visualizes the per-sub-dataset validation accuracy for all remaining models. Across all models, each sub-dataset reaches its maximum accuracy at different training steps, confirming heterogeneous overfitting dynamics discussed in § 2.



(a) `Qwen2.5 0.5B` (b) `Qwen2.5 0.5B`

(c) `Qwen2.5 1.5B` (d) `Qwen2.5 1.5B`

Figure 10: **Heterogeneous learning dynamics.** Multi-task SFT demonstrates underlying sub-datasets overfitting dynamics vary greatly.



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

(c) Qwen2.5 7B (d) Qwen2.5 7B

(e) OLMo2 1B (f) OLMo2 1B

Figure 11: **Heterogeneous learning dynamics.** Multi-task SFT demonstrates underlying sub-datasets overfitting dynamics vary greatly.

C Further Details on SRO SFT and Soft SRO SFT

SRO Alg. 2 is the pseudocode for SRO.

Algorithm 2: SRO

Input : Dataset mixture $\mathcal{D}$, base model θ_0, compute budget C

$\hat{\theta} \leftarrow \theta_0$; // Initialization

/* **Single roll-out search**: Search for per-sub-dataset peaks */

$\theta,\ \{\mathrm{acc}(\mathcal{D}_i, c)\}_{i,c} \leftarrow \text{SFT-ROLL-OUT}\big(\hat{\theta},\ \mathcal{D},\ C\big)$;

$c_i^* \leftarrow \arg\max_c\ \mathrm{acc}(\mathcal{D}_i, c)$; // Optimal compute per sub-dataset

/* **Train from scratch**: Start a new training run and exclude sub-datasets that have exhausted their budget */

$\mathcal{E} \leftarrow \emptyset;\ \hat{\theta} \leftarrow \theta_0;\ c_{\text{current}} \leftarrow 0$; // Initialization

while $\mathcal{D} \setminus \mathcal{E} \neq \emptyset$ **do**

  /* Find the next closest stopping point among active datasets */

  $c_{\text{next}} \leftarrow \min_{\mathcal{D}_i \in \mathcal{D} \setminus \mathcal{E}} c_i^*$;

  $\Delta c \leftarrow c_{\text{next}} - c_{\text{current}}$;

  /* Roll-out active datasets for the delta compute and update model */

  $\hat{\theta},\ _ \leftarrow \text{SFT-ROLL-OUT}\big(\hat{\theta},\ \mathcal{D} \setminus \mathcal{E},\ \Delta c\big)$;

  /* Update current compute and exclude datasets that just peaked */

  $c_{\text{current}} \leftarrow c_{\text{next}}$;

  $\mathcal{E} \leftarrow \mathcal{E} \cup \{\mathcal{D}_i : c_i^* \leq c_{\text{current}}\}$;

end

Soft SRO Alg. 3 is the pseudocode for Soft SRO.

Algorithm 3: SOFT SRO

Input : Dataset mixture $\mathcal{D}$, base model θ_0, compute budget C

$\hat{\theta} \leftarrow \theta_0$; // Initialization

/* **Single roll-out search**: Approximately search for per-sub-dataset peaks */

$\theta,\ \{\mathrm{acc}(\mathcal{D}_i, c)\}_{i,c} \leftarrow \text{SFT-ROLL-OUT}\big(\hat{\theta},\ \mathcal{D},\ C\big)$;

$c_i^* \leftarrow \arg\max_c\ \mathrm{acc}(\mathcal{D}_i, c)$; // Optimal compute per sub-dataset

/* **Train from scratch**: Start a new training run with a new data mixture accounting for the optimal compute budgets */

$\hat{\theta} \leftarrow \theta_0;\ \mathcal{D}' \leftarrow \emptyset;\ Z \leftarrow \sum_j (c_j^* \cdot |\mathcal{D}_j|)$; // Initialization and normalization factor

for $\mathcal{D}_i \in \mathcal{D}$ **do**

  $r \leftarrow \left(\sum_i |\mathcal{D}_i|\right) \cdot \frac{c_i^* \cdot |\mathcal{D}_i|}{Z}$; // Target number of samples, preserving base proportions

  $\mathcal{D}_i' \leftarrow \emptyset$;

  while $r \geq |\mathcal{D}_i|$ **do**

    /* Add a full copy of $\mathcal{D}_i$ using multiset union */

    $\mathcal{D}_i' \leftarrow \mathcal{D}_i' \uplus \mathcal{D}_i$;

    $r \leftarrow r - |\mathcal{D}_i|$;

  end

  if $r > 0$ **then**

    $\tilde{\mathcal{D}}_i \leftarrow$ Sample $\lfloor r \rfloor$ samples from $\mathcal{D}_i$ without replacement ;

    $\mathcal{D}_i' \leftarrow \mathcal{D}_i' \uplus \tilde{\mathcal{D}}_i$;

  end

  $\mathcal{D}' \leftarrow \mathcal{D}' \uplus \mathcal{D}_i'$; // Add the proportioned sub-dataset to the new mixture

end

$\hat{\theta},\ _ \leftarrow \text{SFT-ROLL-OUT}\big(\hat{\theta},\ \mathcal{D}',\ C\big)$;

—

D Further Experimental Results on Δ Optimal Compute

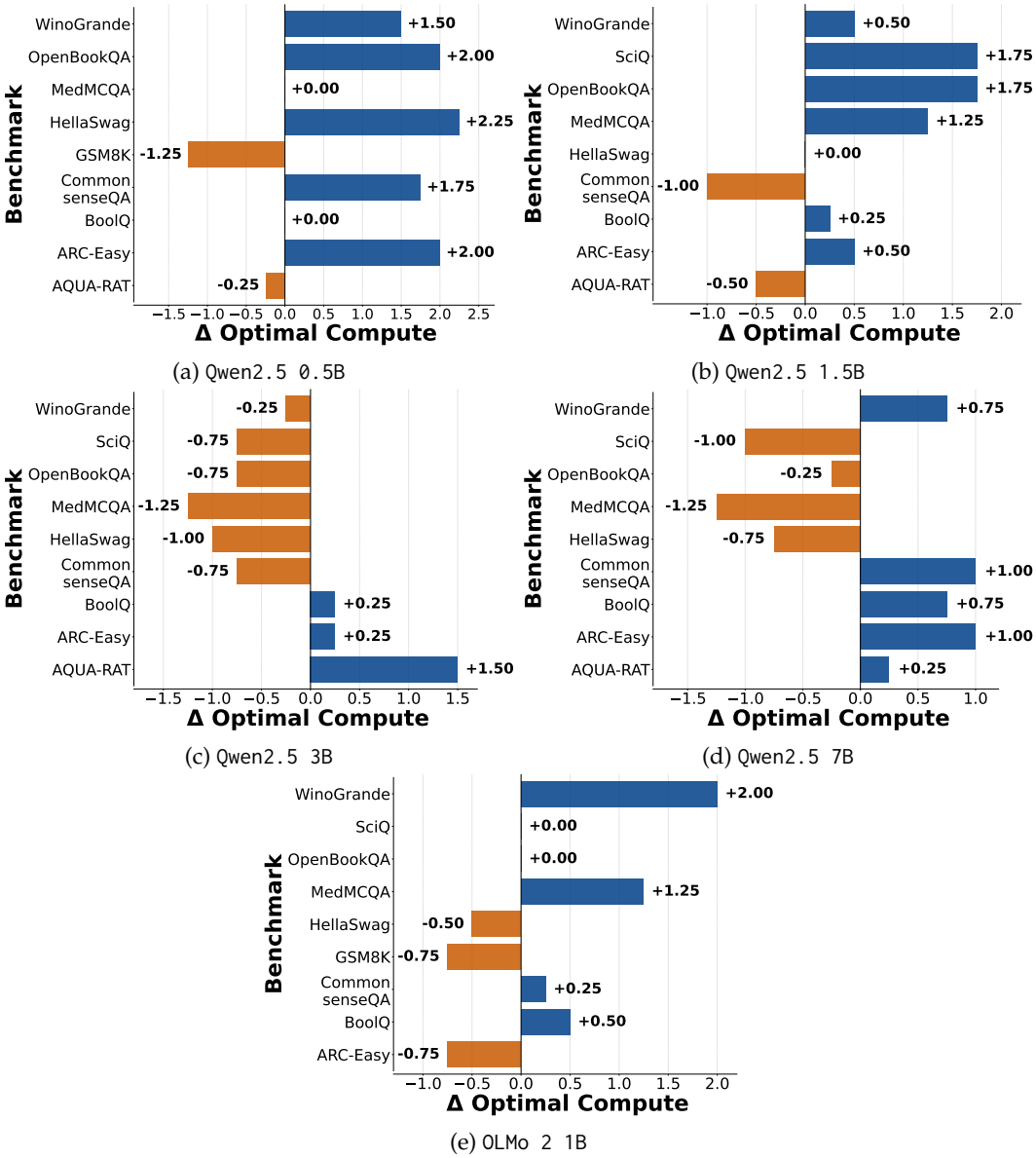


Figure 12: **Divergence of optimal compute upon dataset exclusion.** Excluding a small fraction of the training mixture alters the optimization trajectory, shifting optimal stopping points for remaining tasks. Δ optimal compute varies across individual sub-tasks.

E Further Experimental Details

E.1 Hardware

We use B200, H200, RTX A5000, and RTX 3090s for experiments. For other hardware like CPU and RAM we use commonly available ones, as these hardware did not induce any bottlenecks.

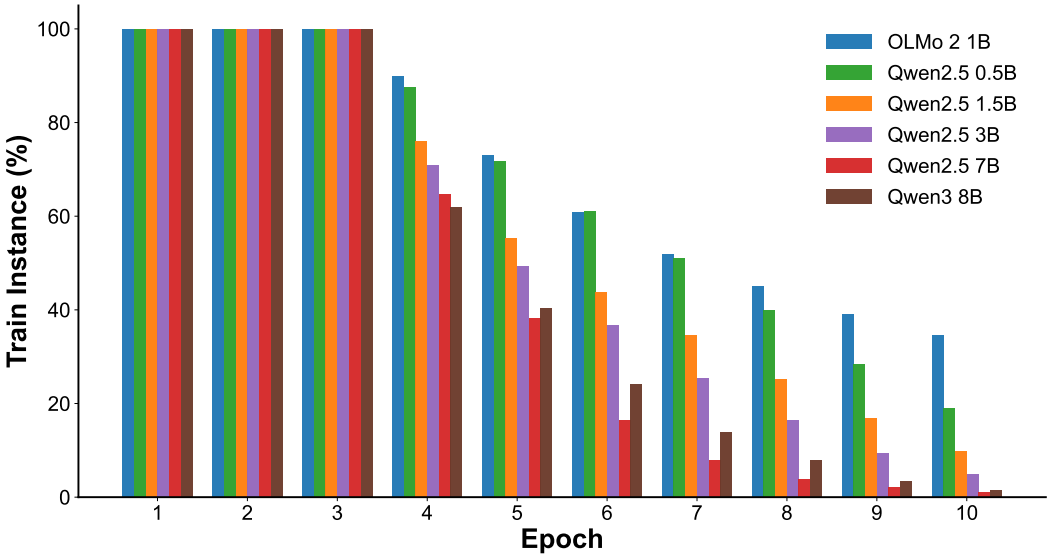


Figure 13: **Training instances across epochs on IES.** Percentage of active training instances per epoch, relative to the initial dataset size at Epoch 1. All models process the complete dataset for the first three epochs, after which the proportion of active instances consistently decreases.

E.2 Common Settings

Default training settings universal across methods are available in Tab. 5. We use a single seed (20) as preliminary experiments with `Qwen2.5 3B` on seeds 20, 30, 40 lead to virtually identical performance gains. Tab. 6 shows that the gains of MSFT is stable (low standard deviation) and thus statistically significant. This likely due to our methods and experiments being non-stochastic in nature.

Table 5: Overlapping hyperparameters.

Hyperparameter	Value
Learning Rate	1×10^{-5}
Learning Rate Schedule	Constant
Batch Size	64
Seed	20
Sub-dataset Size	1800

Method	Acc.			Mean	Std Dev	p-value
	Seed 20	Seed 30	Seed 40			
Average Accuracy Across 10 Benchmarks						
SFT	73.25	73.05	72.65	72.98	0.31	—
MSFT (**ours**)	74.25	74.05	73.25	$73.85_{+0.87}$	0.53	0.023^{*}

Table 6: **Seed stability on `Qwen2.5 3B`.** The subscript in the Mean column shows the difference (Δ) relative to SFT, coloured green for improvement. p-values are from a two-sided paired t-test against SFT ($^{*}p < 0.05$).

E.3 Method-specific Settings

SFT trains for 10 epochs as we observe that in some datasets do not overfit even up to 10 epochs (see Fig. 2 and Appendix B). Continual SFT and MSFT's compute budget is $C = 3$ epochs. DynamixSFT was first run on the settings provided in the paper (Shin et al., 2025), yet we found that further hyperparameter tuning, where sharpness factor $\beta = 5000$ improved performance in our environment so we used this for all reported experiment results. For IES, we adopt the default threshold of $\delta = 0.01$ as proposed in the original paper (Yuan et al., 2025). The cumulative proportion of dropped instances over 10 epochs is visualized in Fig. 13. For SRO SFT and Soft SRO SFT the single search compute budget is set to $C = 10$ as this is conceptually similar to the 10 epochs allocated in SFT.

F Computation of Empirical FLOPS

We calculate computation costs using the standard formula from Kaplan et al. (2020):

$$\text{FLOPS}_{\text{train}} = 6 \times |\theta| \times t, \qquad \text{FLOPS}_{\text{inference}} = 2 \times |\theta| \times t, \tag{5}$$

where $|\theta|$ is the number of model parameters and t is the number of tokens.

F.1 Method-specific FLOPs

Let t_{train} and $t_{\text{validation}}$ denote the total training and validation tokens per unit compute budget (1 epoch) over the full mixture $\mathcal{D}$.

[1] SFT. Standard supervised fine-tuning on all sub-datasets for C units of compute budget.

$$\text{FLOPs}_{\text{SFT}} = \sum_{c=1}^{C} \left[6 \cdot |\theta| \cdot t_{\text{train}} + 2 \cdot |\theta| \cdot t_{\text{validation}}\right].$$

[2] Continual SFT. Sequential training (Scialom et al., 2022): each sub-dataset $\mathcal{D}_i$ is trained independently for C units of compute budget before moving to the next.

$$\text{FLOPs}_{\text{Cont}} = \sum_{i} \left[6 \cdot |\theta| \cdot t_{tr,i} + 2 \cdot |\theta| \cdot t_{\text{validation}}\right] \cdot C,$$

summing over all N sub-datasets trained sequentially.

[3] DynamixSFT. Dynamic mixture optimization (Shin et al., 2025) via multi-armed bandits with 1-step look-ahead. At each update step (1% of total steps), the algorithm samples batches of size $B_{\text{look-ahead}}$ for all N sub-datasets and performs forward-backward passes to estimate look-ahead rewards, incurring $8|\theta|$ FLOPs per token (2 forward pre-loss, 4 backward, 2 forward post-loss). Between updates, training proceeds with current mixture probabilities:

$$\text{FLOPs}_{\text{Dynamix}} = \underbrace{\sum_{c=1}^{C} 6 \cdot |\theta| \cdot t_{\text{train}}}_{\text{training}} + \underbrace{\sum_{t_u} N \cdot 8 \cdot |\theta| \cdot B_{\text{look-ahead}} \cdot t_{\text{avg}}}_{\text{look-ahead}} + \sum_{c=1}^{C} 2 \cdot |\theta| \cdot t_{\text{validation}},$$

where $B_{\text{look-ahead}}$ is batch size for look-ahead, t_{avg} is the average tokens per sample and t_u denotes update steps.

[4] IES. Instance-dependent early stopping (Yuan et al., 2025) computes second-order differences of per-sample loss trajectories to identify mastered instances. Samples satisfying the convergence criterion are excluded from gradient updates (typically from the 3rd unit onward). Training FLOPs decrease as more samples are excluded, while validation always covers the full dataset.

$$\text{FLOPs}_{\text{IES}} = \sum_{c=1}^{C} \left[6 \cdot |\theta| \cdot t_{\text{train}}^{(c)} + 2 \cdot |\theta| \cdot t_{\text{validation}}\right],$$

where $t_{\text{train}}^{(c)} \leq t_{\text{train}}$ reflects the remaining active samples at c.

[5] SRO SFT. Single roll-out searched SFT: a two-step procedure. **Step 1 (Search):** Standard SFT for C units to determine per sub-dataset peak c_i^*, which is also their drop schedule. **Step 2 (Train):** Training with sub-datasets exclusions applied at their respective peak checkpoints; dropped sub-datasets are removed from the active token count.

$$\text{FLOPs}_{\text{SRO}} = \underbrace{\text{FLOPs}_{\text{SFT}}}_{\text{step 1}} + \sum_{c=1}^{C} \left[6 \cdot |\theta| \cdot t_{\text{train}}^{(c)} + 2 \cdot |\theta| \cdot t_{\text{validation}}\right],$$

where $t_{\text{train}}^{(c)} \leq t_{\text{train}}$ denotes training tokens over non-excluded sub-datasets at step c in Step 2.

[6] Soft SRO SFT. **Step 1:** Identical to SRO SFT Step 1, recording per-sub-dataset peak c_i^*. **Step 2:** Rather than hard exclusions, re-trains for C units with per-category sampling weight $w_i = c_i^* / \bar{c}$, where $\bar{c} = \frac{1}{N} \sum_i c_i^*$ is the mean peak across all N sub-datasets. Early-peaking sub-datasets contribute fewer tokens; late-peaking subsets receive more exposure.

$$\text{FLOPs}_{\text{Soft}} = \underbrace{\text{FLOPs}_{\text{SFT}}}_{\text{step 1}} + \sum_{c=1}^{C} \left[6 \cdot |\theta| \cdot \sum_i w_i \cdot \text{tok}_{\text{tr},i} + 2 \cdot |\theta| \cdot t_{\text{validation}}\right].$$

[7] MSFT. MSFT proceeds in S stages indexed by $s = 1, \ldots, S$. At each stage s, the model trains for C units on active subsets $\mathcal{D} \setminus \mathcal{E}_s$, where $\mathcal{E}_s$ is the accumulated exclusion set at stage s. Overfit sub-datasets are added to $\mathcal{E}_{s+1}$ and the model reverts to the earliest overfitting checkpoint (parameter rollback only; no additional FLOPs).

$$\text{FLOPs}_{\text{stage}_s} = \underbrace{6 \cdot |\theta| \cdot C \cdot t_{\text{train}}(\mathcal{D} \backslash \mathcal{E}_s)}_{\text{training on active sets}} + \underbrace{2 \cdot |\theta| \cdot C \cdot t_{\text{validation}}(\mathcal{D} \backslash \mathcal{E}_s)}_{\text{validation on active sets}} + \underbrace{2 \cdot |\theta| \cdot t_{\text{validation}}(\mathcal{E}_s)}_{\text{validation on excluded sets}},$$

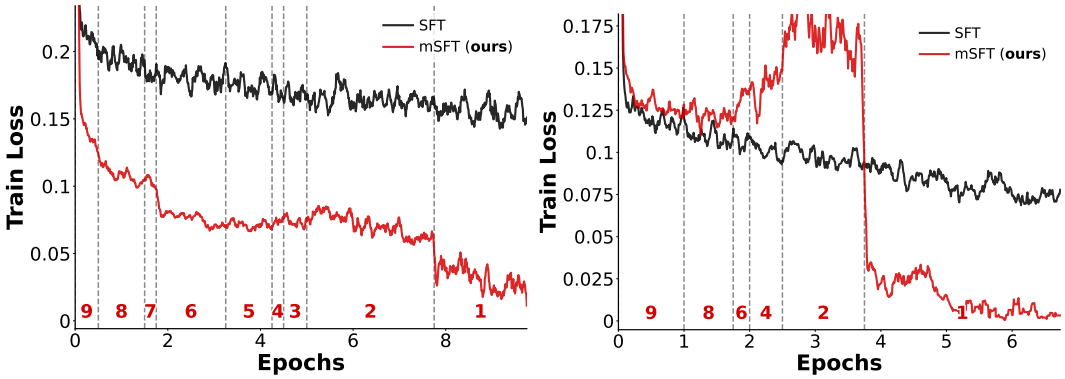
where $t_{\text{train}}(\mathcal{D} \backslash \mathcal{E}_s)$ and $t_{\text{validation}}(\mathcal{D} \backslash \mathcal{E}_s)$ decreases as more sub-datasets are excluded. $t_{\text{validation}}(\mathcal{E}_s)$ denotes validation tokens of excluded sub-datasets. Note that the third term carries no compute budget C: excluded sets are validated only once at the rollback checkpoint to preserve the full validation trajectory, where as active sub-datasets are validated at every checkpoint throughout the stage. Total FLOPs: $\text{FLOPs}_{\text{MSFT}} = \sum_{s=1}^{S} \text{FLOPs}_{\text{stage}_s}$.

Empirical FLOPs comparison. Tab. 7 reports the total FLOPs for each method across six model scales. DynamixSFT incurs substantial look-ahead overhead (94.9% of training FLOPs on average), while IES achieves costs smaller than SFT by dropping parts of samples from 3rd unit of compute budget onward. SRO SFT and Soft SRO SFT require an additional search phase (Step 1), resulting in higher total costs, though Soft SRO mitigates catastrophic forgetting via soft reweighting rather than hard exclusions.

						Soft	MSFT	
Model	**SFT**	**Cont.**	**Dynamix**	**IES**	**SRO**	**SRO**	(C=1)	(C=3)
OLMo 2 1B	153.12	161.53	256.98	113.13	258.65	312.73	74.34	226.23
Qwen2.5 0.5B	57.91	43.36	103.40	37.92	77.20	122.94	29.72	103.17
Qwen2.5 1.5B	219.82	241.71	362.78	143.02	338.22	442.50	113.46	360.72
Qwen2.5 3B	491.72	645.41	778.58	323.68	709.84	937.42	223.73	647.12
Qwen2.5 7B	1170.15	1456.04	1876.63	700.72	1509.68	2070.22	–	1240.94
Qwen3 8B	937.61	637.10	1698.86	449.19	1348.07	1993.78	–	1561.91
Average	505.06	530.86	846.21	294.61	706.94	979.93	–	690.02

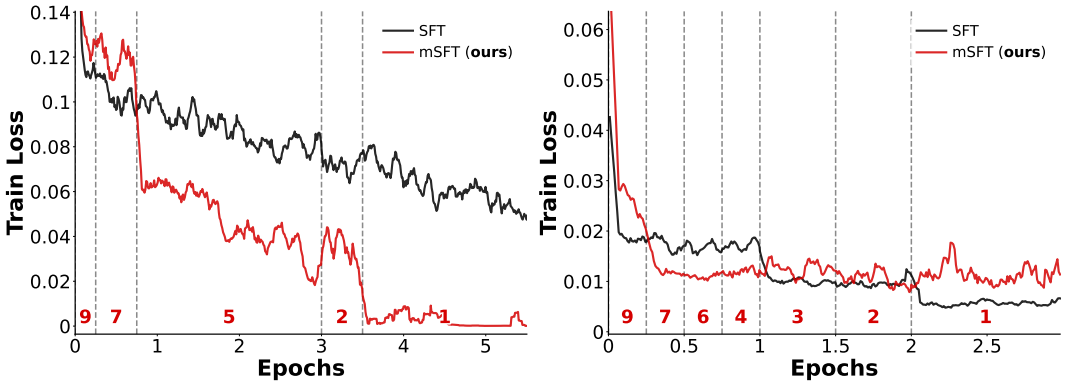
Table 7: **Total PFLOPs for each method across model scales.**

G Further Loss Curves

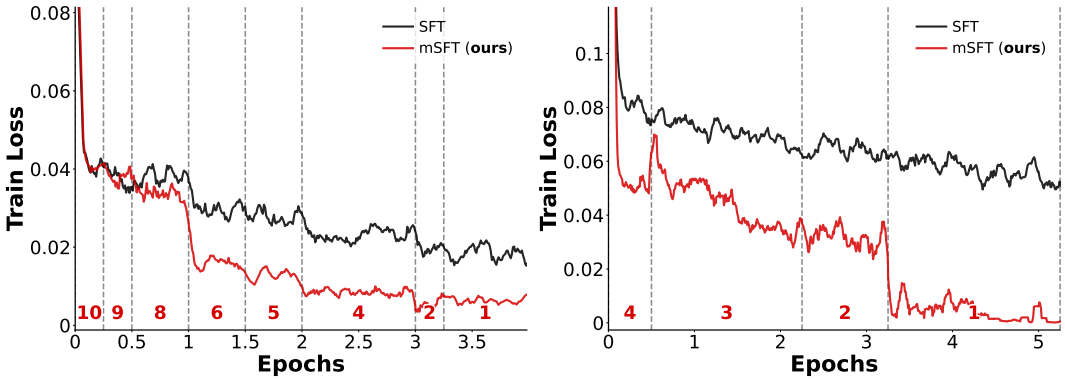


(a) `Olmo 2 1B`, $C = 3,\ N = 10$

(b) `Qwen2.5 1.5B`, $C = 3,\ N = 10$

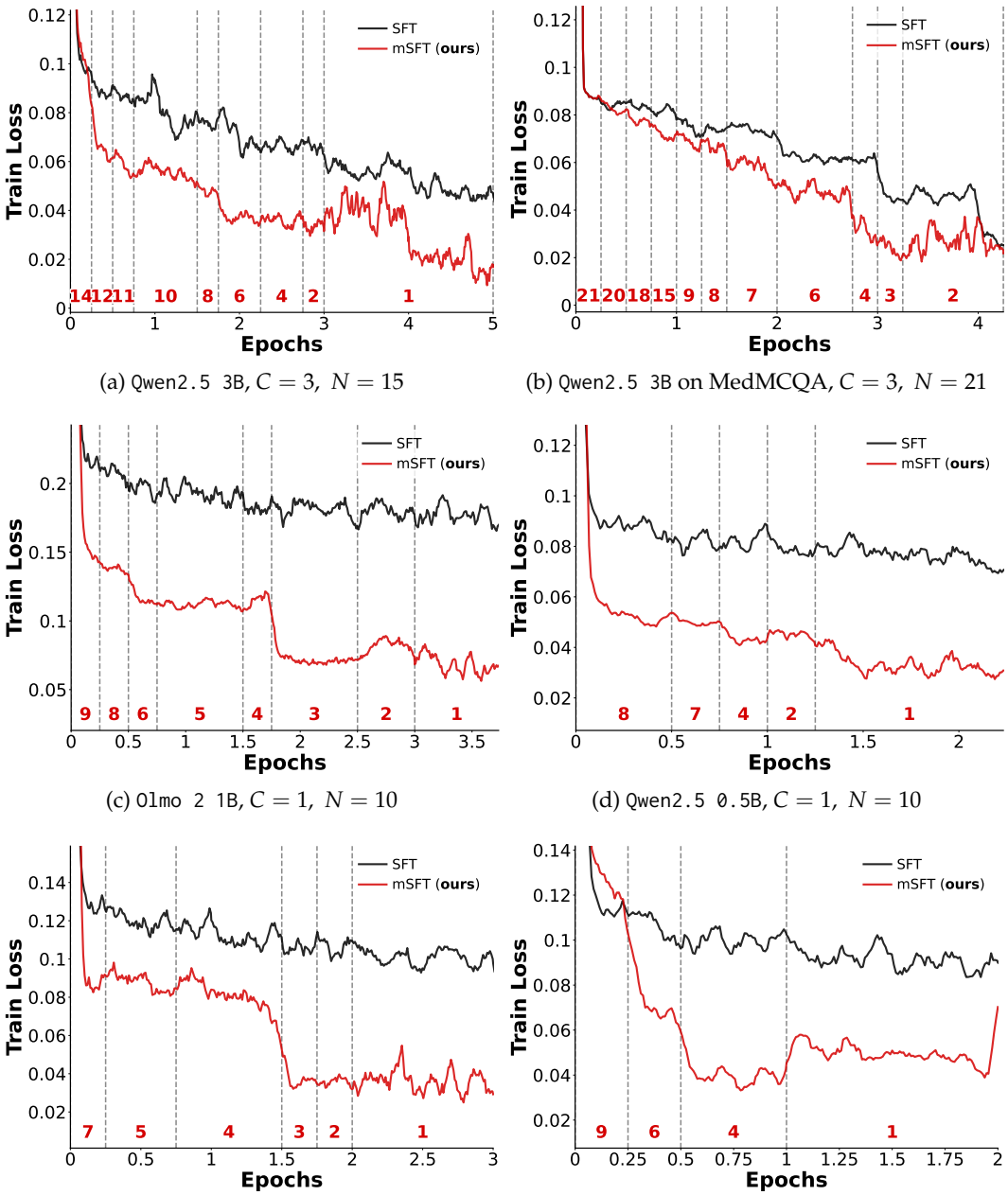


(c) `Qwen2.5 3B`, $C = 3,\ N = 10$ (d) `Qwen2.5 7B`, $C = 3,\ N = 10$



(e) `Qwen3 8B`, $C = 3,\ N = 10$ (f) `Qwen2.5 1.5B`, $C = 3,\ N = 5$

Figure 14: **Training loss curve comparison.** Smoothed with moving average with sliding window 10. Dashed vertical lines denote roll-back where a sub-dataset is excluded. Numerical annotation at the bottom indicate the number of remaining sub-datasets at each interval.



(a) `Qwen2.5 3B`, $C = 3$, $N = 15$

(b) `Qwen2.5 3B` on MedMCQA, $C = 3$, $N = 21$

(c) `Olmo 2 1B`, $C = 1$, $N = 10$

(d) `Qwen2.5 0.5B`, $C = 1$, $N = 10$

(e) `Qwen2.5 1.5B`, $C = 1$, $N = 10$

(f) `Qwen2.5 3B`, $C = 1$, $N = 10$

Figure 15: **Training loss curve comparison.** Smoothed with moving average with sliding window 10. Dashed vertical lines denote roll-back where a sub-dataset is excluded. Numerical annotation at the bottom indicate the number of remaining sub-datasets at each interval.

H MSFT with Efficient Disk Management

Algorithm 4: MSFT with Checkpoint Management

Input : Dataset mixture $\mathcal{D}$, base model θ_0, compute budget C

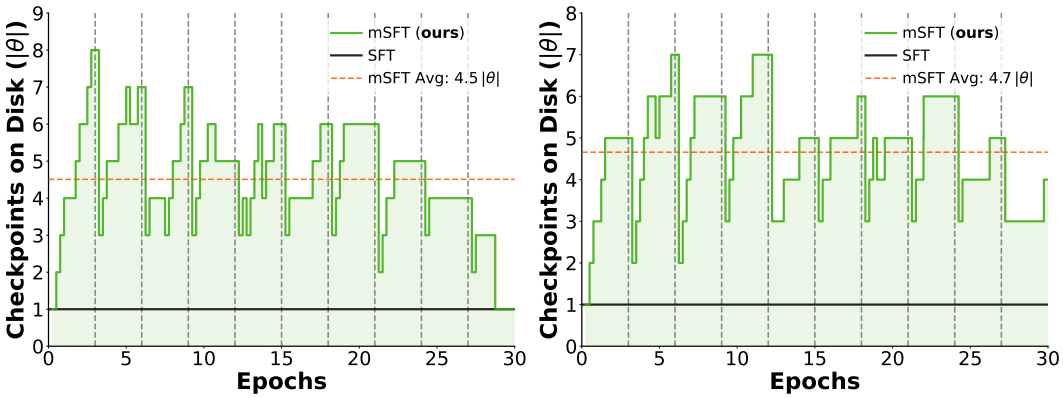
1 $\mathcal{E} \leftarrow \emptyset$; $\hat{\theta} \leftarrow \theta_0$ $\theta^* \leftarrow \theta_0$; $a^* \leftarrow 0$; // Initialization
2 **while** $\mathcal{D} \setminus \mathcal{E} \neq \emptyset$ **do**
    /* **Roll-out**: Search for per-sub-dataset peaks */
3 θ, $\{\mathrm{acc}(\mathcal{D}_i, c)\}_{i,c} \leftarrow$ SFT-ROLL-OUT$(\hat{\theta},\ \mathcal{D} \setminus \mathcal{E},\ C)$;
4 $c_i^* \leftarrow \arg\max_c\ \mathrm{acc}(\mathcal{D}_i, c) \quad \forall \mathcal{D}_i \notin \mathcal{E}$; // Optimal compute per sub-dataset
5 /* During the roll-out, checkpoints $\theta(c_i^*)$ for remaining datasets $\forall \mathcal{D}_i \notin \mathcal{E}$ are written to Disk */
6 $c_{\min}, \mathcal{D}_{\text{exclude}} \leftarrow \arg\min_{\mathcal{D}_i \notin \mathcal{E}}\ c_i^*$;
7 **if** $c_{\min} = C$ **then**
        /* No overfitting: update model and continue */
8 $\hat{\theta} \leftarrow \theta(C)$;
9 **else**
        /* **Roll-back**: Revert to the checkpoint where the sub-dataset overfit */
10 $\mathcal{E} \leftarrow \mathcal{E} \cup \{\mathcal{D}_{\text{exclude}}\}$;
11 $\hat{\theta} \leftarrow$ Load $\theta(c_{\min})$ from Disk ; // Revert to checkpoint at $c_{\min}$
12 **end**
13 /* Update θ^* to be the model parameters of the highest accuracy */
14 $c_{\text{best}} \leftarrow \arg\max_c\ \mathrm{acc}(\mathcal{D}, c)$; $\quad a_{\text{best}} \leftarrow \mathrm{acc}(\mathcal{D}, c_{\text{best}})$;
15 **if** $a_{best} > a^*$ **then**
16 $a^* \leftarrow a_{\text{best}}$; $\theta^* \leftarrow \theta(c_{\text{best}})$;
17 **end**
18 Discard all checkpoints from Disk except $\hat{\theta}$ and θ^*;
19 **end**
20 **return** θ^*

Checkpoint management. Algorithm 4 details the checkpoint management strategy integrated into MSFT, where blue annotations denote disk-management operations added atop the base algorithm. While standard SFT retains only a single checkpoint on disk throughout training, MSFT requires additional storage during the **roll-out** phase: per-dataset peak checkpoints $\theta(c_i^*)_{\mathcal{D}_i \notin \mathcal{E}}$ are persisted as they are identified (line 5), requiring up to $|\mathcal{D} \setminus \mathcal{E}_s|$ checkpoints at stage s. Upon completing each iteration, the algorithm retains only the rollback checkpoint $\hat{\theta}$ and the global best checkpoint θ^* — the model that achieved the highest overall accuracy across all stages — and discards all remaining checkpoints (lines 13–18). The theoretical peak occurs at the second stage, where $|\mathcal{D}| - 1$ live per-dataset peaks coexist with the two retained checkpoints ($\hat{\theta}$ and θ^*), yielding a worst-case of $|\mathcal{D}| + 1$ model copies on disk. Averaging the per-stage peaks across all $|\mathcal{D}|$ stages gives:

$$\frac{1}{|\mathcal{D}|} \sum_{s=1}^{|\mathcal{D}|} \big(\min(|\mathcal{D}| - s + 1,\ E) + 2\big),$$

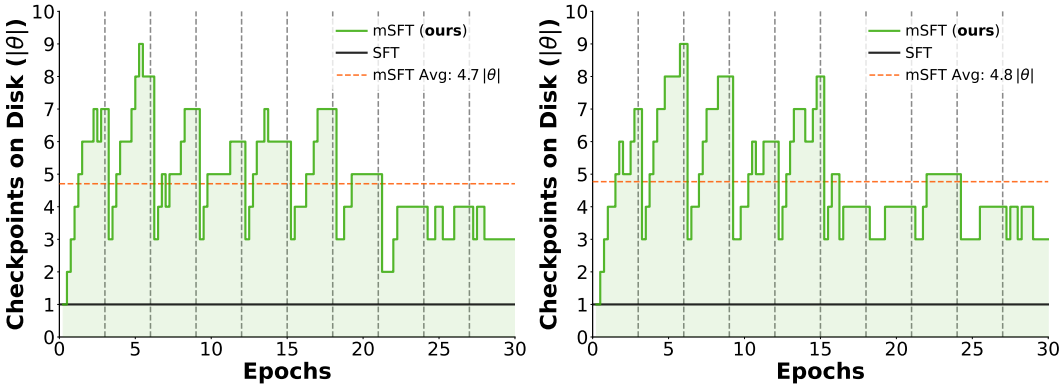
where E is the number of evaluation steps per stage and $+2$ accounts for the retained $\hat{\theta}$ and θ^* (for $s \geq 2$; stage 1 retains none, but the over-count vanishes as $|\mathcal{D}|$ grows). When $E \geq |\mathcal{D}|$, i.e. the evaluation grid is finer than the number of sub-datasets, the min reduces to $|\mathcal{D}| - s + 1$ and the average simplifies to $\frac{|\mathcal{D}|+5}{2}$. For our experiments with $|\mathcal{D}| = 10$ and $C = 3$ epochs evaluated every 0.25 epochs ($E = 12 > |\mathcal{D}|$), this predicts a peak of 11 and an average of 7.5 model copies. In practice, multiple categories often share the same peak epoch, so several per-dataset champions collapse onto a single checkpoint. Empirically, across MSFT runs with compute budgets $C \in \{1, 3\}$ on multiple dataset mixtures, we observe an average disk utilization of $4.44|\theta|$, well below the $|\mathcal{D}| + 1$ theoretical bound. (see Appendix I Figs. 16, 17, 18)

I Disk Storage Footprint



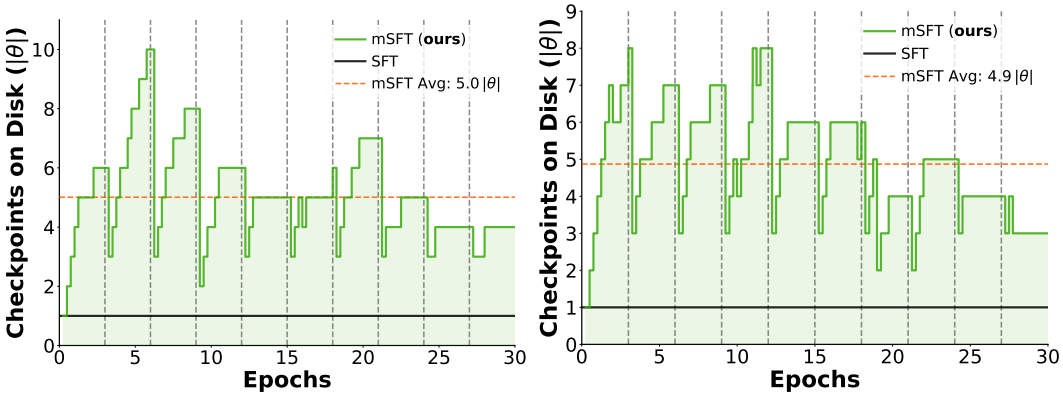
(a) `Olmo 2 1B`, $C = 3,\ N = 10$

(b) `Qwen2.5 0.5B`, $C = 3,\ N = 10$



(c) `Qwen2.5 1.5B`, $C = 3,\ N = 10$

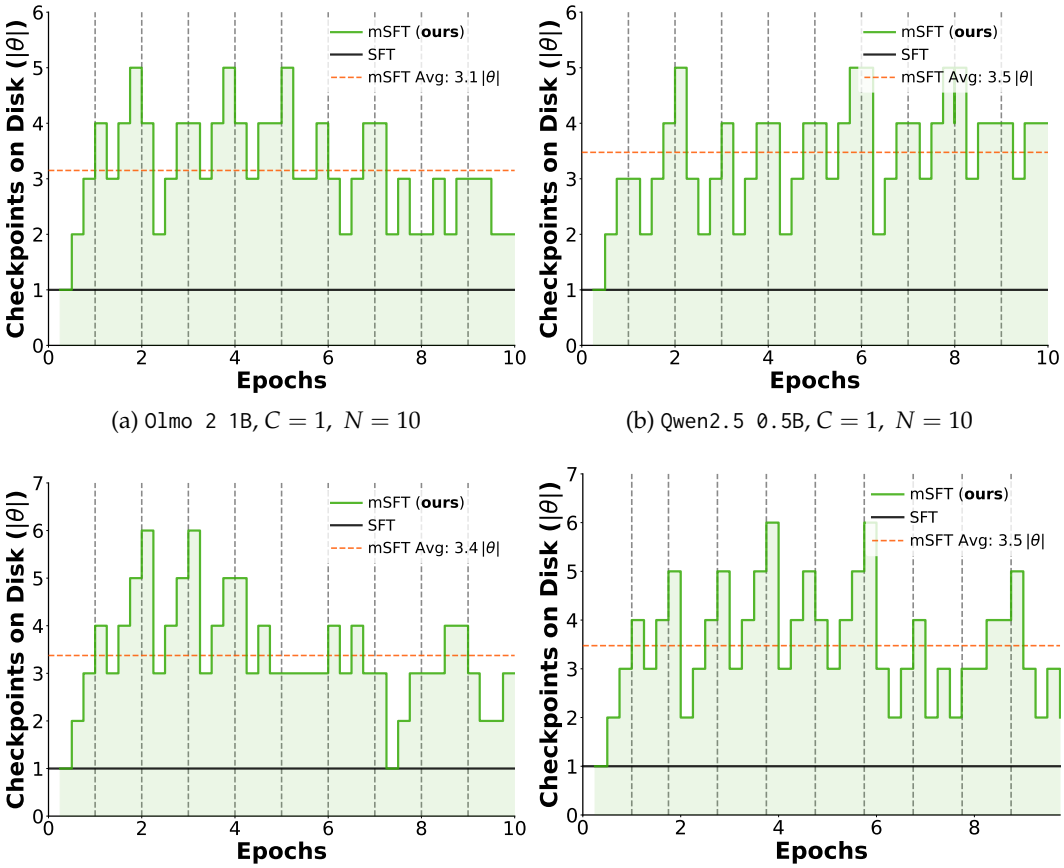
(d) `Qwen2.5 3B`, $C = 3,\ N = 10$



(e) `Qwen2.5 7B`, $C = 3,\ N = 10$

(f) `Qwen3 8B`, $C = 3,\ N = 10$

Figure 16: **Disk utilization across MSFT iteration.** Each point denotes the number of checkpoints on disk at a given evaluation step, measured in multiples of model size $|\theta|$. Dashed vertical lines mark new roll-outs. The orange horizontal line indicates the average utilization across all evaluation steps.



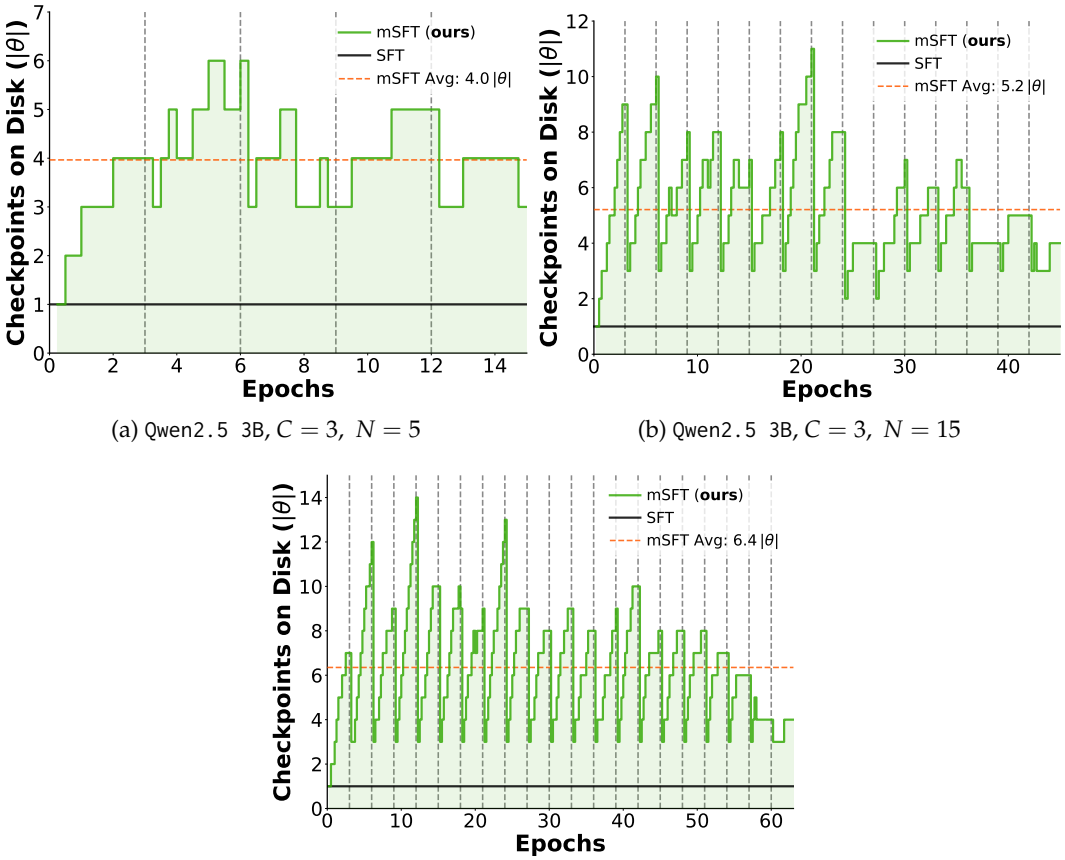
(a) `Olmo 2 1B`, $C = 1,\ N = 10$

(b) `Qwen2.5 0.5B`, $C = 1,\ N = 10$

(c) `Qwen2.5 1.5B`, $C = 1,\ N = 10$

(d) `Qwen2.5 3B`, $C = 1,\ N = 10$

Figure 17: **Disk utilization across MSFT iteration.** Each point denotes the number of checkpoints on disk at a given evaluation step, measured in multiples of model size $|\theta|$. Dashed vertical lines mark new roll-outs. The orange horizontal line indicates the average utilization across all evaluation steps.



(a) `Qwen2.5 3B`, $C = 3,\ N = 5$

(b) `Qwen2.5 3B`, $C = 3,\ N = 15$

(c) `Qwen2.5 3B`, $C = 3,\ N = 21$

Figure 18: **Disk utilization across MSFT iteration.** Each point denotes the number of checkpoints on disk at a given evaluation step, measured in multiples of model size $|\theta|$. Dashed vertical lines mark new roll-outs. The orange horizontal line indicates the average utilization across all evaluation steps.