Revisiting Complete Reasoning Traces for Post-Training

Jaehui Hwang[†] **Sangdoo Yun** **Byeongho Heo** **Dongyoon Han**[†]
NAVER AI Lab
{jaehui.hwang, dongyoon.han}@navercorp.com

Abstract

Large language models (LLMs) are often post-trained on pre-collected reasoning trajectories to improve their reasoning capability. Such trajectories tend to be long due to complex, interwoven paths, which often include detours on the path toward the answer. However, it has been underexplored whether LLMs indeed benefit from learning complete trajectories in post-training, such as supervised fine-tuning (SFT). Starting from our pilot study, we find that full trajectories provide only limited benefit, while partial trajectories are effective even under heavy truncation. We analyze redundancy in reasoning trajectories through attention-based analyses and controlled token-removal studies, both of which show that intermediate tokens contribute minimally to final reasoning quality. This suggests that avoiding redundant information may allow LLMs to internally infer coherent alternatives by inferring missing steps from their internal knowledge, given known trajectory endpoints. Furthermore, we show that training LLMs using endpoints leads to consistent changes in reasoning behavior, and that it also benefits post-training methods based on reinforcement learning or on-policy distillation, highlighting the need to revisit complete reasoning traces. Code is available at here.

1 Introduction

How human learners acquire writing skills may offer a helpful analogy for understanding how large language models (LLMs) acquire reasoning skills. Learners benefit from carefully curated texts (*e.g.,* mentor texts), which are essential for building writing fundamentals (Kim et al., 2021; Culham, 2023; Shubitz, 2023). Yet, as learners advance, they increasingly dismiss misleading and less curated information as unhelpful, gaining little from it; they rather place more value on more essential guidance, which would drive more effective learning. A follow-up question would be: if provided only with essentials to learn, yet disconnected parts, would they learn better and use prior knowledge to internally predict the missing pieces on their own?

LLMs have recently achieved strong results on reasoning tasks driven by large reasoning models (LRMs) (Hurst et al., 2024; Yang et al., 2025, 2024; Guo et al., 2025). The tasks span domains such as mathematical problem solving and logical reasoning, which demand complex processes not known to be achievable solely through pre-training, mid-training, or alignment. LRMs are often trained through reward-based learning (Shao et al., 2024), or by leveraging pre-collected reasoning traces (Muennighoff et al., 2024; Team, 2025; Labs, 2025; Guha et al., 2025) in a manner similar to how human learners are guided through mentor texts. Reasoning traces represent trajectories from a question to its answer and are usually effective supervisory signals for learning reasoning; however, they often contain redundant or unnecessary steps. Following the earlier analogy, one may ask whether a model trained only on essential subpaths could still perform favorably and internally recover the missing pieces with more plausible ones. However, even these are less explored: identifying which parts of the reasoning trajectory are redundant (in a broader sense) and formulating a principle to exploit only the essential ones.

Existing studies have examined the reasoning trajectories of LRMs, which are intentionally separated from their final answers to make the reasoning process explicit. Muennighoff et al. (2024) first introduced the idea of using *entire* reasoning traces as guidance for supervised fine-tuning (SFT), and subsequent datasets such as Guha et al. (2025) have further contributed reasoning traces for learning. Subsequently, however, several studies have noted that the reasoning trajectories produced by LRMs are typically lengthy, less informative, and sometimes misleading (Ma et al., 2025; Cuesta-Ramirez

[†]Corresponding authors.

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 47374–47392
October 24-29, 2026 ©2026 Association for Computational Linguistics

et al., 2025; Wu et al., 2025). Prior work further suggests that LLMs often already know the answer before generating a fully explicit reasoning trace (Lindsey et al., 2025). Ma et al. (2025) and Wang et al. (2025) demonstrate that some problems can be efficiently solved without any explicit reasoning process, albeit with lower performance sometimes. These observations suggest that reasoning trajectories are important for solving complex problems, yet the fully explicit trajectory may not always be necessary. However, understanding where and why such parts can be skipped, particularly for effective training, is still underexplored.

This paper focuses on SFT training using existing reasoning traces for LLMs (Muennighoff et al., 2024; Team, 2025; Labs, 2025; Guha et al., 2025) in the post-training stage. Our insight is that such traces often contain redundancy, and that coarsely trimming contiguous segments enables more effective learning with even fewer training traces. Our pilot study, using LLMs trained on limited reasoning traces, provides initial support for our hypothesis. We further examine this insight through two systematic analyses: (1) interpreting attention patterns motivated by prior work (Ameisen et al., 2025; Clark et al., 2019; Madaan et al., 2023; Geva et al., 2023) and (2) analyzing answer perplexities after replacing segments. We find that answer tokens place little attention on transitional steps, and that skipping these steps preserves answer quality.

This indicates that a pre-trained LLM does not fully rely on the complete reasoning trajectory; some of the transitional steps are largely redundant and non-essential, and skipping them allows the LLM to internally infer the missing parts using its internal knowledge and the known trajectory. Motivated by this, we propose a simple approach that trims a number of steps of reasoning trajectories. We focus on endpoints of trajectories to achieve more effective SFT, as demonstrated on OpenThoughts (Guha et al., 2025) and s1K (Muennighoff et al., 2024), outperforming several competitive filtering-based methods. We further show that this extends beyond SFT, to reinforcement learning (RL)- and distillation-based post-training.

2 Related work

Test-time scaling and overthinking. Muennighoff et al. (2024) introduced a small, curated dataset dubbed s1K, comprising 1,000 reasoning trace examples that satisfy difficulty, diversity, and quality. Their SFT-trained model (*i.e.,* s1-32B) exhibits strong improvements in competition math questions (*e.g.,* AIME24) and shows a clear positive scaling-trend with test-time compute. After it, some recent works have challenged the assumption that longer reasoning chains always improve performance. Ghosal et al. (2025) analyzed test-time scaling and showed that beyond a certain point, longer chains can actually hurt accuracy due to *overthinking* rather than enhancing reasoning. Similarly, Hassid et al. (2025) demonstrated that, when multiple chains are sampled for the same question, the *shortest chain is often more reliable than the longest*, motivating inference-time strategies such as short-m@k and fine-tuning on shorter demonstrations. However, these approaches treat *length itself* as the key factor, without considering whether segments of trajectories contribute unequally.

Reasoning compression and adaptive selectivity. A similar line of work seeks to *compress* reasoning traces or *adaptively decide* how much to reason also for efficiency. At the trajectory level, Hou et al. (2025) pruned tokens *primarily considering sequence length* via reinforcement learning with an explicit token length limit, iteratively tightening the limit to shorten thoughts with minimal accuracy loss. Fan et al. (2025) proposed that *a light instruction model* drafts a high-level outline, and a reasoning model fills in details, which reduces generated tokens while maintaining accuracy, thus enabling difficulty-aware depth adjustment. Yuan et al. (2025) introduces a token-level compression framework that *scores reasoning tokens* and trains on compacted CoT while preserving accuracy. Lin et al. (2024) demonstrated at *pretraining* time that not all tokens are equally useful: they focus loss selectively on *high-utility tokens* (*i.e.,* useful and clean tokens in their terms), improving data efficiency, which is conceptually aligned with compression leading to faster training. These approaches either rely on token-importance scoring, budgeted RL, or guidance policies. By contrast, our approach offers a simple *segment-level* recipe at training time: systematically skip the middle span of machine-generated chains during SFT, while keeping the earlier and later spans, without sophisticated auxiliary scorers or controllers.

3 Background

3.1 Preliminary

Supervised fine-tuning (SFT)-based training. Recent methods with carefully curated

data: Sky-T1 (Team, 2025), s1K (Muennighoff et al., 2024), Bespoke-Stratos (Labs, 2025), and OpenThoughts (Guha et al., 2025) suggest a promising paradigm, beyond RL-based reasoning training (Shao et al., 2024): reasoning models can be effectively trained with curated, machine-generated traces produced by an LRM.

Formally, let $\mathcal{D} = \{(x_i, y_i, r_i)\}_{i=1}^{N}$ denote a dataset of input query and answer pairs (x_i, y_i) augmented with a reasoning trace r_i generated by a LRM T (*e.g.,* DeepSeek-R1 (Guo et al., 2025)). SFT then optimizes parameters θ of a language model M_θ (usually smaller than T *e.g.,* 3B, 7B, or 32B scale) by minimizing the negative log-likelihood: $\mathcal{L}_{\text{SFT}}(\theta) = -\sum_{i=1}^{N} \log p_\theta(y_i, r_i \mid x_i)$. In practice, reasoning trace r_i may be partially truncated if it exceeds the maximum token length allowed during training, typically 16k or 32k tokens.

In this setting, reasoning traces provided by T act as rich supervision signals, enabling smaller models M_θ to acquire strong reasoning ability without resorting to RL-based objectives (*e.g.,* GRPO (Shao et al., 2024)). This setup is particularly practical: large-scale traces are already well-formed by powerful teacher models, and SFT alone has been shown to yield competitive performance in smaller models.

Problem setting. We focus on the post-training stage's SFT with pre-collected reasoning traces (Guha et al., 2025; Muennighoff et al., 2024), applied to LLMs ($\simeq$32B parameters). While RL often surpasses SFT (Chu et al., 2025), we presume that applying SFT is reasonable in this setting, both in terms of simplicity and expected performance. Muennighoff et al. (2024)'s findings implicitly supported the assumption: when large teacher models (*e.g.,* DeepSeek-R1 671B (Guo et al., 2025)) produce trainable reasoning traces (while some redundancy exists), it is realistic to utilize these traces in smaller models through SFT. Furthermore, since *SFT typically precedes RL methods* in RLHF procedures (Ouyang et al., 2022; Liu et al., 2024a; Achiam et al., 2023; Groeneveld et al., 2024) or is employed successively (OLMo et al., 2024; Yang et al., 2025), understanding the mechanics of SFT in relation to reasoning can give valuable insights.

3.2 Pilot Study: Are All Traces Performance-Relevant?

We begin with a pilot study that explores which parts of reasoning trajectories contribute most to effective SFT. We adopt an experimental setup for prototyping, using s1K-1.1 (Muennighoff et al., 2024), which consists of 1K curated reasoning questions and train Qwen2.5-32B-Instruct (Yang et al., 2024) and Qwen3-8B-Base (Yang et al., 2025). We evaluate the resulting models on AIME24 (OpenAI, 2024), GPQA-Diamond (GPQA-D) (Rein et al., 2024), and MATH500 (Lightman et al., 2023), which are used as benchmarks for assessing the effectiveness of reasoning supervision. Our goal is to rapidly examine how retaining different parts of the reasoning trajectory affects SFT, and we test four representative settings: (1) Complete (full) trajectory, which uses the complete reasoning trajectory, with coarse truncation applied at the s1 context limit (20k tokens); (2) Prefix-only, which retains only the beginning part of the trajectory; (3) Suffix-only, which retains only the ending part of the trajectory; and (4) Both, which skips the intermediate part of the trajectory and retains both the beginning and ending parts, with a total retained length matched to the other settings. To retain meaningful segments, we define reasoning steps by splitting trajectories at double newline characters (`\n\n`), which tend to correspond to natural semantic breakpoints in reasoning (see §B.2). In this pilot study, we fix the total number of retained steps to 200 for settings (2)–(4).

Table 1 reports the average performance across the three benchmarks. SFT with retaining both the prefix and suffix of reasoning trajectories consistently outperforms both prefix-only and suffix-only variants. Moreover, it achieves a higher average performance than training with the full reasoning trajectory. In contrast, trimming the beginning or end of the trajectory incurs degraded or marginal gains, suggesting that these segments contain more critical information for effective reasoning supervision. These provide evidence that *not all parts of a reasoning trajectory contribute equally* during SFT, and that using complete trajectories may not be optimal. Furthermore, they suggest that coarse truncation would continue to be effective, even

	Full	Prefix-only	Suffix-only	Both
Qwen2.5-32B	73.51	73.67	71.83	75.19
Qwen3-8B	63.91	60.99	56.98	64.48

Table 1: **Average performance for segment variants** across AIME24, GPQA-D, and MATH. Both (Prefix+Suffix) retained trajectories achieve the best overall performance.

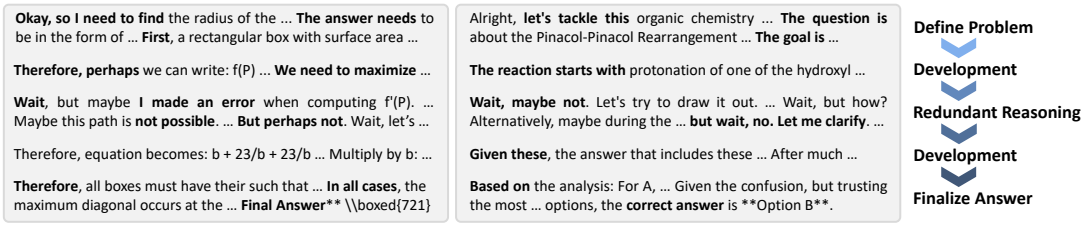


Figure 1: **Two example reasoning trajectories** from SFT training datasets (left and middle) and the corresponding high-level reasoning process highlights redundant intermediate reasoning segments (right).

without counting on hand-tuned or complicated filtering methods.

In the following sections, we further ground the previous experiments by analyzing why truncating substantial transitional segments of reasoning trajectories can be particularly effective, and demonstrating how this insight leads to a more effective SFT paradigm.

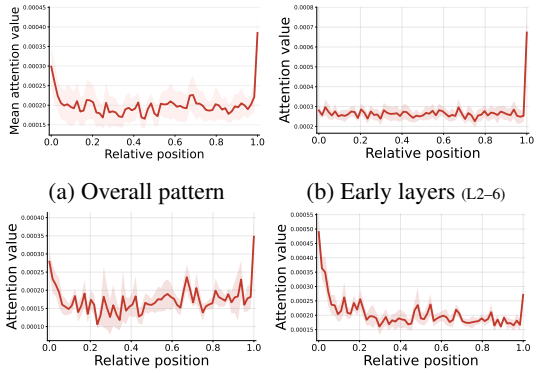
4 Revisiting Reasoning Trajectories

This section first analyzes the impact of reasoning trajectory segments that LLMs process. We then propose a simple segment-level supervision strategy for SFT that retains only the beginning and ending segments of a reasoning trace under context length constraints. Finally, we show that this simple strategy can match or outperform more complex approaches, and that it extends beyond SFT to other post-training objectives.

4.1 How LLMs Process Reasoning Segments?

Figure 1 illustrates example instances and the overall structure of reasoning trajectories. Such trajectories typically proceed from problem definition, through exploratory reasoning, to final answer consolidation. To investigate the functional roles of different parts in reasoning trajectories, we examine representative examples across reasoning tasks. Intriguingly, as noted in prior work (Guo et al., 2025; Ma et al., 2025; Cuesta-Ramirez et al., 2025; Wu et al., 2025), we often observe redundant traces (*e.g.,* repeated checks, backtracking, or unnecessary elaborations) in the middle of a trajectory. A quantitative breakdown of what each region contains is provided in §B.3. We now systematically investigate whether such human-perceived redundancy matters for LLMs.

Interpreting attention weights. Following Ameisen et al. (2025), Clark et al. (2019), Madaan et al. (2023), and Geva et al. (2023) that interpreted models using attention-based metrics, we analyze attention weights to investigate how different parts



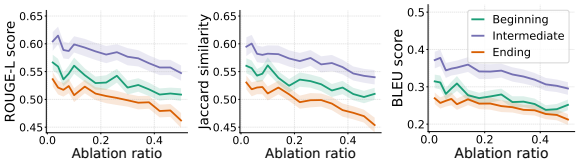
(a) Overall pattern (b) Early layers (L2–6)

(c) Middle layers (L25–35) (d) Late layers (L60–64)

Figure 2: **Averaged 1D attention weights across reasoning trajectories** during answer generation. (a) The overall average highlights strong attention peaks at the beginning and end of trajectories. (b–d) Layer-specific patterns reveal a progression: early layers attend candidate answers, intermediate layers jointly reference problem-definition and reasoning steps, and final layers concentrate on the beginning to format and consolidate the answer.

of reasoning trajectories contribute to answer generation. We expect these patterns to reveal how models prioritize tokens, offering insights into information flow within transformer architectures.

Figure 2 provides two complementary insights. First, the overall (*i.e.,* averaged across layers) trend in Figure 2(a) highlights *strong attention peaks at the beginning and ending of trajectories*, whereas intermediate steps are weakly attended. This suggests that intermediate steps generally contribute less to answer generation, indicating that these tokens could be redundant within the overall reasoning trajectory. Second, the layer-wise patterns in Figure 2(b–d) reveal a progression across the model depth: early layers (pattern 1) shift focus toward candidate answer tokens, intermediate layers (pattern 2) balance attention between early problem-definition and later reasoning steps, and final layers (pattern 3) concentrate strongly on the beginning to format and consolidate the final answer. Taken together, these suggest how information flow evolves through the network and provide evidence that in-



(a) ROUGE-L (b) Jaccard (c) BLEU

Figure 3: **Segment ablation results** for each segment of reasoning traces. All three metrics consistently indicate that intermediate segments have a limited influence.

termediate reasoning steps are de-emphasized as generation progresses toward the later layer.

Segment ablation-based analysis. Inspired by the attention knockout method (Geva et al., 2023), we probe the importance of specific trajectory segments by removing them from the input and re-generating answers, without any fine-tuning. Specifically, we truncate the beginning (*e.g.,* the first 0–10% of tokens), the intermediate (*e.g.,* a centered span such as 45–55%), or the ending (*e.g.,* the last 90–100%) and compare the resulting answers against those from the full trajectories. This allows us to retest which parts of reasoning trajectories are causally important for answer generation. Details on the experimental setup and evaluation metric are provided in Appendix B.1.

Based upon the observational insights from attention analysis, Figure 3 shows the average results over samples across different segment ablation ratios. In the results, *removing the intermediate segments consistently yields the highest similarity* to answers generated with the full trajectory. In contrast, removing the beginning and ending parts results in lower similarity, suggesting that these segments contain critical information for formulating correct answers. This provides quantitative evidence for our hypothesis that intermediate steps are often redundant and do not substantially improve the quality of the final answer.

Answer perplexity-based analysis. Previously, we argued that LLMs may (internally) infer missing reasoning trajectories with more plausible alternatives, and that the final answer can be affected by whether such segments are retained or replaced with LLM-generated outputs. Observing this effect with a pretrained model without SFT suggests that this behavior may already emerge at even early (post-) training stages.

Specifically, we replace the beginning, middle, or ending segments of reasoning trajectories with self-generated content produced by the pretrained model (not SFT post-trained), and evaluate the perplexity of the original answer tokens under the modified trajectory. If a replaced segment was not critical for answer generation, we expect the answer perplexity to decrease; for segments that were critical, the answer perplexity should increase.

	Replaced Part			No
	Begin	Middle	End	replacement
Answer PPL	1.56	**1.52**	1.58	1.54
ΔPPL (vs. Full)	+0.02	-0.02	+0.04	-

Table 2: **Answer perplexities** under segment-wise replacement of reasoning trajectories. Replacing intermediate reasoning steps preserves answer confidence, while replacing the beginning or ending degrades it.

Table 2 shows that the answer perplexity decreases only when intermediate segments of the reasoning trajectory are replaced. In contrast, replacing beginning or ending parts consistently leads to higher perplexity. This suggests that intermediate reasoning steps provide limited supervision for predicting the final answer, whereas the early and late segments carry more essential information for effective reasoning supervision; furthermore, LLMs may self-fill missing parts more effectively than processing redundant original segments. These results are consistent with the LLM-judge results in Section 5, which show that generated reasoning trajectories outperform redundant original ones.

Lessons from §4.1. ❶ LLM processes particular reasoning segments distinctly; they may self-clarify missing redundant parts during training. ❷ This segment-level distinction is clearly observable even under coarse segment-level truncation of reasoning trajectories, leading to variations in reasoning capability.

4.2 Learning from Reasoning Trajectory Segments

Inspired by the findings in §3, and grounded in the analyses presented in §4.1, we propose a simple yet effective segment-level supervision strategy for reasoning SFT. Specifically, given machine-generated reasoning trajectories, we preserve the early and late segments while removing the intermediate segments. The retained segments are determined based on step count; we simply target removing approximately 20% of total tokens across the dataset, as performance is insensitive to this ratio (see Figure B). Further analysis of how the retained steps affect performance is presented in §A. Detailed settings are described in §C.3.

Experimental settings. We conduct experi-

Method	AIME24	GPQA-D	MATH	Avg
Qwen2.5-32B-Instruct & s1K-1.1				
Full	64.44	61.95	94.13	73.51
Prefix	62.22	**63.80**	**95.00**	73.67
Suffix	60.00	61.28	94.20	71.83
Both	**68.89**	62.29	94.40	**75.19**
Qwen3-8B-Base & OpenThoughts3-100K				
Full	50.00	53.87	93.47	65.78
Prefix	40.00	53.20	93.27	62.16
Suffix	41.11	51.35	87.13	59.86
Both	**52.22**	**56.40**	**93.87**	**67.50**
Qwen3-4B-Base & OpenThoughts3-100K				
Full	36.67	48.65	**91.60**	58.97
Prefix	38.89	46.46	89.13	58.16
Suffix	28.89	44.95	84.20	52.68
Both	**40.00**	**48.99**	91.13	**60.04**

Table 3: **Performance of segment-level SFT** on three benchmarks. **Boldface** indicates the best performance in each benchmark, and underline indicates performance better than baseline, standard SFT with full trajectories.

ments on two reasoning SFT datasets: s1K-1.1 (Muennighoff et al., 2024) and OpenThoughts3-1.2M (Guha et al., 2025). For OpenThoughts3, we subsample 100K examples from the full 1.2M corpus, and further restrict to instances where both the begin-of-think (`<think>`) and end-of-think (`</think>`) tokens are present. On s1K-1.1, we fine-tune `Qwen2.5-32B-Instruct` and `Qwen3-8B-Base` for 5 epochs, while on OpenThoughts3 we fine-tune `Qwen3-8B-Base` and `Qwen3-4B-Base` for 1 epoch. More detailed settings are described in Appendix C.3. We also utilize models from the Llama family (Grattafiori et al., 2024), and the corresponding details are mentioned in Appendix C.4. We mainly use AIME24 (OpenAI, 2024), GPQA-D (Rein et al., 2024), and MATH (Lightman et al., 2023) to evaluate reasoning performance; TruthfulQA (Lin et al., 2022), MMLU (Hendrycks et al., 2021), HellaSwag (Zellers et al., 2019), and WinoGrande (Sakaguchi et al., 2021) to assess general-purpose language performance; and LiveCodeBench (Jain et al., 2024) and CodeElo (Quan et al., 2025) to evaluate code generation performance.

Results. Table 3 reports the performance of this coarse segment-level learning strategy across multiple model scales and datasets. Consistent with the results in Table 1, retaining only the beginning and ending parts of reasoning trajectories does not degrade performance compared to using full trajectories. In several settings, this strategy even leads to improved performance over the baseline.

Notably, preserving both prefix and suffix segments of the reasoning trajectory is consistently more effective than retaining only the beginning or only the ending. This indicates that, when reasoning trajectories must be shortened due to context-length constraints, prioritizing the retention of early and late reasoning steps is a more reliable strategy than uniformly using the entire trajectory. These results align with our earlier analyses (§4.1), which showed that intermediate reasoning steps are weakly attended, causally less important, and often redundant for answer generation.

Benchmark	Baseline	SFT	`E-SFT`
General language-centric			
TruthfulQA-MC1	37.09	32.56	33.17
TruthfulQA-MC2	53.42	49.68	49.65
MMLU	73.19	72.53	72.90
HellaSwag	55.82	54.77	54.86
WinoGrande	71.43	70.40	70.64
Code generation			
LiveCodeBench	4.30	38.40	38.47
CodeElo	3.32	11.97	11.59

Table 4: **`E-SFT` on language-centric and code generation benchmarks** using Qwen3-4B-Base and OpenThoughts3-100K. "Baseline" and "SFT" stand for the original Qwen3-4B-Base model without any post-training (SFT) and the model trained on the standard SFT without any modification, respectively.

Overall, these findings suggest that longer reasoning trajectories are not necessarily more informative for SFT, and that redundant intermediate segments need not be included. We refer to this approach as Endpoint-based SFT (`E-SFT`), in which the model is trained on endpoints of reasoning trajectories that retain the beginning and ending segments rather than the full trajectories.

Additional results on general benchmarks. We further examine the effect of `E-SFT` on general language-centric datasets, which are not the primary focus of reasoning-specialized post-training. As shown in Table 4, since the baseline without any reasoning-oriented SFT already demonstrates strong results, domain-specific SFT tends to reduce performance on general-language benchmarks, resulting in noticeable degradation after training. However, the model trained with `E-SFT` generally outperforms the model trained with standard SFT. This indicates that skipping redundant intermediate reasoning steps helps preserve general linguistic ability during domain-specific fine-tuning.

In addition, OpenThoughts3 contains code-related instances. When evaluated on code-generation benchmarks, the model trained with `E-SFT` achieves performance comparable to other

approaches. These results collectively suggest that E-SFT does not compromise model quality across different evaluation domains. Furthermore, the generality of E-SFT across the other model families is demonstrated in §C.4.

4.3 Comparison with Various Filtering Methods

E-SFT can be viewed as one of the trajectory-reduction methods for reasoning training. To demonstrate the effectiveness of our method as a filtering approach, we compare it with various trajectory-reduction methods (Xie et al., 2023; Li et al., 2024b,a; Yu et al., 2025a). It is noteworthy that our goal is not to propose a new trajectory-reduction technique, but rather to rethink the standard SFT practice of using complete reasoning trajectories or naively truncating them based on a fixed token limit.

Compared filtering methods. We compare E-SFT to the setting that uses the original SFT dataset without any processing, as well as to several alternative trajectory-reduction methods. These methods include: (1) Random Step Selection: we randomly sample $2n$ steps from the original trajectory, preserving the same amount of content but without structural consideration. (2) Similarity-based filtering: we compute Jaccard similarity between each step and the preceding 5 steps of the trajectory. Steps exceeding a predefined similarity threshold are filtered out to reduce redundancy, targeting repetitive reasoning patterns. (3) LLM-based step compression: we use external LLMs – Claude Sonnet* (Anthropic, 2025) and Gemini (gemini-2.5-flash) (Comanici et al., 2025) – to compress reasoning trajectories while preserving essential reasoning content. In the main text we report results with Claude, while detailed results with both LLMs are provided in Appendix C.4. (4) Perplexity-based Step Filtering: we utilize perplexity to filter out steps with high perplexity or with extreme values (either high or low) (Xie et al., 2023; Li et al., 2024b,a). (5) LS-Mixture SFT: proposed in (Yu et al., 2025a), where the authors incorporate LLM-based compression into a mixture-style SFT pipeline. Note that methods (3)–(5) require additional computational resources to reduce trajectories, while methods (1)–(2) do not.

Experimental results. Table 5 shows the comparison results of E-SFT (ours) against resource-free filtering strategies, and ours consistently achieves the best performance. In contrast, similarity-based filtering performs less reliably, as it can misclassify important early steps as redundant due to high lexical overlap, leading to the removal of critical definitions or problem setups.

Method	32B	8B
Standard SFT	73.51	63.91
Random	70.92	58.20
Similarity	73.92	59.94
Ours	**75.19**	**64.48**

Table 5: **Comparison with resource-free filtering strategies** on s1K-1.1.

Method	Avg.
Standard SFT	73.51
LLM-based	60.08
PPL-extreme	72.80
PPL-high	74.43
LS-Mixture	71.90
Ours	**75.19**

Table 6: **Comparison with resource-intensive filtering strategies** on s1K-1.1 (Qwen2.5-32B).

Then, Table 6 reports the comparison results of E-SFT (ours) against resource-intensive filtering strategies. LLM-based compression performs noticeably worse despite using extra computational costs of running LLM, because it may disrupt patterns of word usage and the underlying reasoning structure, and does not provide precise control over the compression ratio. Perplexity-based filtering and LS-Mixture SFT achieve comparatively strong performance among these baselines, but neither surpasses E-SFT. Moreover, these methods require non-negligible computational resources; for example, the perplexity-based approach alone takes more than two hours on a single H100 node.

Lessons from §4.2, §4.3. Rather than relying on sophisticated filtering strategies, a simple SFT design that prioritizes learning from the sub-segments of reasoning trajectories also works well.

4.4 Beyond SFT: Generalization to RL- and Distillation-based Post Training

In the previous sections, E-SFT has mainly targeted SFT, with the core message that the full reasoning trajectory is not necessary. Beyond SFT, we also explore whether this message can be adopted by other training methods, such as group relative policy op-

*We utilize the claude-sonnet-4-20250514 version of Claude Sonnet 4.

Method	AIME24	MATH500	GPQA-D	Avg.
Qwen3-1.7B-Base				
Baseline	7.8	39.3	30.3	25.8
Ours	**8.9**	**60.7**	**33.8**	**34.5**
Qwen3-1.7B				
Baseline	26.7	80.8	46.0	51.2
Ours	**27.8**	**82.4**	**47.0**	**52.4**

Table 7: **Applying E-SFT to GRPO** on DAPO-17k. "Baseline" denotes GRPO without masking, and "Ours" masks the middle 20% of the reasoning trajectory from the training objective.

timization (GRPO) (Shao et al., 2024), a representative reinforcement learning-based post-training method, and on-policy distillation (OPD) (Agarwal et al., 2024; Gu et al., 2023; Lu and Lab, 2025), a widely used post-training method that trains a student on a teacher's distribution over its own generated trajectories.

E-SFT and GRPO. We train two types of models: a base model that is not post-trained after pretraining, Qwen3-1.7B-Base, and an instruct-tuned model, Qwen3-1.7B. We utilize the DAPO-17k dataset (Yu et al., 2025b). Similar to E-SFT, we mask the middle 20% of the reasoning trajectory when the trajectory is explicitly separated by special tokens (<think> and </think> for Qwen3). Note that, unlike E-SFT in §4.2 where the middle segments are removed from the trajectory, here masking means that the corresponding tokens remain in the context and are still scored by the reward, but do not contribute to the per-token gradient. Since Qwen3-1.7B-Base does not have a separated thinking part, we instead mask the middle 20% of the whole rollout. Detailed settings are described in §C.7.

As shown in Table 7, on both models, masking the middle part of the reasoning trajectory outperforms the GRPO baseline across all three benchmarks. The gain is larger for Qwen3-1.7B-Base (+8.7 on average, mostly from MATH500), which starts from a lower baseline and thus has more room for improvement, while the instruct-tuned Qwen3-1.7B still shows a consistent improvement (+1.2 on average). These results suggest that the core message of E-SFT also carries over to GRPO, where the learning signal comes from rewards rather than from teacher trajectories.

E-SFT and OPD. We further apply the same idea to on-policy distillation, using Qwen3-1.7B as the student and Qwen3-8B as the teacher, trained on OpenThoughts3. Here we try two variants of the mask to test whether masking granularity matters:

Method	AIME24	MATH500	GPQA-D	Avg.
Baseline	47.3	**87.0**	49.2	61.2
Ours (token)	49.8	86.7	49.3	61.9
Ours (step)	**52.2**	86.4	**51.7**	**63.4**

Table 8: **Applying E-SFT to OPD** on OpenThoughts3, with Qwen3-1.7B as the student and Qwen3-8B as the teacher. "Baseline" denotes OPD without masking, and both variants of "Ours" mask the middle 20% from the training objective, at the token or step level.

a token-level mask that excludes the middle 20% of tokens, and a step-level mask that masks the middle steps while keeping the first and last ones, following the same segment-level design as E-SFT in §4.2. Detailed settings are described in §C.7.

As shown in Table 8, in both variants, masking the middle 20% improves performance compared to using the full trajectory, and the step-level mask works better than the token-level one (+2.2 vs. +0.7 on average). This suggests that reasoning steps, rather than individual tokens, are the more natural unit for removing redundancy, in line with the segment-level view in §4.1. Note that SFT with teacher-generated traces is itself a kind of distillation, but here we show that a similar masking still works well when the objective is changed, from likelihood on a fixed corpus to reward-based learning on its own rollouts (GRPO) and to matching the teacher on-policy (OPD).

Lessons from §4.4. The benefit of skipping middle reasoning segments is not limited to the SFT objective. Beyond SFT, it carries over to RL-based post-training methods and on-policy distillation.

5 Why Does the Proposed Method Work?

In §4, we provide empirical evidence for E-SFT, mainly on the standard SFT setting. We further validate our claims through analytical investigations that explain why E-SFT works. Specifically, we analyze its impact from three complementary perspectives: (1) changes in generation behavior, (2) reasoning quality and token-length patterns, and (3) training loss dynamics during training.

Does E-SFT actually improve reasoning quality? We examine whether E-SFT causes changes in the actual reasoning behavior, as illustrated by the examples in Figure 4. In the upper row, where the baseline model produces unnecessarily lengthy and redundant output, E-SFT suppresses such detours, enabling the model to reach a more accurate so-

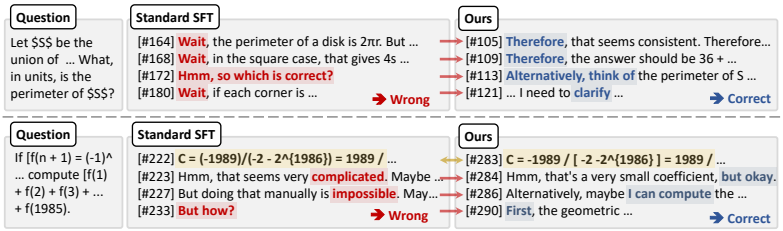


Figure 4: **Changes in generated reasoning traces by E-SFT.** The examples show that SFT with E-SFT reduces redundant reasoning and helps the model find more accurate reasoning pathways.

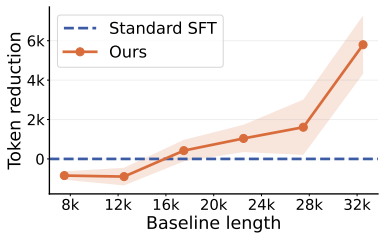


Figure 5: **Token length changes** compared to "Baseline length" from a standard SFT model.

lution with fewer tokens. In the lower row, although both models arrive at the same intermediate equations, the baseline fails to solve the problem, whereas E-SFT succeeds by more effectively completing the missing steps in the reasoning process.

To assess reasoning quality quantitatively, we conduct an LLM-as-judge evaluation using GPT-OSS-120B, which compares reasoning trajectories from the original s1K-1.1 with those generated by models trained with standard SFT and with E-SFT, and selects the one with less unnecessary repetition and smoother logical flow. Trajectories generated by E-SFT are preferred in 22% of cases (8% for the original data), with 70% judged as equivalent; standard SFT is preferred in 19% of cases (9%), with 72% considered equivalent. These results suggest an improvement in reasoning quality. This further indicates that even the initial pretrained model can perform better than using redundant trajectories, and that training with E-SFT yields improvements beyond this initial baseline. Additional details are provided in §C.6.

How does E-SFT change reasoning behavior? To better understand how E-SFT improves reasoning quality, we focus on changes in the token length of thinking trajectories. As shown in Figure 5, the number of reasoning tokens with E-SFT adapts to the baseline trajectories without E-SFT. When original trajectories are excessively long (>15k tokens), the output length decreases; for relatively shorter ones, token usage increases. Together with the LLM-as-judge results above, these observations suggest that E-SFT improves reasoning quality by (1) suppressing redundant reasoning patterns and (2) more effectively completing intermediate reasoning when needed.

Does E-SFT lead to a better convergence point? We further analyze why E-SFT is effective by examining the loss curves during training on OpenThoughts3, with and without E-SFT. As shown in Figure 6, our method achieves consistently lower perplexity during training while using fewer tokens compared to standard SFT. This finding supports our claim in §4.1 that intermediate segments are redundant and could be unnecessary for training; retaining such segments inflates the training loss, makes optimization harder for LLMs, and leads to convergence to suboptimal points. It suggests that, by trimming such redundancy, the model can allocate its capacity to more informative reasoning signals, which in turn facilitates more stable learning dynamics.

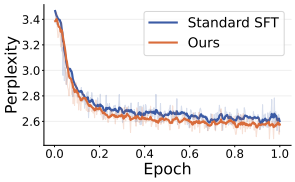


Figure 6: **Lowered perplexity** using E-SFT vs. SFT baseline during training.

6 Conclusion

We have revisited the necessity of complete reasoning trajectories and explored an effective SFT approach. In our pilot study, we have shown that using full trajectories, particularly the intermediate segments of reasoning trajectories, does not yield clear benefits. Motivated by this observation, we conducted thorough analyses of intermediate reasoning traces, which are redundant and not helpful to SFT training. Based on these findings, we introduced E-SFT, which uses incomplete reasoning trajectories and skips learning signals from the middle segments of reasoning trajectories. This approach improved reasoning performance and showed that the actual quality of generated responses is enhanced. Beyond SFT, we further masked the middle segments in GRPO and on-policy distillation, which improves reasoning performance under different training objectives as well. Overall, our findings suggest that complete reasoning trajectories are not always necessary supervision, not only for SFT but also for RL- and distillation-based post-training.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. 2024. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations*, volume 2024, pages 21246–21263.

Emmanuel Ameisen, Jack Lindsey, Adam Pearce, Wes Gurnee, Nicholas L. Turner, Brian Chen, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, Jonathan Marcus, Michael Sklar, Adly Templeton, Trenton Bricken, Callum McDougall, Hoagy Cunningham, Thomas Henighan, Adam Jermyn, Andy Jones, and 8 others. 2025. Circuit tracing: Revealing computational graphs in language models. *Transformer Circuits Thread*.

Anthropic. 2025. Claude 4. https://www.anthropic.com/claude/sonnet.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V Le, Sergey Levine, and Yi Ma. 2025. Sft memorizes, rl generalizes: A comparative study of foundation model post-training. *arXiv preprint arXiv:2501.17161*.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. 2019. What does bert look at? an analysis of bert's attention. *arXiv preprint arXiv:1906.04341*.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and 1 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

Jhouben Cuesta-Ramirez, Samuel Beaussant, and Mehdi Mounsif. 2025. Large reasoning models are not thinking straight: on the unreliability of thinking trajectories. *arXiv preprint arXiv:2507.00711*.

Ruth Culham. 2023. *Writing thief: Using mentor texts to teach the craft of writing*. Routledge.

Tri Dao. 2024. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Siqi Fan, Peng Han, Shuo Shang, Yequan Wang, and Aixin Sun. 2025. Cothink: Token-efficient reasoning via instruct models guiding reasoning models. *arXiv preprint arXiv:2505.22017*.

Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. 2023. Dissecting recall of factual associations in auto-regressive language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*.

Soumya Suvra Ghosal, Souradip Chakraborty, Avinash Reddy, Yifu Lu, Mengdi Wang, Dinesh Manocha, Furong Huang, Mohammad Ghavamzadeh, and Amrit Singh Bedi. 2025. Does thinking more always help? understanding test-time scaling in reasoning models. *arXiv preprint arXiv:2506.04210*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Dirk Groeneveld, Iz Beltagy, Pete Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Harsh Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, and 1 others. 2024. Olmo: Accelerating the science of language models. *arXiv preprint arXiv:2402.00838*.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. 2023. Minillm: On-policy distillation of large language models. *arXiv preprint arXiv:2306.08543*.

Etash Guha, Ryan Marten, Sedrick Keh, Negin Raoof, Georgios Smyrnis, Hritik Bansal, Marianna Nezhurina, Jean Mercat, Trung Vu, Zayne Sprague, and 1 others. 2025. Openthoughts: Data recipes for reasoning models. *arXiv preprint arXiv:2506.04178*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Michael Hassid, Gabriel Synnaeve, Yossi Adi, and Roy Schwartz. 2025. Don't overthink it. preferring shorter thinking chains for improved llm reasoning. *arXiv preprint arXiv:2505.17813*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Bairu Hou, Yang Zhang, Jiabao Ji, Yujian Liu, Kaizhi Qian, Jacob Andreas, and Shiyu Chang. 2025. Thinkprune: Pruning long chain-of-thought of llms via reinforcement learning. *arXiv preprint arXiv:2504.01296*.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2024. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*.

Young-Suk Grace Kim, Dandan Yang, Marcela Reyes, and Carol Connor. 2021. Writing instruction improves students' writing skills differentially depending on focal instruction and children: A meta-analysis for primary grade students. *Educational Research Review*, 34:100408.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Bespoke Labs. 2025. Bespoke-stratos: The unreasonable effectiveness of reasoning distillation. www.bespokelabs.ai/blog/bespoke-stratos-the-unreasonable-effectiveness-of-reasoning-distillation. Accessed: 2025-01-22.

Ming Li, Yong Zhang, Shwai He, Zhitao Li, Hongyu Zhao, Jianzong Wang, Ning Cheng, and Tianyi Zhou. 2024a. Superfiltering: Weak-to-strong data filtering for fast instruction-tuning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Ming Li, Yong Zhang, Zhitao Li, Jiuhai Chen, Lichang Chen, Ning Cheng, Jianzong Wang, Tianyi Zhou, and Jing Xiao. 2024b. From quantity to quality: Boosting llm performance with self-guided data selection for instruction tuning. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7602–7635.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *arXiv preprint arXiv:2305.20050*.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Zhenghao Lin, Zhibin Gou, Yeyun Gong, Xiao Liu, Ruochen Xu, Chen Lin, Yujiu Yang, Jian Jiao, Nan Duan, and Weizhu Chen. 2024. Not all tokens are what you need for pretraining. In *NeurIPS*.

Jack Lindsey, Wes Gurnee, Emmanuel Ameisen, Brian Chen, Adam Pearce, Nicholas L. Turner, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, Jonathan Marcus, Michael Sklar, Adly Templeton, Trenton Bricken, Callum McDougall, Hoagy Cunningham, Thomas Henighan, Adam Jermyn, Andy Jones, and 8 others. 2025. On the biology of a large language model. *Transformer Circuits Thread*.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024a. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024b. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*.

Kevin Lu and Thinking Machines Lab. 2025. On-policy distillation. *Thinking Machines Lab: Connectionism*. Https://thinkingmachines.ai/blog/on-policy-distillation.

Wenjie Ma, Jingxuan He, Charlie Snell, Tyler Griggs, Sewon Min, and Matei Zaharia. 2025. Reasoning models can be effective without thinking. *arXiv preprint arXiv:2504.09858*.

Aman Madaan, Katherine Hermann, and Amir Yazdanbakhsh. 2023. What makes chain-of-thought prompting effective? a counterfactual study. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 1448–1535.

Christopher D Manning. 2008. *Introduction to information retrieval*. Syngress Publishing,.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candes, and Tatsunori Hashimoto. 2024. s1: Simple test-time scaling. In *Workshop on Reasoning and Planning for Large Language Models*.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, and 1 others. 2024. 2 olmo 2 furious. *arXiv preprint arXiv:2501.00656*.

OpenAI. 2024. Learning to reason with llms. OpenAI blog.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Shanghaoran Quan, Jiaxi Yang, Bowen Yu, Bo Zheng, Dayiheng Liu, An Yang, Xuancheng Ren, Bofei Gao, Yibo Miao, Yunlong Feng, and 1 others. 2025. Codeelo: Benchmarking competition-level code generation of llms with human-comparable elo ratings. *arXiv preprint arXiv:2501.01257*.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2024. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: an adversarial winograd schema challenge at scale. *Commun. ACM*, 64(9).

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Stacey Shubitz. 2023. *Craft moves: Lesson sets for teaching writing with mentor texts*. Routledge.

NovaSky Team. 2025. Sky-t1: Train your own o1 preview model within $450. https://novasky-ai.github.io/posts/sky-t1. Accessed: 2025-01-09.

Chenlong Wang, Yuanning Feng, Dongping Chen, Zhaoyang Chu, Ranjay Krishna, and Tianyi Zhou. 2025. Wait, we don't need to" wait"! removing thinking tokens improves reasoning efficiency. *arXiv preprint arXiv:2506.08343*.

Yuyang Wu, Yifei Wang, Ziyu Ye, Tianqi Du, Stefanie Jegelka, and Yisen Wang. 2025. When more is less: Understanding chain-of-thought length in llms. *arXiv preprint arXiv:2502.07266*.

Sang Michael Xie, Shibani Santurkar, Tengyu Ma, and Percy S Liang. 2023. Data selection for language models via importance resampling. *Advances in Neural Information Processing Systems*, 36:34201–34227.

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, and 41 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

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, and 22 others. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Bin Yu, Hang Yuan, Haotian Li, Xueyin Xu, Yuliang Wei, Bailing Wang, Weizhen Qi, and Kai Chen. 2025a. Long-short chain-of-thought mixture supervised fine-tuning eliciting efficient reasoning in large language models. *arXiv preprint arXiv:2505.03469*.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, and 16 others. 2025b. Dapo: An open-source llm reinforcement learning system at scale. *Preprint*, arXiv:2503.14476.

Hang Yuan, Bin Yu, Haotian Li, Shijun Yang, Christina Dan Wang, Zhou Yu, Xueyin Xu, Weizhen Qi, and Kai Chen. 2025. Not all tokens are what you need in thinking. *arXiv preprint arXiv:2505.17827*.

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

Appendix

This appendix provides additional experimental details and results that complement the main paper. Appendix A discusses model scale and performance sensitivity of E-SFT, including associated limitations, and its connection to the "Lost in the Middle" phenomenon. In Appendix B, we elaborate on the experimental settings and extended results of our segment ablation analyses, and provide further analyses on how reasoning steps are delimited and what the trimmed region contains. Appendix C presents further implementation details and additional results of E-SFT, including comparisons on the OpenThoughts3-100K dataset and results with alternative LLM-based compression methods. Finally, Appendix D presents an inference-time application of E-SFT.

A Discussion and Limitation

While E-SFT shows strong empirical performance and leads to practical improvements in the generated reasoning process, its effectiveness could vary across settings. In this section, we discuss when E-SFT works well and when it may be less effective, with a particular focus on model scale and the amount of trajectory removal.

Model scales and E-SFT. Figure A plots the relationship between general performance (MMLU-Pro and MMLU-redux) without any additional training and reasoning performance gain from applying E-SFT instead of standard SFT. Each point corresponds to a different base model, with results shown for both s1K-1.1 and OpenThoughts3-100K. The detailed experimental settings are provided in Appendix C.5. We find that the effectiveness of E-SFT could depend on the original capability of the base model. For s1K-1.1, improvements appear only when the base performance exceeds roughly 60–65, whereas the larger ($\times$100) and more diverse OpenThoughts3-100K dataset shifts this threshold lower, allowing benefits to emerge even for weaker base models.

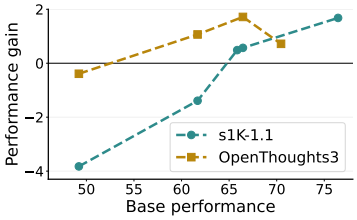


Figure A: **Performance gains** by E-SFT on OpenThoughts3 and s1K-1.1.

This trend aligns with our earlier discussion: E-SFT enhances the model's ability to complete intermediate reasoning steps, but this requires a sufficient baseline capacity. When models are too weak, they lack the ability to reliably infer the trimmed parts, and the method brings little to no benefit. By contrast, once a model has reached a moderate competence level, trimming redundant reasoning forces it to allocate capacity toward informative steps, thereby yielding measurable gains. This pattern closely parallels our analogy to human learners: while novices depend on full guidance, more advanced learners can benefit from streamlined instruction and effectively interpolate the missing reasoning by leveraging prior knowledge.

Performance sensitivity with E-SFT. We next conduct a sensitivity test on the performance impact of the remaining trajectory segments under E-SFT to address a natural question: how the amount of retained trajectory should be tuned, even for such simple truncation. To investigate this effect, we vary the total number of retained steps in the beginning and ending segments when training Qwen2.5-32B-Instruct on s1K-1.1 and Qwen3-4B-Base on OpenThoughts-100K.

As shown in Figure B, we find that performance is not extremely sensitive to the segment size; however, excessive truncation consistently degrades performance (*i.e.,* low retained steps), while overly conservative ones limit the potential benefits (*i.e.,* high retained steps). This suggests that there exists an appropriate operating range for trajectory removal, which can be viewed as a cutoff-length hyperparameter. In practice, without a principled cutoff rule, a simple heuristic of filtering approximately 20% of the original reasoning trajectories works well. We find that this choice provides a reasonable trade-off across datasets, acting like a near-universal regime, and that the same heuristic generalizes across either step count or token count. Additional descriptions on this design choice are provided in §C.3.

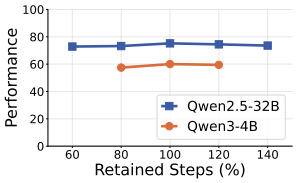
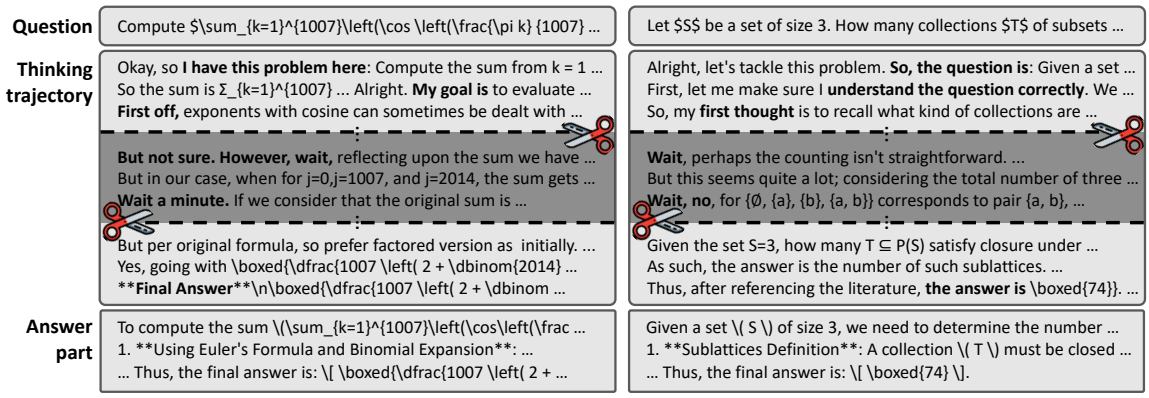
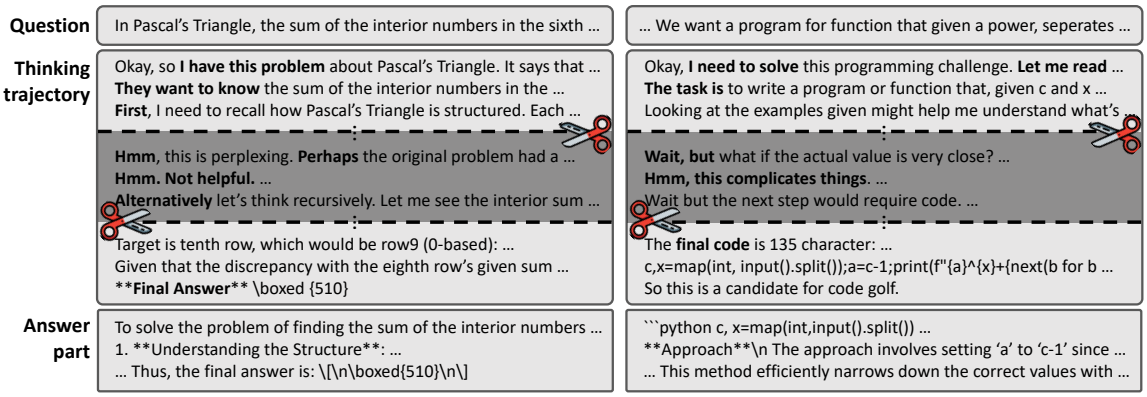


Figure B: Performance vs. different retained step ratios (relative to the default setting).

Attention weight analysis. Figure 2 shows a clear tendency in the attention pattern, suggesting a possible connection to the "Lost in the Middle" effect (Liu et al., 2024b). While there are some differences, there may be a shared connection. First, our analysis shows that LRMs pay less attention to the



(a) s1K-1.1



(b) OpenThoughts3-100K

Figure C: Examples of E-SFT applied to s1K-1.1 and OpenThoughts3-100K datasets.

intermediate parts of reasoning trajectories, leading us to conclude that these segments contain many redundant steps that the model naturally deprioritizes. On the other hand, the "Lost in the Middle" phenomenon demonstrates that LLMs struggle to retrieve crucial information when it appears in the middle of a long context, indicating that even important content in this region receives insufficient attention.

Despite these differences, both observations point to a shared underlying pattern: LLMs generally assign insufficient attention to the middle tokens of long contexts. Specifically, in (Liu et al., 2024b), the authors explain that the "Lost in the Middle" effect is linked to the structure of SFT datasets for instruction tuning, in which task specifications and instructions are often placed at the beginning. Reasoning trajectories exhibit a similar structure: the problem setup and conclusions naturally appear at the beginning and end, while the middle often consists of decomposed but redundant reasoning steps. This distribution of information may cause LRMs to develop weaker attention to the middle, which in turn allows redundant middle content to proliferate during generation. Thus, while our finding differs in mechanism, redundancy-driven attention drop rather than attention-driven failure, the two phenomena may be linked through the underlying structure of the training data.

B Analyses of Reasoning Trajectory Segments

This appendix provides additional analyses on the reasoning trajectories themselves, complementing §3.2 and §4.1. We first describe the detailed settings of the segment ablation analysis in §4.1. We then examine two questions that underlie our design: whether "\n\n" is a reasonable delimiter for reasoning steps, and what kind of content the trimmed middle region actually contains.

B.1 Details of Segment Ablation Analysis

We ablate specific trajectory segments and generate answers under these conditions to verify again whether reasoning trajectories contain redundancy with respect to answer generation. By observing how answer quality changes under such interventions, we can assess whether particular segments play a causal role in generating answers. Our anal-

Region	Plan.	Ded.	Verif.	Backtr.	Reform.	Concl.	No agr.
s1K-1.1							
Early (retained)	33	44	2	5	14	0	1
Middle (trimmed)	11	54	9	14	7	2	3
Late (retained)	6	31	12	13	5	30	4
OpenThoughts3							
Early (retained)	49	23	2	9	13	0	4
Middle (trimmed)	26	39	7	13	8	1	5
Late (retained)	10	29	13	8	7	28	5

Table A: **Functional composition of reasoning steps** in the retained and trimmed regions (%), labeled by three LLM judges under majority vote.

ysis is conducted on 100 problems from the GPQA-D dataset using the `S1.1-32B` (Muennighoff et al., 2024) model[†]. For each problem, a complete reasoning trajectory is first generated. We then remove three different segments: the beginning segment, an intermediate segment centered in the middle, and the ending segment. To systematically examine the effect of segment length, we vary the segment ablation ratio (e.g., 10%, 20%). For instance, with a 10% ratio, the beginning corresponds to tokens 0–10%, the intermediate to 45–55%, and the ending to 90–100% of the trajectory. With a 20% ratio, the corresponding spans become 0–20%, 40–60%, and 80–100%, respectively. The model is subsequently prompted to generate answers from these ablated trajectories as well as from the full trajectory. Finally, the textual similarity between answers from ablated and full trajectories is measured using Jaccard similarity (Manning, 2008) and ROUGE-L (Lin, 2004).

B.2 Do "\n\n" marks correspond to semantic breakpoints?

Throughout the paper, we define reasoning steps by splitting trajectories at double newline characters. To verify that these positions correspond to semantically meaningful boundaries, we sample 20 reasoning trajectories per dataset (s1K-1.1 and OpenThoughts3) and 10 "\n\n" positions per trajectory, yielding 200 positions per dataset. For each sampled position, we compare two cases: whether a semantic transition occurs across the "\n\n", i.e., between the tokens immediately before and after it, and whether one occurs within a single step, i.e., between the tokens immediately after the previous "\n\n" and those immediately before the selected one. Both cases use a length-matched span of about 35 tokens on each side. Each case is labeled as "Continuation" or "Transition" by three LLM judges (Claude-Sonnet-4.6, GPT-5.4, and GLM-5.2) under a majority vote.

Transitions occur across "\n\n" in 33.5% and 28.0% of cases on s1K-1.1 and OpenThoughts3, respectively, compared to only 2.5% and 1.0% within a step. Semantic breakpoints are thus far more likely to coincide with "\n\n" than to appear inside a step, indicating that steps defined this way tend to remain internally consistent. We note that a "\n\n" does not always mark a semantic breakpoint; it serves as a simple and reliable proxy rather than an exact segmentation rule.

B.3 What is in the trimmed region?

To characterize the content that `E-SFT` removes, we classify the function of each step into six categories: planning, deduction, verification, backtracking, reformulation, and conclusion. For each dataset, we sample 30 trajectories with more than 300 steps and label 12 steps per region: the first 12 steps (early, retained), 12 randomly sampled steps from the trimmed region (middle), and the last 12 steps (late, retained). Labels are again assigned by three LLM judges (GPT-5.4, Claude-Sonnet-4.6, and GLM-5.2); a step is assigned a label when at least two judges agree, and is otherwise counted as no agreement.

Table A shows a clear structural pattern. The retained ends carry structurally essential content: planning is concentrated in the early region (33% and 49%) and conclusion in the late region (30% and 28%, versus near zero elsewhere), so removing either end would discard the problem setup or the final answer. In contrast, the trimmed middle region has the highest share of deduction (54% and 39%) and also shows more backtracking than the retained regions (14% and 13%).

Note that this analysis does not determine whether a given step is actually redundant, which is a difficult judgment in itself; our goal here is only to characterize the functional composition of

[†] https://huggingface.co/simplescaling/s1.1-32B

each region. With this caveat, the co-occurrence of frequent deduction and backtracking in the middle suggests that much of this forward reasoning consists of exploratory attempts that are later revised or discarded, and is therefore more likely to be redundant for forming the final answer than the problem setup and conclusion preserved at the two ends.

C Details of E-SFT experiments

C.1 SFT and Evaluation Frameworks

For the **s1K-1.1 experiments**, we conduct supervised fine-tuning using the s1 GitHub repository[‡]. Evaluation relies on the built-in implementation of lm-evaluation-harness[§] within the same repository. The dataset is s1K-1.1[¶].

For the **OpenThought3 experiments**, we follow the instructions of the OpenThought GitHub repository[‖] and train models using Llama-Factory[**]. Evaluation is performed with lm-evaluation-harness[†]. The dataset is based on OpenThoughts3-1.2M[††], from which we randomly sample 100K examples containing both <think> and </think> tokens. Training Qwen2.5-32B-Instruct on the 100K subset of OpenThoughts3-1.2M for 1 epoch took approximately 20 hours on 16 NVIDIA H100 GPUs.

All models, datasets, and codebases used in this research are publicly available open-source artifacts released under permissive licenses and are used solely for academic research.

C.2 LLM-based compression method

We also experiment with compressing the s1K-1.1 dataset using two external LLMs: Claude Sonnet (`claude-sonnet-4-20250514` version of Claude Sonnet 4) (Anthropic, 2025) and Gemini (`gemini-2.5-flash`) (Comanici et al., 2025).

To ensure that only the middle portion is shortened while respecting its local context, we operate on the reasoning trajectory with a block-wise, context-aware paraphrasing pipeline. Each trajectory is split into "blocks" by "\n\n", and we iterate over non-overlapping windows of 10 consecutive blocks. For every window, we construct a prompt that exposes the three preceding and three following blocks as `[Context Before]` and `[Context After]`, while designating the 10-block window as `[Center Section]`. The LLM is instructed to compress the `[Center Section]` only, preserving the logical structure and final conclusion. We instruct the LLMs with a fixed paraphrasing prompt, which explicitly conditions on surrounding context while compressing only the center section. The full prompt is shown in the box below.

Paraphrasing prompt

You will be given a step-by-step reasoning process written by a large language model.

Please paraphrase the **center section only**, while preserving the logical structure.
Try to reduce its length to approximately half of the original, but make sure to keep all essential reasoning steps and the final conclusion.
You may refer to the surrounding context for understanding, but do not modify them.

`[Context Before]`
{before}

`[Center Section]`
{center}

`[Context After]`
{after}

Paraphrased (center section only):

C.3 Implementation details and experimental settings

In E-SFT, the middle segment of each reasoning trajectory is removed while preserving both the beginning and ending segments. Concretely, we preserve the first and last n steps and remove the middle (total steps $- 2n$) steps. We set n=100 for s1K-1.1 and n=200 for OpenThoughts3. We first chose n=100 to remove approximately 20% of the total tokens on s1K-1.1, and found that the preserved first and last steps naturally matched its average trajectory length (234 steps). Based on the

[‡] https://github.com/simplescaling/s1
[§] https://github.com/EleutherAI/lm-evaluation-harness
[¶] https://huggingface.co/datasets/simplescaling/s1K-1.1_tokenized
[‖] https://github.com/open-thoughts/open-thoughts
[**] https://github.com/hiyouga/LLaMA-Factory
[††] https://huggingface.co/datasets/open-thoughts/OpenThoughts3-1.2M

Method	AIME24		GPQA-D		MATH		Average	
	Value	Δ	Value	Δ	Value	Δ	Value	Δ
Qwen2.5-32B-Instruct & s1K-1.1								
Standard SFT	64.44	–	61.95	–	94.13	–	73.51	–
Prefix	62.22	-2.22	63.80	+1.85	95.00	+0.87	73.67	+0.16
Suffix	60.00	-4.44	61.28	-0.67	94.20	+0.07	71.83	-1.68
Random	56.67	-7.77	61.62	-0.33	94.47	+0.34	70.92	-2.59
Sim-based	64.44	0.00	62.79	+0.84	94.53	+0.40	73.92	+0.41
LLM-based (Claude)	33.33	-31.11	57.91	-4.04	89.00	-5.13	60.08	-13.43
LLM-based (Gemini)	17.78	-46.66	48.15	-13.80	82.73	-11.40	49.55	-23.96
Perplexity-extreme	60.00	-4.44	63.13	+1.18	**95.27**	+1.14	72.80	-0.71
Perplexity-high	64.44	0.00	**64.65**	+2.70	94.20	+0.07	74.43	+0.92
LS-Mixture SFT (Mix)	60.00	-4.44	61.10	-0.85	94.60	+0.47	71.90	-1.61
Ours	**68.89**	+4.45	62.29	+0.34	94.40	+0.27	**75.19**	+1.68
Qwen3-8B-Base & s1K-1.1								
Standard SFT	42.22	–	57.41	–	**92.10**	–	63.91	–
Prefix	36.67	-5.55	56.40	-1.01	89.90	-2.20	60.99	-2.92
Suffix	27.78	-14.44	53.87	-3.54	89.30	-2.80	56.98	-6.93
Random	28.33	-13.89	57.41	0.00	88.85	-3.25	58.20	-5.71
Sim-based	34.44	-7.78	55.39	-2.02	90.00	-2.10	59.94	-3.97
LLM-based (Claude)	15.56	-26.66	47.47	-9.94	82.33	-9.77	48.45	-15.46
LLM-based (Gemini)	11.11	-31.11	42.93	-14.48	78.13	-13.97	44.06	-19.85
Ours	**44.44**	+2.22	**57.41**	0.00	91.60	-0.50	**64.48**	+0.57
Qwen3-8B-Base & OpenThoughts3-100K								
Standard SFT	50.00	–	53.87	–	93.47	–	65.78	–
Prefix	40.00	-10.00	53.20	-0.67	93.27	-0.20	62.16	-3.62
Suffix	41.11	-8.89	51.35	-2.52	87.13	-6.34	59.86	-5.92
Random	46.67	-3.33	53.12	-0.75	92.54	-0.93	64.11	-1.67
Ours	**52.22**	+2.22	**56.40**	+2.53	**93.87**	+0.40	**67.50**	+1.72
Qwen3-4B-Base & OpenThoughts3-100K								
Standard SFT	36.67	–	48.65	–	**91.60**	–	58.97	–
Prefix	38.89	+2.22	46.46	-2.19	89.13	-2.47	58.16	-0.81
Suffix	28.89	-7.78	44.95	-3.70	84.20	-7.40	52.68	-6.29
Random	35.00	-1.67	47.48	-1.17	89.87	-1.73	57.45	-1.52
Ours	**40.00**	+3.33	**48.99**	+0.34	91.13	-0.47	**60.04**	+1.07

Table B: **Detailed performance of `E-SFT` on three benchmarks (AIME24, GPQA-D, MATH).** Bold indicates the best performance in each benchmark, and underline indicates performance better than baseline.

same principle, we set n=200 for OpenThoughts3 (average 461 steps), which likewise corresponds to removing about 20% of the tokens. Figure C shows examples applied `E-SFT` on s1K-1.1 and OpenThoughts3 datasets. For evaluation, we use AIME24, GPQA-D, and MATH benchmarks, generating with temperature 0.6. Each instance is evaluated three times, and we report averaged results.

C.4 Detailed results of `E-SFT` and Llama family

Table B presents detailed results by task, model, and method, complementing Table 3-6. Overall, the results confirm that `E-SFT` consistently achieves strong performance.

Table C shows the results on the Llama family using OpenThoughts3-100K for SFT. We fine-tune the models for 1 epoch on the OpenThoughts3-100K dataset, following the same setting used for the Qwen-series models. `E-SFT` maintains performance on Llama-3.2-3B-Instruct and provides a modest improvement on Llama-3.1-8B-Instruct (+1.34% on average). These results suggest that `E-SFT` generalizes beyond the Qwen-based models and remains effective across different model families without introducing performance degradation.

C.5 Experimental details of Figure A

To examine under which conditions `E-SFT` is most effective, we analyze the correlation between the general task performance of the pretrained models and the performance gain of `E-SFT` over full-trajectory SFT. For this experiment, we train `Qwen2.5-32B-Instruct`, `Qwen2.5-7B-Instruct`, `Qwen3-8B-Base`, and `Qwen3-4B-Base` on s1K-1.1, and `Qwen3-8B-Base`

Method	AIME24		GPQA-D		MATH		Average	
	Value	Δ	Value	Δ	Value	Δ	Value	Δ
Llama-3.2-3B-Instruct & OpenThoughts3-100K								
Base	7.67	–	29.65	–	63.20	–	33.50	–
Ours (Step-level)	6.67	-1.00	33.84	+4.19	60.00	-3.20	33.50	0.00
Llama-3.1-8B-Instruct & OpenThoughts3-100K								
Base	17.67	–	37.54	–	74.40	–	43.20	–
Ours (Step-level)	20.00	+2.33	38.22	+0.68	75.40	+1.00	44.54	+1.34

Table C: **Performance of `E-SFT` on the Llama-family.**

and `Qwen3-4B-Base` on OpenThoughts3-100K. The results are then compared to investigate how model capacity and initialization relate to the observed performance gains.

C.6 Experimental details of quantitative comparison using external LLM

We use an external LLM, `GPT-OSS-120B`, to assess the quality of model reasoning. Using the prompt shown below, we compare pairs of responses to questions from s1K-1.1. For each comparison, the judge evaluates the same pair of trajectories twice, once in each order, and is required to select one trajectory in each evaluation. If the preference changes when the order is swapped, we treat the comparison as equivalent.

Comparison prompt

Given the following math problem and two reasoning approaches, determine which one reads like it was written more naturally by someone genuinely working through the problem.

Problem:
{question}

Document A:
{Reasoning trajectory a}

Document B:
{Reasoning trajectory b}

Final Answer (same for both):
{answer}

Which document has less unnecessary repetition and flows more smoothly? Please respond with either "Document A" or "Document B" and briefly explain your reasoning.

C.7 Experimental details of §4.4

Both GRPO and OPD are trained for 100 steps with a learning rate of 1e-6. For GRPO, we use a batch size of 32 with 8 rollouts per prompt, a mini-batch size of 8, a maximum context length of 8k tokens, an entropy coefficient of 0, and a low-variance KL loss with coefficient 0.001, applied as a separate loss term rather than through the reward. For OPD, we use an effective batch size of 8 (per-device batch size 1 with 8 gradient accumulation steps), one generation per prompt, a sampling temperature of 1.0, β=0, a generation loss weight of 0.1, and a maximum context length of 16k tokens.

Model	Full	25%	33%	50%	75%
Qwen3-8B-Base	65.78	65.74	65.79	65.81	65.76
Qwen2.5-32B-Instruct	74.20	74.21	74.20	74.20	74.16

Table D: **Performance of `E-SFT` on decoding.** Values denote the averaged accuracy across AIME24, GPQA-D, and MATH.

D Decoding with `E-SFT` for efficient inference

Inspired by `E-SFT`, we additionally explore a simple inference-time application that reduces computational overhead by truncating the middle segment of the reasoning trajectory as a straightforward extension.

Method. Specifically, the model first generates the entire reasoning trajectory up to the point where it would normally begin producing the final answer. Before the answer part is generated, we truncate the intermediate chunk of the reasoning trajectory according to a specified cutting ratio, while preserving both the beginning and ending segments. We consider four ratios, removing 25, 33, 50, 75% of the middle part. This truncation reduces the computation required for answer generation and yields practical speed improvements in end-to-end evaluation.

Formally, given a trajectory of length L tokens, we remove $\lfloor r \cdot L \rfloor$ tokens centered at position $L/2$, thereby preserving both the first $(1-r)/2$ and last $(1-r)/2$ fractions of the trajectory. For example, with r=0.5, the central 50% of tokens (i.e., from 25% to 75% of the trajectory) is removed. This design allows the model to generate the final answer after observing both the initial problem setup and the concluding consolidation region, while discarding potentially redundant intermediate reasoning.

The efficiency benefit naturally follows from this decoding behavior. For long-form reasoning tasks, truncating the middle part shortens the effective context seen by answer tokens and downstream modules. For example, in GPQA Diamond settings, models typically generate around 8.1k tokens of intermediate reasoning. In a naive two-stage pipeline, this entire trace must be re-prefilled during the second pass. In contrast, applying a 33% truncation reduces the reasoning length to roughly 5.4k tokens, which theoretically lowers second-pass prefill computation by about 46% and reduces attention computation during answer generation (for around 0.5k tokens) to 68%, while maintaining comparable answer quality. Consistent with this analysis, we observe a 4.5% reduction in end-to-end evaluation time on GPQA Diamond using `Qwen2.5-32B-Instruct` fine-tuned on s1K-1.1, despite the use of highly optimized LLM-generation libraries such as vLLM (Kwon et al., 2023) and FlashAttention (Dao et al., 2022; Dao, 2024).

Experimental results. The results are obtained using `Qwen3-8B-Base` trained on OpenThought3-100K, with additional experiments conducted using `Qwen2.5-32B-Instruct` trained on s1K-1.1. Table D reports the inference-time trimming results on these SFT-trained models. Trimming 25–75% of the intermediate steps yields nearly identical average performance compared to using full reasoning trajectories, indicating that inference-time trimming can reduce computational cost for long reasoning traces without harming task accuracy.

E The use of LLMs

LLMs were primarily used for minor language editing, including adjustments to word choices and clarity. They played no role in the research design, analysis, interpretation, or manuscript preparation, and all scientific contributions are fully our own.