

# Distilling the Essence: Efficient Reasoning Distillation via Sequence Truncation

Wei-Rui Chen<sup>1,2,\*</sup>, Vignesh Kothapalli<sup>2</sup>, Ata Fatahibaarzi<sup>2</sup>, Hejian Sang<sup>2</sup>,  
Shao Tang<sup>2</sup>, Qingquan Song<sup>2</sup>, Zhipeng Wang<sup>2\*</sup>, Muhammad Abdul-Mageed<sup>1,3,\*</sup>

<sup>1</sup>The University of British Columbia, <sup>2</sup>LinkedIn,

<sup>3</sup>Canada Research Chair in NLP and ML

weirui.chen@ubc.ca, zhipwang@linkedin.com, muhammad.mageed@ubc.ca

## Abstract

Distilling the capabilities from a large reasoning model (LRM) to a smaller student model often involves training on substantial amounts of reasoning data. However, knowledge distillation (KD) over lengthy sequences with prompt (P), chain-of-thought (CoT), and answer (A) sections makes the process computationally expensive. In this work, we investigate how the allocation of supervision across different sections (P, CoT, A) affects student performance. Our analysis shows that selective KD over only the CoT tokens can be effective when the prompt and answer information is encompassed by it. Building on this insight, we establish a truncation protocol to quantify computation-quality tradeoffs as a function of sequence length. We observe that beyond a specific length, longer training sequences provide marginal returns for downstream performance but require substantially higher memory and FLOPs. To this end, training on only the first 50% of tokens of every training sequence can retain, on average,  $\approx 91\%$  of full-sequence performance on math benchmarks while reducing training time, memory usage, and FLOPs by about 50% each. Codes are available at <https://github.com/weiruichen01/distilling-the-essence>.

## 1 Introduction

Distilling the capabilities of LRMs to smaller counterparts has paved a path for cost-effective, practical deployments in recent years (Hinton et al., 2015; Hsieh et al., 2023; Mukherjee et al., 2023; Mitra et al., 2023; Li et al., 2023; Guo et al., 2025). However, datasets curated for this purpose typically comprise long *chain-of-thought* (CoT) traces (Wei et al., 2023; Kojima et al., 2023) with thousands of tokens and impose significant computational overheads during training. To mitigate such issues, we ask if it is possible to extract meaningful distillation signals from truncated training sequences.

Given an input prompt (P), the CoT trace from a capable LRM reflects the multi-step approach

to arrive at an answer (A). Essentially, CoT explicitly showcases the model’s ability to break down a complex task into sub-problems and its self-reflective phases of corrections. Since smaller models are typically incapable of generating such CoT traces (Wei et al., 2022), distilling this ability to them can aid in solving reasoning-intensive tasks in resource-constrained settings (Chernyshev et al., 2025; Zhang et al., 2025; OpenAI, 2024; Snell et al., 2024; Guo et al., 2025).

In the long-CoT regime, modern LRMs adopt templates that explicitly separate the prompt, a long CoT segment, and the answer (Guo et al., 2025; Guha et al., 2025; Labs, 2025). Although a growing body of work advocates for the effectiveness of longer CoT traces for reasoning tasks (Chen et al., 2025a), their utility tends to be task dependent and longer chains do not always translate into higher accuracy (Chen et al., 2025b; Sui et al., 2025; Wu et al., 2025; Liu and Wang, 2025; Wu et al., 2025). This raises a fundamental yet under-explored question for distillation: ‘which segment (prompt, CoT, answer) is the most effective for supervision, and what fraction of the sequence is sufficient to transfer reasoning abilities efficiently?’

In essence, our work addresses the two related research questions:

- **RQ1:** Under controlled experimental conditions, which sections of an example, *prompt* (P), *CoT*, and *answer* (A), are actually responsible for transferring reasoning capability to the student?
- **RQ2:** Can training with *early and partial* segments match or approach the performance of training with *full* sequences, thereby reducing training cost without sacrificing accuracy?

For RQ1, we show that inclusion of the CoT is the dominant factor for effective distillation. In particular, masking (1) the answer segment (P+CoT),

(2) the prompt segment (CoT+A), (3) or both (CoT-only) during loss calculation results in minimal performance drop over loss calculation with the full sequence (P+CoT+A). However, masking the CoT segment (in both A-only and P+A settings) results in drastic performance degradation of the student model.

For RQ2, token-budget scaling reveals that beyond a certain length, longer training sequences result in performance plateaus on the downstream math tasks but demand compute resources to scale drastically. In particular, using only the first half of tokens retains  $\approx 91\%$  of full-sequence performance on average. In contrast, allocating the same budget to the second half of the sequence results in significantly lower accuracy, suggesting that the most critical reasoning signals are concentrated in the early tokens. Collectively, our results position sequence truncation as a new efficiency axis for reasoning distillation, offering a practical path to reduce training time and memory usage. To the best of our knowledge, we are the first to systematically study the supervision of sequence segments and training-time “overthinking” from the perspective of sequence truncation.

## 2 Related Work

### 2.1 Long Chain-of-Thought

A key development in improving Large Language Model (LLM) reasoning is CoT prompting, which encourages models to generate explicit intermediate reasoning steps before giving an answer (Wei et al., 2023). Such prompting substantially boosts performance on reasoning-intensive tasks in mathematics, science, and code generation (Kojima et al., 2023). Early CoT work typically produced short, highly compressed rationales. The o1 family of models (OpenAI, 2024) demonstrated that allocating substantially more inference-time compute can yield large gains on difficult benchmarks, and subsequent test-time scaling studies (Snell et al., 2024) systematically verified such benefits. Following Ye et al. (2025), we refer to such concise chains as *short CoT* and to o1-style, slow-thinking approaches as *long CoT*.

*Short CoT* typically concatenates the reasoning trace with the answer, using a (P, CoT + A) template in which both rationale and answer appear in a single segment with no explicit token-level boundary (Wei et al., 2023; Kojima et al., 2023; Wang et al., 2023). More recent works in-

stead split the response into distinct CoT and answer segments, and formulate a (P, CoT, A) template (Guo et al., 2025; Min et al., 2024). For example, DeepSeek-R1 introduces an explicit template in the token stream: the model is trained to place its long CoT inside `<think> ... </think>` and the answer afterward (Guo et al., 2025). STILL-2 (Min et al., 2024), OpenThoughts (Guha et al., 2025), and Bespoke-Stratos (Labs, 2025) adopt similar templates that split and place CoT in `<begin_of_thought> ... <end_of_thought>`. Such separation allows long reasoning steps to be included in the CoT segment, whereas a concise final solution is placed in the answer (A) segment. Beyond the common practice of utilizing the full sequence (P, CoT, A) for distilling reasoning capabilities to smaller models (Guo et al., 2025), our work takes a systematic approach to isolating the utility of each of these segments.

### 2.2 Distilling Reasoning Signals

In the era of LRMs, training and serving full-sized models is extremely compute-intensive. To mitigate such bottlenecks, Knowledge Distillation (KD) (Hinton et al., 2015) offers an efficient way to transfer the knowledge of a powerful teacher LRM to a smaller student model (Hsieh et al., 2023; Mukherjee et al., 2023; Mitra et al., 2023). In the *short CoT* regime, supervision from the teacher model is oftentimes emphasized over the response and conditioned on the prompt, as in Orca (Mukherjee et al., 2023), Orca2 (Mitra et al., 2023), Lion (Jiang et al., 2023), and MiniLLM (Gu et al., 2024). With some exceptions, prompts are typically conditioned on but are not included in the loss computation.

In the *long CoT* setting, there are two main aspects to consider. The choice of segments for supervision and the choice of reasoning datasets for distillation. For the first aspect, supervision (i.e., loss calculation) can be placed solely on the CoT segment (Shen et al., 2025), on both CoT and answer segments (Li et al., 2025b), or on the entire sequence (prompt, CoT, and answer triplet) (Xu et al., 2025b,a). For the second aspect of choosing datasets, we note that several works have challenged the idea that “longer CoT implies better performance” and introduced the notion of reasoning with *overthinking* (Chen et al., 2025b; Sui et al., 2025; Liu and Wang, 2025; Marjanović et al., 2025). Utilizing sequences that contain such overthinking signals for distillation can be detrimental

to model performance (Wu et al., 2025). Prior research on LLM reasoning has also noted that not all parts of a CoT are equally valuable. Marjanović et al. (2025) observe that “DeepSeek-R1 often undergoes multiple cycles of self-verification, even when it has already arrived at the correct answer.” This observation suggests that later tokens may not carry significantly useful reasoning signals. These observations motivate our controlled investigation into whether distillation of early and partial sequences can match or approach the efficacy of full-length sequences. In particular, we focus on computation-quality tradeoffs with increasingly long sequences.

### 3 Experiments

#### 3.1 Formulation

We employ supervised knowledge distillation (Hinton et al., 2015; Sanh et al., 2020)<sup>1</sup> to transfer the capabilities of a large reasoning teacher model, denoted as  $\mathcal{T}$ , to a smaller student model that did not undergo any post-training,  $\mathcal{S}$ . The training objective for the student model is a combination of a cross-entropy loss against the ground-truth labels and a distillation loss that encourages the student to align with the teacher’s output distribution. This composite loss function,  $\mathcal{L}$ , for a given input sequence  $x$  from the training corpus, is given by:

$$\mathcal{L} = \lambda \mathcal{L}_{\text{soft}} + (1 - \lambda) \mathcal{L}_{\text{hard}}$$

where  $\mathcal{L}_{\text{hard}}$  is the hard label loss,  $\mathcal{L}_{\text{soft}}$  is the soft distillation loss, and  $\lambda \in [0, 1]$  is a hyperparameter that balances the two terms. In our experiments, we set  $\lambda = 0.5$ , same as in Gu et al. (2024), ensuring a balanced contribution from both the soft distillation loss and the hard label loss.<sup>2</sup> This approach allows the student to learn not only from the correct labels but also from the nuanced knowledge encapsulated in the teacher’s predictions.

The hard loss,  $\mathcal{L}_{\text{hard}}$ , is the conventional negative log-likelihood of the ground-truth sequence of  $T$  tokens  $y = (y_1, \dots, y_T)$ . It is computed as:

$$\mathcal{L}_{\text{hard}} = - \sum_{t=1}^T \log P_{\mathcal{S}}(y_t | y_{<t}, x)$$

where  $P_{\mathcal{S}}(y_t | y_{<t}, x)$  is the probability assigned by the student model to the true token  $y_t$  at timestep

<sup>1</sup>For method choice, see Appendix A.4 and A.5.3 for details.

<sup>2</sup>An ablation of different  $\lambda$ s is included in Appendix A.4

$t$ . The soft loss,  $\mathcal{L}_{\text{soft}}$ , uses the forward Kullback-Leibler (KL) divergence to measure the discrepancy between the student’s and teacher’s output probability distributions. It is defined as the sum of KL divergences over all timesteps:

$$\mathcal{L}_{\text{soft}} = \sum_{t=1}^T D_{\text{KL}}(\sigma(z_{\mathcal{T}}^{(t)}) || \sigma(z_{\mathcal{S}}^{(t)}))$$

where  $z_{\mathcal{T}}^{(t)}$  and  $z_{\mathcal{S}}^{(t)}$  are the logit vectors produced by the teacher and student models at timestep  $t$ , respectively, and  $\sigma(\cdot)$  is the softmax function.

#### 3.2 Experimental Setup

**Models.** We select the Qwen3 family (Team, 2025b) because its post-trained models are natively trained with two dedicated tokens, `<think>` and `</think>`, that explicitly delimit the CoT. In contrast to model families that permit reasoning without explicit start/end markers, such as Gemma3 (Team et al., 2025) and Mistral (Lin, 2025), Qwen3 offers a controlled testbed for comparing masking and truncation strategies. We employ two post-trained teachers (Qwen3-32B and Qwen3-8B) and four base student models (Qwen3-8B, 4B, 1.7B, and 0.6B-Base), which are out-of-the-box pre-trained checkpoints without any post-training. To reflect practical distillation scenarios, we consider the seven teacher-student pairs as shown in Table 1.

Teacher	Student	Teacher	Student
Qwen3-32B	Qwen3-8B-Base	Qwen3-8B	Qwen3-4B-Base
Qwen3-32B	Qwen3-4B-Base	Qwen3-8B	Qwen3-1.7B-Base
Qwen3-32B	Qwen3-1.7B-Base	Qwen3-8B	Qwen3-0.6B-Base
Qwen3-32B	Qwen3-0.6B-Base		

Table 1: Seven teacher–student pairs used in our distillation experiments.

**Datasets.** We distill from five reasoning corpora: OpenThoughts-114k<sup>3</sup> (Openthoughts) (Guha et al., 2025), Bespoke-Stratos-17k<sup>4</sup> (Bespoke) (Labs, 2025), Synthetic-1<sup>5</sup> (Mattern et al., 2025), Llama-Nemotron-Post-Training-Dataset<sup>6</sup> (Nemotron) (Bercovich et al., 2025) and SkyT1-17k<sup>7</sup> (SkyT1) (Team, 2025a). The first four datasets were generated by DeepSeek-R1, whereas the last was produced by QwQ-32B-Preview (Team,

<sup>3</sup><https://huggingface.co/datasets/open-thoughts/OpenThoughts-114k>

<sup>4</sup><https://huggingface.co/datasets/bespokelabs/Bespoke-Stratos-17k>

<sup>5</sup><https://huggingface.co/datasets/PrimeIntellect/SYNTHETIC-1>

<sup>6</sup><https://huggingface.co/datasets/nvidia/Llama-Nemotron-Post-Trainin-Dataset>

<sup>7</sup>[https://huggingface.co/datasets/NovaSky-AI/Sky-T1\\_data\\_17k](https://huggingface.co/datasets/NovaSky-AI/Sky-T1_data_17k)

2024). We use OpenThoughts and Bespoke across all experiments for RQ1 and RQ2. To assess whether the RQ2 results generalize beyond these datasets, we further include Synthetic-1, Nemotron, and SkyT1 exclusively in the RQ2 analysis. Each sequence/example is a linearized dialogue tree: Prompts (P) are user messages; CoT traces and Answers (A) are assistant messages. System messages are removed. Annotation tags `<begin_of_thought>` and `<end_of_thought>` are replaced with `<think>` and `</think>` to match the Qwen3 vocabulary. `<think>` and `</think>` are used to delimit (prompt, CoT, Answer). Token count statistics of each section for the five datasets are listed in Table 2. Full examples of each dataset and their token distribution of each of the three sections (P, CoT, A) are included in Appendix A.1.

Sec	Openthoughts		Bespoke		Synthetic-1		Nemotron		SkyT1	
	Mean	%	Mean	%	Mean	%	Mean	%	Mean	%
Full	2497.5	-	2165.3	-	1854.0	-	2437.8	-	2277.1	-
P	116.6	4.7	120.5	5.6	100.2	5.4	87.3	3.6	139.1	6.1
CoT	1910.4	76.6	1657.3	76.6	1402.8	75.7	2073.3	85.1	1632.1	71.7
A	468.5	18.8	385.5	17.8	349.1	18.8	275.2	11.3	503.9	22.1

Table 2: Token count statistics for five training datasets. “%” is the average proportion of tokens in each section relative to the full sequence. Distributions are included in Appendix A.1.2.

To reduce memory usage and ensure that full-length training sequences include all sections (P, CoT, A), we retain only examples with tokenized length below 4k. After filtering, we then sample 40k, 8k, 16k, 16k, and 8k instances from Openthoughts, Bespoke, Synthetic-1, Nemotron, and SkyT1, respectively, for training, and reserve 1k from each corpus for validation.

**Training.** Across all settings, we train the student models with knowledge distillation, using Fmchisel framework<sup>8</sup> (Behdin et al., 2025b,a), for two epochs with AdamW optimizer (learning rate  $4 \times 10^{-6}$ , weight decay 0.05). We use a batch size of 1, with gradient accumulation, across eight Nvidia H100 (80GB) GPUs so the effective batch size is 8. See Appendix A.5.2 for detailed compute resource usage. The checkpoint with the lowest validation loss is selected for evaluation.

**Evaluation.** We focus our evaluation on mathematical reasoning and consider two widely adopted benchmarks: AIME24<sup>9</sup> (MAA, 2024) and AIME25<sup>10</sup> (MAA, 2025). To form the baselines,

<sup>8</sup><https://github.com/linkedin/fmchisel>

<sup>9</sup>[https://huggingface.co/datasets/huggingfaceH4/aime\\_2024](https://huggingface.co/datasets/huggingfaceH4/aime_2024)

<sup>10</sup>[https://huggingface.co/datasets/yentinglin/aime\\_2025](https://huggingface.co/datasets/yentinglin/aime_2025)

we compare student model performance before and after distillation. During evaluation, we set the maximum generation length to 32,768 new tokens, with decoding hyperparameters temperature 0.6 and top- $p = 0.95$ , aligning with the recommended values in Qwen3 Huggingface page<sup>11</sup>. We report results with the largest sampling budget (64 samples) for both benchmarks using lighteval<sup>12</sup>.

### 3.3 Methods

**Section-wise Supervision (RQ1).** The supervision comprises three sections: P, CoT, and A and we vary which sections are involved in loss calculation. The six conditions are A, P+A, CoT, CoT+A, P+CoT, and P+CoT+A (full-length with all sections included). Following the training setup of previous work (Mukherjee et al., 2023; Mitra et al., 2023; Jiang et al., 2023; Gu et al., 2024; Shi et al., 2024; Shen et al., 2025; Li et al., 2025b), we apply a loss mask to all tokens outside the supervised region before computing the objective, while still feeding the entire P+CoT+A sequence through the model in the forward pass. If a section is not included, tokens within that section are conditioned but not involved in loss computation.

**Early vs. Late Tokens (RQ2).** To investigate whether useful reasoning signals are concentrated in specific parts of the CoT, we design two complementary experiments targeting compute efficiency.

*Sequence-Length Scaling Behavior.* First, we explore how performance scales with the number of tokens trained. Unlike loss masking, which still incurs forward-pass costs, we examine the performance when tokens are removed entirely. We introduce a dynamic truncation protocol, *Lead-Span Proportion (LSP)*. For each example with tokenized inputs  $\mathbf{x}$  and labels  $\mathbf{y}$ , LSP retains the first  $p \in \{0.1, 0.2, \dots, 1.0\}$  fraction of tokens from the concatenation of  $\mathbf{x}$  and  $\mathbf{y}$ , and removes the rightmost  $(1-p)$  fraction before training. The truncated tokens, therefore, never enter the model: they are neither seen in the forward pass nor included in the loss. This protocol preserves example-specific variability in kept length, avoids exclusion of answer tokens that can arise under a global max-length cutoff, and retains diversity of sequence lengths.

*Budget Location Ablation.* Second, to determine where the token budget should be spent, we fix the training budget at 50% of the tokens (a choice guided by performance saturation, compute effi-

<sup>11</sup><https://huggingface.co/Qwen/Qwen3-8B>

<sup>12</sup><https://github.com/huggingface/lighteval>

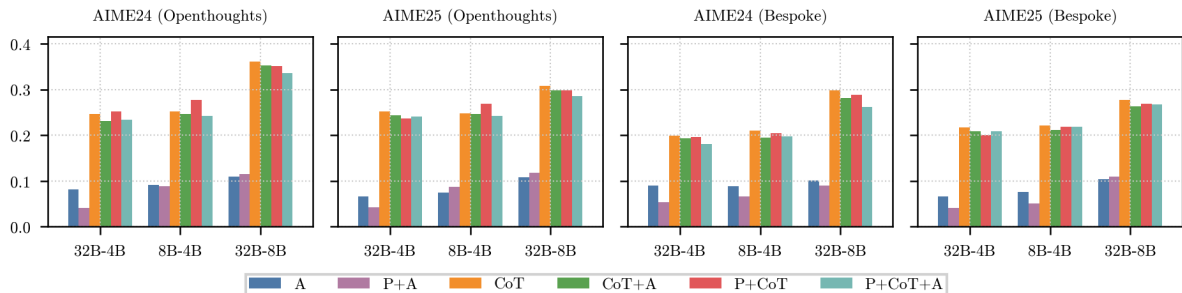


Figure 1: Accuracy on AIME24 and AIME25 under different supervision regions for two different training datasets. Supervising CoT tokens is crucial, with setups that ignore CoT scoring lowest and only modest differences among CoT-inclusive variants.

ciency, and experimental symmetry, as detailed in Section 4.2). We compare two conditions: *Left 50%*, which keeps the first half of tokens and drops the rest, and *Right 50%*, which keeps the final half and drops the beginning. This ablation keeps the compute budget fixed while shifting the kept segment of tokens, enabling a clean test of whether early tokens carry richer learning signal than later ones. Better performance in the *Left 50%* setting would be consistent with the hypothesis that useful signal concentrates early in the trajectory.

## 4 Results and Analysis

### 4.1 Section-wise Supervision (RQ1)

**Sub-optimality of full sequence distillation.** As can be seen in Figure 1, a consistent pattern emerges across benchmarks and teacher-student pairs: settings that supervise with CoT consistently outperform those that do not. On both benchmarks, answer-only supervision (A) yields the lowest accuracy, and extending the loss to the prompt (P+A) brings only modest gains. Once CoT tokens receive supervision, whether alone (CoT), paired with the answer (CoT+A), paired with the prompt (P+CoT), or combined with both (P+CoT+A), performance improves substantially, with only minor differences among these CoT-inclusive variants.

Overall, the results indicate that supervising the rationale tokens is crucial, while spreading loss to the prompt and answer offers only modest, sometimes negligible, additional benefits. Thus, full-token supervision does not reliably outperform more selective objectives that focus on the reasoning process itself.

**CoT implicitly encompasses the prompt and answer sections.** To understand the limited gains from supervising P and A, we manually examined

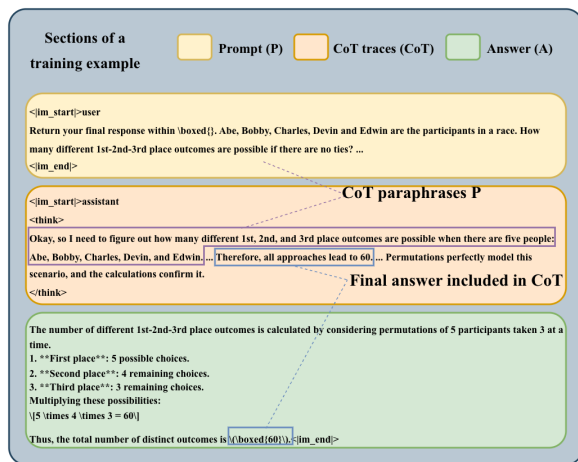


Figure 2: A training example. The text is continuous but divided into three sections to illustrate the idea of Prompt (P), CoT traces (CoT), and Answer (A). We observe that the information in P and A is often covered within CoT. Some intermediate text in CoT has been omitted here for readability. The full example is provided as Figure 7 in Appendix A.1.

the data and observed that the CoT often encapsulates information from both the prompt and the answer as illustrated in Figure 2. Typically, the CoT begins by paraphrasing the question to contextualize the problem, then proceeds with reflection-based reasoning steps before deriving the final answer. The answer section typically summarizes the CoT and reiterates the final answer near the end.

**Linguistic analysis.** To verify whether P and A are commonly covered by the CoT in our training datasets, we conduct a linguistic analysis on 100 randomly sampled training examples using GPT-5.1 as a judge. We prompt the model to verify that: (1) all substantive information in P and A is already present in the CoT; and (2) P and A do not introduce any new substantive statements absent from

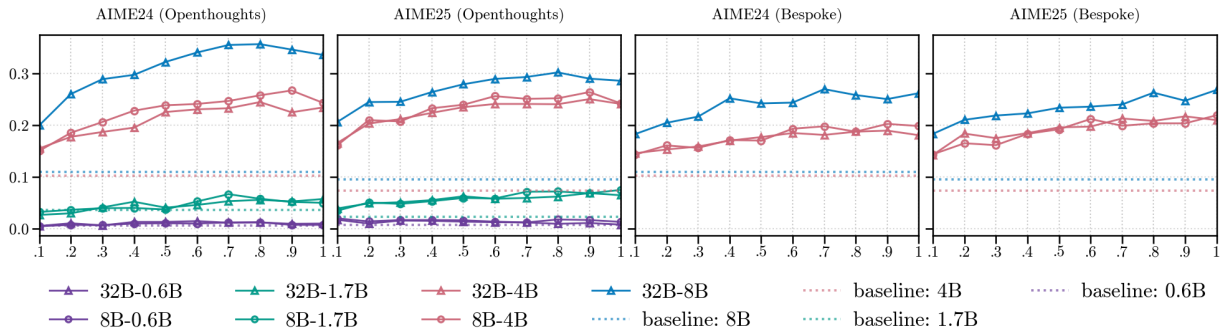


Figure 3: Sequence-length scaling behavior of accuracy against different Lead-Span Proportions (LSP) shows that performance is not linear. The vast majority of performance can be retained when training with only a portion of tokens for each training example. Since student models smaller than 4B performed near baseline on OpenThoughts, we restricted our experiments on Bespoke to student models larger than or equal to 4B.

the CoT. This ensures that, semantically, the CoT entails both P and A. We also checked whether the final answer in A is equivalent to the final answer derived in CoT.

For OpenThoughts, we found that in 99% and 89% of the 100 sampled examples, the CoT semantically entails P and A, respectively. For Bespoke, these figures are 97% and 93%. The few exceptions where CoT does not entail A are due to new information introduced in A. They primarily involve open-ended science or coding questions where the answer is not easily stored in a latex box<sup>13</sup>. However, in 100% and 99% of examples in OpenThoughts and Bespoke respectively, the final answer in A is equivalent to the final answer in the CoT section. That indicates that despite these examples having new information in answer section, their final answer still aligns with that in CoT section. This explains why adding supervision on P and A provides minimal benefit over CoT alone. For details of the design and results of linguistic analysis, refer to Appendix A.2.

## 4.2 Early vs. Late Tokens (RQ2)

### Marginal benefits of longer reasoning sequences.

Figure 3 shows that across all distillation settings, LSP exhibits similar scaling behavior of the student model performance. They show nearly monotonic accuracy improvement as LSP increases and then saturates. The flattening near the right edge suggests those trimmed tokens are not required for effective distillation, implying substantial compute savings with minimal performance drop.

**Role of model size.** The 32B-8B setting is consistently the top performer, and the ranking is largely

determined by student size:  $8B > 4B > 1.7B > 0.6B$ . For a fixed student size, curves from different teachers track closely. For the 4B student model, it performs comparably and sometimes better when distilling from 8B rather than 32B model, which echoes the findings of Li et al. (2025c) that smaller models do not necessarily learn better from much larger teachers. For small students (0.6B and 1.7B), training with LSP does not reliably beat the baselines. Their curves hew close to the dotted baselines across  $p$ , suggesting limited capacity to absorb additional supervision. The benefits of LSP mainly materialize once the student has enough scale (4B and 8B). Therefore, in later experiments, these two small students will not be involved.

Pairs	Openthoughts		Bespoke	
	AIME24	AIME25	AIME24	AIME25
32B-4B	0.5	0.5	0.6	0.2
8B-4B	0.4	0.2	0.2	0.2
32B-8B	0.6	0.5	0.4	0.5

Table 3: Knees for different experimental setups to indicate the most cost-effective LSP.

**Deciding the most cost-effective budget.** To anchor the token budget allocation, we select  $p = 0.5$  as our target. This decision is driven by three factors: performance saturation, computational efficiency, and experimental rigor. First, we identified the most cost-effective point (“knee”) in the accuracy-vs-LSP curves with Satopaa et al. (2011)<sup>14</sup>, as shown in Table 3. We observe that across all settings, the knee consistently occurs at  $p \leq 0.6$ , with the majority at or below  $p = 0.5$ . Second, as detailed in Section 4.3,  $p = 0.5$  serves

<sup>13</sup>Example: What is the effect of different types of amino acid residues in the binding site of protein on the stability of DNA-protein complex in molecular dynamics simulations?

<sup>14</sup><https://github.com/arykevi/kneed>

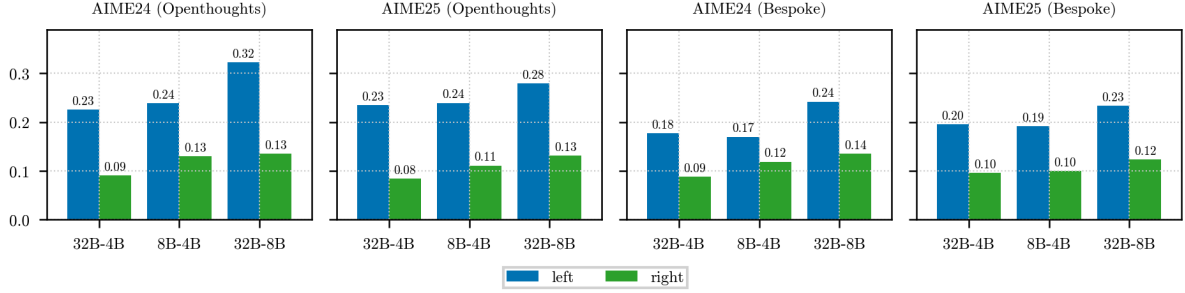


Figure 4: Impact of 50% budget allocation strategies (left, right) on model accuracy. Retaining the left portion of the sequence consistently outperforms the right, indicating that early tokens are more valuable for learning than the later tokens. This ablation of training budget verifies that the efficiency of using 50% tokens is not a general trait when reducing the number of training tokens.

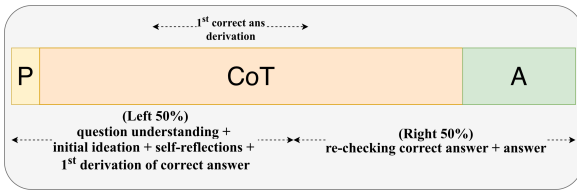


Figure 5: First half of tokens tends to contain denser useful information than the second half, in line with stronger performance when training on the left 50% of tokens. Complete examples, each for one of the five datasets, are included in Appendix A.1.1.

as a critical boundary before computational costs begin to rise drastically. Third, selecting  $p = 0.5$  enables a clean, symmetrical ablation between the Left 50% and Right 50% of the sequence. A budget of  $p > 0.5$  would necessitate overlapping tokens. As seen in Table 4, across all 30 settings (3 teacher-student pairs, 5 datasets, and 2 benchmarks), training with the first half of tokens retains on average  $\approx 91\%$  of the accuracy when compared to training with all tokens, with retention being as high as  $\approx 99\%$  for AIME25 with 8B-4B setup.

Pairs	OpenThoughts		Bespoke		Synthetic-1		Nemotron		SkyT1	
	24	25	24	25	24	25	24	25	24	25
32B-4B	96.21	97.40	97.98	93.28	79.28	91.77	96.56	90.71	87.41	89.65
8B-4B	98.07	98.92	85.79	87.62	82.54	92.12	91.50	91.62	75.23	89.01
32B-8B	95.96	97.63	92.43	87.16	95.23	97.06	92.21	96.63	80.12	82.66

Table 4: The performance retention rate (%) training with the first half of the tokens of each example in five datasets, compared to training with full-length examples. Performance is evaluated on AIME24 (“24”) and AIME25 (“25”). See Appendix A.5.1 for the raw values used to compute the retention rate.

**Why Early Tokens Suffice?** The fact that the first 50% of tokens yields similar performance compared to full-length raises a mechanistic question:

does the early segment already contain the key reasoning signal? To probe this, we analyze where the *first derivation of the correct answer* appears inside the CoT, treating it as a proxy for the earliest point at which the reasoning becomes sufficient. For each of 100 randomly sampled training examples, we prompt GPT-5.1 to identify the first derivation of the correct answer, and then compute its relative position within the sequence.

Across datasets, the first correct derivation tends to occur near the midpoint of the whole sequence; see Appendix A.2 for details. We illustrate this phenomenon in Figure 5. This near-midpoint tendency aligns with the accuracy–vs–LSP curves exhibiting knees at  $p \leq 0.6$  (Table 3) and with the strong performance retention at  $p = 0.5$  (Table 4). Together, these results suggest that an early budget captures most of the learnable signal for distillation.

Qualitative analysis of the content preceding the first derivation shows that this segment contains question understanding, initial ideation, several rounds of self-reflections, and substantial progress toward the final answer. In many examples, the remaining tokens are then largely dominated by verification, minor algebraic clean-up, and re-statements of an already-correct answer, which appear less informative for distillation than the earlier reasoning steps.

We operationalize self-reflection, a key contributor to the capabilities of reasoning models (Muenighoff et al., 2025; Li et al., 2025a), by counting occurrences of lexical cues (e.g., “wait”, and related variants; see full list in Appendix A.2.1) that appear before the first correct derivation point. On average, we observe 2.2 (OpenThoughts), 2.4 (Bespoke), 2.8 (Synthetic-1), 5.1 (Nemotron), and 1.9 (SkyT1) self-reflection cues before the first correct

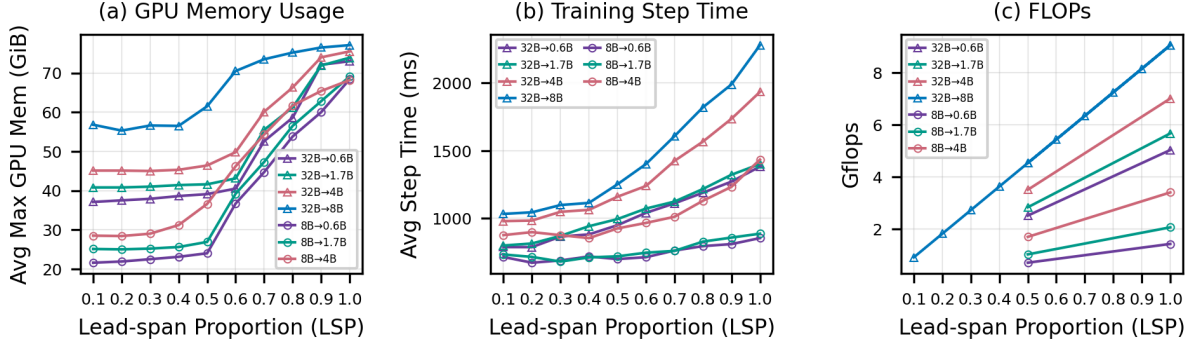


Figure 6: GPU memory usage, time spent for each training step, and FLOPs as functions of LSP. Triangles indicate a 32B teacher, circles indicate an 8B teacher, and matching colors indicate the same student model. Due to the linear behavior of flops, not all but only selected LSPs (0.5, 1.0) for some teacher-student pairs are profiled.

derivation.

**Ablation Studies** Having established that the left 50% often encapsulates the core reasoning process, we use the Left vs. Right ablation to verify that the efficiency gains are specific to this self-contained initial segment, rather than a general property of reducing the token count. We compare training with the Left 50% against the Right 50%. As shown in Figure 4, training with the left 50% of the sequence consistently and significantly outperforms training with the right 50%. This is further supported by our loss analysis (Appendix A.3), which reveals that the loss on the left 50% is comparable to the full sequence, whereas the loss on the right 50% is relatively higher, indicating that the initial segment is more amenable to learning.

Furthermore, we investigate the sensitivity of our method to the distillation weight  $\lambda$ . Our experimental results show that the approach is robust, as performance remains consistently strong across  $\lambda \in \{0.25, 0.5, 0.75\}$ . This consistency demonstrates that the effectiveness stems from the proposed method itself, rather than relying on the specific  $\lambda = 0.5$  selection. For detailed ablation results, see Appendix A.4.

### 4.3 Compute Efficiency

Computationally, our 50% training protocol yields substantial resource savings, as illustrated in Figure 6. We observe a critical inflection point at LSP = 0.5 for both GPU memory usage<sup>15</sup> and training step time, beyond which costs escalate rapidly due to the quadratic complexity of attention mechanisms.

**Memory Efficiency.** Peak GPU memory usage

<sup>15</sup>GPU memory is measured with the NVIDIA Data Center GPU Manager (DCGM) metric DCGM\_FI\_DEV\_FB\_USED to reflect practical device-level VRAM usage.

remains relatively modest for  $LSP \leq 0.5$  but spikes sharply thereafter. For instance, in the 8B→4B setting, processing the full sequence ( $p = 1.0$ ) consumes up to 68.2 GiB, necessitating an 80GB H100 GPU. By truncating to 50%, peak memory drops to 36.6 GiB, fitting comfortably on a standard 40GB A100 GPU. This reduction lowers the hardware barrier significantly, enabling distillation on more accessible infrastructure.

**Training Speed.** Training step time follows a similar trajectory. While larger students generally require more time, we also observe that teacher size imposes a distinct overhead; for the same 4B student, distilling from a 32B teacher is consistently slower than from an 8B teacher. By capping the sequence at 50%, we achieve a roughly  $2\times$  reduction in both training step time and FLOPs, confirming that  $p = 0.5$  is a practical sweet spot for resource-constrained distillation.

## 5 Conclusion

In this work, we demonstrate that the reasoning trace itself is the dominant carrier of supervisory signal in knowledge distillation. By systematically varying the supervised sections of training examples, we found that models trained on the reasoning trace alone achieve performance comparable to full-sequence supervision. Moreover, our investigation into token budget allocation reveals that critical reasoning behaviors are concentrated in early tokens; truncating examples to just their first 50% of tokens retains  $\approx 91\%$  of downstream accuracy on math tasks while reducing training time and memory usage by  $\approx 50\%$ . This establishes sequence truncation as a powerful efficiency lever, positioning partial-sequence training as a cost-effective strategy for scaling reasoning distillation.

## Limitations

Our study focuses on mathematical reasoning tasks. While mathematics provides a rigorous testbed for reasoning with verifiable ground truth, extending these findings to other reasoning domains (e.g., commonsense reasoning, and code generation) remains an avenue for future work. Our conclusions are drawn from highly structured datasets, and may not directly generalize to less structured or interactive reasoning settings.

We conduct experiments within the Qwen3 model family, using Qwen3-32B and Qwen3-8B as teachers and Qwen3-8B/4B/1.7B/0.6B as students. This controlled setup isolates the effects of sequence truncation from confounding factors introduced by architectural heterogeneity across model families. Investigating cross-family distillation (e.g., Qwen to Gemma) represents a natural extension.

In our LSP comparisons, we train on the same set of examples across all settings, and only change how much of each example is kept. This means the total number of training tokens is not matched across LSP values: for instance, LSP = 0.5 uses about half as many training tokens as LSP = 1.0. Therefore, our study does not test an equal-token-budget scenario (e.g., using shorter sequences but proportionally more examples). Evaluating this fixed-budget tradeoff is an important direction for future work.

## Ethical Considerations

This work investigates efficient training protocols for knowledge distillation. All datasets and models used are publicly available and were created by prior work for research purposes.

Our findings on sequence truncation have potential positive implications for democratizing access to reasoning model training, as they substantially reduce computational requirements. By demonstrating that comparable performance can be achieved with approximately half the training compute and memory, this work may lower barriers for researchers and practitioners with limited computational resources.

We acknowledge that improvements in reasoning model distillation could, in principle, be applied to both beneficial and harmful use cases. However, the efficiency techniques we propose do not fundamentally alter model capabilities. They reduce training costs for existing distillation paradigms.

We believe the benefits of making reasoning distillation more accessible outweigh potential risks.

## Acknowledgments

Muhammad Abdul-Mageed acknowledges support from Canada Research Chairs (CRC), the Natural Sciences and Engineering Research Council of Canada (NSERC; RGPIN-2018-04267), the Social Sciences and Humanities Research Council of Canada (SSHRC; 895-2020-1004; 895-2021-1008), Canadian Foundation for Innovation (CFI; 37771), Digital Research Alliance of Canada,<sup>16</sup> and UBC Advanced Research Computing-Sockeye.<sup>17</sup>

## References

- 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 *The Twelfth International Conference on Learning Representations*.
- David D. Baek and Max Tegmark. 2025. [Towards understanding distilled reasoning models: A representational approach](#). *Preprint*, arXiv:2503.03730.
- Kayhan Behdin, Ata Fatahibaarzi, Qingquan Song, Yun Dai, Aman Gupta, Zhipeng Wang, Hejian Sang, Shao Tang, Gregory Dexter, Sirou Zhu, and 1 others. 2025a. [Scaling down, serving fast: Compressing and deploying efficient llms for recommendation systems](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 1687–1702.
- Kayhan Behdin, Ata Fatahibaarzi, Dai Yun, Qingquan Song, Vignesh Kothapalli, Shao Tang, Hejian Sang, Aman Gupta, Zhipeng Wang, Gregory Dexter, Sirou Zhu, and Siyu Zhu. 2025b. [fmchisel](#).
- Akhiad Bercovich, Itay Levy, Izik Golan, Mohammad Dabbah, Ran El-Yaniv, Omri Puny, Ido Galil, Zach Moshe, Tomer Ronen, Najeeb Nabwani, Ido Shahaf, Oren Tropp, Ehud Karpas, Ran Zilberstein, Jiaqi Zeng, Soumye Singhal, Alexander Bukharin, Yian Zhang, Tugrul Konuk, and 114 others. 2025. [Llama-nemotron: Efficient reasoning models](#). *Preprint*, arXiv:2505.00949.
- Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wanxiang Che. 2025a. [Towards reasoning era: A survey of long chain-of-thought for reasoning large language models](#). *arXiv preprint arXiv:2503.09567*.

<sup>16</sup><https://alliancecan.ca>

<sup>17</sup><https://arc.ubc.ca/ubc-arc-sockeye>

- Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. 2025b. [Do not think that much for 2+3=? on the overthinking of o1-like llms](#). *Preprint*, arXiv:2412.21187.
- Konstantin Chernyshev, Vitaliy Polshkov, Vlad Stepanov, Alex Myasnikov, Ekaterina Artemova, Alexei Miasnikov, and Sergei Tilga. 2025. [U-MATH: A university-level benchmark for evaluating mathematical skills in large language models](#). In *Proceedings of the Fourth Workshop on Generation, Evaluation and Metrics (GEM<sup>2</sup>)*, pages 974–1001, Vienna, Austria and virtual meeting. Association for Computational Linguistics.
- Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. 2024. [Minillm: Knowledge distillation of large language models](#). In *ICLR*.
- Etash Guha, Ryan Marten, Sedrick Keh, Negin Raoof, Georgios Smyrnis, Hritik Bansal, Marianna Nezhurina, Jean Mercat, Trung Vu, Zayne Sprague, Ashima Suvarna, Benjamin Feuer, Liangyu Chen, Zaid Khan, Eric Frankel, Sachin Grover, Caroline Choi, Niklas Muennighoff, Shiye Su, and 31 others. 2025. [Openthoughts: Data recipes for reasoning models](#). *Preprint*, arXiv:2506.04178.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shiron Ma, Xiao Bi, and 1 others. 2025. [Deepseek-r1 incentivizes reasoning in llms through reinforcement learning](#). *Nature*, 645(8081):633–638.
- Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. [Distilling the knowledge in a neural network](#). *arXiv preprint arXiv:1503.02531*.
- Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alexander Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. 2023. [Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes](#). In *Findings of ACL*.
- Yuxin Jiang, Chunkit Chan, Mingyang Chen, and Wei Wang. 2023. [Lion: Adversarial distillation of proprietary large language models](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3134–3154, Singapore. Association for Computational Linguistics.
- Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. [Scaling laws for neural language models](#). *Preprint*, arXiv:2001.08361.
- Jongwoo Ko, Sungnyun Kim, Tianyi Chen, and Se-Young Yun. 2024. [Distillm: Towards streamlined distillation for large language models](#). *Preprint*, arXiv:2402.03898.
- Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2023. [Large language models are zero-shot reasoners](#). *Preprint*, arXiv:2205.11916.
- Bespoke Labs. 2025. [Bespoke-stratos: The unreasonable effectiveness of reasoning distillation](#). <https://www.bespokelabs.ai/blog/bespoke-stratos-the-unreasonable-effectiveness-of-reasoning-distillation>. Accessed: 2025-01-22.
- Dacheng Li, Shiyi Cao, Tyler Griggs, Shu Liu, Xiangxi Mo, Eric Tang, Sumanth Hegde, Kourosh Hakhmaneshi, Shishir G. Patil, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. 2025a. [Llms can easily learn to reason from demonstrations structure, not content, is what matters!](#) *Preprint*, arXiv:2502.07374.
- Liunian Harold Li, Jack Hessel, Youngjae Yu, Xiang Ren, Kai-Wei Chang, and Yejin Choi. 2023. [Symbolic chain-of-thought distillation: Small models can also “think” step-by-step](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2665–2679, Toronto, Canada. Association for Computational Linguistics.
- Yang Li, Youssef Emad, Karthik Padthe, Jack Lanchantin, Weizhe Yuan, Thao Nguyen, Jason Weston, Shang-Wen Li, Dong Wang, Ilya Kulikov, and Xian Li. 2025b. [Naturalthoughts: Selecting and distilling reasoning traces for general reasoning tasks](#). *Preprint*, arXiv:2507.01921.
- Yuetai Li, Xiang Yue, Zhangchen Xu, Fengqing Jiang, Luyao Niu, Bill Yuchen Lin, Bhaskar Ramasubramanian, and Radha Poovendran. 2025c. [Small models struggle to learn from strong reasoners](#). In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 25366–25394, Vienna, Austria. Association for Computational Linguistics.
- Yenting Lin. 2025. [Mistral-small-24b-instruct-2501-reasoning](#). *Hugging Face*.
- Xin Liu and Lu Wang. 2025. [Answer convergence as a signal for early stopping in reasoning](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 17896–17907, Suzhou, China. Association for Computational Linguistics.
- Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. 2025. [Understanding r1-zero-like training: A critical perspective](#). *Preprint*, arXiv:2503.20783.
- MAA. 2024. American invitational mathematics examination – AIME 2024. <https://maa.org/maa-invitational-competitions/>. Accessed: 24 November 2025.
- MAA. 2025. American invitational mathematics examination – AIME 2025. <https://maa.org/maa-invitational-competitions/>. Accessed: 24 November 2025.

- Sara Vera Marjanović, Arkil Patel, Vaibhav Adlakha, Milad Aghajohari, Parishad BehnamGhader, Mehar Bhatia, Aditi Khandelwal, Austin Kraft, Benno Krojer, Xing Han Lù, Nicholas Meade, Dongchan Shin, Amirhossein Kazemnejad, Gaurav Kamath, Marius Mosbach, Karolina Stańczak, and Siva Reddy. 2025. *Deepseek-r1 thoughtology: Let’s think about llm reasoning*. *Preprint*, arXiv:2504.07128.
- Justus Mattern, Sami Jaghouar, Manveer Basra, Jan-nik Straube, Matthew Di Ferrante, Felix Gabriel, Jack Min Ong, Vincent Weisser, and Johannes Hagemann. 2025. *Synthetic-1: Two million collaboratively generated reasoning traces from deepseek-r1*.
- Yingqian Min, Zhipeng Chen, Jinhao Jiang, Jie Chen, Jia Deng, Yiwen Hu, Yiru Tang, Jiapeng Wang, Xiaoxue Cheng, Huatong Song, Wayne Xin Zhao, Zheng Liu, Zhongyuan Wang, and Ji-Rong Wen. 2024. *Imitate, explore, and self-improve: A reproduction report on slow-thinking reasoning systems*. *Preprint*, arXiv:2412.09413.
- Arindam Mitra, Luciano Del Corro, Shweti Mahajan, Andres Codas, Clarisse Simoes, Sahaj Agarwal, Xuxi Chen, Anastasia Razdaibiedina, Erik Jones, Kriti Agarwal, Hamid Palangi, Guoqing Zheng, Corby Rosset, Hamed Khanpour, and Ahmed Awadallah. 2023. *Orca 2: Teaching small language models how to reason*. *Preprint*, arXiv:2311.11045.
- Niklas Muennighoff and 1 others. 2025. *s1: Simple test-time scaling*. *arXiv preprint arXiv:2501.19393*.
- Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. 2023. *Orca: Progressive learning from complex explanation traces of gpt-4*. *Preprint*, arXiv:2306.02707.
- OpenAI. 2024. *Learning to reason with llms*.
- Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. *Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter*. *Preprint*, arXiv:1910.01108.
- Ville Satopaa, Jeannie Albrecht, David Irwin, and Barath Raghavan. 2011. Finding a "kneede" in a haystack: Detecting knee points in system behavior. In *2011 31st international conference on distributed computing systems workshops*, pages 166–171. IEEE.
- Zhanming Shen, Zeyu Qin, Zenan Huang, Hao Chen, Jiaqi Hu, Yihong Zhuang, Guoshan Lu, Gang Chen, and Junbo Zhao. 2025. *Merge-of-thought distillation*. *Preprint*, arXiv:2509.08814.
- Zhengyan Shi, Adam X. Yang, Bin Wu, Laurence Aitchison, Emine Yilmaz, and Aldo Lipani. 2024. *Instruction tuning with loss over instructions*. *Preprint*, arXiv:2405.14394.
- Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. *Scaling llm test-time compute optimally can be more effective than scaling model parameters*. *Preprint*, arXiv:2408.03314.
- Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Na Zou, Hanjie Chen, and Xia Hu. 2025. *Stop overthinking: A survey on efficient reasoning for large language models*. *Preprint*, arXiv:2503.16419.
- Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, and 197 others. 2025. *Gemma 3 technical report*. *Preprint*, arXiv:2503.19786.
- NovaSky Team. 2025a. *Sky-t1: Train your own o1 preview model within \$450*. <https://novasky-ai.github.io/posts/sky-t1>. Accessed: 2025-01-09.
- Qwen Team. 2024. *Qwq: Reflect deeply on the boundaries of the unknown*.
- Qwen Team. 2025b. *Qwen3 technical report*. *Preprint*, arXiv:2505.09388.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. *Self-consistency improves chain of thought reasoning in language models*. *Preprint*, arXiv:2203.11171.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. *Chain-of-thought prompting elicits reasoning in large language models*. *Preprint*, arXiv:2201.11903.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. *Chain-of-thought prompting elicits reasoning in large language models*. *arXiv preprint arXiv:2201.11903*.
- Ronald J Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280.
- 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*. *Preprint*, arXiv:2502.07266.
- Jillian Xu, Dylan Zhou, Vinay Shukla, Yang Yang, Junrui Ruan, Shuhuai Lin, Wenfei Zou, Yinxiao Liu, and Karthik Lakshmanan. 2025a. *Dual-head reasoning distillation: Improving classifier accuracy with train-time-only reasoning*. *Preprint*, arXiv:2509.21487.

Jingxian Xu, Mengyu Zhou, Weichang Liu, Hanbing Liu, Shi Han, and Dongmei Zhang. 2025b. [TwT: Thinking without tokens by habitual reasoning distillation with multi-teachers' guidance](#). In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 16475–16489, Suzhou, China. Association for Computational Linguistics.

Edward Yeo, Yuxuan Tong, Morry Niu, Graham Neubig, and Xiang Yue. 2025. [Demystifying long chain-of-thought reasoning in llms](#). *Preprint*, arXiv:2502.03373.

Jie Zhang, Cezara Petrucci, Kristina Nikolić, and Florian Tramèr. 2025. [Realmath: A continuous benchmark for evaluating language models on research-level mathematics](#). *Preprint*, arXiv:2505.12575.

# A Appendix

## A.1 Training Dataset Information

### A.1.1 Full Training Examples

A complete training example

Prompt (P)     CoT traces (CoT)     Answer (A)

<jim\_start>user  
Return your final response within `\boxed{\}`. Abe, Bobby, Charles, Devin and Edwin are the participants in a race. How many different 1st-2nd-3rd place outcomes are possible if there are no ties? Two different outcomes to include are Bobby-Devin-Edwin and Devin-Bobby-Edwin.  
<jim\_end>

<jim\_start>assistant  
<think>

Okay, so I need to figure out how many different 1st, 2nd, and 3rd place outcomes are possible when there are five people: Abe, Bobby, Charles, Devin, and Edwin. The problem says there are no ties, and they want to include examples like Bobby-Devin-Edwin and Devin-Bobby-Edwin as different outcomes. Alright, let's start breaking this down.

First, let me make sure I understand the problem correctly. We have five participants in a race, and we need to count the number of possible ways to arrange the top three positions without any ties. Each position is distinct, so the order matters. That means if Bobby comes first and Devin second, that's a different outcome than Devin first and Bobby second. Got it. So, it's a permutation problem where the order of selection matters.

Hmm, permutations versus combinations. Yeah, in combinations, the order doesn't matter, but here it does. So, permutations are the way to go. Let me recall the formula for permutations. The number of permutations of  $n$  things taken  $r$  at a time is given by  $nPr = n! / (n - r)!$ .

In this case,  $n$  is 5 (since there are five participants) and  $r$  is 3 (because we are looking for 1st, 2nd, and 3rd places). So, plugging those numbers into the formula, we get  $5P3 = 5! / (5 - 3)! = 5! / 2!$ .

Calculating that:  $5!$  is  $5 \times 4 \times 3 \times 2 \times 1 = 120$ , and  $2!$  is  $2 \times 1 = 2$ . So 120 divided by 2 is 60. So, there should be 60 different outcomes. Wait, but let me verify this by another method to make sure I didn't jump to a conclusion too quickly.

Alternative approach: For the first place, there are 5 possible participants. Once first place is decided, that person can't be in second or third place. So, for second place, there are 4 remaining participants. Then, third place would have 3 remaining participants. Therefore, the number of different outcomes is  $5 \times 4 \times 3 = 60$ . Yep, same result. So that seems consistent.

Let me think of another way to visualize this. Imagine filling the three positions one by one. For the first position, you can choose any of the five runners. Once you've chosen the first, you have four left for the second, and then three for the third. Multiplying those possibilities together gives the total number of permutations. So 5 options for first, times 4 for second, times 3 for third.  $5 \times 4$  is 20, times 3 is 60. Yeah, that still checks out.

Another check: Let's list smaller numbers to see if the logic holds. Suppose there were 3 people and we want to know how many ways to arrange 1st, 2nd, 3rd. That should be  $3! = 6$ . Using the permutation formula,  $3P3 = 3! / 0! = 6 / 1 = 6$ . Correct. If there are 4 people and we want top 3, then  $4P3 = 4! / 1! = 24$ . Manually,  $4 \times 3 \times 2 = 24$ . Correct. So, applying the same logic,  $5 \times 4 \times 3 = 60$ . That seems right.

Wait, but let me make sure there isn't a restriction I'm missing. The problem states "no ties," so each position must be occupied by a different person. Since we're only selecting three distinct positions, each with a different person, that's exactly what permutations account for. So, no problem there.

Also, the example given in the problem includes two different orderings of Bobby and Devin in the first two positions. That reinforces that order matters. So, swapping two people in different positions counts as a different outcome, which is exactly what permutations do.

Is there a chance that the problem is considering something else, like teams or something? But no, it's a race, and it's individual participants. Each of the top three positions is occupied by one person, no teams involved. So, it's purely a permutation of 5 people taken 3 at a time.

Alternatively, thinking about the problem in terms of slots. First place: 5 choices. Second place: 4 choices (since one person is already in first). Third place: 3 choices (two people already taken). So, total permutations:  $5 \times 4 \times 3 = 60$ .

I think that's solid. Maybe I can try to count a few possibilities manually to verify. Let's say the participants are A, B, C, D, E (Abe, Bobby, Charles, Devin, Edwin). Let's see how many permutations there are for first three.

If first is A, then second can be B, C, D, E. For each of those, third can be the remaining three. So, for first place A:  $4 \times 3 = 12$  possibilities.

Similarly, if first is B, then second can be A, C, D, E, and third the remaining three. Again  $4 \times 3 = 12$ . So each first place choice leads to 12 possibilities. Since there are 5 first place choices, total is  $5 \times 12 = 60$ . Yep, that matches.

Another way: Suppose we list all possible first-second-third triples. For each possible first place (5 options), the number of possible second and third places is the number of permutations of the remaining 4 runners taken 2 at a time. Which is  $4P2 = 4 \times 3 = 12$ . So, again  $5 \times 12 = 60$ .

Therefore, all approaches lead to 60. So, I think that's the correct answer. I don't see any mistakes in my reasoning. The key points are understanding that order matters and that each position is distinct and cannot be occupied by the same person more than once. Permutations perfectly model this scenario, and the calculations confirm it.

</think>

The number of different 1st-2nd-3rd place outcomes is calculated by considering permutations of 5 participants taken 3 at a time.

1. **First place**: 5 possible choices.
2. **Second place**: 4 remaining choices.
3. **Third place**: 3 remaining choices.

Multiplying these possibilities:  
 $\{5 \times 4 \times 3 = 60\}$

Thus, the total number of distinct outcomes is `\boxed{60}`.<jim\_end>

Figure 7: A complete training example from Bespoke with three sections divided for illustration purposes.

A complete training example sampled for linguistic analysis



Prompt (P)



CoT traces (CoT)



Answer (A)

<|im\_start|>user  
Return your final response within \boxed{}. A caterpillar is climbing a 20-meter pole. During the day, it climbs 5 meters, and during the night, it slides down 4 meters. How long will it take for the caterpillar to reach the top of the pole?  
<|im\_end|>

<|im\_start|>assistant  
<think>

Okay, so there's this caterpillar climbing a 20-meter pole. Every day, it climbs up 5 meters, but then at night, it slides down 4 meters. The question is, how long does it take for the caterpillar to reach the top? Hmm, let me try to figure this out step by step.

First, I need to understand the problem properly. The pole is 20 meters high. During the day, the caterpillar ascends 5 meters, but during the night, it slips back 4 meters. So each full day-night cycle, the caterpillar makes a net progress of  $5 - 4 = 1$  meter. Wait, so if it gains 1 meter each day, then to climb 20 meters, it would take 20 days, right? But that seems too straightforward. Maybe there's a catch here.

Let me think again. Let's break it down day by day. On day 1, the caterpillar climbs 5 meters during the day, reaching 5 meters. Then at night, it slides down 4 meters, ending up at 1 meter. So after the first day, it's at 1 meter. Day 2: climbs 5 meters to 6 meters during the day, slides down 4 to 2 meters at night. Day 3: climbs to 7 meters, slides to 3. Day 4: 8 to 4. Day 5: 9 to 5. Day 6: 10 to 6. Day 7: 11 to 7. Day 8: 12 to 8. Day 9: 13 to 9. Day 10: 14 to 10. Day 11: 15 to 11. Day 12: 16 to 12. Day 13: 17 to 13. Day 14: 18 to 14. Day 15: 19 to 15. Day 16: Climbs 5 meters during the day, so  $15 + 5 = 20$  meters. Wait, but once it reaches the top, it doesn't slide down anymore, right? Because it's already out of the pole. So on day 16, it climbs from 15 meters to 20 meters during the day, and doesn't need to slide down. So the total time is 16 days. first derivation of correct answer

Oh! So my initial thought that it's 20 days was wrong because I didn't account for the fact that on the last day, the caterpillar doesn't slide back. So once it reaches or exceeds the 20 meters during the day, it's done. Therefore, the key here is that the last 5 meters climb doesn't require sliding back. So we need to calculate how many days it takes until the caterpillar's daytime climb gets it to the top.

Let me formalize this. The caterpillar effectively makes a net gain of 1 meter per day (5 up during the day, 4 down at night). But on the final day, when it climbs 5 meters, it might reach or surpass the 20 meters, so it doesn't slide back. Therefore, the total distance it needs to cover before the last day is  $20 - 5 = 15$  meters. Because once it's at 15 meters, the next day it can climb 5 meters to reach 20 meters, and that's it.

So the net gain per day is 1 meter. To reach 15 meters, how many days does it take? Since each day it gains 1 meter, after 15 days, it would have reached 15 meters. Wait, but wait. Let me check. Wait, on day 1, it ends at 1 meter. Day 2, 2 meters. Wait, no, no. Each day, the cycle is: climb 5, then slide 4. So each full day-night cycle, it's 1 meter net. So after each full day (including night), it's 1 meter higher. So to get to 15 meters, how many days?

Wait, but let's track it again. Let's start at 0 meters.

Day 1: climb to 5, slide to 1. End of day 1: 1 meter.

Day 2: climb to 6, slide to 2. End of day 2: 2 meters.

Day 3: climb to 7, slide to 3. End of day 3: 3 meters.

...

So each day, after sliding, it's at  $n$  meters after  $n$  days. So after 15 days, it's at 15 meters. Then on day 16, it climbs 5 meters from 15 to 20, and doesn't slide back. So total days: 16.

Wait, but let's confirm this with actual calculation. Let's see:

After day 1: 1m

After day 2: 2m

...

After day 15: 15m

Then on day 16, it climbs from 15m to 20m during the day. Since it's reached the top, it doesn't slide down. So the total time is 16 days.

But wait, another way to think about this is: the caterpillar needs to cover 20 meters, but on the last day, it doesn't slide. So the total distance it needs to cover with sliding is  $20 - 5 = 15$  meters. Since each day it gains 1 meter, it takes 15 days to reach 15 meters. Then on the 16th day, it climbs the remaining 5 meters.

Yes, that's correct. Therefore, the answer is 16 days.

But let me check with another approach to avoid mistakes. Let's model it mathematically.

Let  $D$  be the number of days. During each day, the caterpillar climbs 5 meters, but slides down 4 meters each night except the last day. So the total distance is:

$$5D - 4(D - 1) \geq 20$$

Because on  $D$  days, it climbs 5 meters each day, and slides down 4 meters each night except the last night (since once it reaches the top, it doesn't slide down). Therefore, the number of nights is  $D - 1$ .

So the equation is:

$$5D - 4(D - 1) \geq 20$$

Simplify:

$$5D - 4D + 4 \geq 20$$

$$D + 4 \geq 20$$

$$D \geq 16$$

So  $D$  must be at least 16 days. That confirms the earlier answer.

Therefore, the caterpillar will reach the top on the 16th day.

</think>

The caterpillar climbs 5 meters each day and slides down 4 meters each night. While the net gain is 1 meter per day, the final ascent on the last day avoids sliding. Here's the breakdown:

1. **Effective Daily Progress**:

Daytime climb: +5 meters

Nighttime slide: -4 meters

Net gain per full day-night cycle: **1 meter** (until the final day).

2. **Critical Insight**:

On the day the caterpillar reaches or exceeds 20 meters, it does not slide back. Thus, the final 5-meter climb needs no subsequent night.

3. **Distance Before the Final Day**:

Total pole height = 20 meters

To avoid sliding after the final climb, the caterpillar must be within **15 meters** ( $20 - 5 = 15$ ) before its last ascent.

4. **Days to Reach 15 Meters**:

At 1 meter per day, it takes **15 days** to reach 15 meters (after nighttime slides).

5. **Final Ascent**:

On the **16th day**, the caterpillar climbs from 15 meters to **20 meters** and exits the pole.

**Answer**: The caterpillar reaches the top of the pole on the **16th day**. <|im\_end|>

Figure 8: A complete training example from OpenThoughts sampled for linguistic analysis with three sections divided for illustration purposes. Red boxes indicate self-reflection keywords. Blue underline is the first derivation of correct answer labeled by GPT-5.1.

A complete training example sampled for linguistic analysis

Prompt (P)
 CoT traces (CoT)
 Answer (A)

<|im\_start|>user

If a box of ballpoint pens contains 16 pens and is sold for 24 yuan, using  $y$  to represent the price of the ballpoint pen in yuan and  $x$  to represent the number of ballpoint pens, the relationship between  $y$  and  $x$  is

Return your final response as 'Final Answer:  $\boxed{\langle \text{answer} \rangle}$ ', where  $\langle \text{answer} \rangle$  is the number or mathematical expression of the solution.<|im\_end|>

<|im\_start|>assistant

<think>

Okay, so the problem here is about finding the relationship between the price of ballpoint pens ( $y$ ) and the number of pens ( $x$ ). Let me try to understand what's given.

There's a box with 16 pens sold for 24 yuan. So, if you buy 16 pens, you pay 24 yuan. The question wants me to express  $y$  (the price) in terms of  $x$  (the number of pens). Hmm, so  $y$  is the total price for  $x$  pens, right?

Wait, let me make sure. If  $x$  is the number of pens, then  $y$  should be the total cost for  $x$  pens. So, if 16 pens cost 24 yuan, each pen would cost 24 yuan divided by 16 pens. Let me calculate that. 24 divided by 16... 24/16 simplifies to 3/2 or 1.5. So each pen is 1.5 yuan.

first derivation of correct answer

Therefore, if each pen is 1.5 yuan, then for  $x$  pens, the total price  $y$  would be 1.5 times  $x$ . So the equation would be  $y = 1.5x$ . But maybe they want it in fraction form instead of decimal. 1.5 is the same as 3/2. So,  $y = (3/2)x$ . Let me check the units:  $x$  is number of pens, multiplying by 3/2 yuan per pen gives  $y$  in yuan. That seems right.

Alternatively, maybe they want a direct proportion. If 16 pens cost 24 yuan, then the price per pen is  $24/16 = 3/2$  yuan. So the relationship is  $y = (3/2)x$ . Yeah, that makes sense. So for any number of pens  $x$ , multiplying by 3/2 gives the total cost in yuan. Let me verify with the given values. If  $x$  is 16, then  $y$  should be  $(3/2)*16 = 24$ , which matches. So that's correct.

I think that's the solution. The relationship is  $y$  equals three halves  $x$ , or  $y = (3/2)x$ . Alternatively, written as a fraction, it's the same thing. So the answer should be  $y = (3/2)x$ . Let me check once more. If someone buys 1 pen, it's 1.5 yuan, 2 pens would be 3 yuan, which is 1.5 times 2. Yep, that's consistent. So I think that's the correct relationship.

</think>

The box contains 16 pens and is sold for 24 yuan. To find the relationship between the price  $y$  in yuan and the number of pens  $x$ , we first determine the price per pen.

The price per pen is calculated by dividing the total price by the number of pens:  $\text{Price per pen} = \frac{24 \text{ yuan}}{16 \text{ pens}} = \frac{3}{2} \text{ yuan per pen}$

Thus, the total price  $y$  for  $x$  pens is given by multiplying the price per pen by the number of pens:  $y = \left(\frac{3}{2}\right)x$

Final Answer:  $\boxed{y = \frac{3}{2}x}$ <|im\_end|>

Figure 9: A complete training example from Synthetic-1 sampled for linguistic analysis with three sections divided for illustration purposes. Red boxes indicate self-reflection keywords. Blue underline is the first derivation of correct answer labeled by GPT-5.1.

A complete training example sampled for linguistic analysis



Prompt (P)



CoT traces (CoT)



Answer (A)

<|im\_start|>user

Solve the following math problem. Make sure to put the answer (and only answer) inside `\boxed{}`.  
Solve  $\left[\frac{x+4}{x-2}=3\right]$  for  $x$ .

<|im\_end|>

<|im\_start|>assistant

<think>

Okay, so I need to solve the equation  $(x + 4)/(x - 2) = 3$  for  $x$ . Hmm, let's think about how to approach this. I remember that when you have a fraction equal to a number, you can cross-multiply to get rid of the denominator. Let me try that.

First, I'll write down the equation again:  $(x + 4)/(x - 2) = 3$ . To cross-multiply, I'll multiply both sides by  $(x - 2)$ . That should eliminate the denominator on the left side. So, multiplying both sides by  $(x - 2)$  gives me  $x + 4 = 3*(x - 2)$ . Wait, is that right? Yeah, because if I have  $a/b = c$ , then  $a = c*b$ , right? So that should work.

Now, expanding the right side:  $3*(x - 2)$  is  $3x - 6$ . So now the equation is  $x + 4 = 3x - 6$ . Hmm, okay, now I need to solve for  $x$ . Let me get all the  $x$  terms on one side and constants on the other. Subtract  $x$  from both sides:  $4 = 2x - 6$ . Then, add 6 to both sides:  $4 + 6 = 2x$ , which is  $10 = 2x$ . Then divide both sides by 2:  $x = 5$ . Wait, is that right? Let me check.

first derivation of correct answer

But wait, I should make sure that the solution doesn't make the original denominator zero. The denominator in the original equation is  $x - 2$ . If  $x = 5$ , then  $5 - 2 = 3$ , which is not zero. So that's okay. So  $x = 5$  is a valid solution.

Let me plug  $x = 5$  back into the original equation to verify. Left side:  $(5 + 4)/(5 - 2) = 9/3 = 3$ . Right side is 3. So it checks out. Therefore,  $x = 5$  is the solution.

Wait, did I make any mistakes here? Let's see. Cross-multiplying was the right step. Then expanding  $3*(x - 2)$  to  $3x - 6$ . Correct. Then moving  $x$  to the right side by subtracting  $x$ :  $4 = 2x - 6$ . Then adding 6 to both sides gives  $10 = 2x$ , so  $x = 5$ . All steps seem correct. And checking the solution in the original equation confirms it's correct. So I think that's the right answer. I don't see any other solutions here, and since we ended up with  $x = 5$  which doesn't cause division by zero, that's the only solution.

</think>

`\boxed{5}`  
<|im\_end|>

Figure 10: A complete training example from Nemotron sampled for linguistic analysis with three sections divided for illustration purposes. Red boxes indicate self-reflection keywords. Blue underline is the first derivation of correct answer labeled by GPT-5.1.

A complete training example sampled for linguistic analysis



Prompt (P)



CoT traces (CoT)



Answer (A)

<|im\_start|>user

A right pyramid has a square base with perimeter 24 inches. Its apex is 9 inches from each of the other vertices. What is the height of the pyramid from its peak to the center of its square base, in inches?

<|im\_end|>

<|im\_start|>assistant  
<|think|>

So I have this problem about a right pyramid with a square base. The perimeter of the base is 24 inches, and the apex is 9 inches away from each of the base vertices. I need to find the height of the pyramid from its peak to the center of its square base.

First, I need to understand the structure of the pyramid. It's a right pyramid, which means the apex is directly above the center of the base. The base is a square, and I know its perimeter is 24 inches. Since a square has four equal sides, each side must be 24 divided by 4, which is 6 inches.

So, the base is a square with sides of 6 inches. Now, the apex is 9 inches away from each of the base vertices. That means if I take any vertex of the base, the distance from that vertex to the apex is 9 inches. I need to find the height of the pyramid, which is the vertical distance from the apex to the center of the base.

To visualize this, I can think of the pyramid in three dimensions. The base is a square, and the apex is directly above the center of this square.

Let me denote the base square as ABCD, with center O, and the apex as P.

So, P is directly above O, and I need to find the length PO.

I know that  $PA = PB = PC = PD = 9$  inches, since the apex is 9 inches from each vertex.

First, I need to find the coordinates or at least the distances involved.

Let me place the square base on the xy-plane with center O at (0,0). Since it's a square with side length 6 inches, the coordinates of the vertices can be:

A: (3, 3, 0)

B: (3, -3, 0)

C: (-3, -3, 0)

D: (-3, 3, 0)

Wait, but that would make the side lengths 6 inches, since from (3,3) to (3,-3) is 6 units.

But actually, if sides are 6 inches, the distance from center to a vertex is half of the diagonal.

Let me calculate the diagonal of the base square.

The diagonal of a square with side length  $s$  is  $s\sqrt{2}$ .

So, diagonal =  $6\sqrt{2}$  inches.

Then, the distance from the center O to a vertex is half of the diagonal, which is  $(6\sqrt{2})/2 = 3\sqrt{2}$  inches.

So,  $OA = OB = OC = OD = 3\sqrt{2}$  inches.

Now, in the pyramid,  $PA = PB = PC = PD = 9$  inches.

Since P is directly above O, the coordinates of P would be (0,0,h), where h is the height I need to find.

Now, the distance from P to any vertex, say A (3,3,0), is given by the distance formula:

$PA = \sqrt{(3-0)^2 + (3-0)^2 + (0-h)^2} = \sqrt{9 + 9 + h^2} = \sqrt{18 + h^2} = 9$  inches.

So,  $\sqrt{18 + h^2} = 9$ .

Squaring both sides:  $18 + h^2 = 81$ .

Then,  $h^2 = 81 - 18 = 63$ .

So,  $h = \sqrt{63}$ .

But  $\sqrt{63}$  can be simplified:  $\sqrt{9*7} = 3\sqrt{7}$ .

Therefore, the height of the pyramid is  $3\sqrt{7}$  inches.

first derivation of correct answer

Wait, but I should verify if this makes sense.

Alternatively, I can think about the right triangle formed by PO, OA, and PA.

PO is the height h, OA is  $3\sqrt{2}$ , and PA is 9.

By the Pythagorean theorem:  $PO^2 + OA^2 = PA^2$ .

So,  $h^2 + (3\sqrt{2})^2 = 9^2$ .

That is  $h^2 + 18 = 81$ .

Thus,  $h^2 = 81 - 18 = 63$ .

So,  $h = \sqrt{63} = 3\sqrt{7}$  inches.

Seems consistent.

Alternatively, maybe there's another way to approach this.

Let me consider the slant height.

Wait, but I don't have information about the slant height directly.

Alternatively, perhaps using the properties of similar triangles or something else.

But the Pythagorean theorem seems straightforward and sufficient here.

I think  $3\sqrt{7}$  inches is the correct answer.

<|think|>

To find the height of a right pyramid with a square base, where the perimeter of the base is 24 inches and the apex is 9 inches away from each vertex, we can follow these steps:

1. **Determine the side length of the base:** - The perimeter of the base is 24 inches. Since the base is a square, each side length  $s$  is:  $s = \frac{24}{4} = 6$  inches
2. **Find the distance from the center of the base to a vertex:** - The diagonal of the square base is given by  $s\sqrt{2}$ :  $\text{Diagonal} = 6\sqrt{2}$  inches - The distance from the center O to a vertex (e.g., A) is half of the diagonal:  $OA = \frac{6\sqrt{2}}{2} = 3\sqrt{2}$  inches
3. **Apply the Pythagorean theorem:** - Let P be the apex of the pyramid, and we need to find the height  $h = PO$ . We know  $PA = 9$  inches (the distance from the apex to any vertex). - Using the Pythagorean theorem:  $PA^2 = PO^2 + OA^2$   $9^2 = h^2 + (3\sqrt{2})^2$   $81 = h^2 + 18$  - Thus, solving for  $h$ :  $h^2 = 81 - 18 = 63$   $h = \sqrt{63} = 3\sqrt{7}$  inches

Therefore, the height of the pyramid is  $3\sqrt{7}$  inches. </end\_of\_solution|>

Figure 11: A complete training example from SkyT1 sampled for linguistic analysis with three sections divided for illustration purposes. Red boxes indicate self-reflection keywords. Blue underline is the first derivation of correct answer labeled by GPT-5.1.

## A.1.2 Training Dataset Token Count Distribution

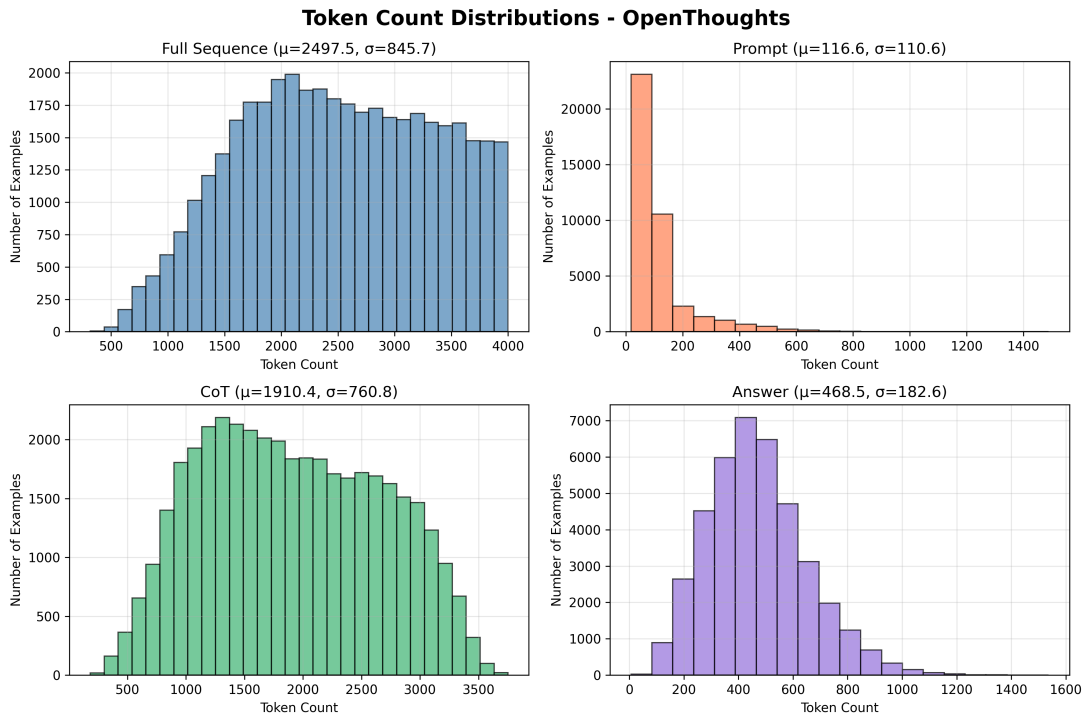


Figure 12: Token count distributions for prompts, chain-of-thought, answers, and full sequences in the sampled 40k training examples from OpenThoughts-114k dataset.

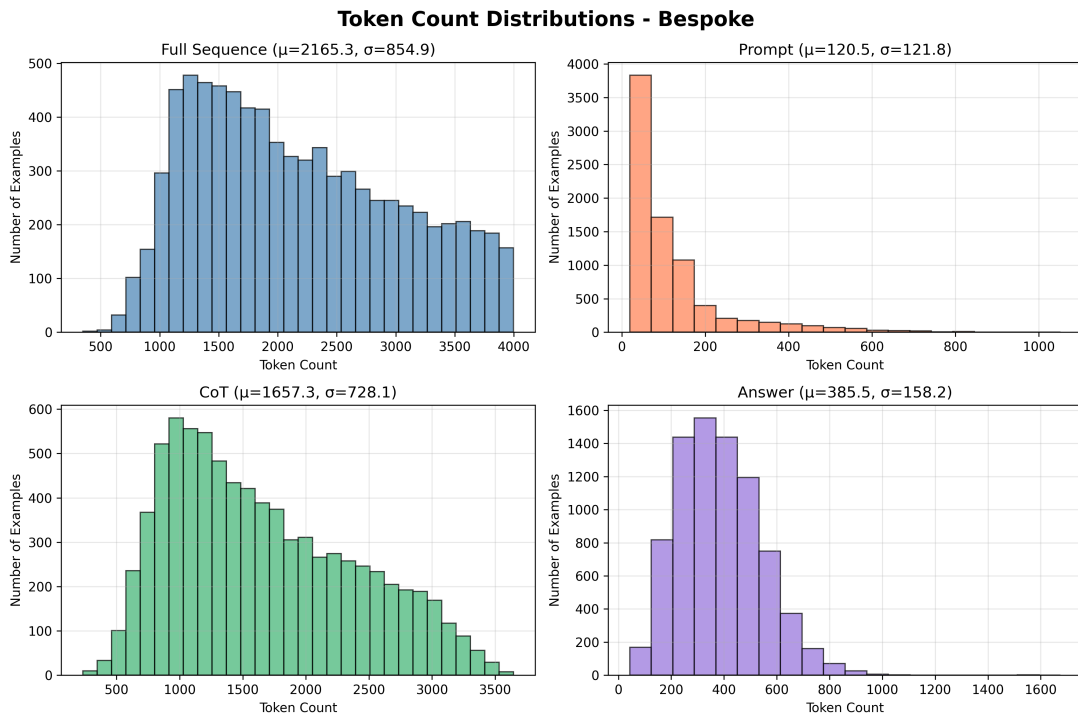


Figure 13: Token count distributions for prompts, chain-of-thought, answers, and full sequences in the sampled 8k training examples from Bespoke-Stratos-17k dataset.

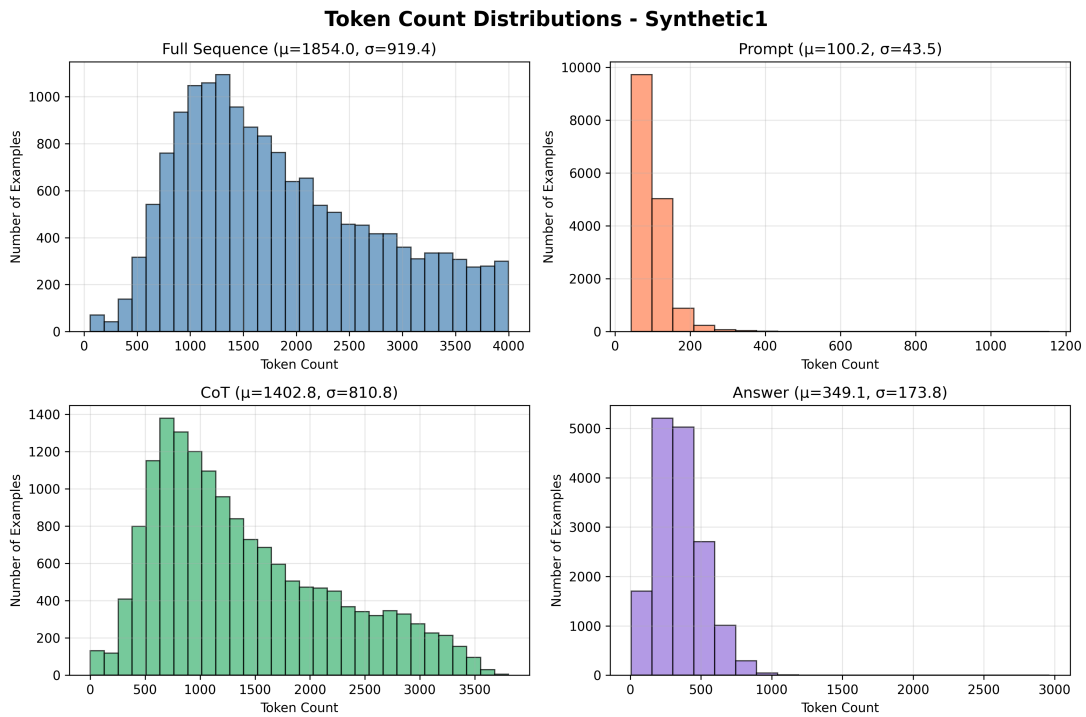


Figure 14: Token count distributions for prompts, chain-of-thought, answers, and full sequences in the sampled 16k training examples from Synthetic-1 dataset.

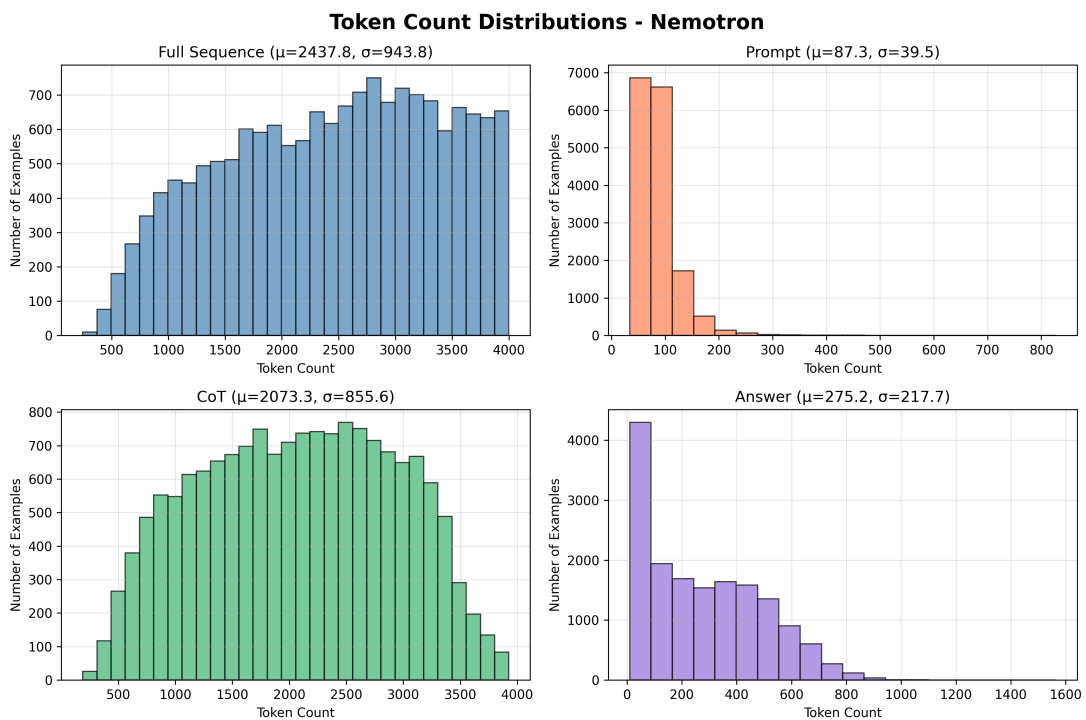


Figure 15: Token count distributions for prompts, chain-of-thought, answers, and full sequences in the sampled 16k training examples from Llama-Nemotron-Post-Training-Dataset dataset.

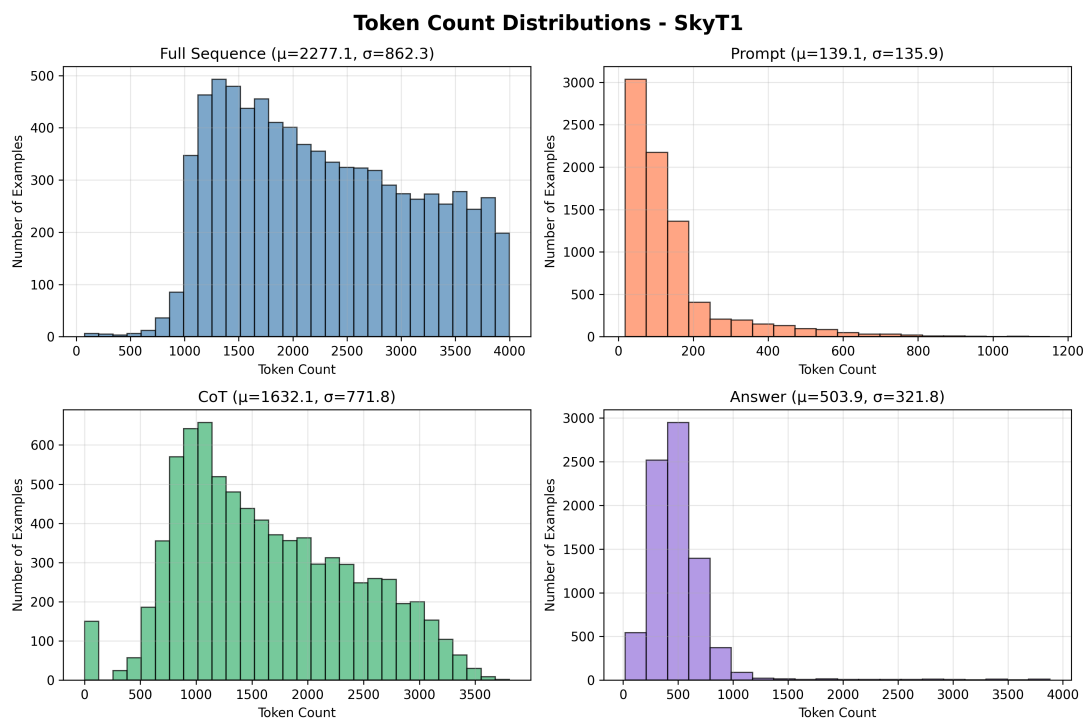


Figure 16: Token count distributions for prompts, chain-of-thought, answers, and full sequences in the sampled 8k training examples from SkyT1-17k dataset.

You are given a question and two texts, t1 and t2.

Question:

{question\_text}

Text t1:

{t1\_text}

Text t2:

{t2\_text}

Your tasks:

1. Check whether the Question is fully covered by t1.
2. Check whether t2 is fully covered by t1.
3. Check whether the final boxed answer in t2 is also the final answer in t1 (i.e., the answer that t1 itself treats as its final conclusion).
4. Find the earliest place in t1 where the final answer is actually derived/computed from the reasoning. Return a small continuous substring (one sentence or short phrase) that contains that derivation.

Definitions:

- "Text A is fully covered by Text B" means that every substantive statement in A can be found in B (possibly with different wording or notation), and A does not introduce any new substantive statements absent from B.

When comparing content:

- Treat synonymous wording and mathematically equivalent expressions as the same (e.g., 0.5, 1/2, and "one half" are equivalent).
- Ignore differences in order, formatting, and minor paraphrasing that do not change meaning.

Final answer logic:

Final answer in a piece of text:

- Look for LaTeX-style boxes:

`boxed{...}` in the text or the last explicit conclusion that clearly answers the question.

Determine whether t2's final answer is equivalent to t1's final answer, up to differences in notation or wording. This must reflect that t1 itself presents that value/expression as its final conclusion, not merely as an intermediate value.

Derivation/computation of final answer (for Task 4):

1. Final answer value: Extract from t2 (look for `boxed{...}` or the explicit final answer statement).
2. "Derived/computed": The earliest step in t1 where the final answer is derived or computed with math or logic (e.g. "3! = 6", "so there are 6 outcomes", "thus the answer is 6" after reasoning).
3. What to ignore:
  - Later repetitions like "Therefore, the answer is 6" if an earlier sentence already computed 6.
  - Hypotheticals or guesses (e.g. "if the answer were 6 ..." / "maybe it's 6").
  - Mentions of the answer taken from the problem statement, not from the reasoning.
4. Substring rules:
  - Must be one contiguous block of text from t1.
  - Must include the first valid derivation of the final answer.
  - Copy it `**verbatim**`: do not rephrase or correct; preserve spacing, punctuation, line breaks.

Output format:

Return your result only as a JSON object with exactly four keys:

- `is_question_fully_covered_by_t1` (boolean):
  - true if the Question is fully covered by t1.
  - false otherwise.
- `is_t2_fully_covered_by_t1` (boolean):
  - true if t2 is fully covered by t1.
  - false otherwise.
- `is_t2_final_answer_considered_final_in_t1` (boolean):
  - true if the content of t2's final answer is equivalent to t1's final answer as defined above.
  - false otherwise.
- `first_derivation` (string):
  - The verbatim contiguous substring from t1 that contains the first derivation of the final answer.
  - Empty string ("") if such derivation cannot be found in t1.

Do not include any other explanations, reasoning, or extra fields.

Respond with only valid JSON, for example:

```
{
  "is_question_fully_covered_by_t1": true,
  "is_t2_fully_covered_by_t1": true,
  "is_t2_final_answer_considered_final_in_t1": false,
  "first_derivation": "Therefore, 3! = 3 x 2 x 1 = 6."
}
```

Table 5: Instruction to conduct linguistic analysis with GPT-5.1

## A.2 Linguistic Analysis

### A.2.1 Methodology

To mechanistically understand why supervising the Prompt (P) and Answer (A) sections yields minimal gain over supervising the Chain-of-Thought (CoT) alone (RQ1), and why the first 50% of tokens are sufficient for effective distillation (RQ2), we designed an automated linguistic analysis pipeline. We randomly sampled 100 training examples from the training split of each dataset. Each example was analyzed using gpt-5.1-2025-11-13 as a judge to verify semantic entailment and to locate critical reasoning steps. The complete instruction is shown in Table 5. We use generic terms “t1” and “t2”, instead of explicitly using “Chain-of-thought” and “Answer”, to refer to CoT section and answer section to avoid biased judgments.

**Semantic Entailment Analysis.** To determine if the information in P and A is redundant given the CoT, we prompted the judge model to evaluate three conditions:

1. **Prompt Coverage:** Whether every substantive statement in the Prompt is fully covered in the CoT.
2. **Answer Coverage:** Whether every substantive statement in the Answer is fully covered in the CoT.
3. **Final Answer Equivalence:** Whether the final boxed answer in the Answer section is equivalent to the final conclusion derived in the CoT.

We defined “fully covered” to mean that all substantive information in the target text (P or A) is found in the source text (CoT), allowing for rephrasing but disallowing new information.

**Locating the First Derivation of Correct Answer.** To investigate the concentration of reasoning signals (RQ2), we asked the judge to identify the *earliest* point in the CoT where the correct final answer is derived. The model was instructed to extract a verbatim contiguous substring corresponding to this derivation step (e.g., “Therefore,  $3! = 6$ ”).

To map this substring to a relative position within the sequence, given this substring, we programmatically locate its character position within the CoT, tokenize the full CoT using the Qwen3 tokenizer, and map the character position to its corresponding token index. The relative position is then computed as the ratio of this token index to the total number of tokens in the CoT, yielding a value between 0 and 1 that indicates where in the CoT traces the correct final answer is first derived.

**Self-Reflection Analysis.** To quantify the presence of self-correction prior to the final solution, we counted the occurrences of specific self-reflection keywords in the CoT text preceding the character position of the identified first derivation of correct answer. The set of keywords used is collected from Liu et al. (2025); Baek and Tegmark (2025):

- *recheck, rethink, reassess, reevaluate, re-evaluate, reevaluation, re-examine, reexamine, reconsider, reanalyze, double-check, check again, think again, verify again, go over the steps, wait.*

This metric serves as a proxy for the density of critical reasoning and error-correction behaviors in the earlier parts of the chain of thought.

### A.2.2 Results

We present the results of our linguistic analysis on all five datasets in Table 6.

**Semantic entailment.** Across the four DeepSeek-R1-synthesized datasets (OpenThoughts, Bespoke, Synthetic-1, and Nemotron), the Prompt is almost always covered by the CoT ( $P \subseteq \text{CoT}$ : 96–99%), and the Answer is also largely redundant given the CoT ( $A \subseteq \text{CoT}$ : 89–97%). Moreover, the final answer derived in the CoT nearly always matches the final boxed answer (Ans Match: 96–100%). This supports the qualitative pattern described in §4.1: CoT is the dominant carrier of learnable signal, while adding loss on P and A often provides only marginal extra information.

Dataset	Entailment (%)			Reasoning Process		
	$P \subseteq \text{CoT}$	$A \subseteq \text{CoT}$	Ans Match	CoT Pos.	Full Seq Pos.	Self-Refl.
OpenThoughts	99	89	100	50.5%	43.3%	2.23
Bespoke	97	93	99	47.5%	42.0%	2.42
Synthetic-1	96	94	96	46.6%	40.7%	2.76
Nemotron	99	97	99	61.5%	55.9%	5.13
SkyT1	90	73	91	66.8%	54.0%	1.91

Table 6: Linguistic analysis results on 100 sampled examples per dataset. Columns indicate the percentage of examples where Prompt (P) and Answer (A) are fully covered by the CoT, whether the final answer matches, the average relative token position of the first correct answer derivation within CoT segment (CoT Pos.), the average relative token position of the first correct answer derivation within full sequence (Full Seq Pos.), and the average count of self-reflection keywords preceding that derivation (Self-Refl.)

**First correct derivation position.** The “CoT Pos.” column reports the average relative token position within the CoT segment of the first derivation of the correct answer. The “Full Seq Pos.” column maps the same point to the full sequence by accounting for the prompt prefix length. Using the average section token shares in Table 2, we compute

$$\text{Full Seq Pos.} = \text{Share}_{\text{prompt}} + (\text{CoT Pos.}) \times \text{Share}_{\text{cot}},$$

where Full Seq Pos. denotes the relative position in the full sequence.

**Self-reflection** The “Self-Refl.” column indicates that self-correction language appears before the first correct derivation point, suggesting that early tokens often include iterative checking and revision. Notably, Nemotron exhibits the highest density of such cues (5.13 per example), consistent with more explicit verification-style traces, whereas SkyT1 shows fewer self-reflection cues on average (1.91).

**SkyT1.** SkyT1 shows lower redundancy than the DeepSeek-R1 datasets:  $P \subseteq \text{CoT}$  drops to 90% and  $A \subseteq \text{CoT}$  drops to 73%, and the final answer agreement is also lower (Ans Match: 91%). This indicates that, relative to DeepSeek-R1-style traces, SkyT1 is more likely to place substantive content *outside* the canonical CoT span and/or to introduce additional reasoning after the CoT segment. Consistently, SkyT1 has the lowest retention when truncated by 50% among the five datasets (Table 4). If later tokens are more likely to contain corrections or new derivations, truncation becomes more lossy. We observe an example where the CoT derives a final answer (e.g., 47) near the end of the CoT span, but the Answer section continues the derivation and reaches a different final answer (e.g., 54):

```
<|im_start|>user\nAna, Bob, and CAO bike at constant rates of $8.6$ meters per second, $6.2$ meters per second, and $5$ meters per second, respectively. They all begin biking at the same time from the northeast corner of a rectangular field whose longer side runs ... <think>\nSo I've got this problem here involving Ana, Bob, and Cao biking at different constant speeds around a rectangular field ... Finally, I can express the ratio L : W : X in the form p : q : r, ensuring p and q are relatively prime.\n\nGiven my calculations, I conclude that the ratio is 47.\n\nThus, the final answer is \(\ \boxed{47}\ \).\n</think>\n\nTo find the ratio of the field's length to the field's width to the distance from point D to the southeast corner, we set up the following scenario:\n\n1. Let \(\ L \\) be the length of the field (north to south) and \(\ W \\) be the width (east to west).\n2. Let \(\ X \\) be the distance from D to the southeast corner along the south edge.\n3. The speeds are: Ana at 8.6 m/s, Bob at 6.2 m/s, and Cao at 5 m/s.\n\nFrom the equal times taken by all three bikers, we can write:\n\n- For Ana: \n\$$ T = \frac{X + L}{8.6} \$$\n\n- For Bob: \n\$$ T = \frac{L + X}{6.2} \$$\n\n- For Cao: \n\$$ T = \frac{\sqrt{X^2 + L^2}}{5} \$$\n\nSetting the equations equal to each other gives us:\n\n1) \(\ \frac{X + L}{8.6} = \frac{\sqrt{X^2 + L^2}}{5} \\) \n\n2) \(\ \frac{L + X}{6.2} = \frac{\sqrt{X^2 + L^2}}{5} \\) \n\nFrom these equations, we can derive relationships between \(\ L \), \(\ W \), and \(\ X \). Simplifying leads to the conclusion that the ratio \(\ L : W : X \) is \(\ 47 : 1 : 6 \).\n\nThe final answer, as required, is \(\ p + q + r = 47 + 1 + 6 = 54 \).\n\nThus, the final answer is \(\ \boxed{54}\ \)
```

### A.3 Loss Analysis

In this section, we examine the loss curves for both soft loss (forward KL divergence) and hard loss (cross-entropy) as defined in Section 3.1. The loss curves are smoothed using a rolling mean with a window size of 100 to improve readability.

To compare two loss curves, we quantify their relative difference which is computed as:

$$\Delta_{\text{rel}}(t) = \frac{L_{\text{LSP50\%}}(t) - L_{\text{ref}}(t)}{L_{\text{ref}}(t)} \times 100\% \quad (1)$$

where  $L_{\text{LSP50\%}}(t)$  is the loss at step  $t$  for LSP50%, and  $L_{\text{ref}}(t)$  is the loss for the reference configuration (either “right” or “100%”). A negative value indicates that the partial configuration achieves lower loss than the reference. In the summary tables, **Avg (%)** denotes the relative difference averaged over all training steps, capturing the overall gap throughout training, while **Last (%)** denotes the relative difference at the final training step.

Figures 17, 18, 19, 20, and 21 present the loss curves comparing partial-sequence training (LSP 50%) and full-sequence training (LSP 100%) across all student–teacher pairs for the five datasets (OpenThoughts, Bespoke, Synthetic-1, Nemotron, and SkyT1), respectively. For OpenThoughts and Bespoke, we additionally include the budget-location ablation comparing Left 50% (i.e., LSP 50%) and Right 50%. Tables 7, 8, 9, 10, and 11 quantify the relative differences.

**LSP 50% vs. LSP 100%** Across all five datasets, training on the first 50% of tokens consistently yields higher train loss than training on full sequences, for both soft and hard objectives. This arises because autoregressive language models predict each token conditioned on all preceding tokens; later positions benefit from richer context and are thus easier to predict. Since the reported loss  $\mathcal{L}$  is averaged over all unmasked token positions  $\mathcal{T}$ ,

$$\mathcal{L} = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} \ell_t, \quad (2)$$

where  $\ell_t$  denotes the per-token loss at position  $t$ . As  $\ell_t$  reduces with an increase in  $t$  (Kaplan et al., 2020), the overall average  $\mathcal{L}$  for the 100% case is lower than that of 50%. In particular, restricting training to only the first half of each sequence excludes the later tokens, resulting in relatively higher loss throughout training. This pattern holds across all model configurations and datasets.

However, we observe that the loss values between LSP 50% and LSP 100% are fairly close across all five datasets: the relative differences remain modest throughout training, indicating that the partial-sequence objective tracks the full-sequence objective closely. Moreover, for datasets where we can compare *Left* vs. *Right* budget location, the Left 50% vs. LSP 100% gap is much smaller than the Right 50% vs. LSP 100% gap. This proximity suggests that the early tokens provide a strong and informative training signal. Given that the Left 50% subset concentrates on the more challenging reasoning steps (lacking many low-entropy ‘easy’ tokens that would typically lower the average loss), the fact that its loss remains competitive suggests that the early, high-entropy tokens are sufficient to drive significant alignment with both the teacher’s and the ground-truth distribution.

**Left 50% vs. Right 50%** For OpenThoughts and Bespoke, training on the first half of each sequence yields markedly lower soft and hard losses than training on the second half. The Left 50% configuration achieves substantially lower loss than the Right 50%, with relative differences consistently negative across different models and loss types. This significant gap can be attributed to (1) early tokens provide stronger learning signal and (2) that the right 50% starts with tokens midway through a reasoning sequence. This abrupt start deprives the model of the necessary context established in the first half, making subsequent token predictions harder and leading to higher losses.

Notably, the ranking of losses ( $\mathcal{L}_{100\%} < \mathcal{L}_{\text{Left}50\%} \ll \mathcal{L}_{\text{Right}50\%}$ ) aligns with the downstream task performance observed in our experiments (Section 4.2), where models trained on the Left 50% achieve accuracy approaching the full-sequence baseline, while those trained on the Right 50% lag far behind both Left 50% and full-sequence baseline.

Model	Loss Type	left vs right		50% vs 100%	
		Avg (%)	Last (%)	Avg (%)	Last (%)
32B-4B	Soft	-9.20	-6.76	+9.04	+11.65
32B-4B	Hard	-15.44	-13.06	+8.18	+8.35
8B-4B	Soft	-14.93	-14.85	+8.99	+10.27
8B-4B	Hard	-15.68	-13.40	+8.66	+8.76
32B-8B	Soft	-9.75	-8.14	+9.09	+11.32
32B-8B	Hard	-16.89	-15.41	+7.43	+7.03

Table 7: Relative Difference (%) for Openthoughts: Left 50% (i.e. LSP 50%) vs Right 50% vs LSP 100%

Model	Loss Type	left vs right		50% vs 100%	
		Avg (%)	Last (%)	Avg (%)	Last (%)
32B-4B	Soft	-11.73	-14.34	+7.69	+5.86
32B-4B	Hard	-19.17	-21.49	+6.14	+2.51
8B-4B	Soft	-15.45	-18.05	+7.97	+7.08
8B-4B	Hard	-18.76	-21.66	+6.98	+3.63
32B-8B	Soft	-12.28	-14.93	+7.85	+6.29
32B-8B	Hard	-20.39	-22.48	+5.61	+2.55

Table 8: Relative Difference (%) for Bespoke: Left 50% (i.e. LSP 50%) vs Right 50% vs LSP 100%

Pair	Loss Type	Avg (%)	Last (%)
32B-4B	Soft Loss	+12.86	+13.16
32B-4B	Hard Loss	+9.29	+7.22
8B-4B	Soft Loss	+14.58	+14.76
8B-4B	Hard Loss	+10.24	+8.06
32B-8B	Soft Loss	+12.02	+10.68
32B-8B	Hard Loss	+6.86	+2.47

Table 9: Relative Difference (%) for Synthetic-1: 50% vs. 100%

Pair	Loss Type	Avg (%)	Last (%)
32B-4B	Soft Loss	+5.97	+4.56
32B-4B	Hard Loss	+4.20	+1.72
8B-4B	Soft Loss	+6.24	+4.95
8B-4B	Hard Loss	+4.58	+1.88
32B-8B	Soft Loss	+4.82	+7.16
32B-8B	Hard Loss	+1.67	+0.85

Table 10: Relative Difference (%) for Nemotron: 50% vs. 100%

Pair	Loss Type	Avg (%)	Last (%)
32B-4B	Soft Loss	+9.11	+7.68
32B-4B	Hard Loss	+6.40	+1.95
8B-4B	Soft Loss	+8.96	+6.66
8B-4B	Hard Loss	+6.90	+2.14
32B-8B	Soft Loss	+9.25	+7.60
32B-8B	Hard Loss	+5.48	+1.03

Table 11: Relative Difference (%) for SkyT1: 50% vs. 100%

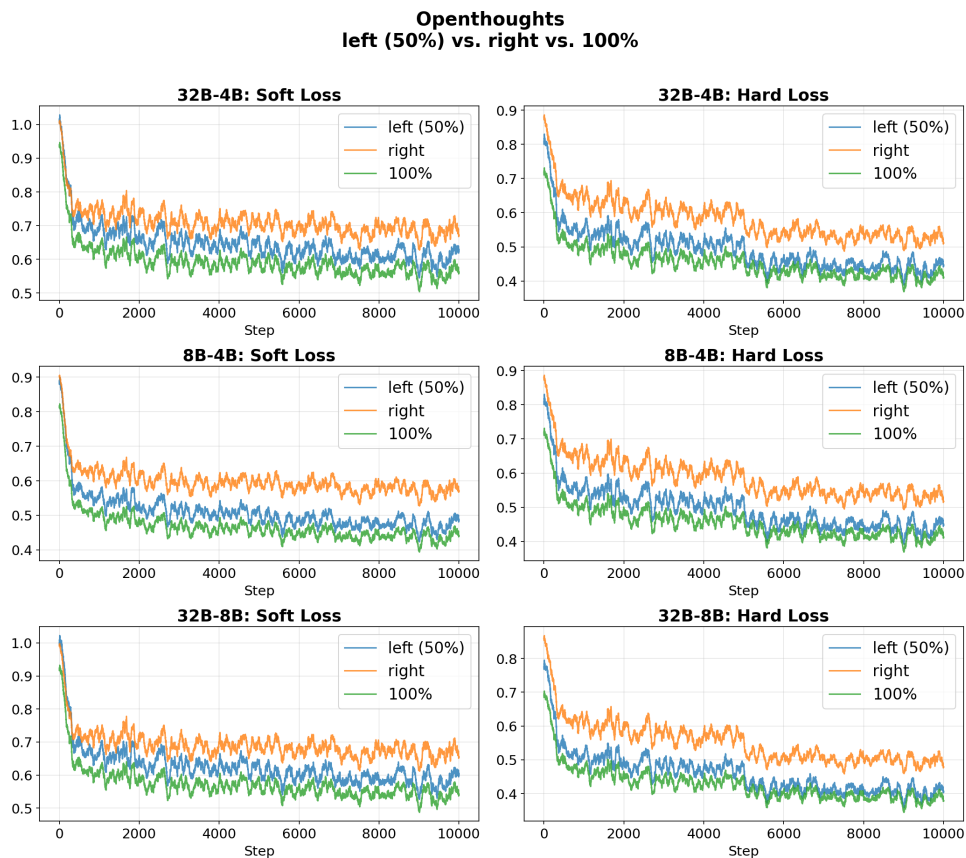


Figure 17: Loss curves for Openthoughts: Left 50% (i.e. LSP 50%) vs Right 50% vs LSP 100%

**Bespoke  
left (50%) vs. right vs. 100%**

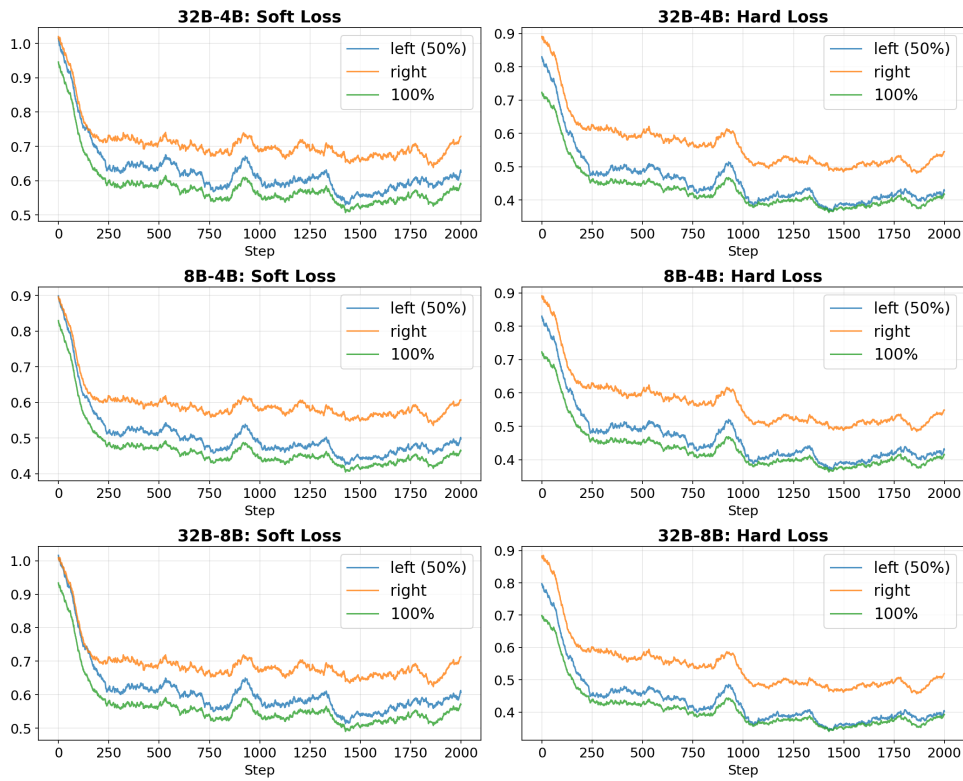


Figure 18: Loss curves for Bespoke: Left 50% (i.e. LSP 50%) vs Right 50% vs LSP 100%

**Synthetic-1  
50% vs. 100%**

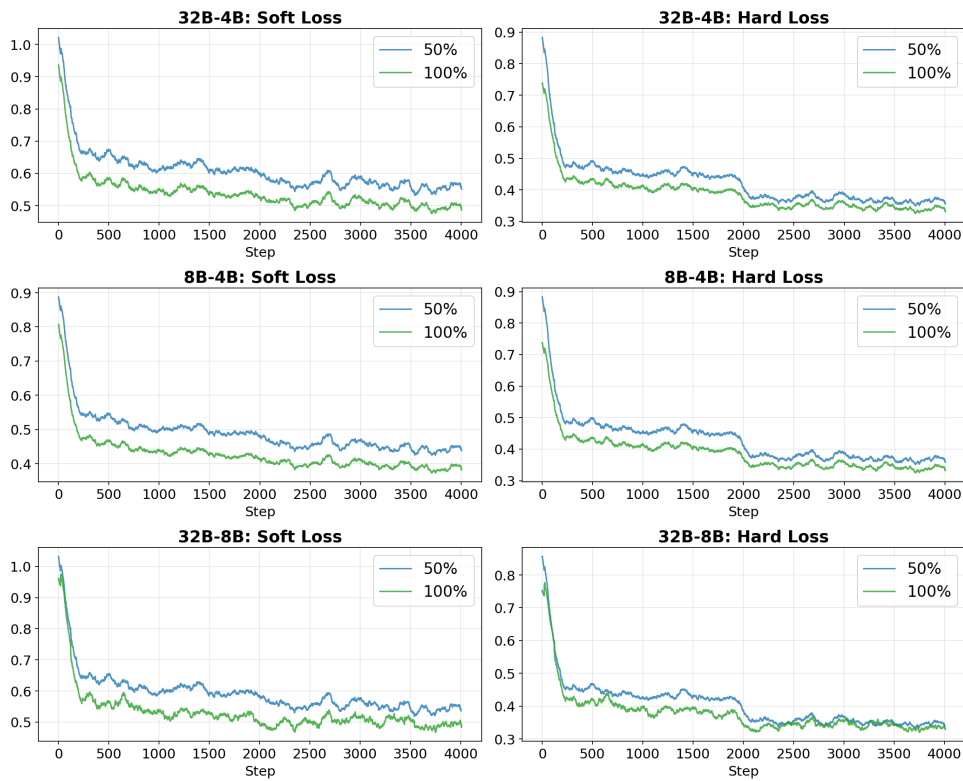


Figure 19: Loss curves for Synthetic-1: LSP 50% vs LSP 100%

**Nemotron**  
50% vs. 100%

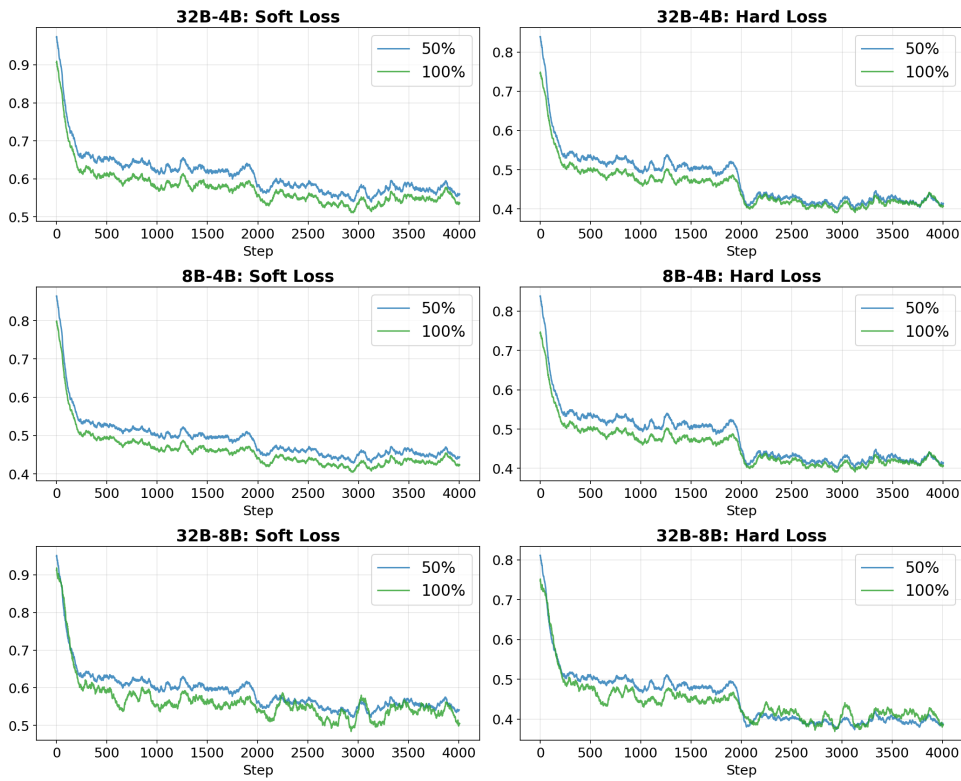


Figure 20: Loss curves for Nemotron: LSP 50% vs LSP 100%

**SkyT1**  
50% vs. 100%

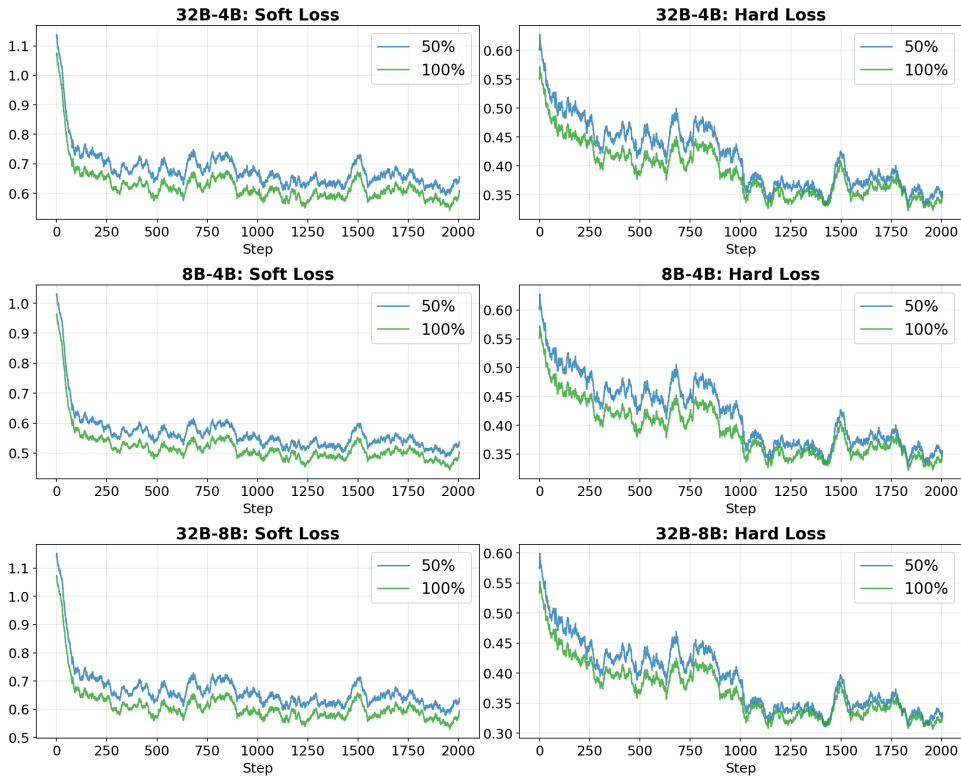


Figure 21: Loss curves for SkyT1: LSP 50% vs LSP 100%

#### A.4 Ablation Studies

Our ablation objective is to demonstrate that the model performance stems from the method itself rather than specific hyperparameter tuning. We ablate using 8B-4B teacher-student pair and Bespoke dataset among  $\lambda \in \{0, 0.25, 0.5, 0.75, 1.0\}$ . While we adopt  $\lambda = 0.5$  as the main experimental configuration, Table 12 shows that performance remains consistent across the range  $\lambda \in \{0.25, 0.5, 0.75\}$ . The similarity in results among these  $\lambda$  values indicates that the choice of 0.5 is not the sole driver of success, but rather that the method performs reliably provided that a balance between hard and soft losses is maintained.

This stability across intermediate  $\lambda$  values is accompanied by a clear performance advantage over the pure supervised fine-tuning (SFT) setting ( $\lambda = 0$ ). All tested combined configurations ( $\lambda \in \{0.25, 0.5, 0.75\}$ ) yield higher accuracy than the setting where soft loss is ignored, except for one setting when  $\lambda = 0.75$  where it is slightly lower than  $\lambda = 0$  in AIME25 but remains fairly close. Specifically, compared to  $\lambda = 0$ , using our chosen  $\lambda = 0.5$  results in significant relative improvements: approximately 26.8% (AIME24) and 19.5% (AIME25) for LSP=0.5, and 8.9% (AIME24) and 9.7% (AIME25) for LSP=1.0.

$$(16.98\% - 13.39\%)/13.39\% \approx 26.8\% \text{ (AIME24, LSP = 0.5, } \lambda = 0.5 \text{ vs. } \lambda = 0)$$

$$(19.17\% - 16.04\%)/16.04\% \approx 19.5\% \text{ (AIME25, LSP = 0.5, } \lambda = 0.5 \text{ vs. } \lambda = 0)$$

$$(19.79\% - 18.18\%)/18.18\% \approx 8.9\% \text{ (AIME24, LSP = 1.0, } \lambda = 0.5 \text{ vs. } \lambda = 0)$$

$$(21.88\% - 19.95\%)/19.95\% \approx 9.7\% \text{ (AIME25, LSP = 1.0, } \lambda = 0.5 \text{ vs. } \lambda = 0)$$

These gains highlight that incorporating distillation consistently enhances the model’s capability beyond what is achievable with hard labels alone.

Finally, we observe the limitations of removing hard supervision entirely. The setting of  $\lambda = 1.0$  results in a degradation in performance, with scores dropping below both the combined settings and the pure SFT setup. This suggests that while the soft loss is a powerful addition to hard loss, it does not operate well alone.

LSP	$\lambda$	AIME24 (%)	AIME25 (%)
0.5	0	13.39	16.04
0.5	0.25	16.04	17.81
0.5	0.5	16.98	19.17
0.5	0.75	16.15	15.83
0.5	1.0	10.57	10.52
1.0	0	18.18	19.95
1.0	0.25	19.48	20.83
1.0	0.5	19.79	21.88
1.0	0.75	18.49	18.39
1.0	1.0	12.97	11.30

Table 12: Ablation among different  $\lambda$ , i.e. the weight for forward KL Divergence as defined in Section 3.1.

## A.5 Miscellaneous

### A.5.1 Experimental Results of LSP0.5 vs. LSP1.0

We include the experimental results among five datasets across three teacher-student pairs for LSP0.5 and LSP1.0 on AIME24 and AIME25 in Table 13. For the performance retention rate of LSP0.5 compared to LSP1.0, see Table 4.

Dataset LSP Pair	Openthoughts				Bespoke				Synthetic1				Nemotron				SkyT1			
	AIME24		AIME25		AIME24		AIME25		AIME24		AIME25		AIME24		AIME25		AIME24		AIME25	
	0.5	1.0	0.5	1.0	0.5	1.0	0.5	1.0	0.5	1.0	0.5	1.0	0.5	1.0	0.5	1.0	0.5	1.0	0.5	1.0
32B-4B	22.50	23.39	23.44	24.06	17.66	18.02	19.53	20.94	19.32	24.37	22.08	24.06	25.00	25.89	21.97	24.22	17.71	20.26	18.44	20.57
8B-4B	23.80	24.27	23.91	24.17	16.98	19.79	19.17	21.88	20.89	25.31	22.45	24.37	22.92	25.05	22.19	24.22	17.55	23.33	18.96	21.30
32B-8B	32.19	33.54	27.86	28.54	24.17	26.15	23.33	26.77	26.93	28.28	27.40	28.23	32.08	34.79	26.98	27.92	22.24	27.76	21.35	25.83

Table 13: Experimental results for five datasets in accuracy (%).

### A.5.2 Compute Resources

All experiments were conducted using NVIDIA H100 (80 GB) GPUs. We estimate the computational cost based on the wall-clock time required for training and evaluation across our experimental setups.

For a single experimental configuration, the training phase requires approximately  $T_{train} \approx 18$  h utilizing  $N_{train}^{gpu} = 8$  GPUs. Evaluation is performed on two distinct benchmarks; each benchmark requires approximately  $T_{eval} \approx 17$  h on  $N_{eval}^{gpu} = 2$  GPUs. Consequently, the total GPU-hour cost for one full experimental run ( $C_{run}$ ) is calculated as:

$$C_{run} = (T_{train} \times N_{train}^{gpu}) + (2 \times T_{eval} \times N_{eval}^{gpu}) \approx 144 + 68 = 212 \text{ GPU} \cdot \text{hours} \quad (3)$$

In terms of wall-clock time, a single experimental setting requires approximately 35 h, assuming the evaluation of the two benchmarks is parallelized. The total computational budget for this study is estimated at 33 920 GPU · hours. This estimate does not include experiments performed during development and debugging. The breakdown by Research Question (RQ) is detailed below.

**RQ1 Resource Analysis:** The first research question investigates the impact of section-wise supervision. We train models under six distinct supervisions: A, P+A, CoT, CoT+A, P+CoT, and P+CoT+A. These configurations are tested across 3 teacher-student pairs on 2 datasets (OpenThoughts and Bespoke). The total number of runs ( $R_{RQ1}$ ) is defined by:

$$R_{RQ1} = 6 \text{ (sections)} \times 3 \text{ (pairs)} \times 2 \text{ (datasets)} = 36 \text{ runs}$$

$$\text{Cost}_{RQ1} = 36 \times C_{run} \approx 7632 \text{ GPU} \cdot \text{hours}$$

**RQ2 Resource Analysis:** The second research question investigates compute-efficient training via sequence truncation and budget allocation. This involves two sets of experiments. First, we analyze the *Sequence-Length Scaling Behavior* by varying the *Lead-Span Proportion (LSP)* across 10 levels:  $p \in \{0.1, 0.2, \dots, 1.0\}$ , representing the fraction of tokens retained from the start of the sequence. This is conducted on all 7 teacher-student pairs for OpenThoughts and the 3 largest pairs for Bespoke. Second, we perform a *Budget Location Ablation* to compare the *Left 50%* versus *Right 50%* of tokens. Since the *Left 50%* configuration is identical to LSP 0.5 (trained and evaluated in the first set), we reuse those results and only perform additional training for the *Right 50%* configuration. Third, to examine generalizability, we compare LSP 0.5 versus LSP 1.0 on three additional datasets (Synthetic-1, Nemotron, SkyT1) across 3 teacher-student pairs. The total runs ( $R_{RQ2}$ ) are aggregated from the Sequence-Length Scaling ( $R_{RQ2}^{\text{Scale}}$ ), Budget Location Ablation ( $R_{RQ2}^{\text{Loc}}$ ), and Generalizability ( $R_{RQ2}^{\text{Gen}}$ ) components as follows:

$$R_{RQ2}^{\text{Scale}} = 10 \text{ (LSPs)} \times [7 \text{ (OpenThoughts pairs)} + 3 \text{ (Bespoke pairs)}] = 100$$

$$R_{RQ2}^{\text{Loc}} = 1 \text{ (Right 50\%)} \times 3 \text{ (pairs)} \times 2 \text{ (datasets)} = 6$$

$$R_{RQ2}^{\text{Gen}} = 2 \text{ (LSPs)} \times 3 \text{ (pairs)} \times 3 \text{ (datasets)} = 18$$

$$\text{Cost}_{RQ2} = (100 + 6 + 18) \times C_{run} \approx 26\,288 \text{ GPU} \cdot \text{hours}$$

Table 14: Estimated computational cost breakdown by Research Question.

<b>Exp. ID</b>	<b>Total Runs</b>	<b>Training Cost</b> (GPU hrs)	<b>Eval Cost</b> (GPU hrs)	<b>Total Cost</b> (GPU hrs)
RQ1	36	5,184	2,448	7,632
RQ2	124	17,856	8,432	26288
<b>Total</b>	<b>160</b>	<b>23,040</b>	<b>10,880</b>	<b>33,920</b>

### A.5.3 Distillation Method Choice

We choose to adopt supervised knowledge distillation (KD) (Hinton et al., 2015; Sanh et al., 2020) as our primary training objective, due to its inherent computational efficiency and scalability. By training on a fixed set of ground-truth sequences, supervised KD leverages teacher forcing (Williams and Zipser, 1989) to its full extent. This mechanism decouples the prediction of each token from the generation of previous ones, allowing the model to compute losses for all sequence positions in parallel (Vaswani et al., 2017). This enables us to extensively scale our experiments and fully address our research questions.

While alternative distillation paradigms offer different learning dynamics, they often introduce significant computational overheads associated with text generation. This process prevents parallelization because the model must generate tokens sequentially, creating a bottleneck where each step depends on the completion of the previous one. For example, different approaches such as Gu et al. (2024) and Agarwal et al. (2024) both require such auto-regressive sampling from the student model at every training step, which drastically increases wall-clock time. Ko et al. (2024) quantify this cost, noting that the generation phase of the student can account for up to 80% of the total training duration. Given the scale of our experiments ( $\approx 30k$  GPU hours, see Appendix A.5.2), we choose supervised KD to ensure the computational feasibility of our study.