TRAINING LARGE LANGUAGE MODELS TO REASON IN PARALLEL WITH GLOBAL FORKING TOKENS

Anonymous authors
Paper under double-blind review

ABSTRACT

Although LLMs have demonstrated improved performance by scaling parallel test-time compute, doing so relies on generating reasoning paths that are both diverse and accurate. For challenging problems, the forking tokens that trigger diverse yet correct reasoning modes are typically deep in the sampling tree. Consequently, common strategies to encourage diversity, such as temperature scaling, encounter a worsened trade-off between diversity and accuracy. Motivated by this challenge, we treat parallel reasoning as a set-of-next-token-prediction problem and incorporate a set-based global loss into Supervised Fine-Turning (SFT) using bipartite matching between global forking tokens and unique reasoning traces. We observe that, whereas naive fine-tuning with multiple reasoning traces collapses these unique reasoning modes, our proposed method, Set Supervised Fine-Turning (SSFT), preserves these modes and produces emergent global forking tokens. Experiments on multiple reasoning benchmarks show our SSFT method consistently outperforms SFT under both pass@1 and cons@k metrics.

1 INTRODUCTION

Large language models have recently improved reasoning by allocating more test-time compute to generate more tokens before producing the final answer (OpenAI, 2025). However, extended sequential scaling can lead to "overthinking", where performance decreases beyond a certain sequence length (Ghosal et al., 2025; Chen et al., 2024a). To mitigate this, another scaling dimension based on repeated parallel sampling and aggregation (Wang et al., 2022; Brown et al., 2024) has shown success in further boosting reasoning performance. However, these methods rely on LLMs generating diverse yet correct solutions; as tasks become harder, a mechanism for increasing diversity is required. Recent work shows that only a minority of tokens in Chain-of-Thought reasoning (Wei et al., 2022) can act as forking tokens that lead to distinct reasoning modes (Wang et al., 2025), so as the problem becomes harder and the generation becomes longer, it can become substantially harder to sample them. Also, common practices to encourage diversity, typically through temperature scaling, inherently entails an diversity-accuracy trade-off, as the forking tokens that trigger diverse yet correct reasoning modes are typically located deeply within the sampling tree. Moreover, recent theoretical work also shows that increasing the temperature alone does not necessarily guarantee greater diversity unless the model is explicitly trained to ensure coverage. (Verine et al., 2025).

Building on these observations, we aim to leverage diverse reasoning traces to train for coverage (Guo et al., 2025; Google, 2025b). We introduced *global forking tokens* prior to generate parallel reasoning traces and frame parallel reasoning as a *set prediction* problem. Specifically, given a question, an LLM, conditioned on a reserved set of tokens in a chosen ordering, generates M reasoning sequences in parallel, each aligned with one of M ground-truth reasoning traces. For each ordering, we compute the total autoregressive loss across the generated sequences. By enumerating all possible orderings, we identify the minimum loss, which defines the set language modeling loss, conditioned on the distinct forking tokens (Equation 3). This formulation naturally incorporates coverage into the training objective and is capable of learning *global forking tokens* that can serve as prompts to trigger reasoning modes that are both diverse and accurate. We operationalize this idea through our Set Supervised Fine-Tuning (SSFT) framework. Our main contributions are summarized below. And we will also release source code to support reproducibility.

- We introduced global forking tokens and incorporate a set-based loss into SFT using bipartite matching between a set of reserved global forking tokens and diverse reasoning traces (Section 2; Figure 1). The goal is that, after training with SSFT, the model can initiate distinct reasoning modes when prompted with different tokens from this set, thus reducing the dependence on sampling forking tokens mid-generation (Section 2.3; Figure 2).
- We empirically demonstrate that, across multiple reasoning benchmarks, a model fine-tuned with SSFT outperforms a standard SFT model trained on the same distilled traces from diverse teacher models, improving $\mathrm{Pass}@1$, $\mathrm{Pass}@k$, and $\mathrm{Cons}@k$ (Section 3, Table 1 and Figure 3). We also find that naively fine-tuning using diverse reasoning traces without the set loss from bipartite matching can result in control tokens initiating the same collapsed reasoning mode (Figure 5). In contrast, the global forking tokens learned by SSFT reliably initiate distinct reasoning (Figure 4).
- To facilitate training with variable-size parallel generation, we propose a scalable training implementation that does not increase VRAM usage. Instead of concatenating diverse reasoning traces, our algorithm can expand these variable-sized parallel generations along the batch dimension under distributed training (Appendix A.3).

2 Learning Global Forking Tokens via Set Supervised Finetuning

Background on Language Modeling with Reasoning. In language modeling, the goal is to train a model $\pi_{\boldsymbol{\theta}}$ to approximate the joint distribution over a sequence of word tokens $\mathbf{x} = \{x_i\}_{i=1}^{\mathrm{T}} \in \mathcal{V}^{\mathrm{T}}$, where T is the sequence length, and each token is within a finite vocabulary set $\mathcal{V}$. An autoregressive model uses the chain rule to represent it as a product of conditionals on the preceding tokens: $\pi_{\boldsymbol{\theta}}(\mathbf{x}) = \prod_{t=1}^{\mathrm{T}} \pi_{\boldsymbol{\theta}}\left(\mathrm{x}_t|\mathbf{x}_{<t}\right)$. This is known as *next-token-prediction* (NTP) (Radford et al., 2019). For reasoning tasks, we break the sequence of word tokens into: (1) an input prompt $\mathbf{x} = \{\mathrm{x}_t\}_{t=1}^{\mathrm{T}_{\mathbf{x}}}$, (2) a special token (or a sequence of tokens) g that initiates reasoning, and (3) a reasoning trace plus the final answer $\mathbf{r} = \{\mathrm{r}_t\}_{t=1}^{\mathrm{T}_{\mathbf{r}}}$. To simplify notation, we combine a verifiable answer and a reasoning trace. A reasoning model autoregressively generates a reasoning path and the final answer: $\pi_{\boldsymbol{\theta}}(\mathbf{r}|\mathbf{x}, \mathrm{g}) = \prod_{t=1}^{\mathrm{T}_{\mathbf{r}}} \pi_{\boldsymbol{\theta}}\left(\mathrm{r}_t|\mathbf{x}, \mathrm{g}, \mathbf{r}_{<t}\right)$. To train a reasoning model, Supervised Fine-tuning (SFT) minimizes the negative log-likelihood of a ground-truth reasoning trace, i.e., $\mathcal{L}(\boldsymbol{\theta}) = -\mathbb{E}_{\mathbf{x},\mathbf{r}}[\sum_t \log \pi_{\boldsymbol{\theta}}(\mathrm{r}_t|\mathbf{x}, \mathrm{g}, \mathbf{r}_{<t})]$.

2.1 Parallel Reasoning as Set of Next Token Prediction

Problem Setup. In this paper, our goal is not only to instill new reasoning capabilities into a model, but also to ensure that prompting with a set of reserved special tokens, in parallel with a question, elicits distinct reasoning traces. We call these *global forking tokens* $\boldsymbol{g} := \{\mathrm{g}^{(i)}\}_{i=1}^{\mathrm{N}}$ instantiated as $\{$<think i>$\}_{i=1}^{\mathrm{N}}$ tags. We use $\mathrm{g}^{(i)}$ interchangeably with <thinki>, depending on context for clarity. We consider a setting with multiple sources of reasoning traces, obtained at low cost without human annotation by distilling from diverse teachers, sampling repeatedly, and potentially filtering with a verifiable metric such as correctness. We adopt this low-cost regime to highlight the effectiveness of our algorithm, though the method extends naturally to settings with well-annotated annotated, human-labeled data. So our problem is to (1) learn to do a set of next-token predictions on multiple distinct yet correct reasoning traces $\mathbf{R} := \{\mathbf{r}^{(j)}\}_{j=1}^{\mathrm{M}}$ in parallel for an input prompt $\mathbf{x}$, and (2) ensure that distinct global forking tokens can uniquely initiate these distinct traces.

To do this, we make a simple change to the NTP loss, which now has two requirements: **(1) Permutation-invariance:** It should not depend on the order of elements in $\mathbf{R}$ and $\boldsymbol{g}$, so we don't penalize a trace that incurs high NTP loss under one forking token if the model predicts it well when conditioned on another. **(2) No shared global forking token:** We want $\{\mathrm{g}^{(i)}\}_{i=1}^{\mathrm{N}}$ to uniquely initiate distinct reasoning traces, so this requirement prevents conditioning on the same $\mathrm{g}^{(i)}$ when generating distinct traces given a question.

To satisfy these requirements, we incorporate a subproblem in language modeling: finding the minimum cost *bipartite matching configuration* between the left vertices $\{\mathrm{g}^{(i)}\}_{i=1}^{\mathrm{N}}$ and the right vertices $\{\mathbf{r}^{(j)}\}_{j=1}^{\mathrm{M}}$ where the cost of each edge between a left vertex i and a right vertex j, is the NTP loss of $\mathbf{r}^{(j)}$ conditioned on $\mathrm{g}^{(i)}$ and an input prompt $\mathbf{x}$. A *matching configuration* is a set of edges connecting the left and right vertices where no two edges share a common vertex. Without loss of generality, we assume $\mathrm{N} \geq \mathrm{M}$ to simplify our notation. The total cost involves all vertices on the smaller side of the bipartite graph, so this allows us to write the summation from 1 to $\min\{\mathrm{N}, \mathrm{M}\}$, which equals M under this assumption. We denote a matching configuration as a finite map $\boldsymbol{\sigma} : \{1, ..., \mathrm{M}\} \to \{1, ..., \mathrm{N}\}$ such that $\boldsymbol{\sigma}(j) = i \iff \mathbf{r}^{(j)}$ is paired with $\mathrm{g}^{(i)}$. Let $\mathfrak{S}_{\mathrm{P}} := \{\boldsymbol{\sigma}_k\}_{k=1}^{\mathrm{P}}$ denote all the $\binom{\mathrm{N}}{\mathrm{M}} \times \mathrm{M}!$ configurations of a bipartite graph. The total cost of each configuration represents the compatibility between $\{\mathbf{r}^{(j)}\}_{j=1}^{\mathrm{M}}$ and $\{\mathrm{g}^{(i)}\}_{i=1}^{\mathrm{N}}$ under this unique matching. Figure 1 shows an example of a matching configuration.

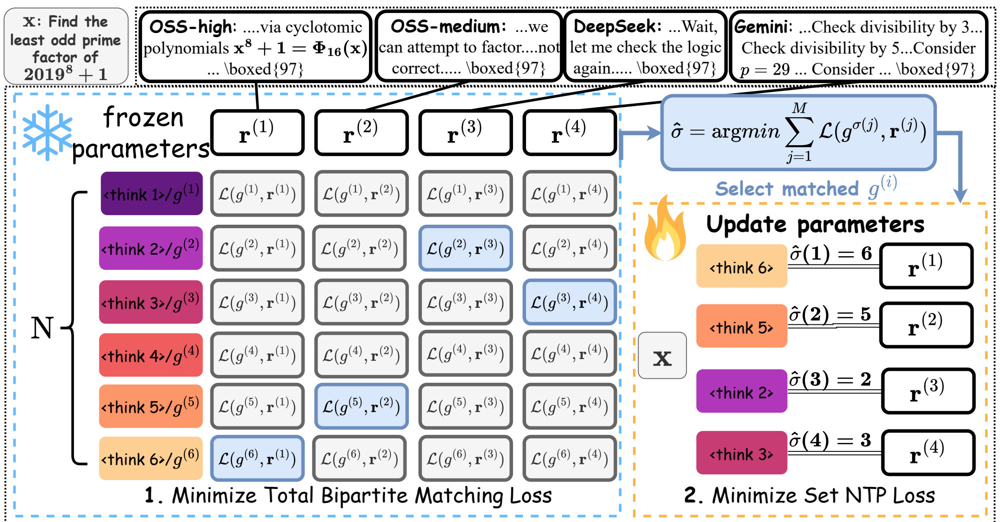


Figure 1: An illustration of one SSFT training step. **Step 1:** We first construct the cost matrix by evaluating all pairwise combinations: for each $\mathbf{r}^{(j)} \in \{\mathbf{r}^{(1)}, \mathbf{r}^{(2)}, \mathbf{r}^{(3)}, \mathbf{r}^{(4)}\}$ and each $\mathrm{g}^{(i)} \in \{\mathrm{g}^{(1)}, \mathrm{g}^{(2)}, \mathrm{g}^{(3)}, \mathrm{g}^{(4)}, \mathrm{g}^{(5)}, \mathrm{g}^{(6)}\}$, we compute the NTP loss of $\mathbf{r}^{(j)}$ conditioned on $\mathrm{g}^{(i)}$ (Equation (2)). Then we use Hungarian algorithm to find $\hat{\boldsymbol{\sigma}}$ that minimizes the total bipartite matching cost. Here, this minimum is the sum of the losses highlighted in blue, which means $\hat{\boldsymbol{\sigma}} = \{(\mathrm{g}^{(6)}, \mathbf{r}^{(1)}), (\mathrm{g}^{(5)}, \mathbf{r}^{(2)}), (\mathrm{g}^{(2)}, \mathbf{r}^{(3)}), (\mathrm{g}^{3}, \mathbf{r}^{(4)})\}$. **Step 2:** We optimize $\boldsymbol{\theta}$ by backpropagating the set of NTP losses for $\mathbf{r}^{(j)}$, each conditioned on $\mathrm{g}^{(\hat{\boldsymbol{\sigma}}(j))}$.

2.2 SSFT: Minimizing Set of NTP Losses under Optimal Bipartite Matching

Under this formulation, we propose Set Supervised Fine-Tuning (SSFT), which performs two operations at each training step: (1) find the minimum-cost bipartite matching that is optimal for uniquely initiating different reasoning traces (2) and then minimize the NTP losses under the matching configuration to instill diverse reasoning modes conditioned on the matched global forking tokens. We show our implementation in Algorithm 1.

For the first step, we first compute all the entries in the cost matrix such as the one in Figure 1 and then apply the Hungarian algorithm (Kuhn, 1955) to efficiently find the optimal $\hat{\boldsymbol{\sigma}}$:

$$\hat{\boldsymbol{\sigma}} = \arg\min_{\boldsymbol{\sigma} \in \mathfrak{S}_{\mathrm{P}}} \sum_{j=1}^{\mathrm{M}} \mathcal{L}_{\text{matching}}\left(\mathrm{g}^{(\boldsymbol{\sigma}(j))}, \mathbf{r}^{(j)}\right), \quad \text{where} \tag{1}$$

$$\mathcal{L}_{\text{matching}}\left(\mathrm{g}^{(i)}, \mathbf{r}^{(j)}\right) = -\mathrm{sg}\left(\frac{1}{\mathrm{T_r}} \sum_{t=1}^{\mathrm{T_r}} \log \pi_{\boldsymbol{\theta}}\left(\mathrm{r}_t^{(j)} | \mathbf{x}, \mathrm{g}^{(i)}, \mathbf{r}_{<t}^{(j)}\right)\right) \tag{2}$$

As noted in Equation 2, each matching cost in Equation 1 is the negative log-likelihood of $\mathbf{r}^{(j)}$ conditioned on $\mathrm{g}^{(\boldsymbol{\sigma}(j))}$ under the current model parameters. Here, $\mathrm{sg}(\cdot)$ is stop-gradient, as the matching process is done by discrete optimization w.r.t. $\boldsymbol{\sigma}$, so we can save VRAM by not storing intermediate activations. We explicitly indicate that length normalization is done to remove biases toward trace length, so that the matching is driven by semantic content.

After solving $\hat{\boldsymbol{\sigma}}$ for each $(\mathbf{x}, \mathbf{R})$, our second step optimizes model parameters $\boldsymbol{\theta}$ by backpropagating on the matching loss in Equation 3. The expectation is replaced by its sample mean over pairs of $(\mathbf{x}, \{\mathbf{r}^{(j)}\}_{j=1}^{\mathrm{M}})$ in a mini-batch. In practice, we may use only the first $\mathrm{L} < \mathrm{T}_{\mathbf{r}}$ tokens in Equation 2 to compute the matching cost and find $\hat{\boldsymbol{\sigma}}$ when the training dataset is such that these early tokens reveal sufficient differences. However, we always optimize the matching loss for the full $\mathrm{T}_{\mathbf{r}}$ length.

$$\mathcal{L}_{\text{Hungarian}}(\boldsymbol{\theta}) = -\mathop{\mathbb{E}}_{\mathbf{x},\mathbf{R}\sim\mathcal{D}}\left[\sum_{j=1}^{\mathrm{M}}\sum_{t=1}^{\mathrm{T}_{\mathbf{r}}} \log \pi_{\boldsymbol{\theta}}\left(\mathrm{r}_t^{(j)}|\mathbf{x}, \mathrm{g}^{(\hat{\boldsymbol{\sigma}}(j))}, \mathbf{r}_{<t}^{(j)}\right)\right] \tag{3}$$

Remarks. The resulting model is not the same as a simple routing of the models independently trained with the nonoverlapping subsets of these traces. Firstly, SSFT allows positive transfer in representation learning within $\{\mathbf{r}^{(j)}\}_{j=1}^{\mathrm{M}}$ even though they are matched to different $\{\mathrm{g}^{(i)}\}_{i=1}^{\mathrm{N}}$. Secondly, it is not optimal to distill reasoning traces from the same fixed sources for every question if the goal is to maximize both diversity and correctness. Our algorithm supports a variable number of target reasoning traces across training steps, with sources that may also change for each $\mathbf{x}$. Thirdly, even if the two sets of traces, $\{\mathbf{r}_a^{(j)}\}_{j=1}^{\mathrm{M}}$ for $\mathbf{x}_a$ and $\{\mathbf{r}_b^{(j)}\}_{j=1}^{\mathrm{M}}$ for $\mathbf{x}_b$, are from the same sources, their optimal configurations $\hat{\boldsymbol{\sigma}}_a$ and $\hat{\boldsymbol{\sigma}}_b$ can still vary because a teacher model can reason differently under different questions. Lastly, we reserve more global forking tokens than the maximum number of traces ($\mathrm{N} > \mathrm{M}$), and empirically observe that all the forking tokens are being matched throughout the process. This is because the extra forking tokens can maximally intra-differentiate similar traces.

2.3 Inference with Learned Global Forking Tokens

We discuss the inference protocols with N global forking tokens.

Inference Process (Cons@k). Our inference protocol with parallel test-time compute is to prompt ith response with `<think(i%N)>` and then do majority voting on their answers. Sharing the KV cache accelerates the generations.

Inference Process (Pass@1). When aggregation is not allowed, we use $\mathrm{g}^{(i)}$ that reasons with more flexibility. Inspired by enumerating dissimilar bipartite matchings to reveal node-level variation (Blumenthal et al., 2022), we choose the learned $\mathrm{g}^{(i)}$ with largest coverage. Note that this token emerges automatically thanks to SSFT, and the other emerging $\{\mathrm{g}^{(i)}\}_{i=1}^{\mathrm{N}}$ still contribute to representation learning and improving $\mathrm{Cons}@k$ with distinct reasoning modes. Concretely, each time an optimal matching $\hat{\boldsymbol{\sigma}} \in \{\boldsymbol{\sigma}_k\}_{k=1}^{\mathrm{P}}$ is computed, we increment a count $c(\boldsymbol{\sigma}_k)$; empirically, only a finite subset $\mathfrak{S}_p := \{\boldsymbol{\sigma}_k\}_{k=1}^{p} \subseteq \mathfrak{S}_{\mathrm{P}}$ continues to accumulate mass late in training, indicating the stable learned matchings. We then take the union of their edges, and select $\mathrm{g}^{(i^\star)}$ that matched to the largest number of distinct traces based on Equation 4 for Pass@1. Figure 2 shows an example of the matchings learned by aggregating all edges in $\mathfrak{S}_p$. More details about $\mathfrak{S}_p$[1] are in Appendix A.2.

$$i^\star = \arg\max_i \Big|\bigcup_{\boldsymbol{\sigma}\in\mathfrak{S}_p}\{j|\boldsymbol{\sigma}(j) = i\}\Big| \tag{4}$$

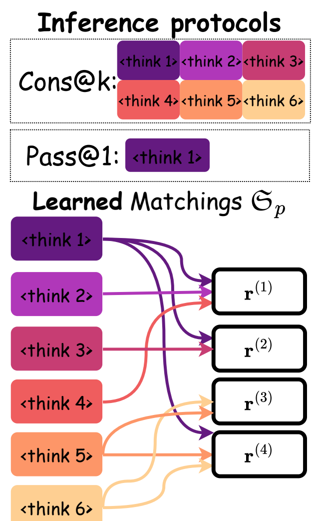


Figure 2: Learned matchings by SSFT-32B in Exp 3, obtained by connecting all edges in $\mathfrak{S}_p$. At test time, for Pass@1, we prompt with $\mathrm{g}^{(i^\star)}$ that has the most connected edges. For $\mathrm{Cons}@k$, we augment ith prompt by `<think(i % N)>`.

3 Experiments

We address the following research questions through experiments: **(RQ1)**: In terms of Pass@1 and $\mathrm{Cons}@k$ accuracy, how does a model trained with SSFT perform on reasoning benchmarks? **(RQ2)**: Does finding the optimal bipartite matching matter in reasoning performance? **(RQ3)**: Does training

[1]We choose the subscripts p and P to emphasize that $\mathfrak{S}_p$ is a subset of $\mathfrak{S}_{\mathrm{P}}$.

with diverse reasoning traces yield better accuracy and coverage under SSFT compared to standard SFT with temperature scaling? **(RQ4)**: Does prompting with distinct $\{\mathrm{g}^{(i)}\}_{i=1}^{\mathrm{N}}$ genuinely make a model generate diverse reasoning traces? **(RQ5)**: Is the performance gain from SSFT conditioned on the traces generated by our procedure and on a high-quality small dataset?

3.1 Experiment Setup

Training Dataset. We use the 1,000 questions from s1k dataset Muennighoff et al. (2025). In addition to the R1 (Guo et al., 2025) and Gemini Flash (Google, 2024) traces provided by s1, we also use Claude Opus 4.0/4.1 (Anthropic, 2025) and GPT-OSS-120B (Agarwal et al., 2025) with high and medium reasoning effort to obtain a pool of distilled targets for the 1,000 questions. For each question, we generate two traces per source to populate the pool. We then sample four traces from this pool. We call this s1k-4mixed-reasoning dataset.

Training Details. We fine-tune Qwen2.5-32B-Instruct (Yang et al., 2025a) for six epochs with a context length of 32,768. We reserve $\mathrm{N} = 6$ global forking tokens and use $\mathrm{M} = 4$ targets per question. To find the optimal bipartite matching for each input prompt, we consider only the first 1,000 tokens when computing the matching cost in Equation 2 for computational efficiency. We call this model SSFT-32B. We also include SSFT but choose a random bipartite matching at each step to fine-tune SSFT-32B (random $\boldsymbol{\sigma}$). Exact details on the pool of diverse distillation targets and selection procedure, as well as training hyperparameters, are provided in Appendix A.4.

Baselines. All of our baselines use Qwen2.5-32B-Instruct as their base model, and only train on the 1k questions. Our baselines include two groups: **(Single-Target △)** models trained with one trace per question and **(Multi-Target ★)** models trained with four traces per question. For **(Single-Target)**, we include s1.1-32B (Muennighoff et al., 2025), which uses 1k DeepSeek-R1 traces. We also fine-tuned an SFT-OSS-distill-32B baseline that trains only on the 1k GPT-OSS traces with high reasoning effort, as these traces achieved the highest correctness on the 1k questions based on an evaluation by Claude 3.5 Sonnet comparing each attempt against the reference answer. For **(Multi-Target)**, we use our s1k-4mixed-reasoning to fine-tune SFT-mixed-distill-32B using standard SFT with one `<think>` token, duplicating each question and treating the four traces as four individual data points. We also include Multiverse-32B (Yang et al., 2025b), which prompts Gemini 2.5-Pro (Google, 2025a) to transform 1k sequential CoTs into parallel CoTs as their training data.

Evaluation Setup. Our evaluation tasks consist of AIME24/AIME25 (Ye et al., 2025), MATH-500 (Hendrycks et al., 2021), and GPQA-Diamond (Rein et al., 2024). We use LightEval (Habib et al., 2023) as our evaluation framework with generation configurations: temperature=0.7 used in (Guha et al., 2025), top_p=0.95, max length=32768. For $\mathrm{Pass@1}$ accuracy without any parallel test-time compute, we select learned $\mathrm{g}^{(1)}$ for SSFT-32B and $\mathrm{g}^{(4)}$ for SSFT-32B (random $\boldsymbol{\sigma}$) based on Equation 4. For each $\mathrm{Pass@1}$ accuracy, we compute the average performance over 32 generations. For $\mathrm{Cons@6}$, which applies each of the six global forking tokens once in our method and uses six generations for the baselines, we compute the average over 11 sets of generations to reduce variance in the results. We refer to this as *Pass@1 of Native Cons@6* using a similar terminology as the concurrent work (Wen et al., 2025). Appendix A.5 presents an example of the parallel generations.

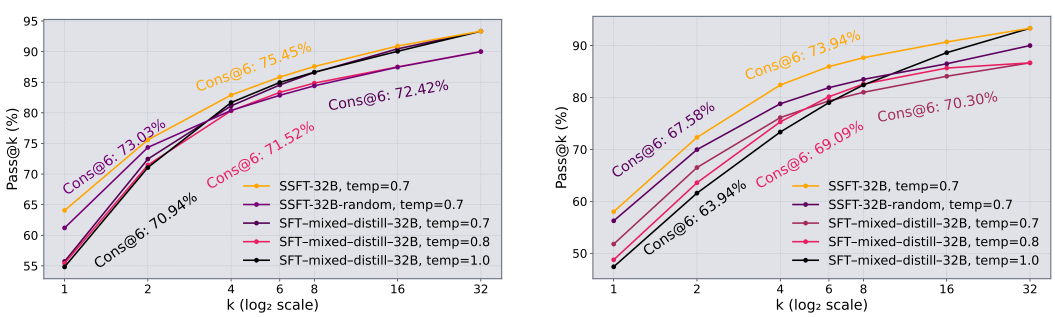
3.2 Evaluating SSFT on Reasoning Benchmarks

For **RQ1**, we see in Table 1 that SSFT delivers the best $\mathrm{Pass@1}$ accuracy, 64.06 on AIME24 and 58.13 on AIME25, outperforming SFT-mixed-32B by 8.33% and 6.57%, respectively. We also observe consistent improvements on all four tasks under parallel test-time compute at two scales, $\mathrm{Cons@6}$ and $\mathrm{Cons@32}$, over SFT-mixed-32B, which was trained on the same reasoning traces. Some notable results are $\mathrm{Cons@6} = 73.94\%, \mathrm{Cons@32} = 86.67\%$ on AIME25. To answer **RQ2**, we observe consistent improvements over SSFT-32B (random $\boldsymbol{\sigma}$), with especially strong gains at $\mathrm{Cons@6}$ on AIME25, where effectiveness with few parallel generations is critical. As shown later in Figures 4 and 5, optimal bipartite matching is essential for preventing collapsing reasoning modes. For **RQ3**, we compare our method against SFT-mixed-32B under various k in $\mathrm{Pass@}k$ accuracy with 32 generations to assess generation coverage. Figure 3 shows that SSFT achieves higher coverage across nearly all values of k. SFT-mixed-32B requires more allowed attempts and higher temperature to match the coverage of SSFT at the cost of lowering its $\mathrm{Pass@1}$ and $\mathrm{Cons@6}$ accuracy.

	AIME 2024	AIME 2025	MATH-500	GPQA-D	Average
Pass@1: Average performance of individual generations					
Qwen2.5-32B-Instruct △	15.80	10.40	80.40	47.00	38.40
s1.1-32B △	54.79	44.27	**92.16**	62.12	63.34
Multiverse-32B ★	53.80	45.80	91.80	60.70	63.03
SFT-OSS-distill-32B△	57.82	48.75	89.54	60.06	64.04
SFT-mixed-distill-32B★	55.73	51.56	88.36	57.50	63.29
SSFT-32B (random $\boldsymbol{\sigma}$) ★	61.77	55.10	89.95	**62.28**	67.28
SSFT-32B ★	**64.06**	**58.13**	90.02	60.39	**68.15**
Pass@1 of Native Cons@6: Average performance of majority voting with 6 parallel generations					
s1.1-32B △	70.30	53.33	95.60	61.45	70.17
SFT-OSS-distill-32B△	72.12	65.45	95.47	61.52	73.64
SFT-mixed-distill-32B★	72.42	70.91	92.10	57.32	73.19
SSFT-32B (random $\boldsymbol{\sigma}$) ★	73.03	67.58	95.67	61.87	74.54
SSFT-32B ★	**75.45**	**73.94**	**96.47**	**63.05**	**77.23**
Cons@32: Majority voting performance with large number of parallel generations					
s1.1-32B △	73.33	63.33	94.80	60.61	73.02
SFT-OSS-distill-32B△	76.66	73.33	96.00	61.60	76.90
SFT-mixed-distill-32B★	80.00	73.33	96.20	60.61	77.54
SSFT-32B (random $\boldsymbol{\sigma}$) ★	80.00	80.00	95.60	**62.63**	79.56
SSFT-32B ★	**83.33**	**86.67**	**96.80**	61.62	**82.11**

△ indicates training with single-target data and ★ indicates training with multi-target data.

Table 1: Performance of SSFT compared to baselines on four reasoning tasks, reported at Pass@1, Cons@6, and Cons@32. SSFT selects `<think1>` for Pass@1 and replaces 6 generations with 6 generations prompted by distinct `<thinki>` for Cons@k. We observe consistent improvements over (i) SFT-OSS-distill-32B, which uses the 1k OSS-high traces; (ii) SFT-mixed-distill-32B, which uses the four mixed traces but treats them as individual data; and (iii) SSFT-32B (random $\boldsymbol{\sigma}$), which trains using Equation 3 but with a randomly chosen $\boldsymbol{\sigma}$.



(a) AIME24

(b) AIME25

Figure 3: Coverage of SSFT compared to SFT-mixed-distill-32B with temperature scaling, reported at Pass@k. For convenience, we also report the Cons@6 accuracy next to each line. In AIME25, SFT-mixed-distill-32B needs to raise the inference temperature to 1 and use more attempts to match the coverage at the cost of lowering its Pass@1 and Cons@6 accuracy, further widening the gaps.

3.3 Evaluating Parallel Reasoning Diversity and Learned Matchings

Addressing **RQ4**, we show that our global forking tokens genuinely initiate distinct reasoning traces and offer **a new mechanism** for leveraging test-time compute.

Emerging Diverse Reasoning Modes. Using the Cons@6 results in Table 1, we form six sets of generations, each prompted by a distinct $\mathrm{g}^{(i)}$. For each set, we show the average accuracy and the distribution of thinking-token counts: Figure 4 for SSFT with optimal bipartite matching and Figure 5 for SSFT with random matching. **Reasoning length:** Length partially indicates the diversity in reasoning, and we see clear differences for SSFT with optimal matching, despite the absence of

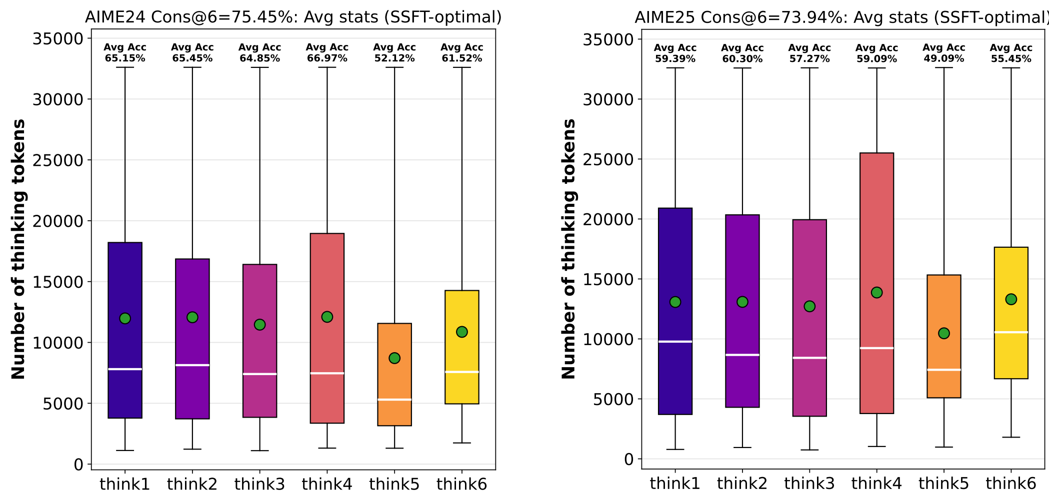


Figure 4: (SSFT, optimal matching). Distribution of thinking-token counts and average performance on AIME24 (left) and AIME25 (right) prompted by a distinct <think1>, . . . , <think6>.

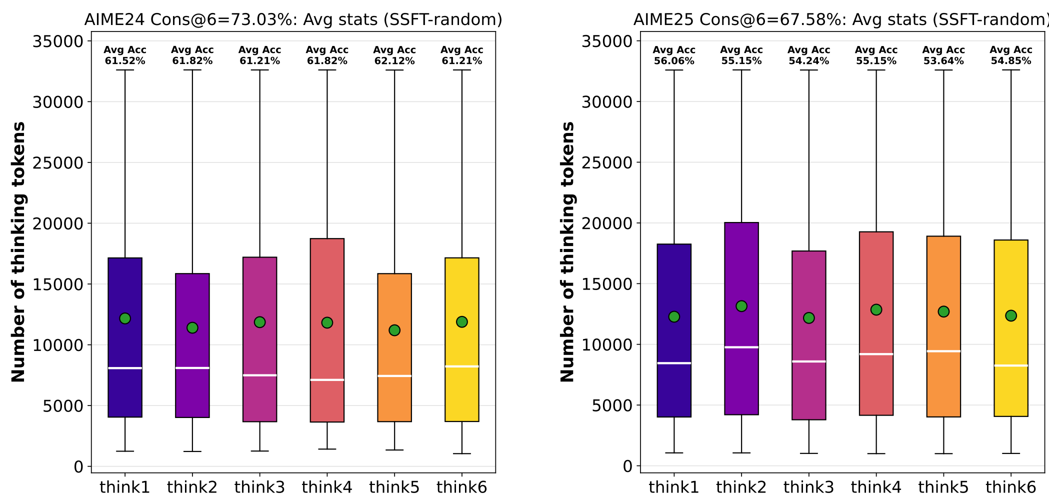
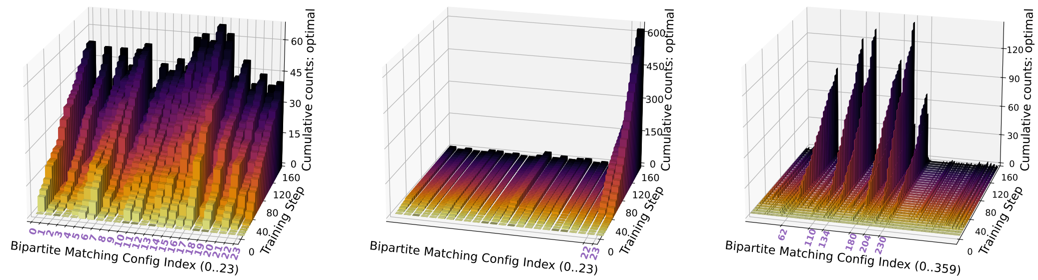


Figure 5: (SSFT, random matching). Distribution of thinking-token counts and average performance on AIME24 (left) and AIME25 (right) prompted by a distinct <think1>, . . . , <think6>.

hand-crafted matching rule or information about these traces. The consistency of these distributions across AIME24 and AIME25 indicates the differences is not from randomness, whereas randomly assigning a <thinki>, as in concurrent work (Wen et al., 2025), does not yield clear or consistent differences in reasoning length, as shown in Figure 5. **Performance:** After finetuning with random matching, prompting with a distinct <thinki> shows no meaningful impact ($\approx 61\%$ on AIME24 and $\approx 55\%$ on AIME25). With optimal matching, SSFT elicits distinct reasoning modes initiated by <think1>, . . . , <think4> that reach around 65% on AIME24 and $\geq 59\%$ on AIME25, with different lengths. Although <think5> and <think6> are weaker due to shorter reasoning modes, the average of these and, especially, Cons@6 performance improve consistently with them.

Visualization of Learned Matchings. We observe in Figure 2 that some $\mathrm{g}^{(i)} \in \boldsymbol{g}$ from SSFT-32B have unique configuration of matched edges with $\{\mathbf{r}^{(j)}\}_{j=1}^{4}$. This is a positive indication that $\{\mathrm{g}^{(i)}\}_{i=1}^{6}$ are likely to initiate distinct reasoning modes. We hypothesize that $\{\mathrm{g}^{(i)}\}_{i=1}^{\mathrm{N}}$ can still yield a unique edge-matching configuration even if a subset of $\{\mathbf{r}^{(j)}\}_{j=1}^{4}$ are difficult to distinguish. For interpretability, we fine-tune using the same four teacher models for each $\mathbf{x}$: GPT-OSS high, medium, R1, and Gemini. We call this dataset s1k-4teachers-reasoning dataset, and ask whether SSFT associate a unique $\mathrm{g}^{(i)}$ to the Gemini and R1 traces which have easily identifiable reasoning patterns, and then have the rest matched to OSS-high/OSS-medium traces in differ-



(a) SSFT under random matching ($\{\mathrm{g}^{(i)}\}_{i=1}^{4}, \{\mathbf{r}^{(j)}\}_{j=1}^{4}$). No correlations learned $\mathfrak{S}_p = \mathfrak{S}_{\mathrm{P}}$

(b) SSFT under optimal matching ($\{\mathrm{g}^{(i)}\}_{i=1}^{4}, \{\mathbf{r}^{(j)}\}_{j=1}^{4}$). $\mathfrak{S}_p = \{22, 23\}$

(c) SSFT under optimal matching ($\{\mathrm{g}^{(i)}\}_{i=1}^{6}, \{\mathbf{r}^{(j)}\}_{j=1}^{4}$). $\mathfrak{S}_p = \{62, 110, 134, 180, 204, 230\}$

Figure 6: Cumulative counts of $\boldsymbol{\sigma}_k \in \mathfrak{S}_{\mathrm{P}}$ computed as optimal over training. Note that $\mathfrak{S}_{\mathrm{P}}$ and $\mathfrak{S}_p$ are defined in Sections 2.1 and 2.3, respectively. Front axis: matching configuration index k. Depth: training step t. Bar height is the cumulative counts. These are the evolution of matchings during training 3 Qwen-32B-Instruct models under 3 bipartite matching settings. In this case study, the $\{\mathbf{r}^{(j)}\}_{j=1}^{4}$ are always (GPT-OSS-high, GPT-OSS-med, R1, Gemini) for each question. Note that *Random matching* method does not minimize Eqn 3 under optimal matching, but we track it. We observe $\mathfrak{S}_p = \mathfrak{S}_{\mathrm{P}}$ with random matching, meaning no correlations learned. But by optimizing Hungarian loss, we see the emergence of $\mathfrak{S}_p \subset \mathfrak{S}_{\mathrm{P}}$.

ent ways (i.e. only matched to OSS-high, only matched to OSS-med, and matched to both). We track source indices only for evaluation; the model still receives an unordered set of traces with no source information for each question. We study 3 bipartite matching settings for SSFT. Figure 6 shows the evolution of matchings learned under these 3 hyperparameters SSFT (a) random matching with four reserved $\mathrm{g}^{(i)}$, (b) optimal matching with four reserved $\mathrm{g}^{(i)}$, and (c) optimal matching with six reserved $\mathrm{g}^{(i)}$. Initially, all the configurations $\boldsymbol{\sigma} \in \mathfrak{S}_{\mathrm{P}}$ accumulate mass as there is no correlations between $\boldsymbol{g}$ sand $\{\mathbf{r}^{(j)}\}_{j=1}^{\mathrm{M}}$. Figure 6a shows SSFT under random matching does not shrink the size of configs computed as optimal, meaning that no correlations are learned between $\{\mathrm{g}^{(i)}\}_{i=1}^{4}$ and $\{\mathbf{r}^{(j)}\}_{j=1}^{4}$. By contrast, Figure 6b and Figure 6c show the emergence of only a strict subset of matching configurations in $\mathfrak{S}_{\mathrm{P}}$. This indicates some correlations between $\{\mathrm{g}^{(i)}\}_{i=1}^{\mathrm{N}}$ and $\{\mathbf{r}^{(j)}\}_{j=1}^{\mathrm{M}}$ are indeed learned through SSFT. We first visualize the learned matchings of the model with four $\mathrm{g}^{(i)}$ in Figure 7b. We observe that $\mathrm{g}^{(1)}$ and $\mathrm{g}^{(2)}$ are uniquely matched to the R1 and Gemini traces, showing that SSFT can indeed uniquely associate $\mathrm{g}^{(i)}$ to sufficiently diverse reasoning traces. Now to confirm our previous hypothesis, we see the unique learned matchings $(\mathrm{g}^{(3)}, (\text{OSS-high, OSS-med})), (\mathrm{g}^{(4)}, \text{OSS-med})$. Furthermore, by connecting all the edges in $\mathfrak{S}_p$ from SSFT with 6 forking tokens (Figure 6c), we also see unique learned matchings $(\mathrm{g}^{(3)}, \text{OSS-high}), (\mathrm{g}^{(4)}, \text{OSS-med}), (\mathrm{g}^{(5)}, (\text{OSS-med}, \text{OSS-high}))$ in Figure 7c. This confirms that the global forking tokens can identify unique correlations even among highly similar traces.

3.4 Ablation Study: Removing High Quality Small Dataset

To test whether our empirical gains are conditioned on the traces generated by our procedure and on a highly optimized small dataset (Muennighoff et al., 2025), we fine-tune on a public dataset that already provides sufficient reasoning traces per question: the 93k math set of Face (2025) (2-4 traces per question). Because this dataset has been successful for fine-tuning Qwen2.5-Math-7B, we adopt that base model and compare SSFT against SFT trained on all available traces. Details on the hyperparameters are in Appendix A.4.4. Addressing **RQ5**, Table 2 shows consistent improvements in both $\mathrm{Pass@1}$ and $\mathrm{Cons@32}$. The results indicate the SSFT is effective for larger sized public dataset with less diverse reasoning traces.

4 Related Work

Test-time Scaling. There has been a surge of work fine-tuning LLMs to reason longer, using reinforcement learning for frontier models (OpenAI, 2024; Shao et al., 2024; xAI, 2025; Yang et al.,

	AIME 2024		AIME 2025		MATH-500		GPQA-D	
Model	Pass@1	Cons@32	Pass@1	Cons@32	Pass@1	Cons@32	Pass@1	Cons@32
Qwen2.5-Math-7B-Instruct	10.42	20.00	9.48	23.33	81.87	87.40	30.29	30.30
SFT-OpenR1-93k-7B	46.15	66.67	34.17	50.00	86.62	90.20	46.35	47.98
SSFT-OpenR1-93k-7B	**51.25**	**73.33**	**35.52**	**56.66**	**89.74**	**93.60**	**46.86**	**48.90**

Table 2: Performance of SSFT versus SFT trained solely on publicly available distillation targets. The setup uses the 93k math questions from Face (2025) with Qwen2.5-Math-7B as the base model. SFT-OpenR1-93k-7B uses the same distillation targets as SFT-mixed-distill-32B in Table 1.

2025a) and supervised fine-tuning for smaller ones (Muennighoff et al., 2025; Hu et al., 2025). These methods enable LLMs to improve reasoning by allocating more test-time compute to sequential, iterative refinement such as self-reflection (Guo et al., 2025; Liu et al., 2025). However, extended sequential reasoning can be more sensitive to the order of reasoning steps and may result in failures (Chen et al., 2024b), and performance can start to degrade beyond a certain length due to "overthinking" (Ghosal et al., 2025). Our goal is to study the effective use of diverse reasoning traces to fine-tune small language models, essential for agentic AI (Belcak et al., 2025).

Parallel Reasoning. Parallel scaling methods such as self-consistency (Wang et al., 2022) and Best-of-N (Lightman et al., 2023) improve LLM performance by generating multiple reasoning paths in parallel and aggregating them. These methods fundamentally require choosing a temperature that can generate diverse reasoning paths, but a recent theoretical work shows that increasing temperature can sometimes fail to increase diversity if language models are not trained towards coverage (Verine et al., 2025). Other search-based methods such as Monte Carlo tree search (MCTS) (Zhang et al., 2024) and Tree of Thoughts (ToT) (Yao et al., 2023) apply heuristic-guided search with an external verifier to do more deliberate search to increase the coverage (Yao et al., 2023; Zhang et al., 2024). However, their dependence on heuristics and domain-specific knowledge can limit their applicable tasks. Regarding training LLMs with parallel reasoning traces, Yang et al. (2025b) proposes training with parallel CoTs decomposed from sequential CoTs, and our concurrent work Wen et al. (2025) proposes to train with multiple reasoning traces distilled from teacher models. These works show native parallel scaling can surpass sequential scaling within certain token limits. However, we aim to show that training on diverse distilled traces with our set language modeling loss enables the model to learn global forking tokens that trigger distinct reasoning modes, improving $\mathrm{Pass@1}$ and $\mathrm{Cons@}k$ over baselines fine-tuned on the same dataset, whether on subsets or the full set.

Set-based Global Loss in Deep Learning. DETR introduces end-to-end object detection with a set global loss (Carion et al., 2020; Minderer et al., 2022), whose success in parallel bounding-box prediction inspires our approach. We are the first to extend this to language modeling: while DETR predicts a set of tokens in parallel to match a list of bounding boxes, we predict a set of sequential reasoning paths initiated by global forking tokens and assess the matchings based on autoregressive losses. We adapt the set-based loss with autoregressive (AR) models, rather than diffusion-based models (Nie et al., 2025), because AR models achieve superior reasoning performance.

5 Conclusion

In this work, we demonstrate that diverse reasoning traces can be leveraged to learn global forking tokens which serve as prompts to initiate distinct reasoning modes that are both diverse and accurate. We proposed Set Supervised Fine-Turning (SSFT), which employs bipartite matching between reserved global forking tokens and diverse reasoning traces to compute a set-based language modeling loss. We show that models trained with SSFT yields improvements in both $\mathrm{Pass@}k$ and $\mathrm{Cons@}k$ accuracy compared to standard supervised fine-tuning (SFT) with temperature scaling. The proposed training method also improves the usage of multi-target data and avoids collapsing distinct reasoning modes, which is especially helpful in the context of distilling from multiple teacher models. For future work, we plan to further investigate how performance scales with a larger number of distillation targets and reserved forking tokens, which jointly determine the size of the underlying bipartite graph.

Reproducibility Statement. To ensure reproductibility of our work, we provide detailed hyperparameter settings in A.4.2, and SSFT algorithm implementations in A.3. We have also included the source code and data in the supplementary materials, which will be open-sourced upon acceptance.

REFERENCES

Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, et al. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025.

Anthropic. Claude 4.1 Opus. `https://claude.ai`, 2025. Generative AI Chatbot. Response to a prompt on *[Date of Access]*.

Peter Belcak, Greg Heinrich, Shizhe Diao, Yonggan Fu, Xin Dong, Saurav Muralidharan, Yingyan Celine Lin, and Pavlo Molchanov. Small language models are the future of agentic ai. *arXiv preprint arXiv:2506.02153*, 2025.

David B. Blumenthal, Sébastien Bougleux, Anton Dignös, and Johann Gamper. Enumerating dissimilar minimum cost perfect and error-correcting bipartite matchings for robust data matching. *Inf. Sci.*, 596:202–221, 2022. URL `https://doi.org/10.1016/j.ins.2022.03.017`.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pp. 213–229. Springer, 2020.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, et al. Do not think that much for 2+ 3=? on the overthinking of o1-like llms. *arXiv preprint arXiv:2412.21187*, 2024a.

Xinyun Chen, Ryan Andrew Chi, Xuezhi Wang, and Denny Zhou. Premise order matters in reasoning with large language models. In *Forty-first International Conference on Machine Learning*, 2024b. URL `https://openreview.net/forum?id=4zAHgkiCQg`.

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023.

Hugging Face. Open r1: A fully open reproduction of deepseek-r1, January 2025. URL `https://github.com/huggingface/open-r1`.

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

Google. Gemini 2.0 flash thinking mode (gemini-2.0-flash-thinking-exp-1219), December 2024. URL `https://cloud.google.com/vertex-ai/generative-ai/docs/thinking-mode`.

Google. Gemini 2.5 (gemini-2.5-pro-preview). `https://blog.google/technology/google-deepmind/gemini-model-thinking-updates-march-2025/`, 2025a. Accessed: 2025-04-22.

Google. Gemini 2.5: Our most intelligent ai model. `https://blog.google/technology/google-deepmind/gemini-model-thinking-updates-march-2025/#gemini-2-5-thinking`, March 2025b.

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

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

Nathan Habib, Clémentine Fourrier, Hynek Kydlíček, Thomas Wolf, and Lewis Tunstall. Lighteval: A lightweight framework for llm evaluation, 2023. URL https://github.com/huggingface/lighteval.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Pin-Lun Hsu, Yun Dai, Vignesh Kothapalli, Qingquan Song, Shao Tang, Siyu Zhu, Steven Shimizu, Shivam Sahni, Haowen Ning, and Yanning Chen. Liger kernel: Efficient triton kernels for llm training. *arXiv preprint arXiv:2410.10989*, 2024.

Xiao Hu, Xingyu Lu, Liyuan Mao, YiFan Zhang, Tianke Zhang, Bin Wen, Fan Yang, Tingting Gao, and Guorui Zhou. Why distillation can outperform zero-rl: The role of flexible reasoning. *arXiv preprint arXiv:2505.21067*, 2025.

H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955. doi: https://doi.org/10.1002/nav.3800020109. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/nav.3800020109.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023. URL https://arxiv.org/abs/2305.20050.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Matthias Minderer, Alexey Gritsenko, Austin Stone, Maxim Neumann, Dirk Weissenborn, Alexey Dosovitskiy, Aravindh Mahendran, Anurag Arnab, Mostafa Dehghani, Zhuoran Shen, et al. Simple open-vocabulary object detection. In *European conference on computer vision*, pp. 728–755. Springer, 2022.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*, 2025.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. *arXiv preprint arXiv:2502.09992*, 2025.

OpenAI. Learning to reason with llms, September 2024. URL https://openai.com/index/learning-to-reason-with-llms/.

OpenAI. Introducing openai o3 and o4-mini. https://openai.com/index/introducing-o3-and-o4-mini/, 2025.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019. URL https://api.semanticscholar.org/CorpusID:160025533.

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Alexandre Verine, Florian Le Bronnec, Kunhao Zheng, Alexandre Allauzen, Yann Chevaleyre, and benjamin negrevergne. Improving diversity in language models: When temperature fails, change the loss. In *Forty-second International Conference on Machine Learning*, 2025. URL `https://openreview.net/forum?id=RsyMfsqzeG`.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen, Jianxin Yang, Zhenru Zhang, et al. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. *arXiv preprint arXiv:2506.01939*, 2025.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Hao Wen, Yifan Su, Feifei Zhang, Yunxin Liu, Yunhao Liu, Ya-Qin Zhang, and Yuanchun Li. Parathinker: Native parallel thinking as a new paradigm to scale llm test-time compute. *arXiv preprint arXiv:2509.04475*, 2025.

xAI. Grok 3 beta the age of reasoning agents. `https://x.ai/news/grok-3`, February 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

Xinyu Yang, Yuwei An, Hongyi Liu, Tianqi Chen, and Beidi Chen. Multiverse: Your language models secretly decide how to parallelize and merge generation. *arXiv preprint arXiv:2506.09991*, 2025b.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

Yixin Ye, Yang Xiao, Tiantian Mi, and Pengfei Liu. Aime-preview: A rigorous and immediate evaluation framework for advanced mathematical reasoning. `https://github.com/GAIR-NLP/AIME-Preview`, 2025. GitHub repository.

Di Zhang, Xiaoshui Huang, Dongzhan Zhou, Yuqiang Li, and Wanli Ouyang. Accessing gpt-4 level mathematical olympiad solutions via monte carlo tree self-refine with llama-3 8b. *arXiv preprint arXiv:2406.07394*, 2024.

A APPENDIX

In this appendix, we provide details omitted from the main text.

A.1 LLM USAGE IN PAPER

We used ChatGPT (OpenAI, GPT-5) in September 2025 for occasional language polishing only. This is done when we really wanted to make sure there are no grammatical errors in a few sentences. No text, code, experiment results, or figures were generated by the LLM. We made our own hypothesis, completed all technical content on our own, and made our conclusions. We also verified all outputs by ourselves. The occasional grammar check is done by typing into LLM chatbox.

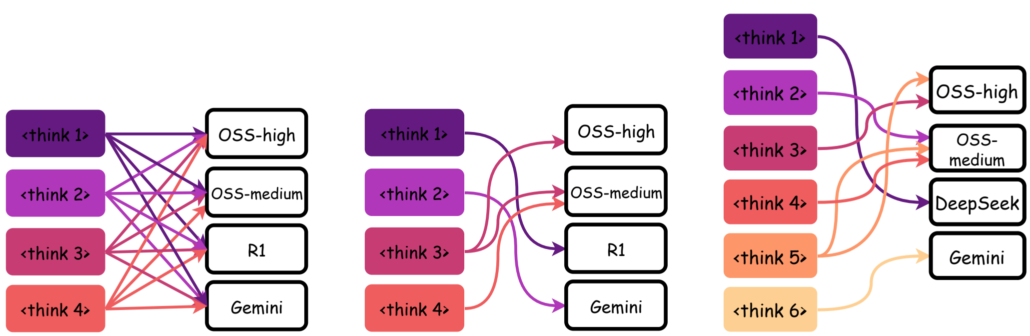
A.2 LEARNED MATCHINGS BETWEEN GLOBAL FORKING TOKENS AND TRACES

Initially, any of the bipartite matching configuration $\boldsymbol{\sigma}_k \in \mathfrak{S}_{\mathrm{P}}$ can be computed as optimal, as the reserved global forking tokens $\{\mathrm{g}^{(i)}\}_{i=1}^{\mathrm{N}}$ have no specific correlations with these traces $\{\mathbf{r}^{(j)}\}_{j=1}^{\mathrm{M}}$. This can be observed in Figures 6 and 7 that $c(\boldsymbol{\sigma}_k)$, the count of configuration $\boldsymbol{\sigma}_k$ being optimal during training, uniformly increases on all configuration indexes. However, as training goes with SSFT, we notice only a subset of $\mathfrak{S}_{\mathrm{P}}$ accumulates mass. This indicates there are some unique correlations learned between $\{\mathrm{g}^{(i)}\}_{i=1}^{\mathrm{N}}$ and $\{\mathbf{r}^{(j)}\}_{j=1}^{\mathrm{M}}$. This subset is denoted as $\mathfrak{S}_p = \{\boldsymbol{\sigma}_k\}_{k=1}^{p}$, and we call the unique edges in $\mathfrak{S}_p$ as *learned matchings*.

A.2.1 HOW TO CHOOSE THE GLOBAL FORKING TOKEN FOR PASS@1

To find $\mathfrak{S}_p$, we can simply track which configurations $\boldsymbol{\sigma}$ still accumulate mass in the last epoch. Then we can connect all the unique edges in $\mathfrak{S}_p$ to visualize learned matchings. However, multiple global forking tokens may share the maximum number of connected edges in the learned matchings. To break the tie, we treat the counts as edge weights and select $\mathrm{g}^{(i)}$ with the largest weighted degree. We provide this implementation in our code.

A.2.2 MORE VISUALIZATIONS ON LEARNED MATCHINGS



(a) Visualization of learned matchings after SSFT with random matching. The fully connected graph shows no correlations were learned

(b) Visualization of learned matchings after SSFT with optimal matching between $\{\mathrm{g}^{(i)}\}_{i=1}^{4}$ and $\{\mathbf{r}^{(j)}\}_{j=1}^{4}$

(c) Visualization of learned matchings after SSFT with optimal matching between $\{\mathrm{g}^{(i)}\}_{i=1}^{6}$ and $\{\mathbf{r}^{(j)}\}_{j=1}^{4}$

Figure 7: These learned matching visualizations are obtained by connecting edges in the subset of configurations $\{\boldsymbol{\sigma}_k\}_{k=1}^{p}$ that still accumulate mass towards the end of training in Figure 6. These models are fine-tuned using the same GPT-OSS-high, GPT-OSS-medium, R1, and Gemini traces for each question, so we can better interpret the learned matchings

A.3 ALGORITHM: SSFT IMPLEMENTATION

Algorithm 1 presents the core SSFT implementation with optimal bipartite matching. **In practice, the nested-loop computation used to populate C is fully vectorized and can be executed in a**

single forward pass. This does not blow up VRAM because (i) we do not store activations for these cost evaluations (no backprop through matching costs), and (ii) We only need to use the first $\mathrm{T_L} < \mathrm{T_r}$ NTP losses to compute the matching cost, as the NTP loss over the first few thousand tokens can already differentiate many reasoning traces in terms of their modes. Nevertheless, our code also supports matching over the full $\mathrm{T_r}$ tokens by chunking the computation into a few batches, so this step does not become a VRAM bottleneck. Fine-tuning on 1k questions with 4 traces each, SSFT (optimal matching) took 6.5 h for 6 epochs, compared to 6.1 h for standard SFT, adding only a small overhead.

The primary VRAM bottleneck in SSFT remains the backpropagation Step 8, regardless of whether we use optimal or random matching, because the effective batch size scales with M. To mitigate this, we split the backward pass into several gradient-accumulation steps. Although our experiments use the same number of reasoning traces per question, we also support variable number of targets using our queue-based batching in Algorithm 2. The complication arises when using distributed training with a variable-sized batch, as different processes require the same per-device batch size to perform collective operations. This is mitigated by padding with PAD" sequences to align batch sizes. Our implementation minimizes the number of "PAD" sequences by storing a variable number of targets in a queue and dequeuing multiple items to form a per-device global batch, so smaller batches can be stitched together instead of always being padded.

A.4 Training details

A.4.1 Training Datasets

32B experiments with questions from s1(Main): We explain the process of generating our training dataset for experiments in Table 1, Figure 3, Figure 4, Figure 5, Figure 6. First, we use the 1000 questions from s1 (Muennighoff et al., 2025) and populate a pool of reasoning traces by distilling from GPT-OSS-120B-high reasoning, GPT-OSS-120B-medium reasoning, DeepSeek R1, Gemini Flash2.0 Thinking, and Claude4/4.1. We use temperature 1.0, maximum length of 32768, and sets high reasoning effort unless specified. We generate two traces per teacher model. We use Claude3.5 Sonnet to extract the answer from the distilled solutions and compare with the ground-truth answer. The correctness of these distilled traces are shown in 3.

For s1k-4mixed-reasoning dataset, we sample 4 traces per question from this pool, so the dataset consists of 1000 questions, each paired with 4 reasoning traces. This dataset was used to fine-tune **SSFT-32B**, **SSFT-32B (random σ)**, and **SFT-mixed-distill-32B**.

For fine-tuning **SFT-OSS-distill-32B** model, we only use the 1000 traces from "Run1" GPT-OSS-120B with high reasoning effort.

For obtaining the visualizations in Figure 6 and Figure 7, we fine-tune models using the same teacher models for all 1000 questions. we always choose the 4 traces from "Run1" of GPT-OSS-120B-high, GPT-OSS-120B-medium, DeepSeek R1, and Gemini Flash2.0 Thinking. As mentioned in Section 3.3 and Figure 6, we fine-tune under 3 bipartite matching hyperparameters to conduct this case study.

	GPT-OSS-120B-high	GPT-OSS-120B-medium	DeepSeekR1	Gemini Flash2.0 Thinking	Claude Opus4/4.1
Run1	796/1000	769/1000	620/1000	538/1000	656/1000
Run2	785/1000	753/1000	641/1000	545/1000	647/1000

Table 3: The number of correct reasoning traces distilled for the 1,000 questions in s1 by different teacher models. This evaluation is done by Sonnet comparing the predictions and the ground-truth answers. We see that GPT-OSS has the highest accuracy for s1 dataset.

Algorithm 1 Set Supervised Fine-tuning (SSFT)

Require:
- $\pi_{\boldsymbol{\theta}}$: base model
- N: Number of global forking tokens $\{\mathrm{g}^{(i)}\}_{i=1}^{\mathrm{N}}$
- $\mathcal{D}$: Dataset with (at most) M reasoning traces per question
- B: Global batch size
- $\mathrm{T_L}$: The first $\mathrm{T_L}$ number of tokens to match in matching cost (Equation 5).

Ensure: Output $\pi_{\boldsymbol{\theta}}$

for each training step **do**
  for $k = 1, ..., \mathrm{B}$ **do**
    Sample an input prompt and the corresponding reasoning traces $(\mathbf{x}_k, \{\mathbf{r}_k^{(j)}\}_{j=1}^{\mathrm{M}}) \sim \mathcal{D}$
    Initialize cost matrix $\mathrm{C} \in \mathbb{R}^{\mathrm{N} \times \mathrm{M}}$
    for $i = 1, ..., \mathrm{N}$ **do**
      for $j = 1, ..., \mathrm{M}$ **do**
        Compute the matching cost between $\mathrm{g}^{(i)}$ and $\mathbf{r}_k^{(j)}$ by Equation 5.

$$\mathcal{L}_{\text{matching}}\left(\mathrm{g}^{(i)}, \mathbf{r}_k^{(j)}\right) = -\mathrm{sg}\left(\frac{1}{\mathrm{T_L}} \sum_{t=1}^{\mathrm{T_L}} \log \pi_{\boldsymbol{\theta}} \left(\mathrm{r}_{k,t}^{(j)} | \mathbf{x}_k, \mathrm{g}^{(i)}, \mathbf{r}_{k,<t}^{(j)}\right)\right) \tag{5}$$

        Store the matching cost in C.

$$\mathrm{C}(i,j) = \mathcal{L}_{\text{matching}}\left(\mathrm{g}^{(i)}, \mathbf{r}_k^{(j)}\right) \tag{6}$$

      end for
    end for
    Compute optimal matching $\hat{\boldsymbol{\sigma}}_k$ between $\{\mathrm{g}^{(i)}\}_{i=1}^{\mathrm{N}}$ and $\{\mathbf{r}_k^{(j)}\}_{j=1}^{\mathrm{M}}$. Hungarian algorithm (Kuhn, 1955) can be applied to C to efficiently compute Equation 7 (Equation 1).

$$\hat{\boldsymbol{\sigma}}_k = \underset{\boldsymbol{\sigma} \in \mathfrak{S}_{\mathrm{P}}}{\arg\min} \sum_{j=1}^{\mathrm{M}} \mathrm{C}(\boldsymbol{\sigma}(j), j) \tag{7}$$

  end for
  Compute the empirical *set language modeling loss* (Equation 8):

$$\mathcal{L}_{\text{Hungarian}}(\boldsymbol{\theta}) = -\frac{1}{\mathrm{B}} \sum_{k=1}^{\mathrm{B}} \left[\sum_{j=1}^{\mathrm{M}} \sum_{t=1}^{\mathrm{T}_{\mathbf{r}}} \log \pi_{\boldsymbol{\theta}} \left(\mathrm{r}_{k,t}^{(j)} | \mathbf{x}_k, \mathrm{g}^{(\hat{\boldsymbol{\sigma}}_k(j))}, \mathbf{r}_{k,<t}^{(j)}\right)\right] \tag{8}$$

  Update model parameters $\boldsymbol{\theta}$ using gradients $\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\text{Hungarian}}(\boldsymbol{\theta})$
end for

Algorithm 2 Queue-based Distributed SSFT with variable number of traces for each question

Require:
- $\pi_{\boldsymbol{\theta}}$: base model
- N: Number of global forking tokens $\{\mathrm{g}^{(i)}\}_{i=1}^{\mathrm{N}}$
- $\mathcal{D}$: Dataset with (at most) M reasoning traces per question, variable m number of traces per question
- B: Global batch size
- $\mathrm{T_L}$: The first $\mathrm{T_L}$ number of tokens to match in matching cost (Equation 5).
- b: Original per-device global batch size ($b \geq \mathrm{M}$), This is "micro batch size*original grad accumulation steps"

Ensure: Output $\pi_{\boldsymbol{\theta}}$

1: **for** each epoch **do**
2: Initialize Queue Q for storing a sequence of $(\mathbf{x}_k, \{\mathbf{r}_k^{(j)}\}_{j=1}^{m})$ where m is a variable number that differs between input questions and different processes (GPUs)
3: Initialize Queue q for storing a sequence of sizes of sets m.
4: **for** every $(\mathbf{x}_k, \{\mathbf{r}_k^{(j)}\}_{j=1}^{m}) \in \mathcal{D}$ **do**
5: $Q \leftarrow Q.enqueue((\mathbf{x}_k, \{\mathbf{r}_k^{(j)}\}_{j=1}^{m}))$
6: $q \leftarrow q.enqueue(m)$
7: $all\_q\_list = All\text{-}gather(q)$
8: Initialize list $temp\_batch$
9: **while** All processes have at least b sequences based on all_q_list **do**
10: **while** $temp\_batch$ does not have at least b sequences **do**
11: $temp\_batch \leftarrow temp\_batch.append(Q.dequeue())$
12: q.deque()
13: **end while**
14: compute the the maximum per_device global batch size b_{max} currently in all processes using all_q_list (inferred, no collective operation)
15: Pad $temp\_batch$ to size b_{max} by appending "pad sequences" as needed.
16: Update all entries in all_q_list based on inferred usage
17: Perform one SSFT training step, SSFT $(\pi_{\boldsymbol{\theta}}, temp\_batch, b_{max})$
18: **end while**
19: **end for**
20: **end for**

A.4.2 Training Hyperparameters

For consistency, we use Qwen2.5-32B-Instruct (Yang et al., 2025a) as the base model for all of our 32B experiments. We use standard fine-tuning hyperparameters: we train for 6 epochs with a global batch size of 32, which is derived from 4 gradient accumulation steps and distributed training with 8 GPUS ($4 \times 8 = 32$). This results in 756 gradient steps. The maximum sequence length is set to 32,768. We train with bfloat16 with a learning rate of $1e-5$ warmed up linearly for 5% and then decayed to 0 using a cosine schedule. We choose AdamW optimizer (Loshchilov & Hutter, 2017) with $\beta_1 = 0.9$, $\beta_2 = 0.95$, and weight decay $1e-4$. We only backpropagates the completion loss, which is the loss on reasoning traces and the answers. Fine-tuning **SSFT-32B** plus loggings took 6.5 hours on 8 NVIDIA B200 GPUs using PyTorch FSDP, Liger Kernel (Hsu et al., 2024) for fused cross entropy loss, and FlashAttention-2 (Dao, 2023) for fused attention computation. Fine-tuning **SSFT-32B (random $\boldsymbol{\sigma}$)** took 6.3 hours, and Fine-tuning **SFT-mixed-distill-32B** took 6.1 hours. Even our baseline **SFT-OSS-distill-32B** with only one trace per question, and our attempt to reproduce s1.1 took 1.66 hours, which is longer than the time reported by Muennighoff et al. (2025). This is due to using 8 GPUs instead of 16 GPUs, hardware and package differences. When training with s1k-4mixed-reasoning, we added one extra epoch from 5 epochs to 6 epochs, since we have 4x reasoning traces, but we did not linearly increase the number of epochs, as these traces can be similar, and the number of distinct questions is still 1,000. Overall, we made sure all of our models are fine-tuned with consistent hyperapameters.

A.4.3 Visualization of SSFT Training Dynamics

Figure 8 shows the standard training dynamics of SSFT with optimal bipartite matching. The resulting model is SSFT-32B. The loss plotted here is Equation 3.

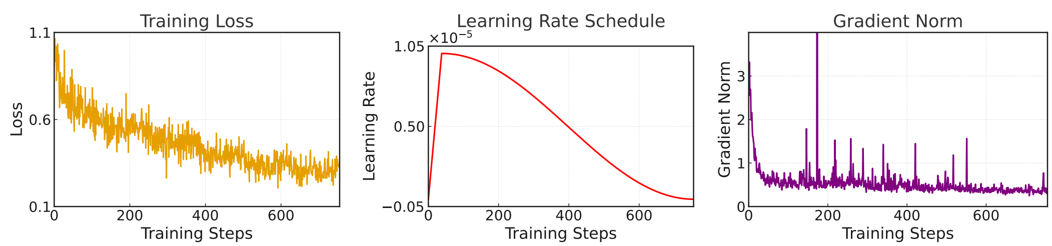
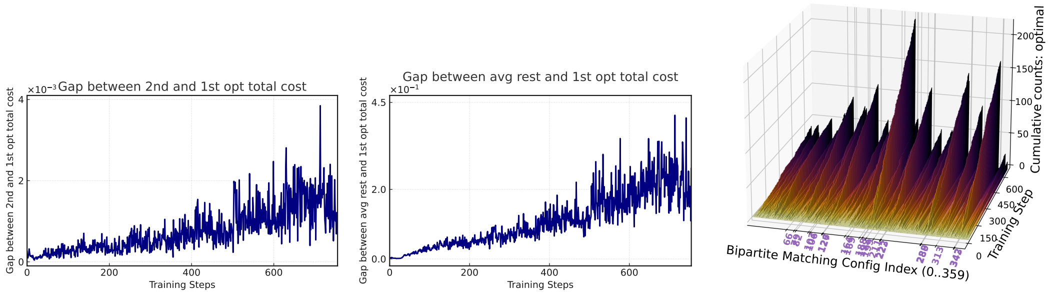


Figure 8: Training dynamics of SSFT-32B on s1k-4mixed-reasoning

Figure 9 shows the evolution of bipartite matching during SSFT. Figures 9a and 9b show that the gap between optimal bipartite matching cost and non-optimal bipartite matching cost under other $\boldsymbol{\sigma}$ keeps widening during training. This means that these reasoning traces are indeed starting to match unique global forking tokens. Even though SSFT effectively optimizes a non-stationary objective which depends on model parameters $\boldsymbol{\theta}$, the widening gap shows the inner discrete optimization is converging as training goes. Figure 9c also confirms that the model learned some unique correlations between $\{\mathrm{g}^{(i)}\}_{i=1}^{6}$ and $\{\mathbf{r}^{(j)}\}_{j=1}^{4}$, as only a subset of matchings are still computed as optimal. Compared to Figure 6, we see more $\boldsymbol{\sigma}$ accumulating mass towards the end. This is due to having more mixed diverse reasoning traces, so the model learned more intricate associations between these global forking tokens and truly diverse reasoning traces.

A.4.4 Ablation Study Training details (Removing High Quality Small Dataset)

For this ablation study, we choose Open-R1-Math220k default split, which has 93,000 math questions and $2 \sim 4$ traces. Since Qwen2.5-Math-7B is a widely fine-tuned model using this dataset, we also choose it as our base model. We train for 3 epochs using 8 A100 GPUs, which took around 4 days. Our hyperparameters are mostly consistent with the recommended hyperparameters by Face (2025). We fine-tune both SSFT-OpenR1-93k-7B and SFT-OpenR1-93k-7B with a maximum length of 32768, learning rate of $4.0e-05$ warmed up linearly for 3% and decayed to 0 following cosine schedule, 8 gradient accumulation steps. For our SSFT method, we reserve $\mathrm{N} = 4$ global forking



(a) The gap between the optimal bipartite matching cost in Equation 1 and second smallest bipartite matching cost. The increase shows the global forking tokens start preferring a specific matching.

(b) The gap between the optimal matching cost in Equation 1 and the average of bipartite matching cost under other $\boldsymbol{\sigma}$. The increase shows the global forking tokens start preferring a specific matching.

(c) Cumulative counts of $\boldsymbol{\sigma}_k$ computed as optimal over training. We observe that only a subset $\mathfrak{S}_p$ accumulates mass towards the end, indicating some unique correlations learned between $\{\mathrm{g}^{(i)}\}_{i=1}^{6}$ and $\{\mathbf{r}^{(j)}\}_{j=1}^{4}$

Figure 9: Dynamics of bipartite matching during the fine-tuning of SSFT-32B on s1k-4mixed-reasoning.

tokens, and use the first 1000 tokens for matching. Again, only the completion loss is used for optimizing the model parameters.

A.5 Examples of SSFT-32B Reasoning Traces initiated by Different Global Forking Tokens

We randomly picked a set of six parallel generations from those where SSFT-32B correctly answered a problem from AIME 2025 under all six global forking tokens. This is question 11.

Problem 11 (AIME25)

A piecewise linear periodic function is defined by

$$f(x) = \begin{cases} x, & \text{if } x \in [-1, 1), \\ 2 - x, & \text{if } x \in [1, 3), \end{cases} \qquad \text{and} \qquad f(x+4) = f(x) \;\text{ for all real } x.$$

The graph of $f(x)$ has a sawtooth pattern. The parabola $x = 34y^2$ intersects the graph of $f(x)$ at finitely many points. The sum of the y-coordinates of these intersection points can be expressed in the form

$$\frac{a + b\sqrt{c}}{d},$$

where a, b, c, d are positive integers, $\gcd(a, b, d) = 1$, and c is squarefree. *Find* $a+b+c+d$.

Here are the 6 parallel generations by SSFT-32B. We see distinct reasoning traces produced by prompting with a unique $\mathrm{g}^{(i)} \in \{\mathrm{g}^{(i)}\}_{i=1}^{6}$. The differences are highlighted in red comments.

`<think1>` — Periodization + Quadratic Families + Vieta/Telescoping

`<|im_start|>assistant`
`<|im_start|>`
<think1>
"write $x = t + 4k$ with $t \in [-1, 3)$" ... ... "Case 1 ($t \in [-1, 1)$, $y = t$) ... Case 2 ($t \in [1, 3)$, $y = 2 - t$)" ... ... $34t^2 - t - 4k = 0$ and $t = \dfrac{1 \pm \sqrt{1 + 544k}}{68}$ ... ... "let

$u = 2 - t$" and $u = \dfrac{-1 \pm \sqrt{273+544k}}{68}$ ... ... *Thus* $u = \dfrac{-1+5\sqrt{185}}{68}$. ... ... "telescopes to rational constant $9/34$" ... ... *([Comment:]* periodize $x = t + 4k$, solve two quadratic families, use Vieta pair-sums; edge-period gives the irrational y)
</think1>
`<|im_start|>answer`
Answer: .... $\boxed{259}$

`<think2>` — Nearest-Multiple-of-4 Reduction

`<|im_start|>assistant`
`<|im_start|>`
<think2>
Start from $y = f(34y^2) \ldots$ ... write $34y^2 = 4k + r$ with $r \in [-1, 3)$ (choose $k = \lfloor \frac{34y^2+1}{4} \rfloor$) ... ... Branch A ($r \in [-1, 1)$): $y = r \Rightarrow 34y^2 - y - 4k = 0 \ldots$ ... Branch B ($r \in [1, 3)$): $y = 2 - r \Rightarrow 34y^2 + y - (2 + 4k) = 0 \ldots$ ... Vieta: pair-sums $+\frac{1}{34}$ and $-\frac{1}{34}$ cancel over full periods ... ... only the edge/partial period remains ($k = 8$, descending branch) ... ... $y = \dfrac{-1+\sqrt{273+544\cdot 8}}{68} = \dfrac{-1+5\sqrt{185}}{68} \ldots \ldots$
([Comment:] reduce to $r \in [-1, 3)$ via nearest multiple of 4; full periods cancel by $\pm\frac{1}{34}$ pair-sums; only the boundary $k = 8$ root contributes)
</think2>
`<|im_start|>answer`
Answer: .... $\boxed{259}$

`<think3>` — Paper-Style Sections; Explicit Pair-Sums and Totals

`<|im_start|>assistant`
`<|im_start|>`
<think3>
"Now we need sum of all y-coordinates" ... ... $\sum_{n=0}^{8} y_{1,n} + \sum_{n=0}^{7} y_{C,n}$ ... ... $y = \dfrac{-1+\sqrt{4625}}{68}$
... ... "Therefore $S_2 = \dfrac{5\sqrt{185}-17}{68}$" ... ... "Total sum $S \ldots$" ... ... *([Comment:]* formal write-up; separates sums by family, explicitly displays the boundary $y = \frac{-1+\sqrt{4625}}{68}$, then totals)
</think3>
`<|im_start|>answer`
Answer: .... $\boxed{259}$

`<think4>` — Set/System Framing Before Algebra

`<|im_start|>assistant`
`<|im_start|>`
<think4>
"the set $\{(x, f(x))\}$ ... Intersecting it with the parabola $x = 34y^2$ requires solving $y = f(x)$, $x = 34y^2$." ... ... (then proceeds to the same two-case algebra) ... ... $u_+ = \dfrac{-1+\sqrt{273+544\cdot 8}}{68} = \dfrac{-1+5\sqrt{185}}{68}$. ... ... *([Comment:]* definition-first framing; after setting up the system, it reaches the same boundary u_+ value on the partial period)
</think4>

```
<|im_start|>answer
```
Answer: .... $\boxed{259}$

`<think5>` — Outline & Symmetry: Full Periods Cancel, Last Period Contributes

```
<|im_start|>assistant
<|im_start|>
```
<think5>
"the sum is zero … That's a key insight." ... ... "for each full period ($k = 0$ to $k = 7$), the sum … is zero … now consider the partial period $k = 8$" ... ... *Compute* $u_{pos}(k{=}8) = \frac{-1+\sqrt{4625}}{68}$. ... ... (term-by-term cancellation examples for $k = 0, 1$) ... ... (*[Comment:]* symmetry explains full-period cancellation; the lone boundary contribution at $k = 8$ supplies the irrational term)
</think5>
```
<|im_start|>answer
```
Answer: .... $\boxed{259}$

`<think6>` — Stream-of-Consciousness Scratchwork

```
<|im_start|>assistant
<|im_start|>
```
<think6>
"Okay, so I need to solve this problem … Let me try to break this down step by step." ... ... "Wait, actually, no. Wait," ... ... $y_{A,8} = \frac{-1+\sqrt{4625}}{68}$. ... ... (*[Comment:]* exploratory self-checks; reaches the same boundary y on the partial period)
</think6>
```
<|im_start|>answer
```
Answer: .... $\boxed{259}$

A.6 Alternative Implementation of Training with Multi-Target Sequences

Figure 10 shows an alternative implementation for training with multiple target parallel sequences. Instead of flattening multi-target data along the batch dimension, we can also concatenate parallel reasoning traces along the sequence dimension, modify the causal attention matrix, and position ids for positional embeddings. While easy to implement, this version extends the sequence length and cannot perform gradient accumulation along the batch dimension to maintain the same VRAM as training with a single reasoning trace. Since most fine-tuning already sets the micro-batch size to 1 due to the increased number of reasoning tokens already present in a single reasoning path, this implementation also cannot further shrink the micro-batch size to accommodate the extended context length. In addition, the code that uses the initial flash-attention implementation, which requires causal attention matrix, is not compatible with this setup. In terms of choosing the number of sequences in a global batch size, this implementation is also less flexible.

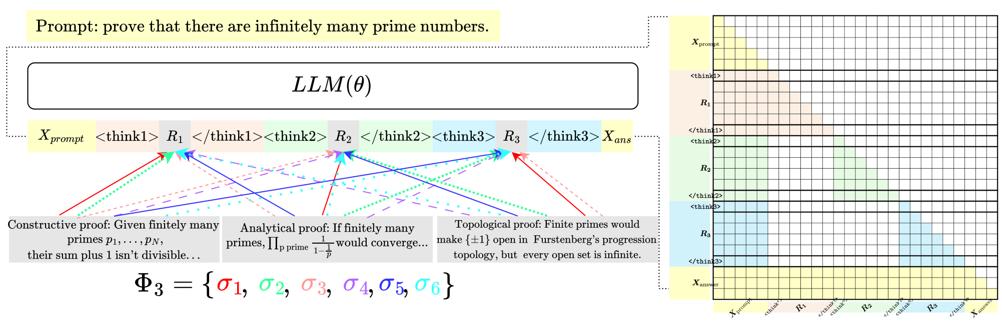


Figure 10: An illustration of training with multiple parallel targets along the sequence dimension.