REASONING COMPRESSION WITH MIXED-POLICY DISTILLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Reasoning-centric large language models (LLMs) achieve strong performance by generating intermediate reasoning trajectories, but often incur excessive token usage and high inference-time decoding cost. We observe that, when solving the same problems, larger reasoning models can often produce more concise traces, whereas smaller reasoning models tend to generate longer and more redundant trajectories. This is especially problematic in real-world deployment, where memory, latency, and serving-cost constraints often favor smaller models. Our observations suggest that reasoning compression can be transferred from large models to small ones rather than enforced through explicit length constraints. Based on this insight, we propose Mixed-Policy Distillation (MPD), a reasoning compression framework that transfers concise reasoning behavior from a larger-sized teacher to a smaller student by distilling teacher-compressed student trajectories. Unlike on-policy distillation, which aligns the student with teacher distributions over verbose student trajectories, or off-policy distillation, which relies on teacher-generated trajectories and may suffer from distribution mismatch, MPD combines the strengths of both. Given a student-sampled trajectory, the teacher rewrites it into a more concise reasoning trace, and the student is trained via KL-based alignment on the compressed trajectory. This preserves student-policy exploration while injecting teacher-guided compression. Experiments on Qwen3-1.7B show that MPD reduces token usage by up to 27.1% while improving performance across multiple reasoning benchmarks, demonstrating an effective approach to efficient small-model reasoning.

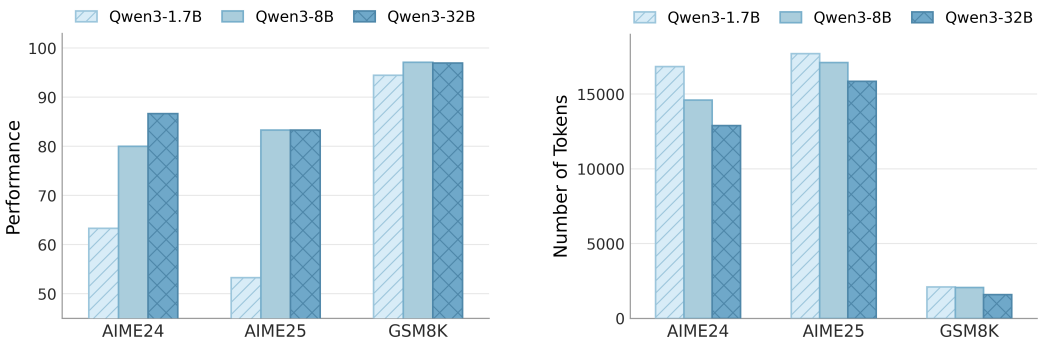
1 INTRODUCTION

LLMs exhibit improved reasoning performance when equipped with explicit chain-of-thought (CoT) traces (Wei et al., 2022; Kojima et al., 2022). Recent reasoning-centric models, such as OpenAI o1, DeepSeek-R1(DeepSeek-AI, 2025), and newer systems like GPT-5.5 and DeepSeek-V4, push this paradigm further by generating intermediate reasoning trajectories that often span tens of thousands of tokens per query. While such extended reasoning improves accuracy, it also incurs substantial inference-time decoding cost, since computation scales with the length of the generated trace. This motivates the need for new paradigms that improve reasoning efficiency by reducing token usage, i.e., through reasoning compression.

Reasoning inefficiency is especially pronounced in smaller models. We observe that, when solving the same reasoning problems, smaller reasoning models tend to produce longer traces with more redundant intermediate steps, whereas larger models often reach correct answers with substantially fewer tokens. For example, Qwen3-32B reduces token usage by around 23% compared with Qwen3-1.7B on the same problems (Fig. 1). This creates a practical tension: real-world deployment often favors smaller models because of memory, latency, and serving-cost constraints, yet these are precisely the models that suffer most from redundant reasoning. At the same time, the concision of larger models suggests a natural source of supervision: concise reasoning behavior can be transferred from a larger model to a smaller one, rather than imposed through explicit length penalties.

Existing reasoning compression methods mainly fall into three categories. Training-free methods modify prompts or decoding strategies to elicit shorter traces (Sui et al., 2025; Wang et al., 2025; Renze & Guven, 2024; Xu et al., 2025), but are prompt-sensitive and provide only shallow control over reasoning length. SFT- and RL-based methods directly optimise models for concise reasoning:

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) Performance.

(b) Token Consumption.

Figure 1: Reasoning performance (a) and token usage (b) across models of different sizes from the same family. Evaluated on GSM8K, AIME24, and AIME25 using Pass@4 as the evaluation metric.

SFT trains on externally shortened traces (Kang et al., 2025; Xia et al., 2025; Wu et al., 2025), creating a train-inference discrepancy because the student does not learn from its own reasoning behaviour, while RL (Jin et al., 2025; Shao et al., 2024; Aggarwal & Welleck, 2025; Hou et al., 2026) penalises length through reward design, often requiring ground-truth answers and risking suppression of useful exploration. More recently, CRISP (Sang et al., 2026) applies on-policy self-distillation to train on the student's own generations and avoid distribution shift. However, it relies on a prompted variant of the student itself rather than actual generations from a stronger teacher, limiting compression quality.

In this paper, we propose **Mixed-Policy Distillation** (MPD), a framework that combines on-policy student generation with external-teacher compression. Given a reasoning trajectory generated by the student, a larger-sized teacher rewrites it into a more concise version, and the student is trained to align with the rewritten trajectory via token-level KL divergence. This creates a mixed-policy signal: the trajectory structure comes from the student, while the compressed content comes from the teacher. Unlike previous methods, MPD preserves the student's own problem-solving strategy while using the teacher to compress and refine that strategy into a more concise reasoning trace. MPD is designed for real-world deployment settings, where smaller models are often preferred, and reasoning compression is most needed for these less capable yet more practical models.

We summarize our contributions as follows:

- We propose Mixed-Policy Distillation, a reasoning compression framework that aligns a student with teacher-compressed versions of its own reasoning trajectories, preserving the student's problem-solving strategy while using a larger-sized teacher to enable concise reasoning.
- We instantiate MPD on OpenMathInstruct-2 (Toshniwal et al., 2025) with Qwen3-1.7B (Team, 2025) as the student and Qwen3-8B as the teacher, training without ground-truth supervision, reward models, or length penalties. On in-domain mathematical reasoning, MPD reduces inference tokens by 27.1% on Qwen3-1.7B while maintaining or improving accuracy.
- We further evaluate MPD on out-of-domain reasoning tasks, including code reasoning and long-context reasoning. MPD consistently reduces token usage with improved performance, demonstrating that the compressed reasoning behavior generalizes beyond the training domain.

2 Related Work and Preliminaries

2.1 Related Work

We discuss notable related work on reasoning trace compression for reasoning-centric LLMs.

Training-Free Reasoning Compression. Large Language Models (LLMs) have demonstrated strong reasoning ability on complex tasks such as mathematics, code generation, and long-context

reasoning (DeepSeek-AI, 2025; OpenAI, 2024; Cobbe et al., 2021; Nguyen et al., 2025; Bai et al., 2025). However, reasoning-centric LLMs often suffer from overthinking, producing unnecessarily verbose reasoning trajectories that increase decoding cost and latency (Chen et al., 2024; Sui et al., 2025; Gekhman et al., 2026). A direct way to reduce reasoning length is to prompt models to produce only essential intermediate steps (Sui et al., 2025; Wang et al., 2025). Concise Chain-of-Thought (Renze & Guven, 2024) encourages shorter CoT traces, while Chain-of-Draft (Xu et al., 2025) asks models to capture only key information. Although simple and broadly applicable, these methods are sensitive to prompt design and remain far from the theoretical efficiency frontier (Lee et al., 2025).

SFT and RL-Based Compression. Another line of work directly optimises models for concise reasoning. SFT-based methods train LLMs on shortened reasoning trajectories (Kang et al., 2025). TokenSkip (Xia et al., 2025) learns controllable CoT compression by omitting non-essential tokens, while LiteCoT (Wu et al., 2025) fine-tunes models on pre-refined concise traces. However, because these compression targets are constructed externally, SFT introduces a train-inference discrepancy: the student learns from curated traces rather than its own reasoning behaviour, and is therefore not explicitly trained to compress or recover from the trajectories it will produce at inference time. RL-based methods instead encourage concise reasoning through length penalties or token budgets (Jin et al., 2025; Shao et al., 2024). L1 (Aggarwal & Welleck, 2025) jointly optimises accuracy and token efficiency, while THINKPRUNE (Hou et al., 2026) trains long-thinking models under a strict token budget. Although effective in reducing reasoning length, these methods often require ground-truth answers to define correctness rewards and may suppress useful exploratory reasoning by directly penalising length. To address these limitations, MPD learns concise reasoning without ground-truth answers, externally curated compression targets, or manually designed length rewards, using the student's own reasoning traces as the basis for training.

Distillation for Concise Reasoning. Distillation offers a natural way to learn reasoning compression by transferring efficient reasoning behaviour from teacher models to students. Off-policy distillation trains the student on teacher-generated reasoning trajectories, which can provide concise and high-quality traces, but may suffer from distribution mismatch because these trajectories do not come from the student's own policy. On-policy distillation instead trains on student-generated trajectories, reducing exposure bias and better matching inference-time reasoning behaviour (Agarwal et al., 2024; Gu et al., 2024; Lu & Lab, 2025; Ye et al., 2026). A recent work, i.e., CRISP (Sang et al., 2026), applies on-policy self-distillation to reasoning compression by inducing conciseness with explicit brevity instructions. However, it does not leverage the natural compression advantage of larger models. Motivated by our observation that larger models often solve the same problems with substantially shorter traces (Fig. 1), MPD further turns the teacher into a trajectory compressor, allowing concise reasoning behaviour to be transferred while keeping training grounded in the student's own inference-time distribution.

2.2 Preliminaries: Off-Policy vs. On-Policy Distillation

We formalize off-policy and on-policy distillation for reasoning trajectory learning. We consider transferring reasoning behaviour from a stronger teacher model $\pi_t(\cdot \mid x; \phi)$ to a weaker student model $\pi_s(\cdot \mid x; \theta)$ with learnable parameters θ, where ϕ denotes the fixed teacher parameters. Given a question $x \sim D$, let $y = (y_1, \ldots, y_n)$ denote a reasoning trajectory together with the final answer. Distillation trains the student to match the teacher's predictive behaviour along such trajectories, typically by minimizing a divergence $\mathrm{Div}(\cdot \| \cdot)$ between the student and teacher distributions.

A key distinction between off-policy and on-policy distillation is the source of the training trajectories. Off-policy distillation trains on trajectories sampled from the teacher, yielding the objective

$$\mathcal{L}_{\text{off}} = \mathbb{E}_{x \sim D,\, y \sim \pi_t(\cdot|x;\phi)} \left[\sum_{i=1}^{n} \mathrm{Div}\left(\pi_s(\cdot \mid x, y_{<i}; \theta) \,\|\, \pi_t(\cdot \mid x, y_{<i}; \phi) \right) \right]. \tag{1}$$

Teacher-generated trajectories are often of higher quality, but they create a train-inference mismatch: the student is optimized under teacher-induced contexts rather than the contexts produced by its own generations (Lu & Lab, 2025). Consequently, the student is not trained to recover from its own mistakes. At inference time, the student must reason over unseen questions under its own generated

contexts. Since off-policy distillation trains it only on teacher-generated trajectories, the student is not explicitly exposed to the kinds of imperfect intermediate states it may produce itself, making recovery difficult and allowing errors to accumulate across reasoning steps.

In contrast, on-policy distillation samples trajectories from the student model:

$$\mathcal{L}_{\text{on}} = \mathbb{E}_{x\sim D,\, y\sim\pi_s(\cdot|x;\theta)} \left[\sum_{i=1}^{n} \text{Div}\left(\pi_s(\cdot \mid x, y_{<i}; \theta) \parallel \pi_t(\cdot \mid x, y_{<i}; \phi)\right) \right]. \tag{2}$$

This brings a key advantage: the student is aligned with the teacher under its own trajectory distribution. Instead of learning only from teacher-induced contexts, the student is trained on the reasoning states it is likely to encounter during inference. This reduces the mismatch between training and inference, making the distillation objective better aligned with the student's actual inference-time behaviour.

In the context of reasoning compression, a teacher should be a model that reasons more efficiently, while the student is more prone to redundant and inefficient intermediate reasoning. Rooting from our observation in Fig. 1, we argue that a smaller reasoning model can be naturally supervised by its larger variant within the same model family, since the larger model provides useful signals for both task performance and token efficiency. We next introduce MPD, a distillation-based framework that mixes on- and off-policy learning: it samples trajectories from the student to preserve the student's inference-time distribution, while using the larger teacher to compress these trajectories into concise reasoning traces. We then show in Sec. 4.2 with empirical evidence to explain why this mixed-policy design outperforms both standard on-policy and off-policy distillation.

3 Reasoning Compression with Mixed-Policy Distillation

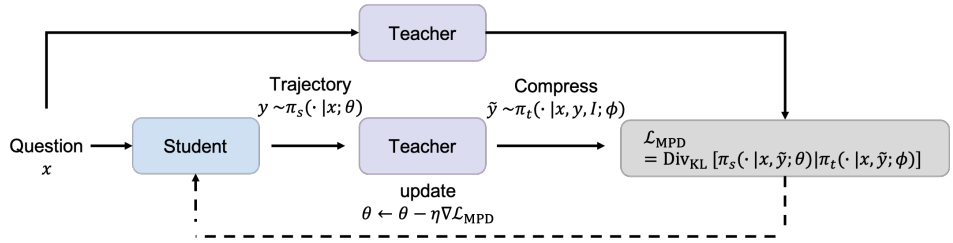


Figure 2: Overview of MPD.

We provide the overview of our proposed MPD in Fig. 2. MPD consists of three stages. First, given a question $x \sim D$, the student model samples a reasoning trajectory with the final answer, $y \sim \pi_s(\cdot \mid x; \theta)$. Second, the larger teacher model compresses this student-generated trajectory into a compressed, semantically equivalent trace, denoted as $\tilde{y} = (\tilde{y}_1, \ldots, \tilde{y}_m)$. Third, the student is trained on the compressed trajectory by minimizing the divergence between the student and teacher distributions conditioned on each prefix $\tilde{y}_{<i}$, using $\pi_t(\cdot \mid x, \tilde{y}_{<i}; \phi)$ as the distillation target.

Motivated by our observation that larger models often use substantially fewer tokens than smaller students when solving the same problems, MPD uses a larger teacher as the source of reasoning compression. Specifically, as shown in Table 1, the teacher is given the same question x together with the student-generated reasoning trace y, and is instructed to rewrite the trace into a more concise form. The compression prompt I is designed to preserve the student's reasoning signal while allowing the teacher to improve its expression. We ask the teacher to remain faithful to the original student-generated reasoning, avoid introducing new solution paths or additional information, and keep only the steps necessary for answering the question. In this way, the compressed trajectory is not an independently generated teacher solution. Instead, **it is a teacher-compressed version of the student's own trajectory, which preserves the student's problem-solving path and therefore keeps the training signal tied to the student policy.** At the same time, the compression process is performed by the larger teacher under its own policy. **When deciding which intermediate steps are necessary, which redundant parts can be removed, and how the remaining reasoning should be rewritten, the teacher injects its stronger ability to organise and express reasoning concisely.**

Table 1: Compression prompt I for the teacher model.

Compression Prompt for Teacher
Based on the provided reasoning, write a summary. The summary should integrate all relevant information from the original text that can help answer the specified question and form a coherent paragraph. Ensure that the summary includes all useful information from the original text for answering the question. Do not add new information. Question: $\{x\}$ Reasoning: $\{y\}$ Please provide the summary:

Therefore, the resulting compressed trace $\tilde{y}$ combines two desirable properties: it remains aligned with the student-generated trajectory y, while reflecting the teacher's stronger compression capability. This is the core mechanism that allows MPD to retain the distributional benefit of student-generated trajectories while transferring concise reasoning behaviour from a larger model.

To verify whether teacher-compressed reasoning traces preserve the student's reasoning strategy, we randomly sample 100 questions from OpenMathInstruct-2 (Toshniwal et al., 2025). For each question, we use Qwen3-1.7B as the student to generate an initial reasoning trajectory, and Qwen3-8B as the teacher to compress it. We then compute sentence embeddings for both the original student trajectory and the teacher-compressed trajectory using Sentence-BERT (Reimers & Gurevych, 2019), and measure their cosine similarity. As a reference, we also compute the cosine similarity between each student-generated trajectory and another student-generated trajectory from a different randomly selected question within the same set of 100 questions. To further isolate the effect of using a larger teacher, we repeat the same compression procedure using also Qwen3-1.7B with the same compression prompt, and report the corresponding embedding similarity. In both teacher-compression and student-compression settings, we also record the compression rate (relative reduction in token usage). This allows us to evaluate whether compression preserves the semantic content of the student's reasoning while reducing trajectory length. We analyse the correctness and faithfulness of compressed trajectories beyond embedding similarity in App. C.

From Table 2, we can observe that our compression prompt substantially shortens the student-generated reasoning traces while preserving their semantic content. Qwen3-8B achieves a higher compression rate (Comp. Rate) than Qwen3-1.7B, reducing the trace length by 93.0% compared with 87.5%, which verifies that larger models are better at expressing reasoning concisely, even when the student is identical to the compressor in the Qwen3-1.7B compression setting. Meanwhile, both compressed traces remain highly similar to the initial student trajectory, with cosine similarity (Sim. w. Init) around 0.81, while their similarity to random trajectories (Sim. w. Rnd) is only around 0.16. This verifies that our compression prompt does not push the compressed trace far away from the original reasoning signal, but mainly removes redundant steps.

Table 2: Compression effect using different sizes of models with our prompt.

Compressor	**Comp. Rate**	**Sim. w. Init**	**Sim. w. Rnd**
Qwen3-1.7B	87.5%	0.8102	0.1660
Qwen3-8B	93.0%	0.8098	0.1659

After obtaining the compressed trajectory $\tilde{y}$, we perform distillation on this trajectory. The MPD training objective is therefore defined as:

$$\mathcal{L}_{\mathrm{MPD}} = \mathbb{E}_{x\sim D,\, y\sim\pi_s(\cdot|x;\theta),\, \tilde{y}\sim\pi_t(\cdot|x,y,I;\phi)} \left[\sum_{i=1}^{m} \mathrm{Div}_{\mathrm{KL}} \left(\pi_s(\cdot \mid x, \tilde{y}_{<i}; \theta) \parallel \pi_t(\cdot \mid x, \tilde{y}_{<i}; \phi) \right) \right], \quad (3)$$

where $\mathrm{Div}_{\mathrm{KL}}(\cdot\|\cdot)$ denotes the KL divergence. **In this way, the student can benefit from the minimal distributional shift of trajectories derived from its own policy, while also leveraging the teacher's policy to provide concise reasoning supervision.** We further provide an algorithm of MPD in Alg. 1.

Algorithm 1 Mixed-Policy Distillation

Require: Teacher model $\pi_t(\cdot;\phi)$, student model $\pi_s(\cdot;\theta)$, input dataset D
Require: Compression instruction I, learning rate η, batch size B, number of training steps K
1: **for** each training step $k = 1, \ldots, K$ **do**
2: Sample a batch of questions $\{x_b\}_{b=1}^{B} \sim D$
3: **for** each question x_b **do**
4: Sample an on-policy reasoning trajectory from the student: $y_b \sim \pi_s(\cdot \mid x_b; \theta)$
5: Compress the student trajectory with the teacher: $\tilde{y}_b \sim \pi_t(\cdot \mid x_b, y_b, I; \phi)$
6: **end for**
7: Update θ by minimizing the MPD objective:

$$\theta \leftarrow \theta - \eta \nabla_\theta \frac{1}{B} \sum_{b=1}^{B} \sum_{i=1}^{|\tilde{y}_b|} \mathrm{Div}_{\mathrm{KL}} \left(\pi_s(\cdot \mid x_b, \tilde{y}_{b,<i}; \theta) \parallel \pi_t(\cdot \mid x_b, \tilde{y}_{b,<i}; \phi) \right)$$

8: **end for**

4 EXPERIMENTS

We report the main comparative results and ablation studies, followed by evaluations of MPD's generalization across model scales and families, a comparison with SFT on compressed reasoning traces, and a perplexity-based analysis (Sec. 4.2). In addition, the appendices provide further results, including extended evaluations across model scales and families, cross-family transfer, and additional comparisons with SFT (App.B); detailed analyses of the correctness and faithfulness of compressed trajectories (App.C); and case studies (App. E).

4.1 EXPERIMENTAL SETUP

Datasets & Evaluation Metrics. We use OpenMathInstruct-2 [1], a massive, open-source mathematical reasoning dataset, and adopt its train_1M split as the training data. For evaluation, following (Wu et al., 2025), we use three mathematical reasoning benchmarks: GSM8K (Cobbe et al., 2021), AIME24 (Zhang & Math-AI, 2024), and AIME25 (Zhang & Math-AI, 2025). To evaluate generalization beyond math, we additionally use CodeMMLU (Nguyen et al., 2025) for code reasoning and LongBench-v2 (Bai et al., 2025) for long-context reasoning. We report Pass@4 (P@4) to measure reasoning performance, defined as whether at least one of four sampled generations answers the question correctly, and the average number of generated tokens (#Tok) to measure compression extent.

MPD Implementation Details. We use Qwen3-8B (Team, 2025) as the teacher model and Qwen3-1.7B as the student model. During training, the student first generates a reasoning trajectory and final answer with a maximum length of 27,000 tokens. The teacher then compresses this trajectory into a concise reasoning trace with a maximum length of 10,000 tokens. All other generation-related hyperparameters follow the default settings of the original Qwen models. We train the student for 200 steps with a batch size of 8 on 4 NVIDIA H100 GPUs. More implementation details, including the complete hyperparameter settings, are provided in App. A. All results are averaged over three runs with different random seeds.

Baselines. We compare MPD with five baselines. First, we consider three training-free methods: Vanilla LLM, where Qwen3-1.7B directly answers the question without explicit compression; Direct Compression (Direct Comp.), which prompts the model to reason concisely (see Table 9 for the full prompt); and Chain-of-Draft(Xu et al., 2025), which prompts the model to keep only concise intermediate thoughts. Second, we compare with LiteCoT (Wu et al., 2025), an SFT-based method that refines reasoning trajectories and fine-tunes the Qwen3-1.7B on the resulting concise traces. Third, we consider CRISP (Sang et al., 2026), an on-policy self-distillation method for concise reasoning that uses the same model as both student and teacher, with the teacher conditioned on a "be concise" instruction and the student aligned on self-generated trajectories via reverse KL. Since

[1] https://huggingface.co/datasets/nvidia/OpenMathInstruct-2

Table 3: Overall performance and token usage. Colors represent the relative improvement (Green) or regression (Red) compared to the Vanilla LLM baseline.

Method	GSM8K		AIME24		AIME25		CodeMMLU		LongBench2		Avg.	
	P@4	#Tok	P@4	#Tok	P@4	#Tok	P@4	#Tok	P@4	#Tok	P@4	#Tok
Vanilla LLM	94.5	2.1k	63.3	16.8k	53.3	17.7k	81.2	3.6k	48.7	0.9k	68.2	8.2k
Direct Comp.	94.1	1.2k	70.0	14.5k	50.0	15.2k	82.0	2.4k	48.1	0.8k	68.9	6.9k
Chain-of-Draft	42.8	0.6k	63.3	11.8k	50.0	13.0k	79.0	2.0k	40.8	1.8k	55.1	5.9k
LiteCoT	52.0	1.2k	66.7	15.6k	46.7	14.6k	82.1	3.4k	36.6	2.0k	56.8	7.3k
CRISP	94.3	1.7k	70.0	14.8k	56.6	16.3k	82.7	4.1k	49.9	1.0k	70.7	7.6k
MPD (Ours)	94.2	1.4k	73.3	13.4k	56.7	12.9k	82.8	2.3k	50.3	0.7k	71.3	6.1k
MPD Rank	3rd	4th	1st	2nd	1st	1st	1st	2nd	1st	1st	1st	2nd

CRISP is originally formulated as self-distillation, we adapt it to our teacher-student setting by using Qwen3-8B as the teacher and Qwen3-1.7B as the student, ensuring a fair comparison with MPD. All reported results are averaged over three runs with different random seeds. We provide implementation details of baseline methods in App. A.

4.2 Comparative Results

Main Results. As shown in Table 3, we have several key findings: (1) MPD outperforms almost all baselines across diverse reasoning tasks[2]. Compared with Vanilla LLM, MPD achieves a 4.5% relative improvement in Pass@4 and a 25.6% relative reduction in token usage on average, showing that reasoning compression can improve efficiency without sacrificing performance. We attribute this to concise reasoning reducing unnecessary iterative reflection, making the model less likely to overthink and allowing it to follow a more direct reasoning path. (2) Training-free compression methods can reduce token usage but often hurt reasoning quality. Chain-of-Draft substantially lowers token usage, but suffers from severe performance degradation, even underperforming Direct Comp., suggesting that forcing the model to produce sparse, "draft-like" intermediate thoughts can disrupt the continuous logical reasoning needed for problem solving (Xu et al., 2026). (3) Fine-tuning on pre-refined traces provides only limited gains. Although LiteCoT performs slightly better than Chain-of-Draft, it still suffers from distribution mismatch caused by static off-policy supervision, since the student is not trained to compress its own reasoning trajectories. (4) On-policy distillation is helpful but insufficient without teacher-guided compression. Our adapted CRISP achieves a competitive Pass@4, benefiting from on-policy training over student-generated trajectories. However, it still trails MPD in both performance and compression because its compression signal mainly comes from brevity prompting rather than teacher-compressed student trajectories. In contrast, MPD distills over teacher-compressed versions of the student's own reasoning traces, combining the inference-time alignment of on-policy learning with the compression advantage of the stronger teacher.

Ablation Studies. To better demonstrate MPD's advantage, we perform ablation studies and report results in Table 5 on four controlled variants: Off-Policy denotes standard off-policy distillation, where the Qwen3-8B teacher generates full reasoning trajectories and the Qwen3-1.7B student is trained to match the teacher on these teacher-generated traces. On-Policy denotes standard on-policy distillation, where the Qwen3-1.7B student generates its own reasoning trajectories and is then trained to match the Qwen3-8B teacher under these student-generated traces. We also consider compressed variants using the same compression prompt as MPD. Off-Policy (C) denotes compressed off-policy distillation, where the teacher both generates and compresses the trajectories before the student is trained on the compressed teacher traces. On-Policy (C) denotes compressed on-policy distillation, where the student generates and compresses its own trajectories and is then trained to match the teacher on the resulting compressed traces. We have several findings: (1) On-Policy achieves good performance by training on the student's own trajectory distribution, but it also produces the highest token usage. This suggests that on-policy learning reduces distribution mismatch, but

[2]MPD does not achieve the best overall rank on GSM8K because GSM8K is a relatively easy benchmark, where performance is already saturated even for the Vanilla LLM baseline.

Table 5: Ablation study on different policy variants. Colors represent the relative improvement (Green) or regression (Red) compared to the Vanilla LLM baseline.

Method	GSM8K		AIME24		AIME25		CodeMMLU		LongBench2		Avg.	
	P@4	#Tok	P@4	#Tok	P@4	#Tok	P@4	#Tok	P@4	#Tok	P@4	#Tok
Off-Policy	94.2	2.8k	70.0	22.9k	53.3	21.9k	82.4	5.4k	51.7	1.0k	70.3	10.6k
Off-Policy (C)	94.6	2.7k	66.6	20.8k	56.6	20.5k	82.2	4.5k	49.0	0.9k	69.8	9.9k
On-Policy	94.0	3.1k	73.3	20.9k	53.3	21.9k	84.2	5.2k	47.9	1.0k	70.5	11.8k
On-Policy (C)	94.1	2.1k	70.0	15.8k	56.7	17.0k	82.8	3.6k	48.3	1.0k	70.5	7.9k
MPD (Ours)	94.2	1.4k	73.3	13.4k	56.7	12.9k	82.8	2.3k	50.3	0.7k	71.3	6.1k
MPD Rank	2rd	1st	1st	1st	1st	1st	2nd	1st	2nd	1st	1st	1st

directly learning from verbose student trajectories can reinforce redundant reasoning patterns. (2) Off-Policy uses shorter teacher-generated trajectories, but does not outperform On-Policy, confirming the mismatch between teacher trajectories and the student's own inference-time behaviour limits effective learning. (3) Explicit compression using MPD's compression prompt consistently reduces token usage compared with the corresponding verbose variants. However, On-Policy (C) relies on the weaker student to compress its own trajectories, which can potentially remove useful reasoning steps and weaken performance, while Off-Policy (C) benefits from teacher compression but still suffers from off-policy supervision. (4) MPD achieves the best trade-off between performance and conciseness by combining the two necessary components: student-generated trajectories provide the on-policy reasoning backbone, and teacher-guided compression removes redundancy while preserving the underlying reasoning structure.

MPD generalizes across different model scales and families. As shown in Table 4, MPD consistently produces more concise reasoning while maintaining or improving performance across student-teacher pairs of different scales and model families. For Qwen3, MPD improves the average P@4 from 68.2 to 69.8 for the 1.7B student paired with a 32B teacher, and from 80.1 to 82.4 for the 8B student paired with the same teacher, while simultaneously reducing the average token usage from 8.2k to 6.3k and from 7.6k to 6.8k, respectively. This indicates that MPD remains effective across student–teacher pairs of different model scales. Similar trends are observed beyond the Qwen3 family. MPD improves or maintains performance for Qwen2.5-1.5B (Yang et al., 2024) with Qwen2.5-14B, Llama 3.2-3B (Team, 2024) with Llama 3.1-8B, and Ministral-3-3B (AI, 2026) with Ministral-3-8B, while consistently reducing the number of generated tokens. These results suggest that the benefits of MPD are not tied to a particular model family or scale, but generalize across different student and teacher configurations. We provide detailed and further evaluations in App. B covering additional model scales, families, and cross-family student-teacher settings.

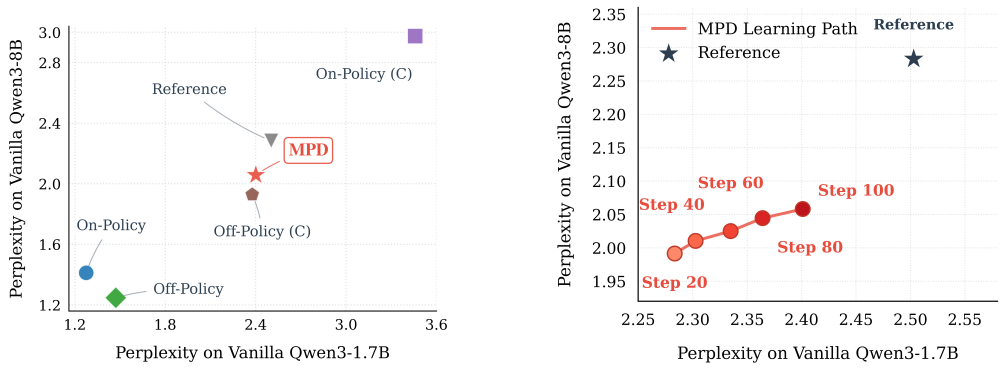
Table 4: MPD across model scales and families. The average results of five evaluation datasets are reported.

Student–Teacher	Vanilla P@4 / #Tok	MPD P@4 / #Tok
Qwen3: 1.7B–32B	68.2 / 8.2k	**69.8 / 6.3k**
Qwen3: 8B–32B	80.1 / 7.6k	**82.4 / 6.8k**
Qwen2.5: 1.5B–14B	42.6 / 1.1k	**44.0 / 0.9k**
Llama 3.2 3B–3.1 8B	44.6 / 1.7k	**45.6 / 1.5k**
Ministral-3: 3B–8B	74.1 / 10.0k	**74.2 / 8.7k**

MPD vs. SFT. We compare MPD with the case when the student is trained by SFT on the teacher's compressed reasoning traces. As shown in Table 6, SFT achieves an average reasoning length to 0.3k, however, the average performance degrades to 43.3. This indicates that SFT causes severe performance collapse against the vanilla student without training and struggles to balance length reduction and correctness. In contrast, MPD reaches an improvement in performance while reasoning concisely, which indicates token-level KL alignment is crucial when learning from concise reasoning traces. We provide detailed per-dataset results and SFT results across training steps in App. B.2.

Table 6: Overall performance under Plain SFT and MPD training settings. The average results of five evaluation datasets are reported.

Method	Avg. P@4	Avg. #Tok
Vanilla	68.2	8.2k
SFT	43.3	**0.3k**
MPD	**71.3**	6.1k



(a) Comparison among different training methods.

(b) MPD learning path.

Figure 3: Perplexity analysis of generated reasoning traces from different training methods under vanilla Qwen models.

Why MPD Outperforms: a Perplexity Perspective. To better understand the effect of MPD, we analyze the perplexity of generated reasoning trajectories under the vanilla Qwen3-1.7B and Qwen3-8B. Specifically, we sample the same 100 questions from OpenMathInstruct-2 and let different trained models generate their own reasoning traces. We then evaluate each set of traces by computing the average perplexity across all questions under vanilla models. We also include the human-inspected reference solutions from OpenMathInstruct-2, which are expected to be non-redundant. As shown in Fig. 3a, the reference solutions do not have the lowest perplexity under either vanilla model, suggesting that concise, human-inspected reasoning is not simply the most likely continuation under the original model distributions. Standard On-Policy and Off-Policy trajectories have much lower perplexity under both vanilla models, indicating that they remain close to typical model-generated reasoning, but they are also far from the reference region. After explicit compression, both On-Policy (C) and Off-Policy (C) move toward the reference, showing that compression helps shift generated traces toward more concise reasoning. However, they do so differently. On-Policy (C) has high perplexity under both vanilla models, likely because the weaker student both generates and compresses its own traces. This can produce compressed trajectories that deviate from ordinary student generations while still lacking the teacher's stronger ability to organize concise reasoning, making them less natural to both vanilla models. Off-Policy (C), in contrast, stays closer to vanilla Qwen3-8B than On-Policy (C), since both generation and compression are performed by the teacher. However, since its trajectories are generated entirely from the teacher policy, they remain less aligned with the vanilla Qwen3-1.7B distribution. MPD is closest to the reference among all variants: it uses student-generated traces to preserve the student-side distributional signal, while applying teacher compression to move the traces toward concise, reference-like reasoning. Finally, Fig. 3b shows the MPD learning path across training checkpoints. As training progresses from Step 20 to Step 100, the generated trajectories gradually move toward the reference solutions. This indicates that MPD provides a consistent learning signal that shifts the student away from redundant reasoning traces and toward more concise, reference-like reasoning behaviour.

5 Conclusion

We propose Mixed-Policy Distillation (MPD), a reasoning compression framework that transfers concise reasoning behaviour from a larger teacher model to a smaller student model without requiring ground-truth answers, reward models, length penalties, or externally curated compressed traces. Motivated by the observation that larger reasoning models often solve the same problems with shorter traces, MPD samples trajectories from the student and uses the larger teacher to rewrite them into more concise reasoning traces. The student is then trained on these teacher-compressed student trajectories through KL-based distillation. Experiments across mathematical reasoning, code reasoning, and long-context reasoning benchmarks show that MPD consistently reduces token usage while maintaining or improving performance. Further analysis demonstrates that MPD's advantage comes from combining student-side exploration with teacher-guided compression, enabling concise reasoning without discarding the key intermediate structure needed for correctness.

AI USE STATEMENT

We use AI tools to assist with language editing and text polishing. All AI-assisted text was reviewed and revised by the authors, who take full responsibility for the final content.

ETHICS STATEMENT

This work focuses on improving the efficiency of reasoning in large language models. The study does not involve human subjects, personally identifiable information, or private user data. All experiments are conducted using existing models and benchmark datasets for research purposes. We follow standard research practices regarding data usage, model evaluation, reproducibility, and responsible reporting. We are not aware of any specific ethical concerns beyond those generally associated with research on large language models.

REPRODUCIBILITY STATEMENT

We provide the implementation details, training configurations, evaluation settings, and benchmark information needed to reproduce our experiments in the main paper and appendix. The appendix further includes the prompts used for LLM-based evaluation, additional ablation and scaling results, and detailed case studies of compressed reasoning trajectories. We also report the model configurations and evaluation protocols used throughout the experiments. Code and supplementary materials will be released to facilitate reproduction of the reported results.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=3zKtaqxLhW.

Pranjal Aggarwal and Sean Welleck. L1: controlling how long A reasoning model thinks with reinforcement learning. *CoRR*, abs/2503.04697, 2025. doi: 10.48550/ARXIV.2503.04697. URL https://doi.org/10.48550/arXiv.2503.04697.

Mistral AI. Ministral 3. *CoRR*, abs/2601.08584, 2026. doi: 10.48550/ARXIV.2601.08584. URL https://doi.org/10.48550/arXiv.2601.08584.

Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. Longbench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pp. 3639–3664. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.acl-long.183/.

Nicolas Boizard, Kevin El Haddad, Céline Hudelot, and Pierre Colombo. Towards cross-tokenizer distillation: the universal logit distillation loss for llms. *Trans. Mach. Learn. Res.*, 2025, 2025. URL https://openreview.net/forum?id=bwRxXiGO9A.

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. Do NOT think that much for 2+3=? on the overthinking of o1-like llms. *CoRR*, abs/2412.21187, 2024. doi: 10.48550/ARXIV.2412.21187. URL https://doi.org/10.48550/arXiv.2412.21187.

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

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *CoRR*, abs/2501.12948, 2025. doi: 10.48550/ARXIV.2501.12948. URL https://doi.org/10.48550/arXiv.2501.12948.

Zorik Gekhman, Roee Aharoni, Eran Ofek, Mor Geva, Roi Reichart, and Jonathan Herzig. Thinking to recall: How reasoning unlocks parametric knowledge in llms. *CoRR*, abs/2603.09906, 2026. doi: 10.48550/ARXIV.2603.09906. URL https://doi.org/10.48550/arXiv.2603.09906.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: Knowledge distillation of large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=5h0qf7IBZZ.

Bairu Hou, Yang Zhang, Jiabao Ji, Yujian Liu, Kaizhi Qian, Jacob Andreas, and Shiyu Chang. Thinkprune: Pruning long chain-of-thought of llms via reinforcement learning. *Trans. Mach. Learn. Res.*, 2026, 2026. URL https://openreview.net/forum?id=V51gPu1uQD.

Zhensheng Jin, Xinze Li, Yifan Ji, Chunyi Peng, Zhenghao Liu, Qi Shi, Yukun Yan, Shuo Wang, Furong Peng, and Ge Yu. Recut: Balancing reasoning length and accuracy in llms via stepwise trails and preference optimization. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025, Suzhou, China, November 4-9, 2025*, pp. 14269–14282. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.findings-emnlp.770/.

Yu Kang, Xianghui Sun, Liangyu Chen, and Wei Zou. C3ot: Generating shorter chain-of-thought without compromising effectiveness. In Toby Walsh, Julie Shah, and Zico Kolter (eds.), *Thirty-Ninth AAAI Conference on Artificial Intelligence, Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence, Fifteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2025, Philadelphia, PA, USA, February 25 - March 4, 2025*, pp. 24312–24320. AAAI Press, 2025. doi: 10.1609/AAAI.V39I23.34608. URL https://doi.org/10.1609/aaai.v39i23.34608.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/8bb0d291acd4acf06ef112099c16f326-Abstract-Conference.html.

Woosuk Kwon. *vLLM: An Efficient Inference Engine for Large Language Models*. PhD thesis, UC Berkeley, 2025.

Ayeong Lee, Ethan Che, and Tianyi Peng. How well do llms compress their own chain-of-thought? A token complexity approach. *CoRR*, abs/2503.01141, 2025. doi: 10.48550/ARXIV.2503.01141. URL https://doi.org/10.48550/arXiv.2503.01141.

Kevin Lu and Thinking Machines Lab. On-policy distillation. https://thinkingmachines.ai/blog/on-policy-distillation/, 2025.

Dung Manh Nguyen, Thang Chau Phan, Nam Le Hai, Tien-Thong Doan, Nam V. Nguyen, Quang Pham, and Nghi D. Q. Bui. Codemmlu: A multi-task benchmark for assessing code understanding & reasoning capabilities of codellms. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=CahIEKCu5Q.

OpenAI. Openai o1 system card. *CoRR*, abs/2412.16720, 2024. doi: 10.48550/ARXIV.2412.16720. URL https://doi.org/10.48550/arXiv.2412.16720.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas

Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 8024–8035, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html.

Carlos Miguel Patiño, Kashif Rasul, Quentin Gallouédec, Ben Burtenshaw, Sergio Paniego, Vaibhav Srivastav, Thibaud Frere, Ed Beeching, Lewis Tunstall, Leandro von Werra, and Thomas Wolf. Unlocking on-policy distillation for any model family. https://huggingface.co/spaces/HuggingFaceH4/on-policy-distillation, 2025.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pp. 3980–3990. Association for Computational Linguistics, 2019. doi: 10.18653/V1/D19-1410. URL https://doi.org/10.18653/v1/D19-1410.

Matthew Renze and Erhan Guven. The benefits of a concise chain of thought on problem-solving in large language models. In *2nd International Conference on Foundation and Large Language Models, FLLM 2024, Dubai, United Arab Emirates, November 26-29, 2024*, pp. 476–483. IEEE, 2024. doi: 10.1109/FLLM63129.2024.10852493. URL https://doi.org/10.1109/FLLM63129.2024.10852493.

Hejian Sang, Yuanda Xu, Zhengze Zhou, Ran He, Zhipeng Wang, and Jiachen Sun. CRISP: Compressed reasoning via iterative self-policy distillation. *arXiv preprint arXiv:2603.05433*, 2026.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *CoRR*, abs/2402.03300, 2024. doi: 10.48550/ARXIV.2402.03300. URL https://doi.org/10.48550/arXiv.2402.03300.

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. Stop overthinking: A survey on efficient reasoning for large language models. *Trans. Mach. Learn. Res.*, 2025, 2025. URL https://openreview.net/forum?id=HvoG8SxggZ.

Gemini Team. Gemini: A family of highly capable multimodal models. *CoRR*, abs/2312.11805, 2023. doi: 10.48550/ARXIV.2312.11805. URL https://doi.org/10.48550/arXiv.2312.11805.

Llama Team. The llama 3 herd of models. *CoRR*, abs/2407.21783, 2024. doi: 10.48550/ARXIV.2407.21783. URL https://doi.org/10.48550/arXiv.2407.21783.

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

Shubham Toshniwal, Wei Du, Ivan Moshkov, Branislav Kisacanin, Alexan Ayrapetyan, and Igor Gitman. Openmathinstruct-2: Accelerating AI for math with massive open-source instruction data. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=mTCbq2QssD.

Chenlong Wang, Yuanning Feng, Dongping Chen, Zhaoyang Chu, Ranjay Krishna, and Tianyi Zhou. Wait, we don't need to "wait"! removing thinking tokens improves reasoning efficiency. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025, Suzhou, China, November 4-9, 2025*, pp. 7459–7482. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.findings-emnlp.394/.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771, 2019. URL http://arxiv.org/abs/1910.03771.

Yifan Wu, Jingze Shi, Bingheng Wu, Jiayi Zhang, Xiaotian Lin, Nan Tang, and Yuyu Luo. Concise reasoning, big gains: Pruning long reasoning trace with difficulty-aware prompting. *CoRR*, abs/2505.19716, 2025. doi: 10.48550/ARXIV.2505.19716. URL https://doi.org/10.48550/arXiv.2505.19716.

Heming Xia, Chak Tou Leong, Wenjie Wang, Yongqi Li, and Wenjie Li. Tokenskip: Controllable chain-of-thought compression in llms. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, EMNLP 2025, Suzhou, China, November 4-9, 2025*, pp. 3351–3363. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.EMNLP-MAIN.165. URL https://doi.org/10.18653/v1/2025.emnlp-main.165.

Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng He. Chain of draft: Thinking faster by writing less. *CoRR*, abs/2502.18600, 2025. doi: 10.48550/ARXIV.2502.18600. URL https://doi.org/10.48550/arXiv.2502.18600.

Yuanda Xu, Hejian Sang, Zhengze Zhou, Ran He, and Zhipeng Wang. Overconfident errors need stronger correction: Asymmetric confidence penalties for reinforcement learning. *CoRR*, abs/2602.21420, 2026. doi: 10.48550/ARXIV.2602.21420. URL https://doi.org/10.48550/arXiv.2602.21420.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *CoRR*, abs/2412.15115, 2024. doi: 10.48550/ARXIV.2412.15115. URL https://doi.org/10.48550/arXiv.2412.15115.

Tianzhu Ye, Li Dong, Xun Wu, Shaohan Huang, and Furu Wei. On-policy context distillation for language models. *CoRR*, abs/2602.12275, 2026. doi: 10.48550/ARXIV.2602.12275. URL https://doi.org/10.48550/arXiv.2602.12275.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2024, 2024.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2025, 2025.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. *CoRR*, abs/2601.18734, 2026. doi: 10.48550/ARXIV.2601.18734. URL https://doi.org/10.48550/arXiv.2601.18734.

APPENDIX

A Additional Implementation & Experimental Detail **14**

B Full Experiment Results **16**

B.1 Full Results across Model Families 16

B.2 Comparison with SFT on Compressed Reasoning Traces 17

B.3 Experiments on Cross-family Transfer 18

C Evaluation of Trajectories after Compression **18**

C.1 Correctness of trajectory 19

C.2 Fine-grained Evaluation of Trajectories using LLM-as-Judge 19

C.3 Case Study of Key Steps Preservation 20

C.4 Evaluation Prompts 21

D Does Shorter Reasoning Correspond to Better Reasoning? **21**

E Case Study: MPD vs. Baselines & Ablations **22**

F Limitations & Future Directions **23**

G Potential Societal Risks **24**

A ADDITIONAL IMPLEMENTATION & EXPERIMENTAL DETAIL

Hardware and Software. All experiments involving LLMs are implemented with PyTorch (Paszke et al., 2019), HuggingFace Transformers (Wolf et al., 2019), and vLLM (Kwon, 2025). Model training is conducted using the TRL[3] library on a SLURM-managed computing cluster. Each training node is equipped with 96 CPU cores and 4 NVIDIA H100 GPUs, each with 94GB of memory. For inference, we use vLLM for efficient batched generation.

Training & Experimental Details. Our training framework is implemented on top of the OPSD repository (Zhao et al., 2026), which inherits from TRL. We provide our training and evaluation configurations for our method in Table 7. The parameter $\beta = 1$ refers to the Reverse KL form (i.e. $D_{\mathrm{KL}}(P_{\text{Student}} \| P_{\text{Teacher}})$) of the JSD loss (Agarwal et al., 2024). The parameter λ from the OPSD framework denotes "the probability of replacing an offline example with a response generated by the student model". We set $\lambda = 1$, so that all training examples are generated by the student model.

Additionally, the OSPD framework contains a `jsd_token_clip` parameter, which "*Clip the JSD loss for each token to a maximum value. This can improve stability by preventing extremely high-loss stylistic tokens from dominating the training signal*", as mentioned in its document. Specifically, this limits the maximum positive range of the term $p_S(v) \log \frac{p_S(v)}{p_T(v)}$ for every token. Then it sums them over the vocabulary. We follow the OPSD framework and set this value by default.

Further, the KL divergence is computed over the full teacher logits, following the default implementation in OPSD. We did not introduce additional regularization terms. Consequently, the objective is $\frac{1}{N} \sum_{i=1}^{N} \sum_v \min \left(p_{S,i}(v) \log \frac{p_{S,i}(v)}{p_{T,i}(v)}, c \right)$, where $c =$ `jsd_token_clip` and N denotes the length of the sequence.

[3] https://github.com/huggingface/trl

Table 7: Hyper-parameter configurations for MPD.

Training & Loss		**Sampling (Distillation)**	
Dataset	OpenMathInstruct-2	Max Completion (Student)	27,000
Student Model	Qwen3-1.7B	Max Completion (Teacher)	10,000
Teacher Model	Qwen3-8B	Max Total Length	30,000
Training Steps	200	Temperature	0.6
Epochs	1	Top-p	0.95
Batch Size (per device)	2	Top-k	20
Learning Rate	5.00E-06	**Evaluation**	
Beta (β)	1	Max Total Length	32,768
Lambda (λ)	1	Temperature	0.6
JSD Token Clip	0.05	Top-p	0.95
Max Grad Norm	0.1	Top-k	20
LoRA Configuration			
Rank (r)	64	Alpha (α)	128

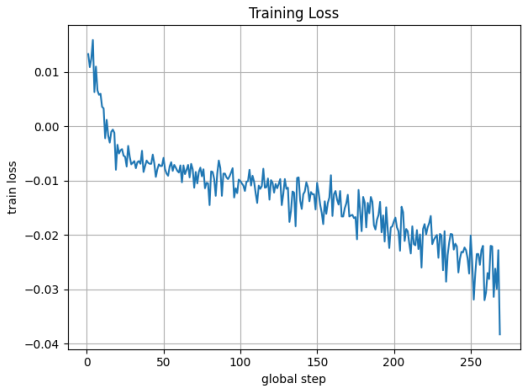


Figure 4: Training loss.

The training process took approximately 18 hours in wall-clock time, corresponding to a total computational cost of roughly 72 GPU-hours. For baseline methods, we follow the official implementations and default hyperparameter settings whenever available. For CRISP (Sang et al., 2026), we re-implemented it based on the OPSD codebase to ensure comparable evaluation. For LiteCoT, we use the LiteCoT training data [4] to train Qwen3-1.7B using the official released code on 1 NVIDIA H100 GPU. The inference hyperparameters, including temperature, top-p and top-k, follow the recommended settings of the corresponding models in the Hugging Face and vLLM documents, with thinking mode enabled (Team, 2025).

Training Loss. We show the loss during the training in Fig. 4. The loss takes negative values in some regions of the curve. As mentioned above, the framework contains a `jsd_token_clip` parameter, which "Clip the JSD loss for each token to a maximum value. This can improve stability by preventing extremely high-loss stylistic tokens from dominating the training signal". Normally, when calculating the standard KL divergence between two logits, the term $p_S(v) \log \frac{p_S(v)}{p_T(v)}$ for every token is first calculated. Then it sums them over the vocabulary. The sum is guaranteed to be non-negative, making the standard KL divergence therefore non-negative. This term for each token *individually*, however, is not guaranteed to be non-negative. With this `jsd_token_clip`, the positive value for this term of each individual token will be limited by this parameter, while the negative values for this

[4] `https://huggingface.co/datasets/SmallDoge/SmallThoughts`

Table 8: Prompt for the student model in answering questions.

Prompt for the Student
Problem: $\{x\}$ Please reason step by step, and put your final answer within \boxed{}.

term are not limited. The resulting objective is then no longer guaranteed to remain non-negative, which explains the negative training loss.

Hyperparameter Search. The maximum completion length is determined by balancing three factors: (1) ensuring that the student model can generate the complete trajectory, (2) reserving sufficient context space for the teacher model to produce the compressed response conditioned on the generated trajectory, and (3) avoiding out-of-memory (OOM) issues during training and inference. The batch size is adjusted according to the sequence length to fit within GPU memory constraints and prevent OOM errors. All other hyperparameters follow the recommended or default settings of the corresponding models and libraries.

Table 9: Prompt for Direct Comp..

Prompt for Direct Comp.
Solve the following problem. Give a full solution, but the final answer MUST be written in the form: \boxed{}. Solve the following math problem concisely and correctly. Be direct – avoid unnecessary elaboration, redundant steps, or restating the problem. Focus only on the key reasoning steps needed to reach the answer. Problem: $\{x\}$

Prompts. We show here the prompts that we used during the training. The student prompt is shown in Table 8. The teacher compression prompt is shown in Table 1. The prompt used in Direct Comp. is shown in Table 9

B Full Experiment Results

We provide additional results across different model families in addition to the training pair Qwen3-1.7B with Qwen3-8B. These results show that our method is generalizable across different model scales, families and generations. Additionally, we provide our exploration of cross-model transfer with our method here.

B.1 Full Results across Model Families

Generalization to scaling model sizes. To evaluate the robustness to the scalability of the models, we conduct experiments with a larger teacher model size, Qwen3-32B as the teacher. Also, we scale the student model size to Qwen3-8B and Qwen3-14B. We show the result in Table 10. Increasing the teacher size to 32B still achieves outstanding token reduction. For the Qwen3-1.7B student, the 32B teacher achieves an average 23.7% reduction in generated tokens (dropping from 8.2k to 6.3k). When scaling up the student model, MPD continues to deliver clear gains. For the Qwen3-8B student distilled by Qwen3-32B, MPD reduces token usage by 10.4% while boosting average performance from 80.1% to 82.4%. Even on the already highly capable Qwen3-14B model, MPD successfully cuts reasoning length while improving accuracy across benchmarks.

Generalization to other model families. To verify whether our method generalizes beyond our setting, we apply the method to other model families, including the Llama family and the Mistral family. Particularly, we set Llama 3.2-3B (Team, 2024) as student with Llama 3.1-8B as teacher, as well as Ministral-3-3B-Reasoning-2512 (AI, 2026) as the student with Ministral-3-8B-Reasoning-2512 as the teacher. In addition, we also conduct experiments with a different generation, the Qwen2.5 family (Yang et al., 2024), across multiple model scales.

Table 10: Overall performance of MDP with scaling student models and teacher models. "–" means that we compare the student-teacher pair with the student vanilla LLM performance.

Student	Teacher	GSM8K P@4 / #Tok	AIME24 P@4 / #Tok	AIME25 P@4 / #Tok	CodeMMLU P@4 / #Tok	LongBench2 P@4 / #Tok	Avg. P@4	Avg. #Tok
Qwen3-1.7B	–	94.5 / 2.1k	63.3 / 16.8k	53.3 / 17.7k	81.2 / 3.6k	48.7 / 0.9k	68.2	8.2k
	Qwen3-32B	94.5 / 1.4k	70.0 / 13.2k	50.0 / 13.6k	83.8 / 2.3k	50.9 / 0.8k	**69.8**	**6.3k**
Qwen3-8B	–	97.1 / 2.1k	80.0 / 14.7k	83.3 / 17.1k	86.9 / 3.4k	53.1 / 1.0k	80.1	7.6k
	Qwen3-32B	97.3 / 1.5k	90.0 / 13.0k	83.3 / 15.9k	87.4 / 2.8k	53.9 / 0.9k	**82.4**	**6.8k**
Qwen3-14B	–	96.8 / 1.7k	90.0 / 13.4k	83.3 / 15.8k	87.8 / 2.9k	56.1 / 0.9k	82.8	6.9k
	Qwen3-32B	96.8 / 1.6k	90.0 / 13.0k	83.3 / 15.8k	87.7 / 2.9k	56.5 / 0.8k	**82.9**	**6.8k**

Table 11: Overall performance of MPD across different model families. "–" means that we compare the student-teacher pair with the student vanilla LLM performance.

Student Model	Teacher Model	GSM8K P@4 / #Tok	AIME24 P@4 / #Tok	AIME25 P@4 / #Tok	CodeMMLU P@4 / #Tok	LongBench2 P@4 / #Tok	Avg. P@4 ↑	Avg. #Tok ↓
Llama 3.2-3B	–	80.5 / 0.3k	13.3 / 4.3k	0.0 / 3.0k	75.1 / 0.5k	53.9 / 0.3k	44.6	1.7k
	Llama 3.1-8B	82.3 / 0.3k	16.7 / 3.8k	0.0 / 2.8k	74.2 / 0.5k	54.9 / 0.3k	**45.6**	**1.5k**
Ministral-3-3B	–	94.7 / 2.0k	63.3 / 19.9k	73.3 / 23.2k	86.2 / 4.5k	53.1 / 0.6k	74.1	10.0k
	Ministral-3-8B	95.6 / 1.4k	70.0 / 18.9k	70.0 / 19.5k	86.1 / 3.5k	49.1 / 0.4k	**74.2**	**8.7k**
Qwen2.5-1.5B	–	86.2 / 0.3k	6.7 / 3.2k	0.0 / 1.5k	74.9 / 0.5k	45.1 / 0.2k	42.6	1.1k
	Qwen2.5-14B	87.4 / 0.3k	13.3 / 2.5k	3.3 / 1.2k	74.4 / 0.4k	41.8 / 0.2k	**44.0**	**0.9k**
Qwen2.5-3B	–	93.0 / 0.4k	13.3 / 2.2k	13.3 / 1.3k	79.4 / 0.4k	61.0 / 0.4k	52.0	0.9k
	Qwen2.5-14B	93.4 / 0.3k	16.7 / 2.1k	16.7 / 1.2k	78.6 / 0.4k	61.4 / 0.5k	**53.4**	**0.9k**
Qwen2.5-7B	–	94.5 / 0.3k	16.7 / 2.0k	16.7 / 0.9k	80.5 / 0.4k	53.1 / 0.4k	52.3	0.8k
	Qwen2.5-32B	93.6 / 0.3k	23.3 / 1.3k	23.3 / 1.0k	78.4 / 0.4k	55.3 / 0.4k	**54.8**	**0.7k**
Qwen2.5-14B	–	96.7 / 0.3k	16.7 / 1.5k	23.3 / 0.9k	84.1 / 0.4k	55.3 / 0.4k	55.2	0.7k
	Qwen2.5-32B	96.5 / 0.3k	23.3 / 1.2k	26.7 / 0.9k	83.4 / 0.3k	55.7 / 0.4k	**57.1**	**0.6k**

The results are shown in Table. 11. On Llama 3.2-3B, MPD successfully reduces token consumption (down to 1.5k) while improving average accuracy from 44.6% to 45.6%. On Ministral-3-3B-Reasoning, MPD achieves substantial token reduction, slashing average token usage from 10.0k to 8.7k, while perfectly preserving reasoning performance.

Across various scales of Qwen2.5, from smaller model pairs (1.5B student with 14B teacher) up to relatively larger model pairs (14B student with 32B teacher), MPD consistently achieves efficient length reduction while matching or enhancing average performance. For instance, Qwen2.5-14B paired with a 32B teacher boosts average accuracy from 55.2% to 57.1% while cutting token usage from 0.7k to 0.6k. These diverse family results demonstrate that MPD is a broadly applicable compression method.

B.2 Comparison with SFT on Compressed Reasoning Traces

We conduct further experiments using the Supervised Fine-Tuning (SFT) training strategy. We trained the student model (Qwen3-1.7B) on MPD's teacher-rewritten compressed traces using SFT while keeping the training environment. The empirical results in Table 12 show that plain SFT severely degrades and collapses on heavily compressed traces, whereas MPD succeeds remarkably. Directly training on MPD's compressed traces with plain SFT leads to rapid reasoning degeneration. From Step 50 to 200, average accuracy collapses from 65.9% to 43.3%, with AIME25 dropping to just 3.3%. This indicates that plain SFT overfits to extreme brevity, shrinking average outputs to merely very small tokens and stripping away necessary step-by-step reasoning.

These results demonstrate that highly compressed reasoning traces are not intrinsically suitable as direct SFT training data. Methods that rely solely on SFT struggle to balance length reduction and mathematical correctness. MPD's framework bridges this gap, proving that token-level KL alignment is crucial when learning from heavily shortened reasoning traces.

Table 12: Comparison of Vanilla, SFT, and MPD across five benchmarks.

Method	GSM8K P@4 / #Tok	AIME24 P@4 / #Tok	AIME25 P@4 / #Tok	CodeMMLU P@4 / #Tok	LongBench2 P@4 / #Tok	Avg. P@4	Avg. #Tok
Qwen3-1.7B (Vanilla)	94.5 / 2.1k	63.3 / 16.8k	53.3 / 17.7k	81.2 / 3.6k	48.7 / 0.9k	68.6	8.2k
SFT (Step 50)	93.9 / 1.6k	66.7 / 12.4k	40.0 / 12.8k	82.1 / 2.6k	46.7 / 0.7k	65.9	6.0k
SFT (Step 100)	87.8 / 0.2k	53.3 / 1.2k	10.0 / 0.9k	73.8 / 0.5k	46.7 / 0.4k	54.3	0.6k
SFT (Step 150)	88.1 / 0.2k	8.3 / 1.0k	6.7 / 1.0k	76.8 / 0.4k	44.7 / 0.3k	44.9	0.6k
SFT (Step 200)	87.1 / 0.2k	3.3 / 0.4k	3.3 / 0.4k	76.4 / 0.2k	46.3 / 0.2k	43.3	0.3k
MPD (Ours)	94.2 / 1.4k	73.3 / 13.4k	56.7 / 12.9k	82.8 / 2.3k	50.3 / 0.7k	71.3	6.1k

Table 13: Performance of MPD under cross-family settings with Llama 3.1-8B as the student model, Qwen3-8B and Qwen3-32B as the teacher.

Teacher Model	GSM8K P@4 / #Tok	AIME24 P@4 / #Tok	AIME25 P@4 / #Tok	CodeMMLU P@4 / #Tok	LongBench2 P@4 / #Tok	Avg. P@4	Avg. #Tok (k)
–	88.3% / 0.4k	20.0% / 8.6k	0.0% / 7.8k	80.1% / 0.6k	**56.5%** / 0.4k	49.0%	3.6k
Qwen3-8B	92.3% / 0.3k	16.7% / 4.3k	0.0% / 3.5k	80.5% / 0.6k	53.5% / 0.2k	48.6%	1.8k
Qwen3-32B	**93.3% / 0.3k**	**20.0% / 3.8k**	**3.3% / 3.0k**	**81.3% / 0.5k**	54.1% / **0.2k**	**50.4%**	**1.5k**

B.3 Experiments on Cross-family Transfer

We further evaluate our method in cross-family settings, where the teacher and student models belong to different model families. Notably, the MPD framework does not require the teacher and student to come from the same model family.

Since different model families typically employ different tokenizers, resulting in different vocabularies, token segmentation, token IDs, and ultimately different output probability spaces (Patiño et al., 2025; Boizard et al., 2025), the teacher and student no longer predict distributions over the same token space, making the logit-level distillation objective itself non-trivial. Therefore, we conduct a preliminary exploration with the existing approach Universal Logit Distillation (ULD) (Patiño et al., 2025) implemented version from Huggingface TRL to bridge token alignment.

ULD handles cross-tokenizer distillation by using the same text as a shared reference. Because the two tokenizers may split the text into different numbers of tokens, this method assigns each token an offset, which refers to the span that the sub-word token covers. It then groups student and teacher tokens that cover the same portion of the original text. ULD also merges the probabilities of multiple sub-tokens when they correspond to one token in another tokenizer. Since this method cannot match all tokens, those unmatched vocabularies are still not able to be used for distribution-based distillation methods, and an additional cross-entropy term is then often added to the training objective as an explicit token-level matching signal.

We use Llama-3.1-8B as the student and Qwen3-8B as well as 32B as the teacher, ULD with default parameters including a cross-entropy coefficient of 0.5, together with our KL-divergence training objective. As shown in Table 13, the trained Llama 3.1-8B model reduces token consumption by more than half while improving performance with Qwen3-32B as the teacher model. Notably, on GSM8K, accuracy increases significantly from 88.3% to 93.3%, confirming that concise reasoning behaviour transferred across families does not sacrifice mathematical rigour. Our results indicate the applicability of MPD under the cross-family, cross-tokenizer scenarios, suggesting that MPD's compression signal is not inherently limited to intra-family teacher-student pairs.

C Evaluation of Trajectories after Compression

To further prove that our teacher compression step preserves fine-grained mathematical constraints, key equations, or logical step dependencies, apart from comparing the similarities of sentence embeddings, we further conduct an additional statistical evaluation on randomly sampled 1000 student-teacher trajectory pairs used in training : (1) correctness of trajectory: verifying whether any correct student reasoning paths were degraded after compression; (2) LLM-as-Judge-based trajectory evaluation with a fine-grained evaluation rubric: using the LLM (Gemini3.1-flash-lite (Team, 2023))

Table 14: Correctness (Accuracy) transition statistics before and after teacher compression. Student and Teacher Accuracy denote the accuracy before and after compression, respectively. False means incorrect and True means correct.

Metric / Category	Ratio (%)
Student Accuracy	**57.98**
Teacher Accuracy	**58.08**
True → True	57.90
True → False	0.00
False → True	0.19
False → False	41.92

to check the preservation of essential constraints, equations, transformations, logical dependencies at a step level and faithfulness.

C.1 Correctness of trajectory

We track the correctness transitions between the student's raw trajectory and the teacher's compressed trajectory. The result is shown in Table 14.

No case turned from correct to incorrect after teacher compression. This provides empirical proof that MPD's teacher compression preserves 100% of the student's original capability in the problems it can solve. The teacher's compressed traces achieved an overall accuracy of 58.08%, slightly surpassing the student's raw accuracy (57.98%). The 0.19% False→True transitions show that condensing messy, overthought steps can occasionally eliminate misleading logical noise. Combining this answer-correctness check with our sentence-level semantic analysis confirms that MPD compresses only the redundant/repetitive reasoning chatter and overthinking while strictly keeping the core mathematical equations, constraints, and logical dependencies intact.

C.2 Fine-grained Evaluation of Trajectories using LLM-as-Judge

To verify whether the compressed traces faithfully follow the student's underlying reasoning path , we employ an LLM-as-Judge auditor to assess faithfulness at two levels of granularity: an overall five-level classification and a multi-dimensional score-based evaluation across five dimensions.

For the overall assessment, we categorize each compressed trace into one of five levels: Faithful, Mostly Faithful, Partially Faithful, Unfaithful, and Contradictory. Faithful indicates that all essential logic, equations, and dependencies are preserved; Mostly Faithful denotes only minor, non-critical omissions; Partially Faithful indicates that the main reasoning path is retained but some dependencies are weakened or omitted; Unfaithful denotes that essential reasoning is replaced by a substantially different reasoning route; and Contradictory indicates that the compressed trace violates an essential constraint or conclusion in the student's original reasoning. The results are shown in Table 15a. Overall, the vast majority of compressed reasoning traces are classified as Faithful, indicating that the compression procedure generally preserves the core reasoning structure of the original student solutions.

In addition to the categorical evaluation, we assess the compressed trajectories along five fine-grained dimensions: Constraint Preservation, Equation Preservation, Logical Consistency, Dependency Preservation, and Faithfulness to Student. These dimensions respectively measure the retention of problem constraints, key mathematical transformations, deductive consistency, inter-step dependencies, and the student's original solution route. As shown in Table 15b, the compressed trajectories achieve consistently high scores across all five dimensions, ranging from 4.88 to 4.92 out of 5.0. Constraint Preservation and Equation Preservation obtain the highest mean scores of 4.92, while Logical Consistency, Dependency Preservation, and Faithfulness to Student remain similarly high at 4.89, 4.88, and 4.88, respectively. Combined with the categorical results, where 93.91% of traces are classified as Faithful, these results indicate that the compressed trajectories preserve both the mathematical structure and the original reasoning path to a high degree despite substantial compression.

Table 15: Faithfulness evaluation of compressed reasoning traces.

(a) Overall classification

Classification	Ratio (%)
Faithful	93.91
Mostly Faithful	2.36
Partially Faithful	2.43
Unfaithful	0.62
Contradictory	0.68

(b) Fine-grained evaluation

Evaluation Dimension	Mean Score (5)
Constraint Preservation	4.92
Equation Preservation	4.92
Logical Consistency	4.89
Dependency Preservation	4.88
Faithfulness to Student	4.88

Table 16: Logical-unit preservation for the mathematic case.

Logical Unit	Student Trace Evidence	Teacher Rewrite Evidence	Preserved?
Target event reduced to complement	The complement of "at least two cards of the same color" is "all four cards are of different colors."	We consider the complement event: all four cards being of different colors.	Yes
Choose color set	Choose 4 different colors from the 5 available: $\binom{5}{4} = 5$.	Choose 4 different colors from the 5 available, which is $\binom{5}{4} = 5$.	Yes
Choose one card per color	Choose one card from each of the 4 chosen colors: $10^4 = 10{,}000$.	Choose one card from each of the 4 colors, giving $10^4 = 10{,}000$ ways.	Yes
Total sample space	The total number of ways to choose 4 cards from 50 is $\binom{50}{4} = 230{,}300$.	The total number of ways to draw any 4 cards from the 50-card deck is $\binom{50}{4} = 230{,}300$.	Yes
Final dependency	$P(\text{at least two same}) = 1 - \frac{500}{2303} = \frac{1803}{2303}$.	$1 - \frac{500}{2303} = \frac{2303-500}{2303} = \frac{1803}{2303}$.	Yes

C.3 Case Study of Key Steps Preservation

We present here two cases showing whether the underlying mathematical reasoning units are faithfully preserved. We compare the original and rewritten traces at the level of individual logical units.

In the first case, we consider the following mathematics problem:

> **Problem:** A deck of 50 cards consists of 5 different colors, with 10 cards of each color. If 4 cards are drawn randomly from the deck, what is the probability that at least 2 cards will be of the same color? Express your answer as a common fraction.
>
> **Answers:** Ground truth: $\frac{1803}{2303}$; Student: $\frac{1803}{2303}$; Teacher: $\frac{1803}{2303}$.
>
> **Compression:** Teacher/student character ratio: 0.020.

We decompose the corresponding reasoning process into five key logical units: (1) reducing the target event to its complement, (2) choosing four distinct colors, (3) choosing one card from each selected color, (4) computing the total number of possible four-card draws, and (5) combining these quantities to derive the final probability. As shown in Table 16, the compressed trace preserves every essential logical unit used in the student's solution. In particular, it retains the reduction to the complement event, the selection of four distinct colors, the choice of one card from each color, the construction of the total sample space, and the final dependency leading to the answer. None of the key equations or reasoning dependencies is lost, while the teacher/student character ratio reaches 0.020. This example illustrates that the compression removes largely redundant exposition while preserving the mathematical structure necessary to reproduce the original solution.

In the second case study, we consider the following geometry problem:

Table 17: Logical-unit preservation for the geometry case.

Logical Unit	Student Trace Evidence	Teacher Rewrite Evidence	Preserved?
Radius relation	For a regular tetrahedron, inradius r and circumradius R satisfy $R = 3r$.	In a regular tetrahedron, the relationship between these two radii is $R = 3r$.	Yes
Objects counted	There are four face-tangent spheres and one inscribed sphere.	The problem introduces five small spheres: four tangent to faces and one inscribed sphere.	Yes
Introduce unknown radius	Let the radius of each face-tangent small sphere be x.	Let the radius of each such small sphere be x.	Yes
Tangency constraint	Tangency to the circumscribed sphere gives $R - x$, while the face-to-center distance gives $r + x$.	Since the small sphere is tangent to the circumscribed sphere, $R - x = r + x$.	Yes
Solve dependency	$R - x = r + x$ and $R = 3r$ imply $x = r$.	$R = 2x + r \Rightarrow 3r = 2x + r \Rightarrow x = r$.	Yes
Volume ratio	$P = \frac{5\left(\frac{4}{3}\pi r^3\right)}{\frac{4}{3}\pi(3r)^3} = \frac{5}{27}$.	$P = \frac{\frac{20}{3}\pi r^3}{36\pi r^3} = \frac{5}{27}$.	Yes

Problem: A tetrahedron with four equilateral triangular faces has a sphere inscribed within it and a sphere circumscribed about it. For each of the four faces, there is a sphere tangent externally to the face at its center and to the circumscribed sphere. A point is chosen at random inside the circumscribed sphere. What is the probability that the point lies inside one of the five small spheres? Express your answer as a common fraction.

Answers: Ground truth: $\frac{5}{27}$; Student: $\frac{5}{27}$; Teacher: $\frac{5}{27}$.

Compression: Teacher/student character ratio: 0.027.

We decompose the reasoning process into six key logical units: (1) using the radius relation between the inradius and circumradius of a regular tetrahedron, (2) identifying the five small spheres involved in the probability calculation, (3) introducing the radius of a face-tangent sphere as an unknown, (4) formulating the tangency constraint relating this radius to the inradius and circumradius, (5) solving this constraint to show that the face-tangent spheres have the same radius as the inscribed sphere, and (6) computing the desired probability from the ratio of the total volume of the five small spheres to the volume of the circumscribed sphere. As shown in Table 17, all six logical units are preserved in the compressed teacher rewrite. In particular, the rewrite retains the critical geometric dependency $R - x = r + x$, together with the regular-tetrahedron relation $R = 3r$, which jointly imply $x = r$. The compressed trace therefore preserves not only the final equations, but also the geometric constraints and intermediate dependencies required to derive the correct probability. This case further illustrates that substantial compression can remove redundant exposition while maintaining the essential mathematical structure of the original reasoning.

C.4 Evaluation Prompts

We provide the prompts used in our LLM-based evaluation below. The system prompt in Table 19 defines the role and evaluation principles of the mathematical reasoning auditor, while the user prompt in Table 23 specifies the input format, evaluation criteria, scoring rubric, and required output schema.

D Does Shorter Reasoning Correspond to Better Reasoning?

We also study how much of MPD's gain comes from producing shorter reasoning traces.

To isolate whether MPD genuinely learns to produce more concise reasoning traces or merely appears shorter by dropping long, failed trajectories, we evaluate the average token length strictly among correct answers only across all five evaluation benchmarks.

Table 18: Average length of correct reasoning of the vanilla model and MPD across benchmarks.

Benchmark	Vanilla Student	MPD	Token Reduction (%)
GSM8K	1.82k	**1.36k**	**-25.01**
AIME 2024	11.27k	**10.35k**	**-8.13**
AIME 2025	10.55k	**7.59k**	**-28.12**
CodeMMLU	2.92k	**2.23k**	**-23.60**
LongBench2	0.90k	**0.88k**	**-2.80**
Overall Average	**5.49k**	**4.48k**	**-18.40**

Table 19: System Prompt for Mathematical Reasoning Auditor.

System Prompt for Mathematical Reasoning Auditor
You are a strict mathematical reasoning auditor. Your task is NOT to solve the problem from scratch and NOT to reward shortness. You compare a student's full trajectory with a teacher's compressed trajectory. The teacher trajectory should be judged as a compression/rewrite of the student trajectory. Focus on whether key mathematical constraints, equations, transformations, case distinctions, and logical dependencies from the student trajectory are preserved. Ignore wording style and harmless omission of routine arithmetic, but penalize omissions that make the teacher's reasoning unsupported, after-the-fact, or logically different. Return only valid JSON with the requested schema.

The empirical comparison between the Vanilla Student and the MPD model on correctly solved problems is summarized in Table 18. Even when restricting the evaluation exclusively to correct solution traces, MPD consistently consumes fewer tokens than the vanilla student model across all five benchmarks, achieving an overall average token reduction of 18.40% and up to 28.12% on challenging tasks (e.g. AIME 2025). This indicates that MPD's efficiency gains are not solely explained by avoiding long failed trajectories. Instead, MPD intrinsically improves the student model's reasoning efficiency by teaching it to eliminate redundant verbal chatter, self-correction loops, and overthinking while strictly maintaining—and in fact boosting—overall solution correctness.

E Case Study: MPD vs. Baselines & Ablations

We present a case study on an AIME24 question comparing MPD with On-Policy, On-Policy (C), and Off-Policy (C) under the setting of ablation studies in Sec. 4.2 . As shown in Table 20, MPD is the only method that is both correct and concise. Compared with On-Policy, which eventually reaches the correct answer but spends 13.3k tokens with repeated verification and unnecessary detours, MPD obtains the same answer using only 6.2k tokens, reducing the reasoning length by 53.38%. On-Policy (C) shortens the response to 7.7k tokens, but drops a crucial geometric constraint and incorrectly concludes that the answer is 80. Off-Policy (C) is even longer, using 22.2k tokens, and still fails by following a confusing parametrization that leads to the same wrong answer. These failures show that compression without reliable reasoning can remove essential structure, while teacher-generated compressed traces alone may not align well with the student's own problem-solving process. MPD combines student-side exploration with teacher-guided compression, thus achieving concise reasoning without discarding the key intermediate structure needed for correctness.

We further examine how the model trained with MPD differs from the vanilla LLM in its reasoning behavior. Table 21 illustrates a case where the Vanilla LLM produces a long but ultimately incorrect reasoning trajectory. The Vanilla response repeatedly assumes that the valid sets B behave like subsets of A, leading to the erroneous equation $2^m - 1 = 2024$. Although the model later recognizes that this interpretation may be problematic, its self-reflection does not recover the correct counting principle. Instead, it continues to reason around the contradiction, eventually hypothesizing a typo in the problem and returning the wrong answer 66. In contrast, MPD identifies the key combinatorial structure directly: for a fixed maximum $x \in A$, there are 2^{x-1} possible sets. This reduces the problem to the binary expansion of 2024, yielding $A = \{4, 6, 7, 8, 9, 10, 11\}$ and the correct answer

Table 20: Case study comparing MPD with On-Policy, On-Policy (C), and Off-Policy (C). Text in **red** marks inefficient or incorrect reasoning, while text in **blue** marks MPD's key correct reasoning.

Case: MPD vs. On-Policy, On-Policy (C), and Off-Policy (C).

Question: Let A, B, C, and D be points on the hyperbola $\frac{x^2}{20} - \frac{y^2}{24} = 1$ such that $ABCD$ is a rhombus whose diagonals intersect at the origin. Find the greatest real number that is less than BD^2 for all such rhombi.

Ground-Truth: $\boxed{480}$

On-Policy Response (13.3k tokens, correct but verbose): Okay, so I need to solve this problem about a rhombus inscribed in a hyperbola. ... **Wait, so BD^2 can take various values, and we need to find the greatest number that is less than BD^2 for all such rhombi. That is, the infimum of BD^2? Wait, no.** ... After a long derivation, OPD obtains $BD^2 = 480 \cdot \frac{t}{t - \frac{600}{11}}$. Now, as $t \to \infty$, the expression approaches 480. Thus, the infimum of BD^2 is 480, and since $BD^2 > 480$ for all such rhombi, the greatest real number less than BD^2 for all such rhombi is $\boxed{480}$.

On-Policy (C) Response (7.7k tokens, incorrect): Okay, so I need to solve this problem about a rhombus inscribed in the hyperbola $x^2/20 - y^2/24 = 1$. ... **Therefore, $BD^2 = 4(r^2 + s^2) = 80 + \frac{22}{3}s^2$. So, since s^2 is non-negative, BD^2 is at least 80. Therefore, the minimal BD^2 is 80.** ... **Therefore, boxed answer is $\boxed{80}$.**

Off-Policy (C) Response (22.2k tokens, incorrect): Okay, so I need to solve this problem about a rhombus inscribed in a hyperbola. ... **This is very confusing. Given that the problem states such a rhombus exists, I must have made a mistake in my reasoning.** ... **Therefore, $BD^2 = 4/k$, where $k \in [-1/24, 1/20]$. Since k is in this range, and $BD^2 = 4/k$, when k is negative, BD^2 is negative, which is impossible.** ... **Therefore, the answer is 80, which is $\boxed{80}$.**

MPD Response (6.2k tokens, correct): Okay, let's try to tackle this problem. ... **BD^2 is $16(5\sec^2\phi + 6\tan^2\phi)$.** The diagonals are perpendicular, giving $\sin\theta\sin\phi = -5/6$. ... **Therefore, $BD^2 = 16(5 + 11\tan^2\phi)$. If $\tan^2\phi \geq 25/11$, then $5 + 11\tan^2\phi \geq 30$, so $BD^2 \geq 16 \cdot 30 = 480$. Therefore, the answer is $\boxed{480}$.**

55. This example shows that longer reasoning is not necessarily more reliable; redundant reflection can amplify an early misconception, while MPD preserves the decisive reasoning steps.

Table 22 shows a complementary case where both methods reach the correct answer, but with substantially different reasoning efficiency. The Vanilla LLM derives $xy = 25$ early, but then spends many additional steps questioning whether the shortcut is valid, setting up a transcendental equation, and attempting numerical verification. Since the problem already states that such $x, y > 1$ exist, these extra steps are unnecessary for determining xy. MPD instead introduces $k = \log_y x$, immediately obtains $x = 10k$ and $y = \frac{5}{2k}$, and directly concludes $xy = 25$. Thus, MPD not only reduces token usage, but also encourages more goal-directed reasoning by retaining the minimal algebraic path needed to solve the problem.

Overall, these case studies suggest that MPD improves reasoning efficiency in two ways. First, it can avoid verbose but unproductive self-reflection that fails to correct an initial error. Second, even when the baseline eventually answers correctly, MPD can remove unnecessary verification detours and reach the answer with a much shorter reasoning trajectory.

F Limitations & Future Directions

Incorporating Correctness Objective. Our training approach provides a dense and efficient learning signal. It ensures that the training trajectories remain aligned with the student's intrinsic reasoning patterns while being concise. However, the correctness of the trajectories is not explicitly incorporated into the training objective. As a result, the model may still learn from regions where both the student and the teacher are incorrect. Incorporating correctness-aware signals into the training process may further improve performance.

Multi-Turn Compression. Effective compression requires the student to generate a complete trajectory, and the teacher's rewriting process also relies on access to the full trajectory. However,

Table 21: Case study where Vanilla LLM produces a long self-reflective but incorrect solution. Text in **red** marks the erroneous assumption, repeated confusion, and wrong final answer; text in **blue** marks MPD's key correct reasoning path.

Case 1: Vanilla LLM is long, self-reflective, and incorrect; MPD is concise and correct.

Question: Alice chooses a set A of positive integers. Then Bob lists all finite nonempty sets B of positive integers with the property that the maximum element of B belongs to A. Bob's list has 2024 sets. Find the sum of the elements of A.

Ground-Truth: $\boxed{55}$

Vanilla LLM Response (26.5k tokens): Okay, let's try to tackle this problem step by step. ... Wait, if A is a set of positive integers, let's suppose that A has elements $a_1, a_2, \ldots, a_m$, sorted in increasing order. Then, for each a_i, the number of elements in A less than a_i is $i - 1$. Therefore, **the total number of subsets B is** $\sum_{i=1}^{m} 2^{i-1} = 2^m - 1$. ... So then, if the problem says that the number of subsets is 2024, then $2^m - 1 = 2024$**. Therefore,** $2^m = 2025$**. But 2025 is not a power of 2.** ... Oh! That's a crucial mistake. So, the problem is not about subsets of A, but about all finite nonempty sets B where the maximum element of B is in A. ... **Wait, I'm stuck. Given that, and since the problem must have a solution, I will assume that there is a typo and the number of sets is 2047, which is** $2^{11} - 1$**.** ... **But this is getting too long, and I don't see how this helps.** Given that, I think the problem must have a typo, and the correct number of sets is 2047, which is $2^{11} - 1$, and the sum of the elements of A is $1 + 2 + \cdots + 11 = 66$. Therefore, the answer is 66. ... **Final Answer:** $\boxed{66}$

MPD Response (5.2k tokens): Okay, let's try to figure out this problem. ... **But wait, if x is the maximum, then the subset must contain x and any combination of elements less than x. So, for each x in A, the number of subsets where the maximum is x is 2^{x-1}.** But then, the total number of subsets is the sum over all elements in A of 2^{x-1}. ... So, the formula seems to hold. Therefore, if A has elements $a_1, a_2, \ldots, a_k$, then the total number of subsets is $\sum_{i=1}^{k} 2^{a_i - 1} = 2024$. ... **Well, since 2024 is a sum of distinct powers of 2, as we saw, the set $\{b_i\}$ would be the exponents in the binary representation of 2024.** ... **If we have the exponents in the binary representation of 2024 as $3, 5, 6, 7, 8, 9, 10$, then the sum is $2^3 + 2^5 + 2^6 + 2^7 + 2^8 + 2^9 + 2^{10}$.** ... Therefore, the corresponding b_i are these exponents, and then $a_i = b_i + 1$. Therefore, the elements of A are $4, 6, 7, 8, 9, 10, 11$. ... **Final Answer:** $\boxed{55}$

in scenarios with limited context windows, the student may fail to produce the complete trajectory, which prevents our method from reaching its full potential. In such cases, multi-step compression can help maintain the quality of the compressed trajectories by allowing the teacher to iteratively refine partial outputs, thereby improving the overall learning effectiveness.

G Potential Societal Risks

Positive Societal Impact. Our work contributes to improving the efficiency of reasoning in large language models. By enabling concise yet effective reasoning, our approach reduces the number of tokens required during inference, which can lower computational cost and energy consumption. This improves the scalability and accessibility of reasoning models, making them more practical for deployment in resource-constrained settings. In addition, by facilitating the transfer of reasoning capabilities to smaller models, our method can help broaden access to advanced AI systems, potentially benefiting applications in education and low-resource environments. More broadly, our work encourages research on efficient reasoning, promoting approaches that balance performance and computational efficiency.

Potential Societal Risks. Methods based on model distillation or self-improving training may introduce certain risks. First, since correctness is not always explicitly enforced in the training objective, models may learn from trajectories that are incorrect, potentially leading to confident yet erroneous outputs. Second, as these approaches rely on stronger models to guide training, any biases or systematic errors present in the teacher model may be inherited or even amplified in the student model. These risks are broadly shared across distillation-based training paradigms. Addressing

Table 22: Case study where both methods answer correctly, but Vanilla LLM spends many extra steps verifying a product already forced by the equations. Text in **red** marks inefficient detours; text in **blue** marks the direct reasoning that quickly leads to the correct answer.

Case 2: Both methods are correct, but MPD reaches the answer much more efficiently.

Question: There exist real numbers x and y, both greater than 1, such that $\log_x(y^x) = \log_y(x^{4y}) = 10$. Find xy.

Ground-Truth: $\boxed{25}$

Vanilla LLM Response (8.5k tokens): Okay, so I need to solve this problem where there are real numbers x and y, both greater than 1. ... So from equation 1, $x = 10/a$. From equation 2, $4y = 10a \Rightarrow y = (10a)/4 = (5a)/2$. ... $xy = (10/a)(5a/2) = 25$. Wait, that's interesting! The a cancels out, so $xy = 25$? But that seems too straightforward. Let me check that again. ... **Suppose that $xy = 25$. Let me check if there exist x and y greater than 1 such that $\log_x(y^x) = \log_y(x^{4y}) = 10$.** ... **This is a transcendental equation in a. It might not have an algebraic solution, but maybe there is a specific value of a that satisfies this equation.** ... Wait, earlier I thought that $xy = 25$ regardless of a. Let me check that again. ... **But to confirm, let me try to find specific x and y. Suppose that $xy = 25$. Let me pick $x = 2$, then $y = 12.5$.** ... Therefore, $a \approx 6.515$. Then $x = 10/a \approx 1.534$, and $y = 5a/2 \approx 16.2875$. ... **Final Answer:** $\boxed{25}$

MPD Response (2.6k tokens): Okay, let's try to solve this problem. So, we have two logarithmic equations. ... Using the power rule, that becomes $x \log_x(y) = 10$. ... Again, using the power rule, that's $4y \log_y(x) = 10$. ... **Now, let me denote $\log_y(x)$ as some variable to make it easier. Let me let $\log_y(x) = k$. Then, the first equation becomes: $x/k = 10 \Rightarrow x = 10k$. The second equation is $4y \cdot k = 10 \Rightarrow y \cdot k = 5/2$.** So, we have: from the first equation, $x = 10k$; from the second equation, $y = (5/2)/k$. ... $xy = 10k \cdot (5/2)/k = 10 \cdot (5/2) = 25$. ... **Final Answer:** $\boxed{25}$

them, for example by incorporating correctness-aware signals or improved verification mechanisms, remains an important direction for future work.

Table 23: User Prompt for Faithfulness Evaluation.

User Prompt for Faithfulness Evaluation

We need to evaluate whether the TEACHER trajectory is a faithful logical compression of the STUDENT trajectory for the same math problem.
Problem:
{problem}
Ground-truth final answer, if available:
{ground_truth}
Student extracted final answer:
{student_answer}
Teacher extracted final answer:
{teacher_answer}
STUDENT trajectory:
<<<STUDENT>>>
{student_trajectory}
<<<END_STUDENT>>>
TEACHER compressed trajectory:
<<<TEACHER>>>
{teacher_trajectory}
<<<END_TEACHER>>>
Audit instructions:
1. First identify the student's essential reasoning units: problem constraints, introduced variables, key equations/inequalities, transformations, case splits, and dependencies needed to reach the answer.
2. Check whether the teacher preserves each essential unit accurately. A compressed trajectory may omit low-level algebra, but it must not skip a necessary constraint or introduce a new unsupported route.
3. Penalize an after-the-fact summary if it states the answer with only generic or incomplete justification, even if the final answer matches.
4. Penalize contradictions, changed equations, changed assumptions, lost domain restrictions, lost case distinctions, or changed dependency order.
5. Do not mark the teacher unfaithful merely because it is shorter or more concise.
Return exactly this JSON schema:

```
{
  "essential_student_units": [
    {"id": "U1", "unit": "short description of a key constraint/equation/dependency"}
  ],
  "unit_preservation": [
    {"id": "U1", "status": "preserved|weakened|missing|contradicted", "explanation": "brief reason"}
  ],
  "scores": {
    "constraint_preservation": 1,
    "equation_preservation": 1,
    "dependency_preservation": 1,
    "logical_consistency": 1,
    "faithfulness_to_student": 1
  },
  "critical_issues": ["brief issue strings; empty if none"],
  "verdict": "faithful|mostly_faithful|partially_faithful|unfaithful|contradictory|uncertain",
  "after_the_fact_summary": false,
  "one_sentence_rationale": "brief rationale"
}
```

Scoring rubric:
1 = severe failure, 2 = major gaps, 3 = partial preservation, 4 = minor omissions only, 5 = all essential logic preserved.
Use "faithful" only when all essential units are preserved and faithfulness_to_student ≥ 4.
Use "mostly_faithful" when only minor non-critical units are weakened/missing.
Use "partially_faithful" when the final answer/path is similar but one or more important dependencies are missing or unsupported.
Use "unfaithful" when essential logic is missing or replaced by a different route.
Use "contradictory" when the teacher contradicts an essential student constraint/equation/dependency.
Use "uncertain" only when the trajectories are too ambiguous to judge.