REASON IN STYLE: DISCOVERING AND CONTROLLING STYLE IN LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Language models learn content and style jointly, making stylistic variation in their outputs difficult to identify and control. We study whether recurring styles in model responses can be discovered without supervision and explicitly controlled. We design an algorithm that learns to separate representations of content and style from language models' outputs and validate its effectiveness on math questions in a controlled setting. By applying this method to over 100K verified traces from nine distinct teacher models, we discover six recurring yet imbalanced styles. We then fine-tune smaller student models to follow these styles when explicitly conditioned on them, using importance weighting to balance the contribution of the styles represented in the corpus. This approach improves Pass@k over standard fine-tuning on the same data across six math reasoning benchmarks, demonstrating that we can diversify the style of answers effectively. We confirm that this also results in strong correspondence between requested and realized styles. We find that style affects correctness: the probability of solving a problem depends on the style we condition on, and different problems benefit from different styles. In summary, our results show that stylistic variation in model-generated data can be discovered in an unsupervised way, and made explicit, providing a source of both control and improved reasoning performance.

1 INTRODUCTION

Language models are trained to predict the next token in a sequence, an objective that does not explicitly distinguish between different aspects of the text being modeled. In particular, it does not separate *what* a response says from *how* it is expressed: semantic content is learned jointly with stylistic properties such as organization, verbosity, formatting, and lexical style. This entanglement becomes important when we want to analyze or control model behavior. A generated response may contain the desired semantic content while also exhibiting stylistic properties inherited from pretraining or post-training data. Separating these factors would allow us to characterize stylistic variation independently of content and control it explicitly during training and generation.

We study this problem in mathematical reasoning, where the distinction between content and style is particularly concrete. The same mathematical problem can be solved correctly while the solution is expressed in different ways. Different language models naturally produce such variation. However, there is no predefined taxonomy specifying which styles are present in a collection of model-generated solutions, and style is not necessarily tied to model identity: different models may produce similar styles, while the same model may produce multiple styles. Our goal is therefore to discover this structure automatically and represent style separately from problem content.

The need to make this stylistic structure explicit becomes particularly important as language models are increasingly post-trained on model-generated data that provides complete reasoning traces (Hsieh et al., 2023; Mukherjee et al., 2023; Feng et al., 2024). Fine-tuned models can inherit the structural and lexical properties of their training traces (Lippmann & Yang, 2025), even though the training objective does not explicitly represent style. Instead, these properties are learned implicitly from the examples themselves, so styles that occur frequently in the training data can dominate the model's behavior. At the same time, stylistic mismatch between the training data and the model's prior can make fine-tuning less effective or even degrade performance (Huang et al., 2026). Discovering and representing style explicitly would allow us to control which stylistic be-

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.

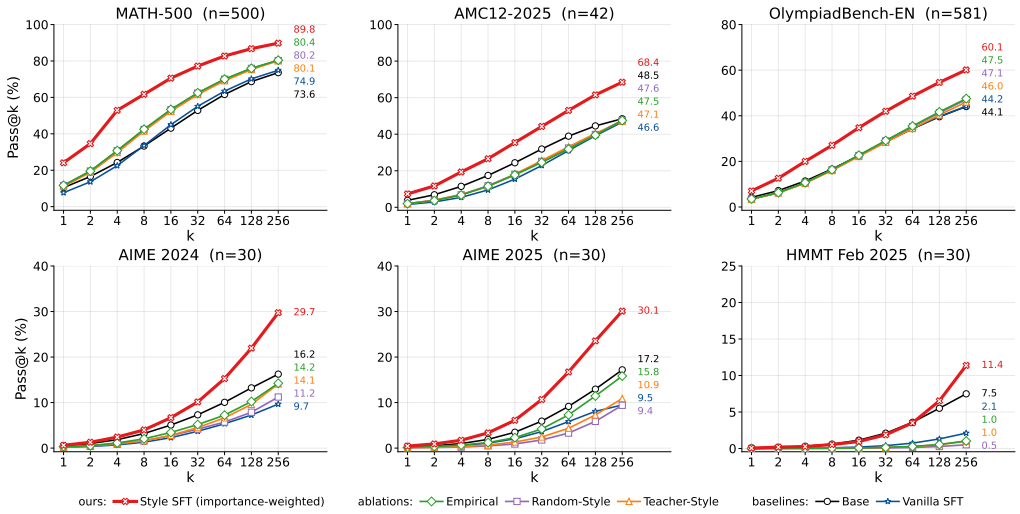


Figure 1: Pass@k for the Qwen3 0.6B student. STYLE SFT (IMPORTANCE-WEIGHTED) is our full method, combining discovered style conditioning with importance-weighted SFT. STYLE SFT (EMPIRICAL) conditions on the discovered style labels while retaining the empirical training distribution. RANDOM-STYLE SFT uses randomly assigned style labels as a control. VANILLA SFT uses the same training traces without style information. Our full method substantially improves Pass@k, while vanilla SFT often provides little benefit or degrades performance.

haviors are transferred during fine-tuning and how strongly they influence the resulting model, rather than permit the model to implicitly inherit them from the data composition.

To isolate style in reasoning traces, we first learn a representation that separates stylistic variation from problem-specific content using an autoencoder with an objective designed for disentanglement. We validate this representation in a controlled setting in which problems are answered in four known writing styles. Although the style labels are never used during training, the learned style representation recovers the hidden styles while the content representation becomes invariant to them. We then apply the same method to a dataset of 111,834 verified mathematical reasoning traces generated by nine models from Hu et al. (2026). Without assuming a predefined style taxonomy, we discover six recurring styles expressed by different models for multiple questions.

We next use these discovered styles to make these models style-aware. We label each training trace with its discovered style, and then fine-tune Qwen3 students of three sizes using style prefixes and importance weighting to balance the contribution of the styles observed for each problem. The resulting students are controllable and exhibit meaningful usage of the discovered style. Moreover, style-aware fine-tuning substantially improves Pass@k over standard fine-tuning on the same dataset across six mathematical reasoning benchmarks (Figure 1). The correctness of a solution appears to depend jointly on problem content and the style in which the model is conditioned to reason.

Taken together, our results provide a first demonstration that stylistic variation in model-generated reasoning can be discovered without supervision, separated from problem content, and exposed as an explicit control variable during fine-tuning and generation. Our approach shifts the role of style in language models from an incidental consequence of the training objective, to a representable and controllable component of model behavior.

2 RELATED WORK

Disentangling and controlling text style. A large body of work on text style transfer aims to modify how text is expressed while preserving its semantic content. Many approaches explicitly separate representations of content and style, including methods based on adversarial objectives, information-theoretic constraints, and structured latent spaces (Shen et al., 2017; John et al., 2019; Cheng et al., 2020; Sha & Lukasiewicz, 2024; Zhu et al., 2024; Han et al., 2024). However, these

methods typically assume a known style attribute, predefined domains, or a fixed set of style categories. Our setting instead assumes no style labels or predefined taxonomy. We use multiple independently generated solutions to the same problem as weak supervision for shared content, allowing recurring variation across solutions to be represented separately as style.

Style in model-generated training data. Model-generated reasoning traces exhibit systematic style variation in how solutions are organized and expressed across models and scales (Chen et al., 2025). Trace-level analyses further identify recurring behaviors such as exploration, verification, and backtracking (Bachmann et al., 2026; Zhang et al., 2026). Lippmann & Yang (2025) show that structural and lexical properties of teacher-generated reasoning traces are transferred to students. Huang et al. (2026) further show that stylistic mismatch between generated training data and the student model can affect fine-tuning performance. These results suggest that the style of training data can shape the behavior of the resulting model. Our work makes the stylistic variation explicit by discovering the styles present in the data.

Controlling solution modes. Sampling multiple solution trajectories can improve reasoning performance (Wang et al., 2023), motivating methods that explicitly encourage or control variation across reasoning paths. Wu et al. (2026) condition generation on distinct reasoning modes using mode-specific prefixes and show that allocating samples across modes can improve Pass@k. They further introduce gradient clustering to discover modes without predefined labels; on multi-teacher data, these clusters closely recover teacher identity. Relatedly, Jia et al. (2026) learn global forking tokens that specialize to different reasoning paths, allowing multiple solution modes to be preserved and elicited during generation. In our setting, teacher identity is not a reliable proxy for style: different teachers can produce similar styles, while a single teacher can produce multiple styles. We therefore learn stylistic structure directly from the traces, and use the discovered styles as controllable variables during fine-tuning and generation.

3 Unsupervised Learning of Style Representations

Our goal is to separate information about the underlying problem from variation in how a solution is expressed. We exploit the fact that independently generated traces (answers) for the same question share problem-specific information but may differ in organization, notation, verbosity, verification, and other stylistic properties. We use an autoencoder with a disentanglement objective to learn separate style and content representations, inspired by Makino et al. (2025).

Architecture. An encoder E maps each trace s to a representation $h(s)$, which is projected by two MLP heads into a content representation $c(s)$ and a style representation $z(s)$: $c(s) \in \mathbb{R}^{d_c}, z(s) \in \mathbb{R}^{d_z}$, with $d_c = d_z = 256$. A decoder reconstructs the original trace from both representations. We concatenate $[c(s); z(s)]$ and project it into $m = 8$ continuous prefix tokens that condition an autoregressive decoder. Reconstruction requires the two representations jointly to retain sufficient information about the trace.

Training objective. We train the model with the following objective:

$$\mathcal{L} = \lambda_c \mathcal{L}_c + \lambda_{\text{orth}} \mathcal{L}_{\text{orth}} + \lambda_{\text{rec}} \mathcal{L}_{\text{rec}} + \lambda_{\text{var}} \mathcal{L}_{\text{var}} + \lambda_{\text{cov}} \mathcal{L}_{\text{cov}} + \lambda_{\text{div}} \mathcal{L}_{\text{div}}. \tag{1}$$

Content agreement ($\mathcal{L}_c$). We use an InfoNCE objective (van den Oord et al., 2018; Chen et al., 2020) in which the two traces generated for the same question form a positive pair and traces from other questions in the minibatch serve as negatives. This encourages c to capture information that is shared across independently generated solutions to the same problem.

Content–style decorrelation ($\mathcal{L}_{\text{orth}}$). We penalize the squared cosine similarity between $c(s)$ and $z(s)$ for the same trace. This discourages the two heads from learning aligned representations and encourages z to capture information complementary to the question-specific content represented in c.

Reconstruction ($\mathcal{L}_{\text{rec}}$). The decoder reconstructs the original trace from the continuous prefix derived from $[c(s); z(s)]$. This ensures that the two representations remain jointly informative about the observed solution rather than discarding trace-level information.

Variance and covariance regularization ($\mathcal{L}_{\text{var}}, \mathcal{L}_{\text{cov}}$). To prevent the style representation from collapsing, we use VICReg-style variance and covariance regularization (Bardes et al., 2022). The variance term encourages each dimension of z to maintain nontrivial variation across the minibatch, while the covariance term discourages redundant dimensions from becoming highly correlated.

Within-question diversity ($\mathcal{L}_{\text{div}}$). Finally, we discourage the style representations of the two solutions to the same question from collapsing together. While $\mathcal{L}_c$ pulls same-question content representations together, $\mathcal{L}_{\text{div}}$ penalizes positive cosine similarity between their corresponding z representations. This prevents z from simply becoming another representation of question identity and leaves room for differences between independently generated solutions to be represented as style.

Together, the losses encourage c to capture information that is stable across solutions to the same problem, while z captures complementary trace-level variation. Full definitions of each objective, loss weights, and implementation details are provided in Appendix A.

4 Style Discovery

4.1 Recovering Known Styles

Before applying our representation to unlabeled teacher traces, we first test whether it can recover a known source of stylistic variation without using style labels during training.

Controlled setup. We prompt Gemini 3.1 Pro (Google DeepMind, 2026) to solve 300 math problems in four explicitly specified writing styles: academic prose, tutor-like explanations, algorithmic step-by-step lists, and equation-dominated solutions. Each problem is generated in all four styles, yielding 1,200 traces and ensuring that style is not confounded with question identity. Style labels are used only for evaluation. We train the autoencoder using the objective described above and obtain a content representation c and a style representation z.

Recovering style without supervision. The learned representation separates stylistic variation from problem content. Clustering z with k-means using $k = 4$ recovers the four ground-truth styles with ARI 0.77 and NMI 0.78. In contrast, clustering c shows essentially no agreement with style. A linear probe similarly predicts style from z with 96.7% accuracy, while accuracy from c is 23.3%, near the 25% chance level. This separation does not emerge from reconstruction alone. With reconstruction-only training, style is encoded in both representations: style ARI is 0.85 from z and 0.91 from c. The frozen pretrained encoder also contains stylistic information, but its representation is poorly organized for unsupervised recovery, with style ARI 0.29. Thus, our objective does not simply preserve style information already present in the encoder; it organizes this information in z while removing it from c. We show additional analysis of the learned embeddings in Appendix B. These results validate z as a representation in which known stylistic variation can be recovered without style supervision. We next apply the same procedure to unlabeled traces from the multi-teacher reasoning corpus.

4.2 Styles in Multi-Teacher Reasoning Data

We use the SCAS dataset from Hu et al. (2026), which contains over 100K verified mathematical reasoning traces generated by nine teacher models. Details on the dataset and autoencoder training are shown in Appendix C. We encode each teacher trace using the learned style representation $z(s)$ and fit a Gaussian mixture model for clustering. We use $K = 6$ clusters, chosen on the validation set, in our main experiments. We order the clusters by mean trace length and denote them $s_1, \ldots, s_6$.

What styles emerge? The discovered styles range from short textbook-like solutions (s_1) to long think-block traces (s_6), with intermediate styles differing in formatting, use of markdown, verification, backtracking, and other stylistic features. Importantly, the styles are imbalanced in the generated training data. The dominant style, s_5, accounts for 41.3% of all traces, whereas each of the remaining styles accounts for only 9.0–15.3%. Thus, although the teacher pool contains multiple ways to express solutions, standard fine-tuning exposes the student to them at different frequencies. Full teacher–style distributions are reported in Appendix C.2.

Styles appear across multiple teachers. For example, llama, gemma, and qwen2.5 frequently produce s_1, whereas phi-4, qwen3, and qwen3.5 mostly s_6. The normalized mutual information between teacher identity and style is only 0.17. In particular, the dominant s_5 cluster contains substantial mass from all nine teachers.

Table 1: Teacher composition of each discovered style (%; rows sum to 100).

Style	gemma	gpt-5	gpt-oss	llama	olmo	phi-4	qwen2.5	qwen3	qwen3.5
s_1	30.2	0.3	0.1	34.5	4.0	0.0	28.6	0.4	1.9
s_2	16.4	10.8	5.5	16.0	12.4	8.1	13.2	6.5	11.0
s_3	0.8	29.6	34.7	0.0	18.8	9.3	2.6	0.3	3.9
s_4	2.4	24.0	23.3	0.1	19.2	18.7	5.2	2.6	4.5
s_5	10.0	10.0	11.9	8.7	12.0	7.9	10.9	16.6	12.1
s_6	1.9	4.9	0.7	3.5	4.0	31.8	0.1	24.2	28.8

Morover, the styles do not identify the underlying mathematical problem. Because each question has nine independently generated teacher traces, we can measure stylistic variation while holding the problem fixed. Only 0.85% of questions have all nine solutions assigned to the same style. A question contains 3.38 distinct styles on average, and 83% of questions contain at least three. Agreement between question identity and style is correspondingly low (AMI $= 0.049$, ARI ≈ 0). Examples of traces for each style are shown in Appendix C.3.

Comparison with LLM-based style discovery. As a simple alternative to learning a style representation, we also ask a language model to discover recurring styles directly from the traces. Because direct prompting is limited by context length, we apply this baseline to a balanced subset of 180 traces, with 20 traces from each of the nine models. The resulting partition is strongly aligned with model identity (ARI $= 0.70$), whereas our discovered styles have much weaker agreement with model identity (ARI $= 0.06$). Thus, direct LLM-based discovery primarily recovers model-specific differences on this subset, rather than the cross-model stylistic structure captured by our autoencoder representation. Details for this experiment are shown in Appendix C.4.

5 STYLE-AWARE FINE-TUNING

We next use the discovered styles to make models explicitly aware of style. We use the same dataset for fine-tuning. We first pass every training trace through the learned autoencoder, extract its style representation z and identify its corresponding style cluster, then assign the resulting style label to the trace. We then prepend this label to the trace during supervised fine-tuning (SFT), allowing the student to condition its generation on a requested style at inference time.

Simply adding style labels, however, leaves the empirical style distribution unchanged. Because the discovered styles are imbalanced, the student still receives more supervision for common styles than for rare ones. In practice, this STYLE SFT (EMPIRICAL) baseline performs similarly to vanilla SFT, suggesting that exposing style labels alone is not sufficient to make effective use of the different modes present in the training data.

To mitigate this, we combine style conditioning with importance weighting: we reweigh the contribution of training traces to the training objective evenly among styles. This gives underrepresented styles greater influence during training while preserving all available reasoning traces. At inference, we generate under each style prefix, exposing multiple style-conditioned solution distributions.

5.1 IMPORTANCE-WEIGHTED STYLE-AWARE TRAINING

The discovered styles are unevenly represented not only globally, but also among the teacher traces available for an individual question. Let S_i denote the set of styles observed among the teacher traces for question x_i. We define a uniform target distribution over these observed styles c, $p(c \mid x_i) = \mathbb{I}(c \in S_i)/|S_i|$, and assign each trace an importance weight $w(c \mid x_i) \propto p(c \mid x_i)/q(c \mid x_i)$. where $q(c \mid x_i)$ is a smoothed estimate of the empirical style distribution for question x_i. Thus, styles that are overrepresented for a question receive smaller weights, while underrepresented styles receive larger weights. We retain every training trace and only change its contribution to the negative log-likelihood objective. We mean-normalize the weights to one within the training set. Full details of the proposal distribution, smoothing, and weighting objective are given in Appendix D.

Training setup. We fine-tune Qwen3-0.6B-Base, Qwen3-1.7B-Base, and Qwen3-4B-Base students on the same multi-teacher corpus using full-parameter SFT. All models are trained with a maximum sequence length of 4096 and an effective batch size of 256 for 40 epochs with a constant

learning rate of 10^{-5}. Style-conditioned models prepend a `[style_i]` prefix. Our importance-weighted objective uses the same training traces as the other methods; only the per-example loss weights differ.

Baselines. Our primary comparison is VANILLA SFT, which uses the same teacher corpus without style prefixes or reweighting. We additionally compare against three controls. STYLE SFT (EMPIRICAL) uses the discovered style prefixes while retaining the empirical training distribution. RANDOM-STYLE SFT replaces each discovered style prefix with one sampled uniformly at random from the six available styles, testing whether gains arise merely from introducing conditioning tokens. TEACHER-STYLE SFT uses teacher identity as the prefix, testing whether the discovered styles provide structure beyond the model that generated each trace. This model uses nine styles instead of six, one corresponding to each teacher.

Evaluation. We evaluate on MATH-500, AMC 12 (2025), OlympiadBench-EN, AIME 2024, AIME 2025, and HMMT February 2025. We sample with temperature 0.6, top-$p = 0.95$, and a maximum generation length of 4096 tokens. For each style-conditioned model, we draw 256 samples under each of the six style prefixes; teacher-conditioned models are evaluated analogously over nine teacher prefixes. Vanilla SFT and the base models produce 256 unprefixed samples per problem. To compare methods at the same sampling budget, our reported mixed-style Pass@k allocates the k samples uniformly across the available style prefixes. If p_c is the empirical success probability under style c and k_c samples are allocated to that style, we compute $\mathrm{Pass@}k = 1 - \prod_c (1 - p_c)^{k_c}$, with the total budget satisfying $\sum_c k_c = k$. We report Pass@k for k up to 256.

6 RESULTS

6.1 STYLE-AWARE FINE-TUNING IMPROVES PASS@k

Across benchmarks, STYLE SFT (IMPORTANCE-WEIGHTED) frequently improves substantially over both vanilla SFT and the base model, as shown in Figure 1. The same pattern holds for the 1.7B and 4B students. Full results for these models are reported in Appendix E. In contrast, vanilla SFT often reduces Pass@k relative to the base model. For the Qwen3 0.6B student, at $k = 256$, vanilla SFT obtains 74.9% on MATH-500, 46.6% on AMC12-2025, and 44.2% on OlympiadBench, while our method reaches 89.8%, 68.5%, and 60.1%, respectively. The difference is even more apparent on the harder competition benchmarks: Pass@256 increases from 9.7% to 29.7% on AIME 2024 and from 9.5% to 30.1% on AIME 2025. On HMMT, vanilla SFT reaches only 2.1%, compared with 11.4% for our method.

The gains cannot be explained by the addition of style prefixes alone. STYLE SFT (EMPIRICAL), which uses the same discovered style labels while preserving their empirical frequencies, remains much closer to vanilla SFT. Likewise, replacing the discovered labels with random style prefixes does not reproduce the gains. These controls suggest that simply introducing discrete conditioning tokens is insufficient; the improvement emerges when style conditioning is combined with reweighting the contributions of the discovered styles during training.

Importance-weighted SFT improves style fidelity. The preceding results show that style prefixes affect performance, but this alone does not establish that they induce the corresponding styles discovered in the teacher traces. We measure whether generations conditioned on `[style_i]` fall into the same style mode as teacher traces assigned to s_i. After generation, we assign each solution a realized style $\hat{s}$ using the nearest teacher-style centroid in our handcrafted style feature space (Table 6) and compute $P(\hat{s} = j \mid \texttt{[style_i]})$. For STYLE SFT (IMPORTANCE-WEIGHTED), the fine-tuned models generally produce the style they were conditioned on: mean diagonal probability is 30.8%, 30.8%, and 32.9% for the 0.6B, 1.7B, and 4B students, respectively, compared with 16.7% chance. Style fidelity is concentrated in a subset of modes: s_1 and s_3 are recovered most reliably, while s_2, s_4, and s_5 are frequently realized as neighboring styles and s_6 shows moderate adherence. In comparison, STYLE SFT (EMPIRICAL) remains close to chance (Appendix E.1). Thus, the learned prefixes provide control over the discovered style structure, and the importance-weighted model exhibits stronger style fidelity than conditioning on the empirical distribution alone. Representative generations for each style prefix are shown in Appendix E.2.

6.2 STYLES INDUCE PROBLEM-DEPENDENT SOLUTION DISTRIBUTIONS

Having established that style-aware fine-tuning improves Pass@k, we next ask whether the discovered styles correspond to functionally different solution distributions. If style acted only as a surface-level control over how an otherwise fixed solution is expressed, changing the style prefix should not substantially affect correctness. We find that this is not the case: style interacts with the content changing the correctness of the answers. We use MATH-500 for the following analyses.

Style conditioning changes solution accuracy. For the Style-SFT models trained with importance-weighted SFT, we generate solutions under each of the six explicit style prefixes and measure $P(\text{correct} \mid s)$, so that style is fixed before generation. As shown in Figure 2, changing only the style prefix produces differences in accuracy for the same trained model. For comparison, the Base and Vanilla SFT models are decoded without style prefixes, and their generations are assigned retrospectively to the nearest teacher-style centroid using the handcrafted features from Table 6.

For the 0.6B Style SFT model, mean correctness ranges from 13.9% under s_6 to 30.6% under s_4; for the 1.7B student, from 22.2% to 43.1%; and for the 4B student, from 25.8% to 52.4%. The spread in correctness across style conditions increases with model size: the difference between the highest and lowest performing style is 16.7 percentage points for 0.6B, 20.9 points for 1.7B, and 26.6 points for 4B. Thus, larger students exhibit more differentiated style-conditioned solution distributions.

A second pattern is that several styles associated with relatively low correctness under Base or Vanilla SFT become effective conditional modes under Style SFT. For example, for the 0.6B model, generations retrospectively assigned to s_4 have 12.8% correctness under Vanilla SFT, compared with 30.6% when s_4 is explicitly requested from the Style-SFT model. At 4B, the corresponding values are 28.2% and 52.4%, with similar patterns for s_3 and s_6. This analysis suggest that style-aware training makes several discovered styles available as useful controlled solution modes.

Style effects occur within individual problems. The aggregate differences above could still be explained by a global ordering in which some styles are simply more effective than others across all questions. To test whether style instead interacts with problem content, we measure style sensitivity separately for each question. Let $p_{x,s} = P(\text{correct} \mid x, s)$, and define $\Delta_x = \max_s(p_{x,s}) - \frac{1}{6}\sum_{s=1}^{6} p_{x,s}$. Mean Δ_x is 0.104, 0.125, and 0.164 for the 0.6B, 1.7B, and 4B students, respectively. For the 0.6B student, 59% of questions have a gap exceeding 5 percentage points and 45% exceed 10 points; for 1.7B and 4B the 5 and 10 percentage point gaps rise to 74% and 55%, and 84% and 71% respectively. Thus, even for a fixed problem, changing the style condition can change the probability of success. This difference also appears in multi-sample performance: when all samples are drawn from a single style, the resulting Pass@k curves differ across style conditions (Appendix F). Thus, the choice of style also affects the coverage obtained from repeated sampling.

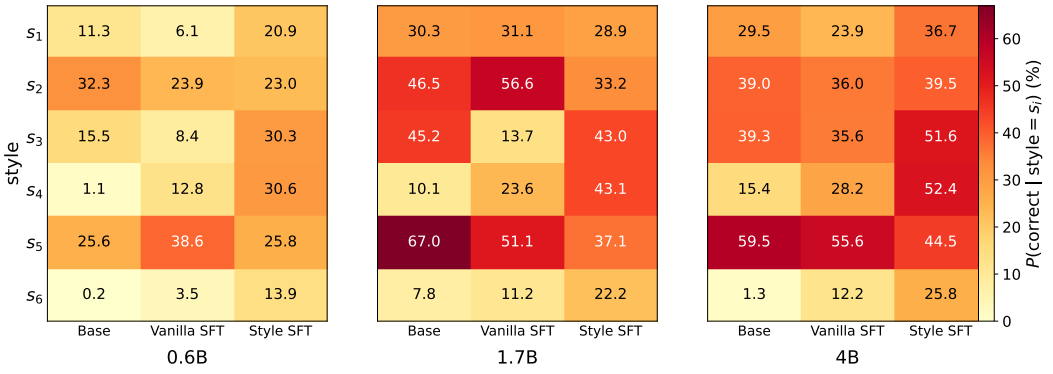


Figure 2: Correctness by style on MATH-500. Changing the style condition produces differences in accuracy at all three model scales.

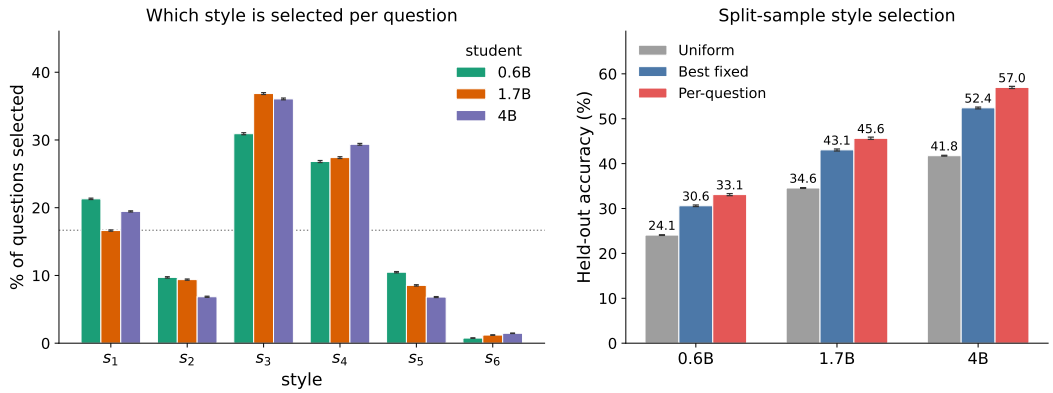


Figure 3: The most useful style depends on the problem. For each of 400 random splits, 128 of the 256 generations per question and style are used for style selection and the remaining 128 for held-out evaluation. Left: fraction of questions for which each style is selected as best; several different styles are preferred across problems. Right: held-out accuracy under a uniform mixture of styles, the globally best fixed style, and a style selected separately for each question. Question-specific selection consistently outperforms the globally best fixed style, showing that style usefulness cannot be explained by a single global ranking.

Different problems favor different styles. Within-question variation alone does not rule out the possibility that one style is consistently best. We test this with a split-sample style-selection experiment. For each question, we generate 256 answers in each of the six styles. For each style, we choose 128 answers to select the best style $s_x^* = \arg\max_s \hat{p}_{x,s}^{select}$ for that question. We then evaluate the remaining 128 generations conditioned on the selected style. We compare this to a globally best style, $s^*_{\text{global}} = \arg\max_s \frac{1}{|\mathcal{X}|} \sum_{x \in \mathcal{X}} \hat{p}^{\text{select}}_{x,s}$, determined by choosing the best style on average across questions. We repeat the procedure over 400 random splits of the 256 generated responses.

The selected styles vary substantially across questions. Although s_3 and s_4 are chosen most often, styles s_1–s_5 are each preferred for a nontrivial subset of problems. More importantly, this variation generalizes to held-out samples. Per-question selection reaches 33.1% accuracy for the 0.6B student compared with 30.6% for the globally best fixed style; for 1.7B the corresponding values are 45.6% and 43.1%, and for 4B they are 57.0% and 52.4%. The per-question advantage is positive at every model size, as shown in Figure 3.

Taken together, these results show that the discovered styles correspond to functionally distinct conditional solution distributions. Changing the style condition alters the probability of success for the same problem, and this variation cannot be explained by a single global ranking of styles: different problems favor different styles on held-out generations. This provides a possible explanation for why mixed-style sampling improves Pass@k: it gives the student access to multiple conditional solution behaviors whose usefulness varies across problems.

7 Discussion

Why does style reweighting help? Style prefixes alone provide relatively little improvement when the empirical training distribution is preserved, whereas importance-weighted style training produces substantially larger gains. One possible explanation is that rare styles receive too little supervision under the empirical distribution for the student to learn them as reliable conditional behaviors. Reweighting increases their contribution to the objective and may therefore make a broader set of style-conditioned solution distributions available at inference time. Importance-weighted training also produces substantially stronger correspondence between requested and realized styles than conditioning on the empirical distribution alone. The benefit of reweighting may come from preventing less common solution modes from being absorbed into the dominant behavior of the model.

More generally, these results suggest that diversity in a multi-teacher corpus is not necessarily preserved by standard training. A dataset may contain multiple useful ways of reasoning about the same

problem, while their unequal frequency causes the student to preferentially learn only the dominant modes. Making this structure explicit and balancing its contribution to training provides one way to preserve more of the variation already present in the training data.

Style-conditioned behaviors become more differentiated with model size. The effect of style conditioning becomes stronger as student size increases. Both the gap between the highest and lowest performing style and the problem-level sensitivity to style increases with scale. On the other side, style fidelity itself does not increase nearly as sharply across model sizes: all three models exhibit comparable correspondence between requested and realized styles.

One possibility is that larger models have sufficient capacity to represent several genuinely distinct conditional solution distributions, whereas smaller models partially collapse these conditions onto a more similar underlying behavior. Under this interpretation, the style prefix selects among increasingly differentiated reasoning behaviors as model capacity grows. Another possibility is that larger models can exploit the structural differences associated with each style more effectively: for example, patterns of verification, decomposition, or intermediate derivation may have little effect for a smaller student but become useful once the model is capable of carrying them out reliably.

Why does correctness depend on style? Conditioning on different styles changes correctness for the same problem, and the most useful style varies across problems. Why this interaction arises remains an open question. One possibility is that a given model finds some forms of reasoning easier to generate or maintain than others. Styles that encourage particular forms of intermediate steps may make some solution trajectories easier to realize reliably. Another possibility is that in the benchmarks we used in evaluation, the problems themselves contain regularities that interact with these forms of reasoning, such that particular styles are better suited to particular classes of questions.

8 Conclusion

Current language models learn content and style jointly, making stylistic variation difficult to identify or control explicitly. In this work, we show that these sources of variation could be separated without style supervision, and that the resulting structure could be used to create more controllable, style-aware language models. We focus on mathematical reasoning and use an autoencoder to learn disentangled representations for the style and content of the problem solutions. We cluster the style representations and discover six styles in traces from nine models. Using these discovered styles, we fine-tune models that can be explicitly conditioned on style, by using an importance-weighted loss that encourages learning of all styles uniformly. Importance-weighted Style SFT improves Pass@k over vanilla SFT across mathematical reasoning benchmarks and produces strong correspondence between requested and realized styles.

Our work has several limitations. The discovered styles depend on the representation-learning objective, the training corpus, and our discretization of the representation into clusters; other datasets or objectives may reveal different structure, and stylistic variation may ultimately be continuous or hierarchical rather than discrete. We also focus on mathematics because it allows both multiple solutions to a question and an objective way to evaluate correctness. We expect our approach to separate style and content to work in other domains that have these properties and leave it for future work to extend it to general text. Moreover, we leave it for future work to study the cause of the interaction between style and correctness and figure out how to best use the learned styles at test time.

Overall, we identify style as a source of structure in data that standard next-token prediction does not represent explicitly. Instead, style is learned together with all other properties of the text, so standard model training simply inherits their distribution from the training traces and may underrepresent or collapse modes. We show that we can discover these styles without labels, represent them explicitly, and control how they influence learning. Our work is a first step toward training style-aware language models, in which style is an explicit and controllable dimension of model behavior rather than an incidental byproduct of the data and training procedure.

AI USE STATEMENT

In this work, we used generative AI tools for generating synthetic dataset, implementing methods, supporting qualitative and thematic data analysis, designing and providing feedback on research methodology or experiments. We have not used generative AI tools for helping developing theoretical models or conceptual frameworks. Formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, assisting with translation, clean and reformat dataset, are not applicable to this work. Additionally, we used generative AI tools for creating or modifying scientific figures or images, suggesting experimental parameters, creating or editing software code, creation of artifacts, draft parts of a research paper, sourcing/searching for information, identify relevant literature, format references. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The code is attached as supplementary material to the submission. The details for the autoencoder training are shown in Appendix A. The fine-tuning implementation is described in Section 5.1.

REFERENCES

Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Nguyen Bach, Amit Bahree, Arushi Behl, Alon Benhaim, Rahul Bhalerao, Sébastien Bubeck, et al. Phi-4-reasoning technical report. *arXiv preprint arXiv:2504.21318*, 2025.

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourrier, Ben Burtenshaw, Hugo Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, Colin Raffel, Leandro von Werra, and Thomas Wolf. Smollm2: When smol goes big – data-centric training of a small language model. In *Second Conference on Language Modeling*, 2025.

Gregor Bachmann, Yichen Jiang, Seyed-Mohsen Moosavi-Dezfooli, and Moin Nabi. The potential of cot for reasoning: A closer look at trace dynamics. In *International Conference on Learning Representations*, 2026.

Adrien Bardes, Jean Ponce, and Yann LeCun. VICReg: Variance-invariance-covariance regularization for self-supervised learning. *ICLR*, 2022.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020.

Yida Chen, Yuning Mao, Xianjun Yang, Suyu Ge, Shengjie Bi, Lijuan Liu, Saghar Hosseini, Liang Tan, Yixin Nie, and Shaoliang Nie. Your thoughts tell who you are: Characterize the reasoning patterns of lrms. *arXiv preprint arXiv:2509.24147*, 2025.

Pengyu Cheng, Martin Renqiang Min, Dinghan Shen, Christopher Malon, Yizhe Zhang, Yitong Li, and Lawrence Carin. Improving disentangled text representation learning with information-theoretic guidance. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 7530–7541. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.673.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

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

Allyson Ettinger, Team Olmo, and Allen Institute for AI. Olmo 3. *arXiv preprint arXiv:2512.13961*, 2025.

Kaituo Feng, Changsheng Li, Xiaolu Zhang, Jun Zhou, Ye Yuan, and Guoren Wang. Keypoint-based progressive chain-of-thought distillation for LLMs. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 13241–13255. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/feng24e.html.

Gemma Team Google DeepMind. Gemma 4 technical report. *arXiv preprint arXiv:2607.02770*, 2026.

Google DeepMind. Gemini 3.1 pro model card, February 2026. URL https://deepmind.google/models/model-cards/gemini-3-1-pro/. Accessed: 2026-09-25.

Jingxuan Han, Quan Wang, Zikang Guo, Benfeng Xu, Licheng Zhang, and Zhendong Mao. Disentangled learning with synthetic parallel data for text style transfer. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pp. 15187–15201. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.811.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. *NeurIPS Datasets and Benchmarks*, 2021.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 8003–8017. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.findings-acl.507.

Zhengyu Hu, Zheyuan Xiao, Linxin Song, Fengqing Jiang, Yuetai Li, Zhihan Xiong, Yue Liu, Junhao Lin, Yao Su, Lijie Hu, Kaize Ding, Teng Xiao, and Radha Poovendran. The strongest teacher is not always the best teacher: Student-centric answer selection. *arXiv preprint arXiv:2605.26872*, 2026.

Zixian Huang, Kaichen Yang, Xu Huang, Feiyang Hao, Qiming Ge, Bowen Li, He Du, Kai Chen, and Qipeng Guo. How to fine-tune a reasoning model? a teacher–student cooperation framework to synthesize student-consistent sft data. *arXiv preprint arXiv:2604.14164*, 2026.

Hugging Face. Open-R1: a fully open reproduction of DeepSeek-R1. https://github.com/huggingface/open-r1, 2025.

Sheng Jia, Xiao Wang, and Shiva Kasiviswanathan. Training large language models to reason in parallel with global forking tokens. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=xBQvvkg4Wc.

Vineet John, Lili Mou, Hareesh Bahuleyan, and Olga Vechtomova. Disentangled representation learning for non-parallel text style transfer. In Anna Korhonen, David Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 424–434, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1041. URL https://aclanthology.org/P19-1041/.

Philip Lippmann and Jie Yang. Style over substance: Distilled language models reason via stylistic replication. In *Proceedings of the Second Conference on Language Modeling*, 2025.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Tianjun Zhang, Erran Li, Raluca Ada Popa, and Ion Stoica. Deepscaler: Surpassing O1-preview with a 1.5B model by scaling RL. Notion Blog, 2025. URL https://pretty-radio-b75.notion.site/DeepScaleR-Surpassing-O1-Preview-with-a-1-5B-Model-by-Scaling-RL-19681902c1468005b

Taro Makino, Ji Won Park, Natasa Tagasovska, Takamasa Kudo, Paula Coelho, Jan-Christian Huetter, Heming Yao, Burkhard Hoeckendorf, Ana Carolina Leote, Stephen Ra, David Richmond, Kyunghyun Cho, Aviv Regev, and Romain Lopez. Supervised contrastive block disentanglement, 2025. URL https://arxiv.org/abs/2502.07281.

Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. Orca: Progressive learning from complex explanation traces of gpt-4. *arXiv preprint arXiv:2306.02707*, 2023.

OpenAI. Gpt-5 system card. Technical report, OpenAI, August 2025a. URL https://cdn.openai.com/gpt-5-system-card.pdf.

OpenAI. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025b.

Qwen Team. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Qwen Team. Qwen3.5-omni technical report. *arXiv preprint arXiv:2604.15804*, 2026.

Lei Sha and Thomas Lukasiewicz. Text attribute control via closed-loop disentanglement. *Transactions of the Association for Computational Linguistics*, 12:190–209, 2024. doi: 10.1162/tacl_a_00640. URL https://aclanthology.org/2024.tacl-1.11/.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. Style transfer from non-parallel text by cross-alignment. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

Chen Henry Wu, Sachin Goyal, and Aditi Raghunathan. Mode-conditioning unlocks superior test-time compute scaling. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=JzkdJQzPw1.

xAI. Grok 4.6 model card, August 2026. URL https://media.x.ai/v1/website/card-4p6-4cd2dc57.pdf. Accessed: 2026-09-25.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bo-Wen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *ICML*, 2021.

Xinliang Frederick Zhang, Anhad Mohananey, Alexandra Chronopoulou, Pinelopi Papalampidi, Somit Gupta, Tsendsuren Munkhdalai, Lu Wang, and Shyam Upadhyay. Do llms really need 10+ thoughts for "find the time 1000 days later"? towards structural understanding of llm overthinking. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, pp. 17005–17030. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.773.

Kangchen Zhu, Zhiliang Tian, Jingyu Wei, Ruifeng Luo, Yiping Song, and Xiaoguang Mao. Styleflow: Disentangle latent representations via normalizing flow for unsupervised text style transfer. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation*, pp. 15384–15397, 2024.

A Style Representation Learning Details

This section gives the full objective and hyperparameters for the representation-learning method described in Section 3.

A.1 ARCHITECTURE

Each minibatch contains N questions, with two independently generated traces for each question, yielding $2N$ traces

$$\{s_{i,1}, s_{i,2}\}_{i=1}^{N}. \tag{2}$$

An encoder E maps each trace s to a representation $h(s)$. Two separate MLP projection heads produce a content representation $c(s)$ and a style representation $z(s)$: $c(s) \in \mathbb{R}^{d_c}, z(s) \in \mathbb{R}^{d_z}$, where $d_c = d_z = 256$.

The two representations are concatenated and projected to a continuous prefix of $m = 8$ tokens. This prefix conditions an autoregressive decoder D, which reconstructs the original trace.

A.2 TRAINING OBJECTIVE

The full objective is

$$\mathcal{L} = \lambda_c \mathcal{L}_c + \lambda_{\text{orth}} \mathcal{L}_{\text{orth}} + \lambda_{\text{rec}} \mathcal{L}_{\text{rec}} + \lambda_{\text{var}} \mathcal{L}_{\text{var}} + \lambda_{\text{cov}} \mathcal{L}_{\text{cov}} + \lambda_{\text{div}} \mathcal{L}_{\text{div}}. \tag{3}$$

Unless otherwise specified, we use

$$\lambda_c = \lambda_{\text{orth}} = \lambda_{\text{rec}} = \lambda_{\text{var}} = \lambda_{\text{div}} = 1, \qquad \lambda_{\text{cov}} = 0.15. \tag{4}$$

The variance, covariance, and within-question diversity objectives operate only on z.

Content agreement. The two traces generated for the same question form a positive pair for the content representation. We normalize each content vector,

$$\bar{c}_i = \frac{c_i}{\|c_i\|}, \tag{5}$$

and define

$$\text{sim}(i, j) = \frac{\bar{c}_i^{\top} \bar{c}_j}{\tau}, \qquad \tau = 0.07. \tag{6}$$

For trace i, let $j^+(i)$ denote the other trace generated for the same question. All remaining traces in the minibatch serve as negatives. The content InfoNCE loss is

$$\mathcal{L}_c = -\frac{1}{2N} \sum_{i=1}^{2N} \log \frac{\exp\left(\text{sim}(i, j^+(i))\right)}{\sum_{k \neq i} \exp\left(\text{sim}(i, k)\right)}. \tag{7}$$

This encourages c to retain information that is stable across independently generated solutions to the same problem.

Content–style decorrelation. To discourage c and z from learning aligned representations of the same trace, we penalize their squared cosine similarity:

$$\mathcal{L}_{\text{orth}} = \frac{1}{2N} \sum_{r=1}^{2N} \left(\frac{c(s_r)^{\top} z(s_r)}{\|c(s_r)\| \, \|z(s_r)\| + \epsilon} \right)^2, \tag{8}$$

with $\epsilon = 10^{-7}$.

This encourages the two heads to occupy different directions in representation space, without requiring either representation to discard information needed for reconstruction.

Reconstruction. The reconstruction objective is the autoregressive cross-entropy of the decoder on the original trace:

$$\mathcal{L}_{\text{rec}} = -\sum_t \log P_D\left(s_t \mid s_{<t}, c(s), z(s)\right). \tag{9}$$

This ensures that the pair $[c(s); z(s)]$ remains jointly informative about the observed trace.

Variance regularization. To prevent collapse of the style representation, we use a VICReg-style variance constraint (Bardes et al., 2022). Let $\sigma_d(z)$ denote the standard deviation of the d-th dimension of z across the $2N$ traces in a minibatch:

$$\sigma_d(z) = \sqrt{\mathrm{Var}_{r=1,\ldots,2N}\left[z(s_r)_d\right] + \varepsilon}, \tag{10}$$

where $\varepsilon = 10^{-4}$ is a small constant for numerical stability.

We impose a lower bound $\gamma = 1$ on the standard deviation of each dimension:

$$\mathcal{L}_{\mathrm{var}} = \frac{1}{d_z}\sum_{d=1}^{d_z} \max\left(0,\, \gamma - \sigma_d(z)\right). \tag{11}$$

This penalizes dimensions of z whose variation across the minibatch falls below γ, preventing the style representation from collapsing to a constant or near-constant representation.

Covariance regularization. Variance regularization prevents individual dimensions from collapsing but does not prevent them from becoming redundant. We therefore penalize the off-diagonal entries of the centered covariance matrix of z:

$$\mathcal{L}_{\mathrm{cov}} = \frac{1}{d_z}\sum_{i\neq j} \mathrm{Cov}(z)_{ij}^2. \tag{12}$$

This follows the redundancy-reduction principle used by VICReg and related methods (Bardes et al., 2022; Zbontar et al., 2021), encouraging variation in z to be distributed across multiple dimensions.

Within-question style diversity. Finally, we prevent z from inheriting the same same-question invariance imposed on c. For question i, let

$$\rho_i = \frac{z(s_{i,1})^\top z(s_{i,2})}{\|z(s_{i,1})\|\,\|z(s_{i,2})\|}. \tag{13}$$

We penalize positive similarity between the two style representations:

$$\mathcal{L}_{\mathrm{div}} = \frac{1}{N}\sum_{i=1}^{N} \mathrm{ReLU}(\rho_i). \tag{14}$$

Thus, the content objective encourages same-question solutions to agree in c, while the diversity objective prevents the corresponding z representations from collapsing together. Combined with reconstruction, this encourages z to retain trace-specific variation that is complementary to the information represented by c.

A.3 Implementation Details

Encoder and decoder. We use `Alibaba-NLP/gte-Qwen2-1.5B-instruct` as the encoder. The pretrained encoder parameters are frozen and adapted with LoRA of rank 8, scaling parameter $\alpha = 16$, and dropout 0.05 on the query, key, value, and output projections. We obtain the trace representation using last-token pooling. The encoder uses scaled dot-product attention and gradient checkpointing.

For reconstruction, we use the post-trained `Qwen/Qwen3-4B` model as the decoder unless noted otherwise. The decoder is adapted with LoRA of rank 16, $\alpha = 32$, and dropout 0.05 on the query, key, value, and output projections. The decoder's base parameters remain frozen, while the LoRA parameters and the continuous-prefix projection are trained.

Projection heads. The content and style heads have identical architectures. Starting from the 1536-dimensional pooled encoder representation, each head is a four-layer MLP:

$$\begin{aligned}\mathrm{LayerNorm}(1536) &\to \mathrm{Linear}(1536, 256) \to \mathrm{GELU} \to \mathrm{Linear}(256, 256) \to \mathrm{GELU} \\ &\to \mathrm{Linear}(256, 256) \to \mathrm{GELU} \to \mathrm{Linear}(256, 256).\end{aligned} \tag{15}$$

There is no final activation or normalization in either head. For the InfoNCE objective, c is normalized explicitly when computing cosine similarity.

Continuous reconstruction prefix. The concatenated representation $[c(s); z(s)] \in \mathbb{R}^{512}$ is mapped by a single learned linear projection to 8×2560 values and reshaped into $m = 8$ continuous embeddings in the decoder's input space. These embeddings are prepended to the decoder input through `inputs_embeds`. The encoder and decoder use their respective tokenizers. When required by the decoder tokenizer, a beginning-of-sequence token is inserted after the continuous prefix and before the trace tokens. The reconstruction cross-entropy is computed only over the original trace tokens, excluding the continuous prefix.

Optimization. We train for four epochs using 8 GPUs with distributed data parallelism and `bf16` mixed precision. The maximum sequence length is 4096 tokens. Each GPU processes one pair of traces from the same question per forward pass, giving 8 question pairs, or $2N = 16$ traces, across devices. Representations are gathered across devices before computing the batch-dependent objectives. We use gradient accumulation over two forward passes, corresponding to an effective optimizer batch of 16 question pairs.

We optimize with AdamW using

$$\beta_1 = 0.9, \qquad \beta_2 = 0.999, \qquad \text{weight decay} = 0.01. \tag{16}$$

The content and style heads, decoder LoRA parameters, and prefix projection use a learning rate of 10^{-4}, while the encoder LoRA parameters use a learning rate of 10^{-6}. We use a constant learning-rate schedule with no warmup and clip the gradient norm at 10.

B Controlled Style Recovery

This appendix provides the full setup and ablations for the controlled style-recovery experiment summarized in Section 4.1. The goal is to verify that the proposed representation can recover a known stylistic factor without using style labels during training, and that stylistic information is isolated in z rather than duplicated across both representations.

B.1 Dataset and prompted styles

We prompt Gemini 3.1 Pro to solve 300 mathematical problems from the DeepScaleR dataset in four explicitly specified writing styles, yielding 1,200 traces in total. Each problem is generated in all four styles, ensuring that style is not confounded with question identity. We use a question-level 1,080/120 train/test split, so no question appears in both sets. Style labels are used only for evaluation.

The four prompted styles are shown in Table 2. They are deliberately chosen to differ strongly in observable surface form, providing a controlled positive test of whether the representation can separate question-specific content from stylistic variation.

Table 2: Four prompted Gemini writing styles used in the controlled experiment.

ID	Name	Prompted surface form
0	Academic	Formal paragraphs; passive voice; frequent use of connective phrases such as "therefore" and "it follows that".
1	ELI5 / tutor	Short sentences; explanatory analogies; encouraging, tutorial-like tone.
2	Algorithmic list	Step-by-step organization, with each sentence explicitly formatted as a numbered or ordered step.
3	Pure equation	Minimal natural language; equation-dominated derivations using symbols such as $\Rightarrow$ and $\therefore$.

B.2 Model and training details

The encoder is GTE–Qwen2-1.5B-Instruct with LoRA rank r=8. The decoder is SmolLM2-135M (Allal et al., 2025), frozen except for LoRA adapters with rank r=16.

We compare three representations or training objectives:

- **Frozen encoder h:** the off-the-shelf GTE representation, without LoRA adaptation or a c/z split.

Table 3: Full disentanglement results on the controlled Gemini setting. Gap is the mean within-group cosine similarity minus the mean between-group similarity. ARI/NMI are from k-means with k=4 against the ground-truth `style_id`. The style probe is a 4-way linear classifier with chance accuracy 25%.

Metric	Frozen GTE h	Recon-only	Full objective
c question gap	—	0.36	**0.93**
c style gap	—	0.09	$\approx$ **0**
z style gap	—	0.10	**1.18**
z question gap	—	0.38	$\mathbf{-0.30}$
z style ARI / NMI	0.29 / 0.36	0.85 / 0.82	0.77 / 0.78
c style ARI / NMI	—	0.91 / 0.87	$\approx$ **0** / 0.00
z style probe	96.7%	96.3%	96.7%
c style probe	—	97.1%	**23.3%**

- **Reconstruction-only:** the same autoencoder architecture trained with $\lambda_{\text{rec}} = 1$ and all auxiliary loss weights set to zero.
- **Full objective:** the complete content–style disentangling objective described in Section 3.

B.3 Evaluation

We evaluate whether the learned representations organize according to question identity and prompted style.

Cosine-similarity gap. For a given grouping variable, we compute the mean cosine similarity between pairs sharing the same label and subtract the mean similarity between pairs with different labels. Higher values indicate stronger clustering by the corresponding attribute.

Unsupervised clustering. We apply k-means with k=4 and compare the resulting partition against the ground-truth style labels using adjusted Rand index (ARI) and normalized mutual information (NMI).

Linear probing. We train a 4-way linear classifier to predict the ground-truth style ID. Chance accuracy is 25%. We additionally evaluate a question-held-out probe in which the classifier is trained and evaluated on disjoint sets of questions.

B.4 Full disentanglement results

Table 3 reports the complete results. Under the full objective, the content representation c strongly organizes traces by question, with a question cosine gap of 0.93, while showing essentially no clustering by style. In contrast, the style representation z has a style gap of 1.18 and a negative question gap of -0.30.

Style can also be recovered directly from z. A linear probe predicts the prompted style with 96.7% accuracy, while the same probe on c achieves only 23.3%, close to the 25% chance level. Similarly, k-means on z recovers the four prompted styles with ARI 0.77 and NMI 0.78, whereas clustering c gives essentially no agreement with style.

Why the auxiliary objectives matter Reconstruction alone is sufficient to preserve substantial stylistic information, but it does not separate that information from content. Under the reconstruction-only objective, both representations strongly encode style: z achieves style ARI 0.85 and probe accuracy 96.3%, while c achieves style ARI 0.91 and probe accuracy 97.1%.

Thus, successful reconstruction does not by itself induce a content–style factorization. Without the auxiliary objectives, the same stylistic information is duplicated across both heads. In contrast, the full objective retains style information in z while removing it from c, reducing the style ARI of c to approximately zero and its probe accuracy to chance.

This distinction is important for the downstream use of the representation: our goal is not merely to encode style somewhere in the latent space, but to obtain a representation in which recurring stylistic variation can be isolated and clustered independently of question-specific content.

Comparison with the pretrained encoder The off-the-shelf encoder already contains substantial information about prompted style. A supervised linear probe predicts the style label from the frozen representation h with 96.7% accuracy. However, this information is not naturally organized into the four style modes: unsupervised k-means on h obtains only ARI 0.29 and NMI 0.36.

This comparison separates *encoding* style information from *organizing* the representation around style. The pretrained encoder makes style linearly accessible, but style labels are required to read it out. The learned z representation instead makes the same structure recoverable without supervision.

Generalization across questions We additionally test whether the learned style structure generalizes to unseen problem instances. A linear style classifier trained on one set of questions predicts style on held-out questions with 95.8% accuracy. This indicates that the information captured in z is not tied to particular mathematical problems, but instead reflects recurring variation that transfers across questions.

Overall, the controlled experiment establishes three properties needed for the main analysis: the proposed representation recovers a known stylistic factor without using its labels, style is isolated in z rather than leaking into c, and the resulting structure generalizes across questions. These results motivate using z to discover recurring styles in the unlabeled multi-teacher reasoning corpus.

C Teacher Styles

This appendix provides additional details on the six styles discovered in the multi-teacher SCAS corpus, including their relationship to teacher identity, question identity, and handcrafted surface features.

C.1 Style discovery setup

We use the subset of the SCAS verified teacher pool containing 12,426 questions with one solution from each of nine teacher models, yielding 111,834 traces in total with maximum length 4096 tokens. The nine teachers are `gemma-4-31b-it` (Gemma Team Google DeepMind, 2026), `gpt-5-chat` (OpenAI, 2025a), `gpt-oss-120b` (OpenAI, 2025b), `llama-3.3-70b-instruct` (Dubey et al., 2024), `olmo-3.1-32b-instruct` (Ettinger et al., 2025), `phi-4-reasoning-plus` (Abdin et al., 2025), `qwen2.5-72b-instruct` (Yang et al., 2024), `qwen3-32b` (Qwen Team, 2025), and `qwen3.5-27b` (Qwen Team, 2026). We split this dataset into 90% train and 10% validation.

The questions are drawn from MATH (Hendrycks et al., 2021), GSM8K (Cobbe et al., 2021), DeepScaleR (Luo et al., 2025), and OpenR1-Math (Hugging Face, 2025). The dataset is available at `https://huggingface.co/datasets/Student-Centric-Answer-Sampling/scas_verified_teacher_pool`.

After training the style autoencoder, we encode each trace and fit a full-covariance Gaussian mixture model to $z(s)$. We use $K{=}6$ for the experiments, chosen by best Pass@k on the validation set. The clusters are ordered by mean trace length and denoted $s_1, \ldots, s_6$, from shortest to longest.

C.2 Analysis of Discovered Teacher Styles

Teacher preferences over styles. Teachers differ substantially in how frequently they generate the six styles. Table 4 reports $P(s \mid \text{teacher})$. Llama, gemma, and qwen2.5 place substantial mass on the short textbook style s_1, whereas phi-4, qwen3, and qwen3.5 generate many more long think-block traces in s_6. The dominant s_5 cluster is comparatively common across all teachers. Despite these preferences, style is not equivalent to teacher identity. The mutual information between GMM style and teacher ID is 0.33 nats, corresponding to normalized mutual information 0.17.

Table 4: Share of each teacher's traces assigned to each $K=6$ style (%; rows sum to 100). The final row gives the overall style distribution across all 111,834 traces.

Teacher	s_1	s_2	s_3	s_4	s_5	s_6
gemma-4-31b-it	41.6	16.1	0.7	2.3	37.0	2.3
gpt-5-chat	0.4	10.6	23.9	22.1	37.1	5.9
gpt-oss-120b	0.1	5.4	28.1	21.4	44.3	0.8
llama-3.3-70b-instruct	47.6	15.7	0.0	0.1	32.4	4.2
olmo-3.1-32b-instruct	5.6	12.1	15.2	17.7	44.7	4.7
phi-4-reasoning-plus	0.0	8.0	7.5	17.2	29.4	37.8
qwen2.5-72b-instruct	39.5	13.0	2.1	4.8	40.5	0.1
qwen3-32b	0.6	6.4	0.3	2.4	61.6	28.8
qwen3.5-27b	2.6	10.8	3.2	4.1	45.0	34.3
All traces	15.3	10.9	9.0	10.2	41.3	13.2

Table 5: Number of distinct styles among the nine teacher solutions to the same question. There are 12,426 aligned questions with nine traces each.

# distinct styles	1	2	3	4	5	6
% of questions	0.8	15.8	36.6	38.3	8.2	0.3
# questions	105	1,963	4,545	4,754	1,021	38

Styles differ from question identity. A potential concern is that the clusters reflect properties of the mathematical problem rather than properties of the generated solution. For example, difficult questions might systematically elicit longer traces, causing all teachers to be assigned to the same cluster. The structure of the dataset allows us to test this directly. For every question, we observe nine independent teacher solutions. Only 0.85% of questions have all nine traces assigned to the same style. The average question contains 3.38 distinct styles, and 83% of questions contain at least three distinct styles.

Observable properties of the discovered styles. To characterize the discovered clusters without using teacher identity, we compute a small set of handcrafted trace-level features inspired by Lippmann & Yang (2025). These include trace length, formatting statistics, and discourse markers:

- number of words;
- number of lines per 100 words;
- equals-sign density;
- density of backtracking or "wait" language; and
- density of verification or checking language.

Except for total word count, density features are measured per 100 words. For each feature, we report the mean within each style and

$$\eta^2 = \frac{\mathrm{SS}_{\text{between styles}}}{\mathrm{SS}_{\text{total}}}, \tag{17}$$

the fraction of feature variance explained by style membership.

Table 6: Handcrafted properties of the six discovered styles. η^2 gives the fraction of variance in each feature explained by style membership.

Feature	s_1	s_2	s_3	s_4	s_5	s_6	η^2
n words	311	330	417	483	486	723	0.09
dens. backtrack	0.01	0.10	0.12	0.20	0.17	0.34	0.07
dens. verification	0.02	0.08	0.04	0.07	0.11	0.20	0.09
dens. equals	6.2	4.7	8.4	9.3	7.0	5.9	0.06
lines / 100w	13.0	17.5	27.9	26.5	19.9	13.8	0.12

The styles are ordered by length, but length alone does not characterize the partition. Formatting is the strongest of the measured separators, with $\eta^2 = 0.12$ for lines per 100 words, while backtracking and verification also differ systematically. At the same time, all effect sizes remain moderate ($\eta^2 \approx 0.06$–0.12), indicating substantial variation within each cluster.

Summary. The discovered styles exhibit three properties relevant to our downstream use of them. First, they correspond to systematic differences in observable reasoning-trace form. Second, they recur across teacher models and across solutions to the same mathematical problem, rather than simply encoding teacher or question identity. Third, their empirical frequencies are highly imbalanced, with the dominant style accounting for more than 40% of the training traces. These properties motivate using the discovered styles as a basis for controlling the composition of the distillation data.

C.3 Examples of Discovered Styles in Teacher Traces

To illustrate the styles discovered in the original teacher corpus, we show solutions to the same mathematical problem assigned to each of the six GMM clusters. Holding the question fixed makes differences in organization, verbosity, formatting, and reasoning presentation easier to compare.

The examples are drawn from the SCAS training corpus and labeled using the style representation and $K = 6$ GMM described in Section 4.2. The style names are descriptive glosses assigned after clustering and are not provided to the teacher models.

Question. For how many integer values of n between 1 and 1000 inclusive does the decimal representation of $\frac{n}{1400}$ terminate?

Gold answer: 142. Style counts among the nine teachers: style_1: 2, style_2: 2, style_3: 1, style_4: 1, style_5: 2, style_6: 1.

Style 1 (short textbook). 2 teacher traces were assigned to this cluster; we show all of them.

Teacher: `qwen2.5-72b-instruct`

> To determine for how many integer values of n between 1 and 1000 inclusive the decimal representation of $\frac{n}{1400}$ terminates, we need to understand when a fraction has a terminating decimal. A fraction in its simplest form has a terminating decimal if and only if the denominator (in its simplest form) has no prime factors other than 2 and 5.
>
> First, we factorize the denominator 1400:
>
> $$1400 = 14 \times 100 = (2 \times 7) \times (2^2 \times 5^2) = 2^3 \times 5^2 \times 7$$
>
> The prime factorization of 1400 includes the prime number 7, which means that $\frac{n}{1400}$ will not have a terminating decimal unless the 7 in the denominator is canceled out by a factor of 7 in the numerator n.
>
> Thus, n must be a multiple of 7. We need to find how many multiples of 7 there are between 1 and 1000 inclusive. This is equivalent to finding the largest integer k such that $7k \leq 1000$:
>
> $$k = \left\lfloor \frac{1000}{7} \right\rfloor = \lfloor 142.857 \rfloor = 142$$
>
> So, there are 142 multiples of 7 between 1 and 1000. Therefore, there are 142 integer values of n for which the decimal representation of $\frac{n}{1400}$ terminates.
>
> The final answer is:
>
> $$\boxed{142}$$

Teacher: `olmo-3.1-32b-instruct`

> To determine for how many integer values of n between 1 and 1000 inclusive the decimal representation of $\frac{n}{1400}$ terminates, we need to recall the condition for a fraction to have a terminating decimal expansion.
>
> A rational number $\frac{a}{b}$ in lowest terms has a terminating decimal if and only if the denominator (after simplifying the fraction) has no prime factors other than 2 or 5. In other words, the denominator's prime factorization contains only the primes 2 and/or 5.

Let's factorize 1400:

$$1400 = 14 \times 100 = (2 \times 7) \times (2^2 \times 5^2) = 2^3 \times 5^2 \times 7$$

So the prime factorization is $2^3 \times 5^2 \times 7$.

For $\frac{n}{1400}$ to have a terminating decimal, after simplifying the fraction, the denominator should have only 2s and 5s as prime factors. That means the factor of 7 in the denominator must be canceled out by a factor of 7 in the numerator n.

Let's formalize this:

Let $d = \gcd(n, 1400)$. Then

$$\frac{n}{1400} = \frac{n/d}{1400/d}$$

For the denominator $1400/d$ to have no prime factors other than 2 and 5, all factors of 7 in 1400 must be canceled by d. That is, d must be divisible by 7.

But more precisely, since 1400 has one factor of 7, in order for $1400/d$ to not have a factor of 7, d must include that 7, i.e., 7 divides d.

But actually, let's think step by step:

Let's write $n = 7k \cdot m$, where k is an integer and m is not divisible by 7 (i.e., k is the highest power of 7 dividing n). But perhaps a better way is:

Let's let $d = \gcd(n, 1400)$. Then the reduced denominator is $\frac{1400}{d}$. We want $\frac{1400}{d}$ to have no factor of 7, i.e., all the 7s in 1400 must be canceled by d. Since 1400 has exactly one factor of 7, d must be divisible by 7.

That is, 7 divides d, i.e., 7 divides $\gcd(n, 1400)$, i.e., 7 divides n.

Thus, n must be divisible by 7.

But wait: Let's check this more carefully.

Let's suppose n is divisible by 7, say n = 7m.

Then:

$$\frac{n}{1400} = \frac{7m}{1400} = \frac{m}{200}$$

Now, 200 factors as $2^3 \times 5^2$, which contains only 2s and 5s. So any fraction with denominator 200 (in lowest terms or not) will have a terminating decimal.

But wait, what if m and 200 have common factors? That's fine, because after simplifying, the denominator will still have only 2s and 5s as factors.

So if n is divisible by 7, then the reduced denominator is a divisor of 200, which is only 2s and 5s.

Now, what if n is NOT divisible by 7?

Then, in the reduced form, the denominator will still have a factor of 7 (since the numerator can't cancel it), so the decimal will not terminate.

Therefore, the decimal expansion of $\frac{n}{1400}$ terminates if and only if n is divisible by 7.

Now we just need to count the number of integers n between 1 and 1000 inclusive that are divisible by 7.

Let's compute:

The smallest such n is 7, the largest is the greatest multiple of 7 less than or equal to 1000.

Compute $\left\lfloor \frac{1000}{7} \right\rfloor = 142$ since $7 \times 142 = 994$, and $7 \times 143 = 1001 > 1000$.

Thus, the multiples of 7 between 1 and 1000 are:

$7 \times 1, 7 \times 2, ..., 7 \times 142$

So there are 142 such integers.

Let's check with a quick example:

For n = 7: $\frac{7}{1400} = \frac{1}{200} = 0.005$ (terminates) For n = 14: $\frac{14}{1400} = \frac{1}{100} = 0.01$ (terminates) For n = 1: $\frac{1}{1400}$ — denominator has 7, so decimal does not terminate.

Thus, the answer is 142.

But wait, let's consider if there are any other restrictions. For example, what if n is divisible by higher powers of 7? But as shown above, even if n is divisible by 49 or 343, after dividing by 7, the denominator becomes 200, and any further common factors only reduce the denominator further, but never introduce new primes.

So the answer is correct.

Final Answer: $\boxed{142}$

Style 2 (short unmarked). 2 teacher traces were assigned to this cluster; we show all of them.

Teacher: `llama-3.3-70b-instruct`

> **Step 1: To determine if the decimal representation of** $\frac{n}{1400}$ terminates, we need to analyze the prime factors of the denominator. The prime factorization of 1400 is $2^3 \cdot 5^2 \cdot 7$. For the decimal representation to terminate, the denominator must only have prime factors of 2 and 5 after simplification with the numerator.
>
> **Step 2: We need to find the values of** n that will simplify the factor of 7 in the denominator. This means n must be a multiple of 7 for the fraction $\frac{n}{1400}$ to simplify to a form where the denominator only contains factors of 2 and 5.
>
> **Step 3: Calculate the number of multiples of 7 between 1 and 1000 inclusive.** To do this, we divide 1000 by 7 and take the floor of the result, which gives us the number of multiples of 7 in the range.
>
> **Step 4: Perform the calculation.** $\left\lfloor \frac{1000}{7} \right\rfloor = \lfloor 142.857 \rfloor = 142$.
>
> The final answer is: $\boxed{142}$

Teacher: `gemma-4-31b-it`

> To determine for how many integer values of n between 1 and 1000 inclusive the decimal representation of $\frac{n}{1400}$ terminates, we follow these steps:
>
> 1. **Condition for a terminating decimal:** A simplified fraction $\frac{a}{b}$ (where $\gcd(a, b) = 1$) has a terminating decimal representation if and only if the prime factorization of the denominator b contains no prime factors other than 2 and 5.
>
> 2. **Analyze the denominator:** The denominator is 1400. We find its prime factorization: $1400 = 14 \times 100 = (2 \times 7) \times (10^2) = 2 \times 7 \times (2 \times 5)^2 = 2 \times 7 \times 2^2 \times 5^2 = 2^3 \times 5^2 \times 7$.
>
> 3. **Determine the condition for n:** The fraction is $\frac{n}{1400} = \frac{n}{2^3 \times 5^2 \times 7}$. For the decimal to terminate, the prime factor 7 in the denominator must be canceled out by a factor of 7 in the numerator n. Therefore, n must be a multiple of 7.
>
> 4. **Count the multiples of 7 between 1 and 1000:** We need to find the number of integers n such that $1 \leq n \leq 1000$ and n is a multiple of 7. This is equivalent to finding the largest integer k such that $7k \leq 1000$. Dividing 1000 by 7: $1000 \div 7 = 142.857...$
>
> The largest integer k is $\lfloor \frac{1000}{7} \rfloor = 142$.
>
> The multiples of 7 are $7 \times 1, 7 \times 2, \ldots, 7 \times 142$. There are 142 such values.

Style 3 (contest markdown). One teacher trace was assigned to this cluster.

Teacher: `gpt-5-chat_2025-10-03`

> Let's restate the problem carefully.
>
> We want the number of integer values n between 1 and 1000 (inclusive) for which the decimal representation of
>
> $$\frac{n}{1400}$$
>
> **terminates**.
>
> —
>
> **Step 1. Recall the condition for a terminating decimal.**
>
> A fraction $\frac{a}{b}$ in lowest terms has a terminating decimal if and only if the denominator b has no prime factors other than 2 or 5.
>
> That is, after simplifying $\frac{a}{b}$, the denominator must be of the form $2^x 5^y$.
>
> —
>
> **Step 2. Factorize the denominator** 1400.
>
> $$1400 = 14 \times 100 = (2 \times 7) \times (2^2 \times 5^2) = 2^3 \times 5^2 \times 7$$
>
> So $1400 = 2^3 \cdot 5^2 \cdot 7$.
>
> —
>
> **Step 3. Simplify the fraction** $\frac{n}{1400}$.

When we reduce $\frac{n}{1400}$ to lowest terms, we can cancel out common factors of 2, 5, or 7 that occur in n.
Let's write

$$n = 2^a \cdot 5^b \cdot 7^c \cdot k$$

where k is not divisible by 2, 5, or 7.
Then

$$\frac{n}{1400} = \frac{2^a \cdot 5^b \cdot 7^c \cdot k}{2^3 \cdot 5^2 \cdot 7} = \frac{k}{2^{3-a} \cdot 5^{2-b} \cdot 7^{1-c}}$$

after canceling common factors (some exponents may become 0 or negative, meaning those primes disappear from the denominator).
—
Step 4. Condition for termination.
For the decimal to terminate, the denominator after simplification must have *no factor of 7* (and only 2's and 5's allowed).
That means the factor of 7 in the denominator must cancel completely.
So we need $c \geq 1$, i.e. n must be **divisible by 7**.
—
Step 5. That's the only requirement.
If n is divisible by 7, then the denominator after cancellation will be $2^x 5^y$, which gives a terminating decimal.
If n is **not** divisible by 7, then a factor of 7 remains in the denominator, making the decimal **non-terminating**.
—
Step 6. Count how many n between 1 and 1000 are multiples of 7.
We need integers n such that $n = 7k$ and $1 \leq n \leq 1000$.
The largest k satisfying $7k \leq 1000$ is $k = \lfloor 1000/7 \rfloor = 142$.
So the positive multiples of 7 up to 1000 are $7, 14, \ldots, 994$. There are **142** of them.
—
Final Answer:

$$\boxed{142}$$

Style 4 (longer markdown). One teacher trace was assigned to this cluster.

Teacher: `gpt-oss-120b`

The fraction $\frac{n}{1400}$ will have a terminating decimal exactly when, after reducing the fraction to lowest terms, the denominator contains no prime factors other than 2 and 5.
Factor the denominator:

$$1400 = 14 \cdot 100 = (2 \cdot 7)(2^2 \cdot 5^2) = 2^3 \cdot 5^2 \cdot 7.$$

The only "bad" prime factor is 7. To eliminate this factor we must cancel it with a factor of 7 in the numerator. Hence n must be divisible by 7.
If $n = 7k$ (with k an integer), then

$$\frac{n}{1400} = \frac{7k}{7 \cdot 2^3 \cdot 5^2} = \frac{k}{2^3 \cdot 5^2},$$

and the remaining denominator consists solely of powers of 2 and 5, guaranteeing a terminating decimal. No further condition on k is needed.
Thus the problem reduces to counting the multiples of 7 between 1 and 1000:

$$\left\lfloor \frac{1000}{7} \right\rfloor = 142.$$

Therefore, there are

$$\boxed{142}$$

integer values of n in the range $1 \leq n \leq 1000$ for which $\frac{n}{1400}$ terminates.

Style 5 (default mix). 2 teacher traces were assigned to this cluster; we show all of them.

Teacher: `qwen3.5-27b`

Here's a thinking process that leads to the solution:

1. **Analyze the Problem:** * We are looking for the number of integers n such that $1 \leq n \leq 1000$. * The condition is that the decimal representation of the fraction $\frac{n}{1400}$ must terminate.

2. **Recall the Condition for Terminating Decimals:** * A fraction $\frac{a}{b}$ (where $\gcd(a,b) = 1$) has a terminating decimal representation if and only if the prime factorization of the denominator b contains *only* the prime factors 2 and 5. * If the fraction is not in simplest form, say $\frac{a}{b}$, we must first reduce it to lowest terms $\frac{a'}{b'}$. Then the condition is that b' must be of the form $2^x 5^y$ for non-negative integers x, y.

3. **Apply to the Specific Fraction:** * The fraction is $\frac{n}{1400}$. * Let's find the prime factorization of the denominator 1400. * $1400 = 14 \times 100$ * $14 = 2 \times 7$ * $100 = 10^2 = (2 \times 5)^2 = 2^2 \times 5^2$ * So, $1400 = 2 \times 7 \times 2^2 \times 5^2 = 2^3 \times 5^2 \times 7^1$.

4. **Simplify the Condition:** * Let the reduced fraction be $\frac{n'}{1400'}$. * We need the denominator $1400'$ (after canceling common factors with n) to have no prime factors other than 2 and 5. * The prime factors of 1400 are 2, 5, and 7. * The factors 2 and 5 are allowed. The factor 7 is *not* allowed. * Therefore, to make the decimal terminate, the factor 7 in the denominator must be canceled out by a factor of 7 in the numerator n. * This means n must be divisible by 7.

5. **Formulate the Counting Problem:** * We need to find the number of integers n in the range $[1, 1000]$ such that n is a multiple of 7. * Wait, is that sufficient? * If n is a multiple of 7, let $n = 7k$. * Then $\frac{n}{1400} = \frac{7k}{1400} = \frac{k}{200}$. * The denominator is 200. $200 = 2^3 \times 5^2$. * The prime factors are only 2 and 5. * So, any multiple of 7 works. * Is it necessary? * Suppose $\frac{n}{1400}$ terminates. * Then the denominator in simplest form must be of the form $2^x 5^y$. * The original denominator is $2^3 \cdot 5^2 \cdot 7$. * For the 7 to disappear from the denominator, it must be a factor of the numerator n. * So yes, n must be divisible by 7.

6. **Calculate the Count:** * We need to count multiples of 7 between 1 and 1000 inclusive. * The multiples are $7 \times 1, 7 \times 2, \ldots, 7 \times k$. * We need $7k \leq 1000$. * $k \leq \frac{1000}{7}$. * Let's perform the division: * $1000 \div 7 = 142.857\ldots$ * $140 \times 7 = 980$. * $142 \times 7 = 994$. * $143 \times 7 = 1001$. * So, the largest integer k is 142. * Since n starts at 1, k starts at 1. * Therefore, there are 142 such values.

7. **Double Check:** * Let $n = 7$. $\frac{7}{1400} = \frac{1}{200} = 0.005$. Terminates. * Let $n = 1$. $\frac{1}{1400} = \frac{1}{2^3 \cdot 5^2 \cdot 7}$. Denominator has factor 7. Repeats. * Let $n = 14$. $\frac{14}{1400} = \frac{1}{100} = 0.01$. Terminates. * The logic holds: we need to eliminate the factor 7 from the denominator. * The denominator is $1400 = 200 \times 7$. * $\frac{n}{1400} = \frac{n}{200 \times 7}$. * For this to terminate, the simplified denominator must not have any prime factors other than 2 and 5. * The prime factors of 1400 are $\{2, 5, 7\}$. * The prime factors of the simplified denominator are $\{2, 5, 7\} \setminus \{p : p \text{ divides } \gcd(n, 1400)\}$. * For the set to be a subset of $\{2, 5\}$, the factor 7 must be removed. * This implies 7 must divide $\gcd(n, 1400)$, which implies 7 must divide n. * So n must be a multiple of 7. * Range is $1 \leq n \leq 1000$. * Count is $\lfloor \frac{1000}{7} \rfloor$. * $1000 = 700 + 280 + 20$. * $1000 = 700 + 280 + 14 + 6$. * $1000 = 7(100 + 40 + 2) + 6$. * $1000 = 7(142) + 6$. * Quotient is 142.

8. **Structure the Answer:** * State the condition for a terminating decimal. * Prime factorize the denominator 1400. * Determine the condition on n (must be divisible by 7). * Calculate the number of multiples of 7 in the range [1, 1000]. * State the final answer.

9. **Refinement:** * Is there any edge case? * What if n shares factors with 200? That doesn't matter, it just changes the power of 2 or 5 in the denominator, which is allowed. * Example: $n = 1400$. $\frac{1400}{1400} = 1$. Terminates. 1400 is a multiple of 7. But n is restricted to 1000. * Example: $n = 700$. $\frac{700}{1400} = \frac{1}{2}$. Terminates. * Example: $n = 14$. $\frac{14}{1400} = \frac{1}{100}$. Terminates. * Example: $n = 2$. $\frac{2}{1400} = \frac{1}{700}$. Denominator has factor 7. Does not terminate. * The logic is solid.

10. **Final Calculation:** * $\lfloor 1000/7 \rfloor = 142$.

* Wait, let's re-read the question carefully. "integer values of n between 1 and 1000 inclusive". Yes. * "decimal representation ... terminate". Yes.
*

Teacher: `qwen3-32b`

¡think¿ Okay, so I need to figure out how many integers n between 1 and 1000 inclusive make the decimal representation of n/1400 terminate. Hmm, terminating decimals... I remember that a fraction will have a terminating decimal if the denominator, after simplifying the fraction to its lowest terms, has no prime factors other than 2 and 5. Is that right? Let me confirm. Yeah, I think that's the rule. So if you can write the denominator as 2^a * 5^b where a and b are non-negative integers, then the decimal terminates. Otherwise, it repeats.

So first, I need to take the fraction n/1400 and simplify it. The key is that after simplifying, the denominator should only have 2s and 5s in its prime factorization. Let me recall that 1400 can be factored into primes. Let me factor 1400.

Starting with dividing by 2: 1400 ÷ 2 = 700. Again by 2: 700 ÷ 2 = 350. Again by 2: 350 ÷ 2 = 175. Now 175 is not divisible by 2. Let me check 5: 175 ÷ 5 = 35. Then 35 ÷ 5 = 7. And 7 is prime. So putting it all together, 1400 factors into 2^3 * 5^2 * 7^1. So the prime factors are 2, 5, and 7.

Therefore, if we have n/1400, to have a terminating decimal, when we reduce the fraction, the denominator must not have any prime factors other than 2 and 5. Which means that any factor of 7 in the denominator must be canceled out by a factor of 7 in the numerator n. Therefore, n must be a multiple of 7. Is that correct? Let me think. If n shares a common factor with 1400, specifically the 7, then when we reduce the fraction, the 7 in the denominator will be canceled.

So for n/1400 to reduce to something with denominator only 2s and 5s, n must be divisible by 7. Let me check. Suppose n is divisible by 7. Then n = 7k for some integer k. Then n/1400 = 7k/(1400) = k/(200). Because 1400 divided by 7 is 200. And 200 factors into 2^3 * 5^2. Which only has 2s and 5s. So yes, if n is divisible by 7, then the reduced denominator is 200, so decimal terminates.

Conversely, if n is not divisible by 7, then when you reduce n/1400, there will still be a 7 in the denominator, making it non-terminating. Therefore, the decimal terminates if and only if n is divisible by 7.

Therefore, the problem reduces to finding the number of integers between 1 and 1000 inclusive that are divisible by 7. But wait, hold on. Is there a catch here? Because maybe when you reduce the fraction, even if n is divisible by 7, maybe there are other prime factors introduced? Wait, no. Because the original denominator is 2^3 * 5^2 * 7. If n is divisible by 7, then when you divide numerator and denominator by 7, you get denominator 2^3 * 5^2. Which is only 2s and 5s. So regardless of what else n is, as long as it cancels out the 7, the rest of the denominator is already okay.

But hold on, what if n has other common factors with the original denominator? For example, suppose n is divisible by 7 and also by 2 or 5. Then when you reduce the fraction, you would reduce the powers of 2 and 5 in the denominator. But even if you do that, the denominator still only has 2s and 5s. So even if n has common factors with 2 or 5, it doesn't affect the termination. Because the presence of 2s and 5s in the denominator is allowed. So the only problem is the 7.

Therefore, my conclusion is that n must be divisible by 7. Therefore, the number of such n between 1 and 1000 inclusive is equal to the number of multiples of 7 in that range.

But let me check with an example. Let's take n = 7. Then 7/1400 = 1/200 = 0.005, which terminates. Good. How about n = 14? 14/1400 = 1/100 = 0.01, terminates. n = 21: 21/1400 = 3/200 = 0.015, terminates. Okay. What if n is 7* something that also has 2s or 5s? Let's say n=14, which is 2*7. Then 14/1400 = 1/100. Which is 0.01. Still terminates. If n=35, which is 5*7. 35/1400 = 1/40 = 0.025. Terminates. If n=70, which is 2*5*7. 70/1400 = 1/20 = 0.05. Terminates. So even though n has factors of 2 and 5, as long as it cancels the 7, the decimal still terminates. Therefore, my conclusion is correct: n must be divisible by 7.

Therefore, the answer is the number of multiples of 7 between 1 and 1000 inclusive.

Now, how do we compute that? Well, the formula is floor(1000/7). But let me verify. The number of multiples of 7 between 1 and N inclusive is floor((N)/7). However, we need to check whether 1000 is divisible by 7. Let me compute 7*142 = 994. 7*143 = 1001. So 143*7 is 1001, which is over 1000. Therefore, the number of multiples of 7 between 1 and 1000 is 142. Wait, 7*142=994. Then 994 + 7 = 1001. So the last multiple of 7 less than or equal to 1000 is 994. Therefore, the number of multiples is 142. But wait, starting from 7*1=7, up to 7*142=994. So yes, 142 numbers. But let me check with the formula: floor(1000/7). Let me compute 1000 divided by 7.

7*142 = 994. 1000 - 994 = 6. So 1000/7 is 142 + 6/7, so floor(1000/7) is 142. Therefore, the number of multiples is 142.

But wait, hold on. The question says between 1 and 1000 inclusive. So n starts at 1. The first multiple of 7 is 7*1=7. So yes, so 142 numbers. Therefore, the answer is 142?

> But let me check with another approach. Let me think if there's a possibility that even if n is divisible by 7, when you reduce n/1400, maybe the denominator could have other primes? Wait, suppose n is divisible by 7 but also by some other primes? Wait, no. Because if n has other primes, say 3, then when you reduce the fraction, would that prime remain in the denominator? Let me think.
>
> Suppose n is 7*3 = 21. Then n/1400 = 21/1400 = 3/200. The denominator is 200, which is 2^3 * 5^2. So even though the numerator has a 3, which is a prime not in 2,5, since it's in the numerator, it doesn't affect the denominator. Therefore, the denominator still only has 2s and 5s. Therefore, decimal terminates.
>
> Another example: n=7*3* something. Let's take n=7*3*2=42. Then 42/1400 = 3*2/(200) = 6/200 = 3/100 = 0.03. Terminates.
>
> So even if n has other prime factors besides 2 and 5 and 7, as long as it cancels the 7 in the denominator, the decimal will terminate. Therefore,

Style 6 (long think-block). One teacher trace was assigned to this cluster.

Teacher: `phi-4-reasoning-plus`

> ¡think¿We are asked: For how many integer values of n between 1 and 1000 inclusive does the decimal representation of n/1400 terminate?
>
> We need to consider fraction n/1400, n is integer between 1 and 1000. We need to find number of n for which the fraction's decimal representation terminates. A rational fraction terminates if and only if after reduction, the denominator has only factors 2 and 5. Alternatively, we can say: fraction n/1400 terminates if and only if n is divisible by all prime factors of denominator aside from 2 and 5. Let's check factorization of 1400: 1400 = 14*100 = (2*7)* (2^2*5^2)= 2^3 * 5^2 *7. So the fraction is n / (2^3 * 5^2 * 7). For fraction to be terminating, after reducing, the denominator must be of the form 2^a * 5^b only. That means that the factor of 7 in denominator must be canceled out by factor in numerator. So n must be divisible by 7. But wait, is that sufficient? Let's check: if n is divisible by 7, then factor of 7 cancels out, leaving denominator of form 2^3 * 5^2 which is terminating. But is it necessary? Yes, if n is not divisible by 7, then the factor 7 remains in denominator and the fraction will be of the form something/7, which does not terminate (since 7 is not of the form 2^a 5^b). So the necessary and sufficient condition: n must be divisible by 7.
>
> But wait, check: Actually, it's not exactly necessary that n must be divisible by 7. Wait, check: if n is not divisible by 7, then the denominator after reduction remains with factor 7, so it's not terminating. So indeed, n must be divisible by 7. So the answer: count multiples of 7 in the interval [1,1000]. How many multiples of 7 are there between 1 and 1000? The smallest multiple is 7, largest multiple is floor(1000/7)*7. floor(1000/7) = 142 because 7*142=994, 7*143=1001 ¿1000. So count = 142. But wait, double-check: Actually, is that the complete answer? Let's check: n/1400. But wait, is that sufficient? Let's test: n=7. Then fraction equals 7/1400 = 1/200. That terminates (1/200 = 0.005). Also n=14, then fraction equals 14/1400 = 1/100, which terminates. So yes.
>
> Thus answer: 142.
>
> Wait, but check: n=0? But n must be between 1 and 1000 inclusive. So yes.
>
> Double-check: If n is divisible by 7, then n=7k. Then fraction equals (7k)/1400 = k/200, and since 200=2^3*5^2, so that fraction terminates. So answer: 142.
>
> So answer: 142.
>
> But wait, we must check if there's any possibility that n might have factors that cancel some factors of 2 or 5? But that's not necessary. Actually, wait, let me re-read the condition for a fraction a/b in simplest form to have terminating decimal representation. It is that b's prime factors can only be 2 and 5. That is exactly what we are doing. So yes.
>
> Thus answer: 142.
>
> Double-check: The number of multiples of 7 between 1 and 1000 equals floor(1000/7) = 142.
>
> Thus answer: 142.
>
> I'll now produce answer: 142.
>
> I'll now produce answer in plain text.
>
> I'll produce answer in plain text with explanation: "The fraction n/1400 terminates if and only if the reduced denominator has no prime factors other than 2 and 5. Since 1400=2^3 * 5^2 *7, the factor 7 must be cancelled by n. Therefore, n must be divisible by 7. The number of multiples of 7 between 1 and 1000 is floor(1000/7)=142. Therefore, the answer is 142."

I'll produce answer with explanation.

I'll now produce answer.¡/think¿To determine when the decimal representation of n/1400 terminates, we start by factoring the denominator:

1400 = 14 × 100 = (2 × 7) × ($2^2 \times 5^2$) = $2^3 \times 5^2 \times 7$.

A fraction in lowest terms has a terminating decimal if and only if its denominator has no prime factors other than 2 and 5. In the fraction n/1400, the factor 7 must be canceled out by n. This means n must be divisible by 7.

Now, we need to count how many integers n between 1 and 1000 (inclusive) are divisible by 7. The smallest multiple of 7 in this range is 7, and the largest is found by dividing 1000 by 7:

1000 ÷ 7 142.857, so the largest integer multiple is 7 × 142 = 994.

Thus, there are 142 multiples of 7 between 1 and 1000.

Therefore, there are 142 integer values of n between 1 and 1000 for which n/1400 has a terminating decimal representation.

C.4 LLM BASED STYLE DISCOVERY

As a human-readable check on the autoencoder taxonomy, we asked a long-context LLM to invent writing-style clusters from unlabeled SCAS teacher writeups. The judge was Cursor Grok 4.6 (`cursor-grok-4.6-high-fast` xAI (2026)), run as a fresh Cursor agent with no access to teacher IDs, question text, or GMM labels. We report the primary 180-trace protocol below (a 45-trace pilot on five style-balanced questions gave the same seven-style inventory).

Sample. 180 traces: 20 per teacher, distinct questions (excluding the five-question pilot), IDs shuffled as `T001–T180`. Long writeups were truncated in the middle (200 opening words + 80 closing words); the prompt told the model to judge style from opening and closing form.

Prompt (instructions only). The writeups then followed as `===== Txxx =====` blocks.

```
You are given unlabeled math solution writeups.  Do not
assume how many styles exist or who wrote them.  Some
writeups are truncated in the middle; judge style from the
opening and closing form.

Task:
1.  Infer writing-style clusters from formatting, voice,
structure, and surface form --- not from the math content or
the final numeric answer.
2.  Decide the number of styles yourself.  Name each style
with a short label and 2--4 sentence description of the
distinctive cues.
3.  Assign every ID to exactly one style.
4.  If two writeups solve different problems but share the
same manner of writing, they belong together.

Return ONLY:
- A list of styles:  label, description, member IDs
- A complete ID to style assignment
- A brief note on any IDs that were hard to place

Do not mention teachers, models, clusters, autoencoders, or
GMM.
```

Agreement. On this sample, chance-corrected agreement is LLM–teacher ARI=0.70, LLM–autoencoder ARI=0.07, autoencoder–teacher ARI=0.06. The LLM inventory tracks teacher surface fingerprints much more than the K=6 GMM bins.

Model output: seven styles. Verbatim from the Grok 4.6 response (member counts only; full ID lists omitted).

1. **Exploratory first-person monologue** (n=20). Opens in a talking-to-yourself voice (`Okay, so I need to...`, `Hmm`, `Let me think`, `Alright`). Often wrapped in a

Table 7: Blind LLM styles on the 180-trace sample (20 writeups per teacher) vs. the oracle teacher that produced each writeup. Most styles recover a single teacher at 18–20/20; the markdown-tutor bin merges gpt-5 and olmo; the instructional walkthrough merges gemma and qwen2.5.

LLM style	n	Teacher mix (of 20 per teacher)
Exploratory first-person monologue	20	qwen3 20/20
Telegraphic "We are asked" think-block	20	phi-4 20/20
Structured request-analysis outline	18	qwen3.5 18/20
H2-numbered recipe with stock closer	20	llama 20/20
Restating markdown tutor	34	gpt-5 18/20, olmo 16/20
Compact contest exposition	25	gpt-oss 20/20; also gpt-5 2, qwen3.5 2, qwen2.5 1
Instructional "To find…" walkthrough	43	gemma 20/20, qwen2.5 19/20; also olmo 4

Table 8: Same 180-trace sample, transposed: for each teacher, how many of its 20 writeups the LLM assigned to each invented style. Pure diagonals (e.g. llama→H2 recipe, gpt-oss→contest) are the recoverable fingerprints; olmo and gpt-5 share the markdown-tutor bin.

Teacher	mono.	think	outline	H2	md tutor	contest	walk.
gemma-4-31b-it	0	0	0	0	0	0	20
gpt-5-chat	0	0	0	0	18	2	0
gpt-oss-120b	0	0	0	0	0	20	0
llama-3.3-70b-instruct	0	0	0	20	0	0	0
olmo-3.1-32b-instruct	0	0	0	0	16	0	4
phi-4-reasoning-plus	0	20	0	0	0	0	0
qwen2.5-72b-instruct	0	0	0	0	0	1	19
qwen3-32b	20	0	0	0	0	0	0
qwen3.5-27b	0	0	18	0	0	2	0

Column keys match Table 7: mono. = exploratory monologue; think = telegraphic think-block; outline = request-analysis; H2 = H2-numbered recipe; md tutor = restating markdown; contest = compact contest; walk. = instructional walkthrough.

`<think>` block. The writeup rambles, restates the problem, second-guesses, and narrates memory of formulas before settling. Closings, when present, are an afterthought rather than a fixed template.

2. **Telegraphic "We are asked" think-block** (n=20). Opens `<think>We are asked:` or `<think>We are given:` with almost no space after the tag. Dense, repetitive inner speech (`We'll solve`, `Thus answer:`, `I'll produce answer` / `We'll produce answer`). After the block (or after a truncation cut) a cleaner rewrite often appears. Surface form is compressed and formulaic, not conversational.
3. **Structured request-analysis outline** (n=18). Opens with a meta header such as `Here's a thinking process that leads to the solution`, `Here's my thought process`, `Thinking Process:`, or `The user wants to...`. Numbered stages with nested bullets: **Analyze the Request**, **Understand the Goal**, Given / Constraints / Strategy / Execution. Treats the problem as a request to be dissected before computing.
4. **H2-numbered recipe with stock closer** (n=20). Every piece starts at `## Step 1:` (exactly two hashes) and walks through `## Step 2`, `## Step 3`, ... in running prose with inline `$...$` math. Always ends with the same closer: `The final answer is: $\boxed{...}$`. No checkmarks, no `<think>`, no "Let's restate."
5. **Restating markdown tutor** (n=34). Friendly coach voice: `Let's restate the problem carefully`, `Let's carefully go step-by-step`, `Let's break down the problem`. Heavy markdown: `---` rules, `### Given` / `### Step`, bold key phrases, and often a leading checkmark before **Final Answer** and `\boxed{}`. Even when the emoji is missing, the restating/step-card layout is the same.
6. **Compact contest exposition** (n=25). Starts in the middle of the math: `Let ...`, `The ...`, `First ...`, or a display `\[`. Tight declarative sentences,

`aligned` / `\qquad` / `\Longrightarrow`, lists of values, then a lone `\boxed{}`. Little student-facing scaffolding and almost no step theater.

7. **Instructional "To find…" walkthrough** (n=43). Opens `To find` / `To solve` / `To determine` / `To compute` (or a close `We are given` / `Let $x$ be` plus "we can"). Numbered `1. **Bold action:**` lists or sequential "we start / next / thus" prose. Homework-helper voice; closer is `\boxed{}`, `**Answer:**`, or a plain concluding sentence. Distinct from style 5 by the task-announcing lead-in and the absence of restating cards / checkmarks.

Hard-to-place notes (model). The judge flagged borderline openings that mix tutor restating with first-person self-correction, `To determine...` leads that continue as markdown cards, short we-voice walkthroughs that resemble contest setups, and `## Step` headings that close like style 5 rather than the stock "The final answer is:" recipe. Assignments followed the dominant surface cues (opening voice, headers, closer).

D IMPORTANCE-WEIGHTED SFT VARIANTS

We compare three ways of reweighting the same set of $N = 111{,}834$ training traces. In all cases, the underlying examples are unchanged; only their contribution to the token-level negative log-likelihood differs. Importance weights are mean-normalized to one before training, and we use $\varepsilon = 0.01$ for smoothing.

Let $C = 6$ denote the number of discovered styles. Question x_i has K_i teacher traces with style assignments $c_i^1, \ldots, c_i^{K_i}$. We define

$$n_{i,c} = \sum_{k=1}^{K_i} \mathbb{I}(c_i^k = c), \tag{18}$$

and

$$S_i = \{c : n_{i,c} > 0\}, \tag{19}$$

where $n_{i,c}$ is the number of traces for question x_i assigned to style c, and S_i is the set of styles actually observed for that question.

For the two question-conditional variants, we use the same smoothed proposal distribution. Before normalization across styles,

$$q_{\text{raw}}(c \mid x_i) = \frac{\sum_{k=1}^{K_i} \left(\mathbb{I}(c_i^k = c) + \varepsilon\right)}{K_i(1+\varepsilon)}. \tag{20}$$

Since

$$\sum_{c'=1}^{C} q_{\text{raw}}(c' \mid x_i) = \frac{1 + C\varepsilon}{1+\varepsilon}, \tag{21}$$

renormalizing over the C styles gives

$$q(c \mid x_i) = \frac{n_{i,c} + K_i\varepsilon}{K_i(1 + C\varepsilon)}. \tag{22}$$

Each training trace receives an importance weight

$$w = \frac{p}{q}, \tag{23}$$

where p is the target style distribution for the corresponding arm. Before optimization, we mean-normalize the weights,

$$w_j \leftarrow \frac{w_j}{\bar{w}}, \tag{24}$$

and optimize the weighted token-level negative log-likelihood

$$\mathcal{L} = \frac{1}{|B|} \sum_{j \in B} w_j \, \ell(x_j, (c_j, r_j), y_j)\,. \tag{25}$$

D.1 PER-QUESTION UNIFORM TARGET OVER ALL STYLES: IS-REBAL

In IS-REBAL, the target distribution is uniform over all C discovered styles for every question:

$$p_{\text{rebal}}(c \mid x_i) = \frac{1}{C}. \tag{26}$$

The corresponding importance weight is

$$w_{\text{rebal}}(c \mid x_i) = \frac{1/C}{q(c \mid x_i)}. \tag{27}$$

This can be viewed as approximating the expectation under a per-question uniform style distribution:

$$\mathbb{E}_{c\sim p_{\text{rebal}}(c|x_i)}\left[\ell\right] = \mathbb{E}_{c\sim q(c|x_i)}\left[\frac{p_{\text{rebal}}(c \mid x_i)}{q(c \mid x_i)}\ell\right]. \tag{28}$$

Because the finite training set contains no trace for styles absent from S_i, the Monte Carlo estimator only receives contributions from observed styles. The target nevertheless assigns mass to all C styles, while smoothing ensures $q(c \mid x_i) > 0$ for every style. Consequently, the raw target mass represented by the available traces for question x_i is

$$\frac{|S_i|}{C}. \tag{29}$$

Thus, IS-REBAL is the most aggressive question-conditional balancing scheme: it treats all six styles as equally desirable for every question, even when only a subset of them is represented in the teacher traces.

D.2 PER-QUESTION UNIFORM TARGET OVER OBSERVED STYLES: IS-OBS

In IS-OBS, we instead define the target distribution only over styles that actually occur among the teacher traces for question x_i:

$$p_{\text{obs}}(c \mid x_i) = \frac{\mathbb{I}(c \in S_i)}{|S_i|}. \tag{30}$$

The importance weight is therefore

$$w_{\text{obs}}(c \mid x_i) = \frac{\mathbb{I}(c \in S_i)/|S_i|}{q(c \mid x_i)}. \tag{31}$$

Equivalently,

$$\mathbb{E}_{c\sim p_{\text{obs}}(c|x_i)}\left[\ell\right] = \mathbb{E}_{c\sim q(c|x_i)}\left[\frac{p_{\text{obs}}(c \mid x_i)}{q(c \mid x_i)}\ell\right]. \tag{32}$$

This variant equalizes only the styles for which a trace is actually available. For example, if a question has eight traces in s_5 and one trace in s_2, then

$$S_i = \{s_2, s_5\}, \tag{33}$$

and the target assigns probability $1/2$ to each of these styles and zero to the remaining four. Thus, IS-OBS rebalances within the observed support of each question without assigning target mass to unobserved styles.

D.3 DATASET-WIDE UNIFORM TARGET: IS-GLOBAL

The third variant ignores question identity and reweights styles according to their global frequency in the training corpus. Let

$$N_c = \sum_{i,k} \mathbb{I}(c_i^k = c), \tag{34}$$

and

$$N = \sum_{c=1}^{C} N_c. \tag{35}$$

We define the smoothed global proposal

$$q(c) = \frac{N_c + \varepsilon}{N + C\varepsilon}, \tag{36}$$

and use a uniform target over styles,

$$p_{\text{global}}(c) = \frac{1}{C}. \tag{37}$$

The resulting weight is

$$w_{\text{global}}(c) = \frac{1/C}{q(c)} \propto \frac{1}{N_c + \varepsilon}. \tag{38}$$

The corresponding importance-sampling identity is

$$\mathbb{E}_{c\sim p_{\text{global}}(c)}\left[\ell\right] = \mathbb{E}_{c\sim q(c)}\left[\frac{p_{\text{global}}(c)}{q(c)}\ell\right]. \tag{39}$$

Unlike the two question-conditional variants, IS-GLOBAL assigns one weight to each style for the entire corpus. Frequent styles such as s_5 are downweighted and rarer styles are upweighted, but the weighting does not depend on which styles are available for a particular question.

D.4 Summary of weighting schemes

The three variants differ only in the target distribution used to define the importance ratio:

$$\begin{aligned} \text{IS-REBAL}: &\quad p(c \mid x_i) = \text{Unif}(\{1,\dots,C\}), \\ \text{IS-OBS}: &\quad p(c \mid x_i) = \text{Unif}(S_i), \\ \text{IS-GLOBAL}: &\quad p(c) = \text{Unif}(\{1,\dots,C\}). \end{aligned} \tag{40}$$

All three retain the full set of training traces and differ only in how those traces are weighted during SFT. We report the downstream results of these three weighting schemes for Qwen3 0.6B (Figure 4), 1.7B (Figure 5), and 4B (Figure 6). We used IS-obs for the experiments in the main paper and the rest of experiments in the appendix.

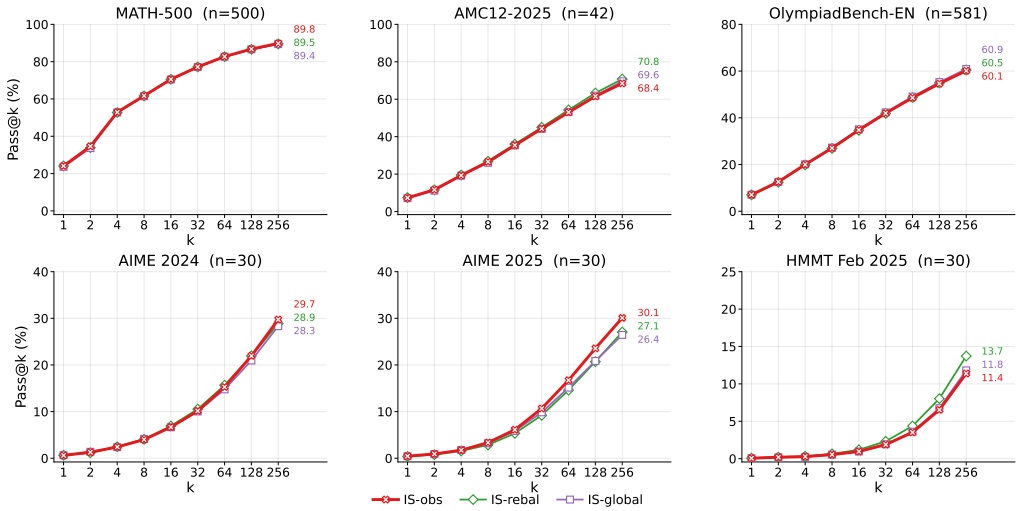


Figure 4: Qwen3 0.6B importance sampling variants

E Full results with Qwen3 1.7B and 4B

We report the Pass@k for Qwen3 1.7B in Figure 7 and 4B Figure 8.

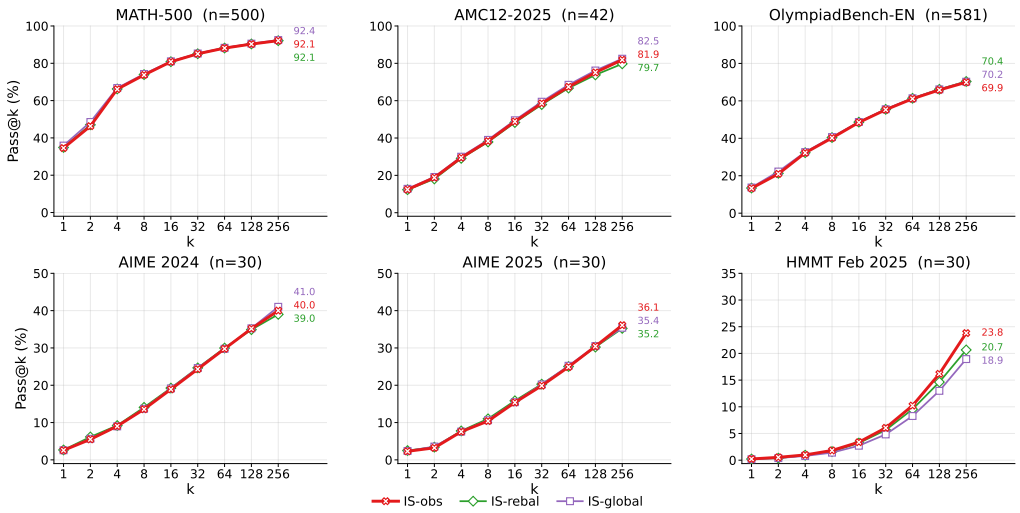


Figure 5: Qwen3 1.7B importance sampling variants

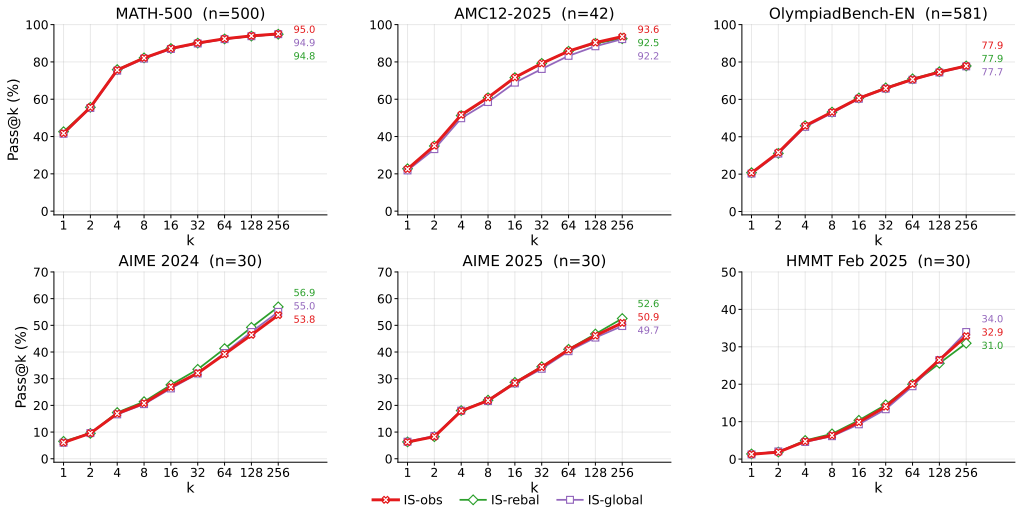


Figure 6: Qwen3 4B importance sampling variants

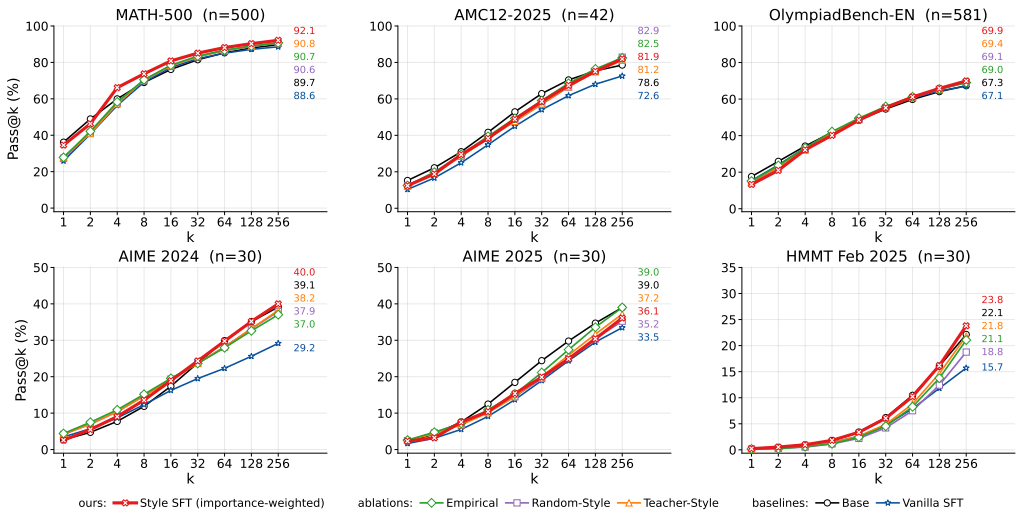


Figure 7: Pass@k Qwen3 1.7B

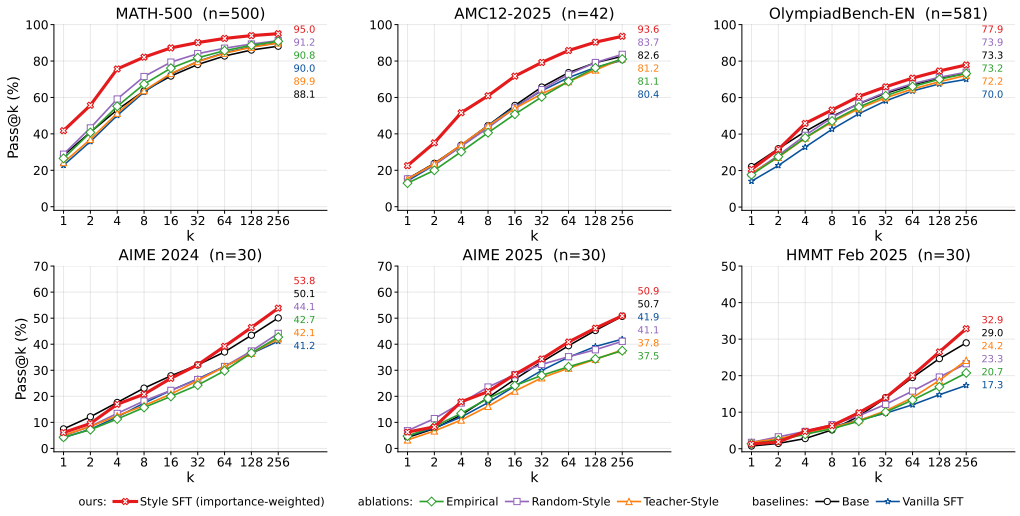


Figure 8: Pass@k Qwen3 4B

E.1 STYLE-PREFIX ADHERENCE

We evaluate whether the discrete style prefixes learned during SFT induce generations that resemble the corresponding styles discovered in the teacher traces. This analysis distinguishes between a prefix acting as an arbitrary control token and a prefix that actually steers generation toward the associated teacher-style mode.

For each MATH-500 generation produced under a requested prefix `[style_i]`, we assign a realized style $\hat{s}$ using the same handcrafted style feature space used to characterize the teacher clusters. The feature representation consists of length-normalized structural features from Table 6 together with log word count. We standardize generated traces using statistics computed from the teacher corpus and assign each generation to its nearest teacher-style centroid. We then compute the row-normalized confusion matrix

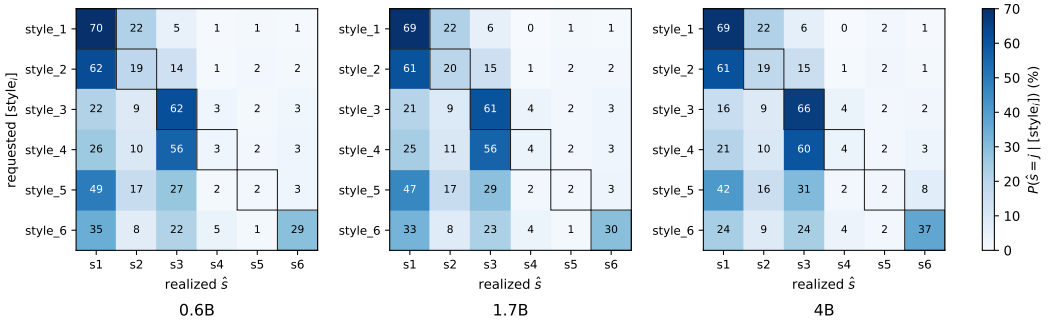
$$P\left(\hat{s} = j \mid \texttt{[style_i]}\right), \tag{41}$$

where rows correspond to the requested prefix and columns to the realized teacher-style assignment. Perfect control would place all mass on the diagonal, while a uniform assignment over the six styles would give diagonal mass of $1/6 = 16.7\%$.

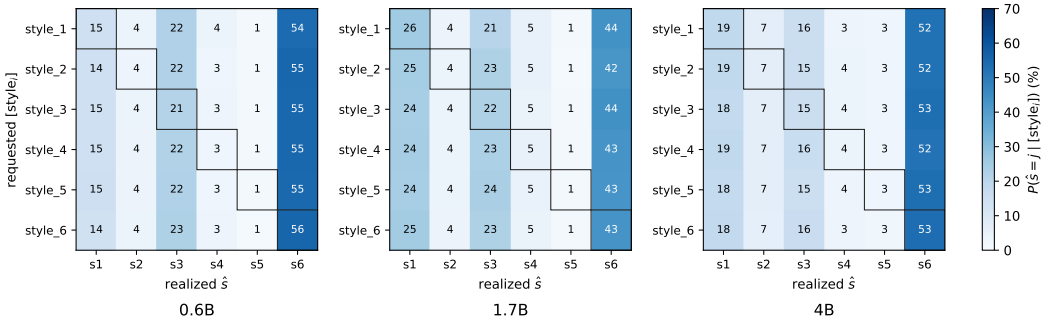
Figure 9 compares the full importance-weighted model with STYLE SFT (EMPIRICAL). For STYLE SFT (IMPORTANCE-WEIGHTED), mean diagonal agreement is 30.8%, 30.8%, and 32.9% for the 0.6B, 1.7B, and 4B students, respectively, approximately twice the chance level. Controllability is strongly style-dependent. Requested s_1 and s_3 are realized reliably, with roughly 60–70$\%$ of generations assigned to the corresponding teacher mode. Style s_6 shows moderate adherence, while s_2, s_4, and s_5 are frequently assigned to neighboring modes, particularly s_1 and s_3.

In contrast, STYLE SFT (EMPIRICAL) remains close to chance overall. Thus, simply attaching style labels to the empirical training distribution does not cause the student to reproduce the corresponding teacher-style structure reliably. Importance-weighted training substantially strengthens the association between the requested prefix and the realized style, although the six modes are not perfectly separable in the student.

These results support a partial-controllability interpretation of the style prefixes. The strongest prefixes recover identifiable teacher-style modes, while others remain conflated. We therefore treat the prefixes as controls associated with the discovered style structure rather than as perfect realizations of six independent styles.



(a) STYLE SFT (IMPORTANCE-WEIGHTED)



(b) STYLE SFT (EMPIRICAL)

Figure 9: Requested versus realized style on MATH-500. Rows denote the forced prefix `[style_i]` and columns denote the realized style $\hat{s}$ assigned by the nearest teacher-style centroid in the handcrafted style feature space. Values report $P(\hat{s} = j \mid \texttt{[style_i]})$. Diagonal mass measures adherence to the requested teacher-style mode, while off-diagonal mass reveals which styles are conflated. (a) Importance-weighted Style SFT yields substantially greater style fidelity than (b) empirical Style SFT, although adherence remains uneven across modes.

E.2 EXAMPLES OF STYLE-CONDITIONED GENERATIONS

To qualitatively illustrate the discovered styles, we show generations from the same MATH-500 problem under each of the six style prefixes. Holding the question fixed makes differences in organization, verbosity, and formatting easier to compare. These examples are generated by the Qwen3-0.6B student trained with importance-weighted Style SFT.

Question. Bill walks $\frac{1}{2}$ mile south, then $\frac{3}{4}$ mile east, and finally $\frac{1}{2}$ mile south. How many miles is he, in a direct line, from his starting point? Express your answer as a decimal to the nearest hundredth.

Gold answer. $\boxed{1.25}$

---

s_1 **— short textbook.**

Step 1: Calculate the total distance Bill walks south.
Bill walks $\frac{1}{2}$ mile south and then $\frac{1}{2}$ mile south. So, the total distance he walks south is $\frac{1}{2} + \frac{1}{2} = 1$ mile.
Step 2: Calculate the total distance Bill walks east.
Bill walks $\frac{3}{4}$ mile east.
Step 3: Apply the Pythagorean theorem.
The direct distance is

$$\sqrt{1^2 + \left(\frac{3}{4}\right)^2} = \sqrt{\frac{25}{16}} = \frac{5}{4} = 1.25.$$

The final answer is: $\boxed{1.25}$.

---

s_2 — short unmarked.

Bill walks $\frac{1}{2}$ mile south and then another $\frac{1}{2}$ mile south, giving a net displacement of 1 mile south. He also walks $\frac{3}{4}$ mile east.
The distance from the starting point is therefore

$$d = \sqrt{(-1)^2 + \left(\frac{3}{4}\right)^2} = \sqrt{1 + \frac{9}{16}} = \sqrt{\frac{25}{16}} = \frac{5}{4} = 1.25.$$

The final answer is: $\boxed{1.25}$.

---

s_3 — contest markdown.

Let Bill start at the origin $(0, 0)$.

1. After walking $\frac{1}{2}$ mile south:
$$\left(0, -\frac{1}{2}\right).$$
2. After walking $\frac{3}{4}$ mile east:
$$\left(\frac{3}{4}, -\frac{1}{2}\right).$$
3. After walking another $\frac{1}{2}$ mile south:
$$\left(\frac{3}{4}, -1\right).$$

Hence the distance from the origin is

$$\sqrt{\left(\frac{3}{4}\right)^2 + (-1)^2} = \sqrt{\frac{25}{16}} = \frac{5}{4} = 1.25.$$

Thus,

$$\boxed{1.25}.$$

---

s_4 — longer markdown.

Bill's net displacement is the sum of his movements:

- South: $-\frac{1}{2}$ mile
- East: $\frac{3}{4}$ mile
- South: $-\frac{1}{2}$ mile

The total southward displacement is

$$-\frac{1}{2} - \frac{1}{2} = -1,$$

while the eastward displacement is $\frac{3}{4}$ mile.
The straight-line distance from the starting point is therefore

$$\sqrt{(-1)^2 + \left(\frac{3}{4}\right)^2} = \sqrt{1 + \frac{9}{16}} = \sqrt{\frac{25}{16}} = \frac{5}{4} = 1.25.$$

$$\boxed{1.25}$$

---

s_5 **— default mix.**

Let's break down Bill's journey step by step. Suppose he starts at the origin $(0, 0)$.
First leg. He walks $\frac{1}{2}$ mile south, reaching

$$\left(0, -\frac{1}{2}\right).$$

Second leg. He walks $\frac{3}{4}$ mile east, reaching

$$\left(\frac{3}{4}, -\frac{1}{2}\right).$$

Third leg. He walks another $\frac{1}{2}$ mile south, reaching

$$\left(\frac{3}{4}, -1\right).$$

Using the distance formula,

$$d = \sqrt{\left(\frac{3}{4}\right)^2 + (-1)^2} = \sqrt{\frac{25}{16}} = \frac{5}{4} = 1.25.$$

Final answer:

$$\boxed{1.25}$$

---

s_6 **— long think-block.**

Let's carefully go step-by-step.
Step 1: Represent coordinates.
Bill's starting point is $(0, 0)$.
Step 2: First movement.
He walks $\frac{1}{2}$ mile south, taking him to

$$\left(0, -\frac{1}{2}\right).$$

Step 3: Second movement.
He walks $\frac{3}{4}$ mile east, taking him to

$$\left(\frac{3}{4}, -\frac{1}{2}\right).$$

Step 4: Third movement.
He walks another $\frac{1}{2}$ mile south:

$$\left(\frac{3}{4}, -\frac{1}{2} - \frac{1}{2}\right) = \left(\frac{3}{4}, -1\right).$$

Step 5: Distance from the starting point.

$$\text{Distance} = \sqrt{\left(\frac{3}{4}\right)^2 + (-1)^2} = \sqrt{\frac{9}{16} + 1} = \sqrt{\frac{25}{16}} = \frac{5}{4} = 1.25.$$

Final answer:

$$\boxed{1.25}$$

F FIXED STYLE PASS@k

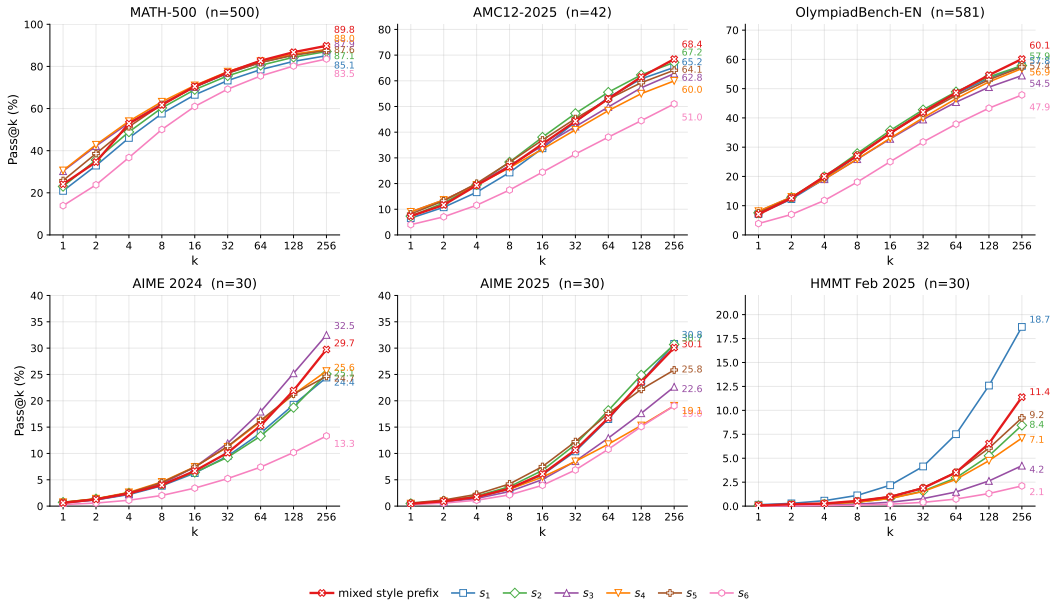


Figure 10: Qwen3 0.6B conditioning individual styles

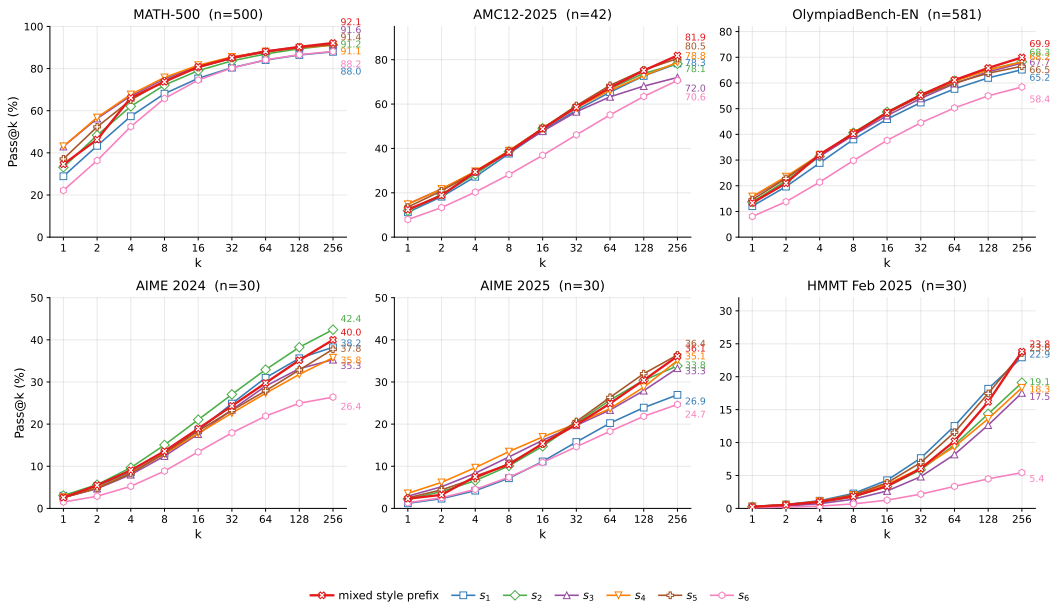


Figure 11: Qwen3 1.7B conditioning individual styles

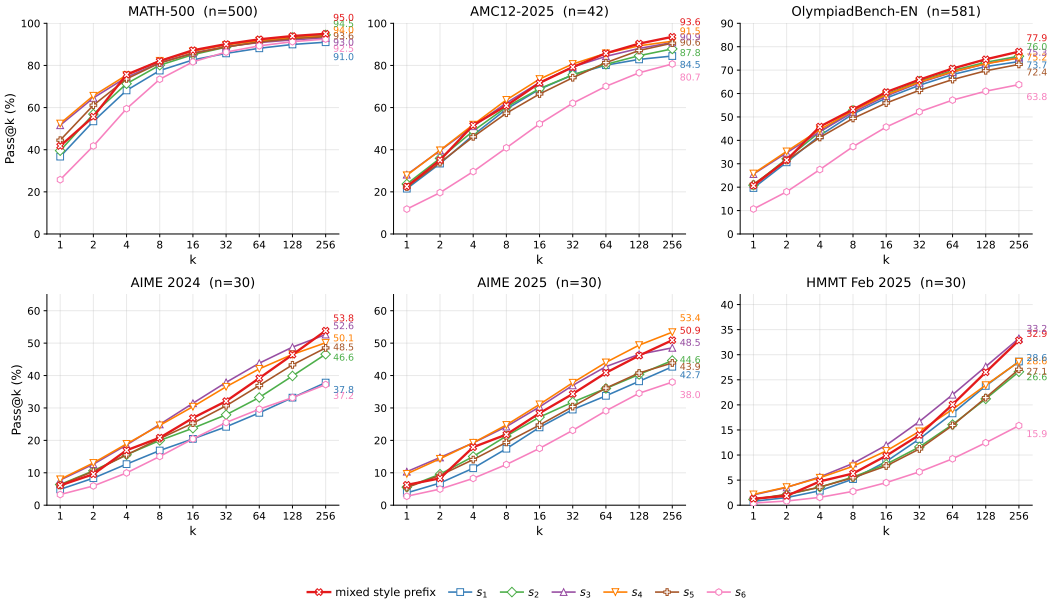


Figure 12: Qwen3 4B conditioning individual styles