TRAINING-INDUCED REASONING DIRECTIONS: UNCOVERING REUSABLE STRUCTURE IN POST-TRAINING REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Reasoning post-training substantially improves the problem-solving ability of large language models, yet the internal structure that supports these gains remains poorly understood. We introduce *Training-Induced Reasoning Directions* (TIRDs), fixed activation directions extracted from computational differences between models before and after post-training. First, TIRD isolates reusable capability-relevant information: injecting it into the Base model improves reasoning accuracy by up to 50.9 percentage points, while suppressing it substantially degrades the post-trained model. Second, TIRD reveals how this information supports reasoning during generation: it remains functionally important throughout the reasoning process and provides strong predictive support for key reasoning decisions, whose probabilities drop by over 50 percentage points on average under suppression. Third, we introduce AdaTIRD for difficulty-adaptive reasoning allocation. AdaTIRD reduces generation by up to 27.8% while preserving performance close to the trained model, without additional post-training. Together, these results reveal reusable functional structure in post-training reasoning capability that can be isolated, understood, and directly controlled at inference time. Our code is available at https://anonymous.4open.science/r/TIRD-A36B

1 INTRODUCTION

Post-training has become a central stage for improving the reasoning ability of large language models. Reinforcement Learning (RL) has driven major advances in recent reasoning models (Guo et al., 2025; Yang et al., 2025), while On-Policy Distillation (OPD) has emerged as an efficient paradigm for transferring and improving reasoning capability (Agarwal et al., 2024; Xiao et al., 2026). Despite these gains, the internal structure of the acquired capability remains poorly understood. Post-training changes a large number of parameters and affects computation throughout generation. It is therefore unclear whether the resulting capability depends only on many context-specific changes, or whether part of it is carried by reusable internal information.

Identifying such structure would provide a more direct way to study post-training capability. It would allow us to isolate a functionally important part of the training effect, understand how it supports reasoning during generation, and directly regulate its contribution at inference time.

Recent work suggests that compact functional structure may exist in post-training updates. Several studies report sparse or low-rank structure in RL-induced parameter changes (Mukherjee et al., 2025; Chen et al., 2026). AlphaRL (Cai et al., 2025) and EffOPD (Cai et al., 2026) further show strong concentration into dominant directions under RL and OPD, respectively. These findings motivate our hypothesis that compact structure in parameter updates may be reflected by a recurring directional pattern in model computation across questions and generation positions.

We test this hypothesis by comparing a model before post-training with its corresponding post-trained model across many reasoning examples. We aggregate their training-induced computational differences to extract fixed activation directions that capture recurring structure across contexts. We call these directions *Training-Induced Reasoning Directions* (TIRDs). Using TIRD as the central representation, we study what information it captures, how that information supports reasoning dur-

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.

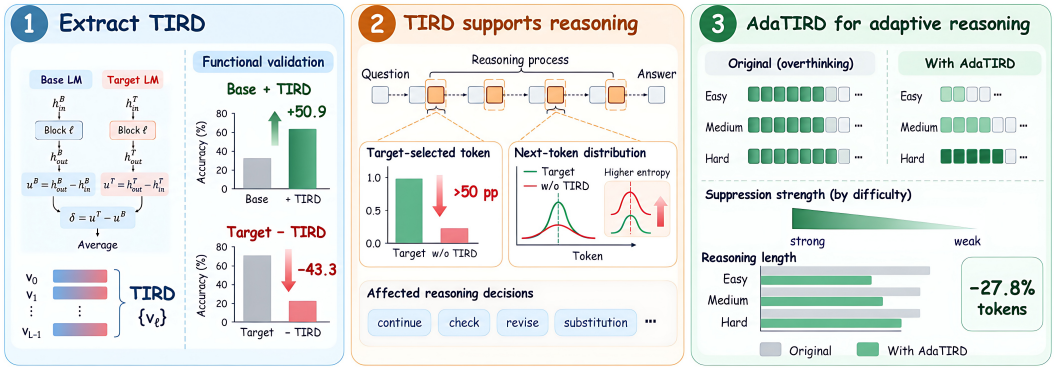

Figure 1: **TIRD for understanding and controlling post-training reasoning.** Left: TIRD captures reusable capability-relevant information from post-training computational changes. Middle: TIRD supports key reasoning decisions during generation. Right: AdaTIRD adapts TIRD suppression to problem difficulty for more efficient reasoning.

ing generation, and how it can be used for inference-time reasoning allocation. Figure 1 summarizes these three parts of our work.

Our work makes three contributions. **First, TIRD identifies reusable capability-relevant information induced by post-training.** Injecting TIRD into Base improves reasoning accuracy by up to 50.9 percentage points, while suppressing it in Target causes substantial performance degradation. **Second, TIRD reveals how this information supports reasoning during generation.** Suppressing TIRD lowers the probability of key reasoning decisions by over 50 percentage points on average, showing that TIRD provides strong predictive support as reasoning unfolds. **Third, we introduce AdaTIRD for difficulty-adaptive reasoning allocation.** AdaTIRD reduces generation by up to 27.8% while preserving most of the trained model's performance, without additional post-training.

2 Training-Induced Reasoning Directions

Post-training changes model computation in ways that vary across questions and generation positions. We ask whether part of these changes follows a shared direction that remains functionally useful across different reasoning contexts. Training-Induced Reasoning Directions (TIRDs) provide a direct test of this possibility. Section 2.1 constructs directions from matched Base–Target computations and defines the corresponding activation intervention. Section 2.2 tests whether these directions recover post-training gains, whether their effects depend on orientation rather than perturbation magnitude, and whether the trained model itself relies on the same directional information. Section 2.3 examines whether the identified functional effect generalizes beyond the task distribution used for extraction.

2.1 TIRD Extraction and Injection

We first construct directions that summarize computational changes repeatedly expressed across different reasoning contexts. Let B denote the Base checkpoint before training and T the Target checkpoint after RL or OPD. Comparing independently generated responses would mix changes caused by training with changes caused by different textual trajectories. We therefore let T generate a reference response $y_i = (y_{i,1}, \ldots, y_{i,n_i})$ to question x_i, and replay the same question–response sequence through both models.

At the position predicting response token $y_{i,t}$, let $h^{M,\text{in}}_{i,t,\ell}$ and $h^{M,\text{out}}_{i,t,\ell}$ denote the input and output of layer ℓ in model M, computed from the shared prefix $(x_i, y_{i,<t})$. We define the update contributed by the current block and its change after training as

$$u^{M}_{i,t,\ell} = h^{M,\text{out}}_{i,t,\ell} - h^{M,\text{in}}_{i,t,\ell}, \qquad \delta_{i,t,\ell} = u^{T}_{i,t,\ell} - u^{B}_{i,t,\ell}. \tag{1}$$

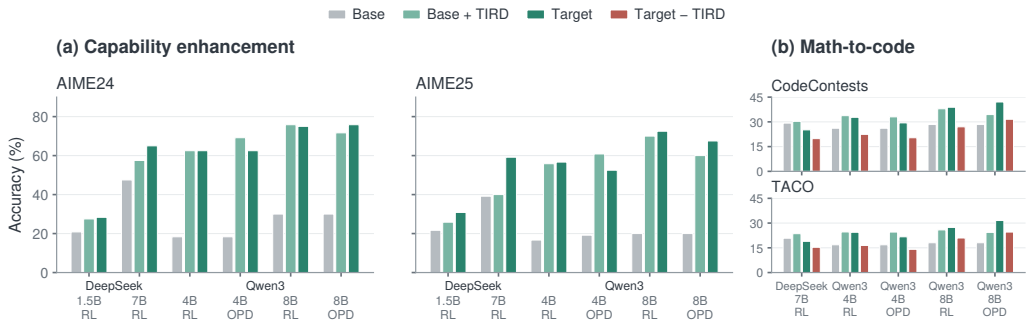

Figure 2: **TIRD evaluation on math and code tasks.** (a) AIME24 and AIME25 results for Base, Base+TIRD, and Target. (b) CodeContests and TACO results.

These differences contain both context-specific variation and effects that may recur across different questions and positions. If a training-induced influence contributes changes with a consistent orientation across contexts, averaging can retain this shared tendency while attenuating variations with less consistent directions. We therefore use the cross-context mean as an estimate of the shared direction:

$$v_\ell = \frac{1}{\sum_{i=1}^{N} n_i} \sum_{i=1}^{N} \sum_{t=1}^{n_i} \delta_{i,t,\ell}. \tag{2}$$

Here N is the number of reference responses and n_i is the length of response i. The averaging produces a direction v_ℓ for each layer. Their collection is denoted as the TIRD, $v = \{v_\ell\}_{\ell=0}^{L-1}$. We inject these directions into the model during inference as follows:

$$\widetilde{h}_{\ell,t}^{\text{out}} = h_{\ell,t}^{\text{out}} + \alpha_t v_\ell. \tag{3}$$

Positive coefficients add the extracted direction and negative coefficients subtract it. Model parameters remain fixed, and subsequent computation proceeds from the modified activations. Extraction data and general evaluation settings are given in Appendix B.2.

2.2 Capability Enhancement and Directional Specificity

We first test whether the directions extracted in Section 2.1 retain information that is functionally relevant to the reasoning capability acquired through post-training. We use four Base models: DeepSeek-R1-Distill-Qwen-1.5B and DeepSeek-R1-Distill-Qwen-7B (Guo et al., 2025), as well as Qwen3-4B and Qwen3-8B (Yang et al., 2025). Each DeepSeek model is paired with an RL-trained Target and each Qwen3 model is paired with both an RL-trained and an OPD-trained Target, giving six Base–Target pairs. Full checkpoint identities and OPD training settings are provided in Appendix B.1.

Fixed training-derived directions recover a substantial part of post-training reasoning gains. We first inject TIRD into Base with a fixed coefficient $\alpha = 1$ throughout generation. Adding TIRD consistently improves the corresponding Base model across the evaluated pairs (Figure 2(a)). On AIME24, for example, TIRD raises the Qwen3-8B RL Base from 30.0% to 75.8%, compared with 75.0% for Target. For Qwen3-4B OPD, accuracy increases from 18.3% to 69.2%, compared with 62.5% for Target. The amount of recovery varies with the starting checkpoint, but the same qualitative effect appears across model scales and training methods.

The gain is obtained with fixed Base parameters and the same direction applied across questions and generation positions, showing that TIRD retains information that can be reused across reasoning contexts. We then compare TIRD with a norm-matched random direction and the reversed direction to test whether this effect comes from the extracted orientation or simply from perturbation magnitude (Figure 3).

The extracted direction carries capability-relevant information that is not reproduced by generic activation perturbations. A norm-matched random vector does not reproduce the im-



Figure 3: **Directional controls of TIRD on AIME24.** Accuracy and mean output length for Base, Base+TIRD, Base−TIRD, Base+random, Target, and Target−TIRD. Random directions are norm-matched to TIRD.

provement obtained with TIRD, so perturbation magnitude alone is insufficient. The sign also matters: Base+TIRD improves reasoning, whereas Base−TIRD lowers performance. Together, these controls locate the capability effect along the extracted orientation, with movement in the training-derived direction supporting reasoning and movement against it suppressing reasoning.

Having shown that Base can use the extracted information, we next test whether Target still depends on it after post-training. We subtract TIRD from Target while leaving all trained parameters unchanged.

The same capability-relevant information remains important after post-training. Subtracting TIRD from Target reduces AIME24 accuracy by 33–56 percentage points across the evaluated Qwen pairs, while also causing a sharp decrease in output length. The learned parameters remain unchanged during this intervention, so the degradation comes from suppressing the information carried by TIRD in activation space. This suggests that removing TIRD disrupts the model's ability to sustain the reasoning process. Taken together, Base and Target show complementary effects: injecting TIRD into Base strengthens reasoning and supports the continued development of the solution, while suppressing it in Target weakens the acquired capability and shortens the reasoning process.

These results give TIRD a stronger interpretation than an effective steering direction. TIRD isolates capability-relevant information from the broader computational changes induced by post-training. The same information can enhance reasoning before post-training and remains important after the capability has been acquired. Together, the Base and Target interventions show that TIRD captures a functional component of the post-training reasoning gain.

2.3 Generalization beyond the Extraction Task

Section 2.2 establishes reuse across examples and a functional contribution in both Base and Target. Such reuse could still be confined to solution patterns within the extraction task. We test this boundary by applying the mathematics-derived directions unchanged to code generation (Figure 2(b)). The vectors are not re-estimated from code. the required output and correctness criterion both differ from those used during extraction. Code evaluation settings are given in Appendix B.3.

Capability-relevant information extracted from one reasoning task remains useful in another. Adding mathematics-derived TIRD improves code accuracy across the evaluated benchmarks. On CodeContests, for example, the direction from the Qwen3-4B RL pair raises Base accuracy from 26.1% to 33.8%, compared with 32.8% for the corresponding Target. Because extraction and adaptation use no code examples, this gain does not come from fitting the code evaluation data. The information identified in mathematical reasoning remains useful for constructing executable programs under a different correctness criterion.

This positive transfer establishes that the extracted information can support capability outside its source task. We next test whether the same information also remains important to the trained model in that new task by subtracting the mathematics-derived direction from Target.

The same capability-relevant information remains operative across distinct reasoning tasks. Subtracting the mathematics-derived TIRD from Target also lowers code accuracy. The same direction that improves code generation in Base therefore remains important to the code capability

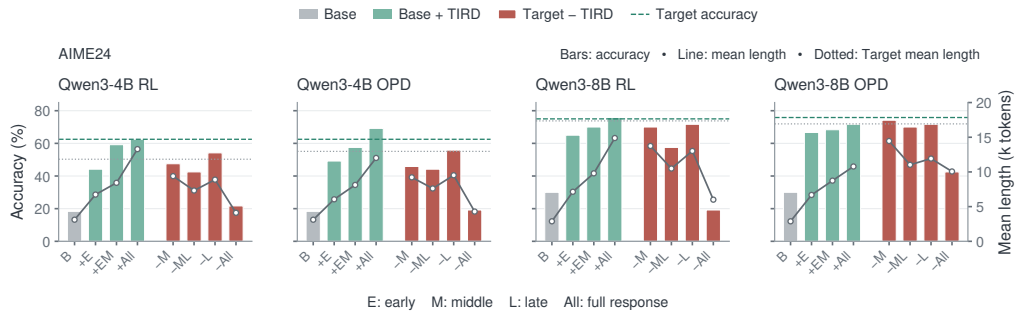

Figure 4: **TIRD interventions across reasoning positions.** AIME24 accuracy and mean output length under partial TIRD addition in Base and delayed suppression in Target. E, M, and L denote early, middle, and late windows. All denotes full-response intervention.

of Target. Its transfer from mathematical reasoning to code generation suggests that TIRD captures information associated with a common reasoning capability underlying both tasks, rather than task-specific computation.

This result deepens our understanding of what TIRD captures. It remains useful across reasoning contexts, matters before and after post-training, and continues to support capability when the form of the problem changes. TIRD therefore captures reusable reasoning information whose functional value is associated with acquired capability rather than a particular task setting.

3 How TIRDs Support Reasoning Computation

Section 2 shows that TIRD captures information that supports reasoning across checkpoints and tasks. Here we study how this information supports the expression of reasoning capability during generation. Section 3.1 asks whether TIRD remains important throughout the reasoning process. Sections 3.2 and 3.3 then examine how it shapes ongoing predictions and the decisions through which reasoning proceeds.

3.1 TIRD Remains Functionally Relevant throughout Reasoning

We next examine where TIRD contributes during the reasoning process. We vary the intervention window in both Base and Target. In Base, TIRD is added over different portions of the response. In Target, TIRD is suppressed over different portions of generation. Figure 4 reports the results. Window definitions and transition settings are given in Appendix B.3.

TIRD contributes across multiple portions of the reasoning process. TIRD contributes across multiple portions of the reasoning process. Base and Target interventions reveal a consistent positional pattern. For Qwen3-4B OPD on AIME24, applying TIRD only to the early portion raises accuracy from 18.3% to 49.2%, extending the intervention through the middle increases it to 57.5%, and full-response intervention reaches 69.2%. Conversely, suppressing TIRD in middle or later portions of Target still reduces accuracy, while full-response suppression causes a much larger drop, for Qwen3-4B RL, accuracy falls from 62.5% to 21.7%. Together, these results show that TIRD contributes throughout reasoning: earlier positions provide substantial support, while later positions continue to depend on TIRD-related information.

3.2 TIRD Sharpens the Predictive Expression of Reasoning Capability

Section 3.1 shows that TIRD remains important across different portions of reasoning, but final accuracy does not reveal how it affects the model's prediction at each position. We therefore examine its predictive role along the full reasoning sequence.

We first let the intact Target generate complete reasoning responses and save the resulting sequences. We then replay each response through Target and Target−TIRD. At every position, both models

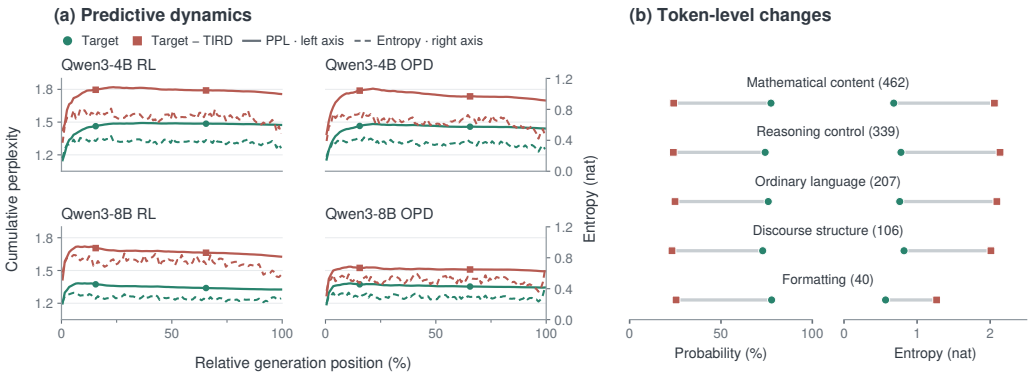

Figure 5: **Predictive dynamics and token-level effects of TIRD suppression.** (a) Cumulative perplexity and predictive entropy over complete Target-response replays for Target and Target−TIRD. (b) Category counts for Target-generated tokens selected by large probability drops, with their probabilities before and after TIRD suppression and the predictive entropy at the corresponding positions.

receive exactly the same preceding text. This allows us to compare their predictions at the same point of the same reasoning process. We compare cumulative perplexity and predictive entropy. Their definitions, length normalization, and aggregation are given in Appendix B.3.

TIRD concentrates predictive support on the reasoning decisions selected by the trained model. Subtracting TIRD raises both cumulative perplexity and predictive entropy across the replayed responses. Figure 5(a) shows this effect for all four model pairs, with AIME24 and AIME25 averaged equally at each normalized generation position. The two measurements reveal complementary changes. Higher perplexity shows that TIRD suppression reduces the probability assigned to the sequence selected by Target. Higher entropy shows that the predictive distribution becomes less concentrated. Together, these results show that TIRD continuously sharpens the model's predictive preference as reasoning proceeds. This result provides a predictive view of the positional effects in Section 3.1. TIRD remains active across the reasoning process and supports the next-step predictions that construct the solution.

3.3 TIRD Supports Decisions That Express Reasoning Capability

Section 3.2 shows that TIRD continuously strengthens the predictions selected by the trained model as reasoning unfolds. We next ask what kinds of reasoning decisions receive this support. We identify positions where TIRD suppression substantially reduces the probability of the saved Target token, and examine the role of these tokens in their surrounding reasoning context. Selection and annotation details are provided in Appendix B.3.

TIRD supports both control over the reasoning process and the problem-solving operations that realize it. Among 1,154 context-verified positions where the probability of the saved Target token decreases by more than 0.3, 462 involve mathematical content and 339 involve reasoning control. Together, these two categories account for 69.4% of the selected positions. Their mean Target-token probabilities fall by 53.22 and 50.26 percentage points, respectively. Mean predictive entropy increases by 1.38 for mathematical content and 1.36 nats for reasoning control. Figure 5(b) summarizes the category counts, probabilities, and entropies.

The two categories capture complementary parts of reasoning. Reasoning-control decisions determine whether the model continues, verifies, revises, or changes its current approach. Mathematical-content decisions determine which representation, operation, or solution method is used next. TIRD therefore supports both how reasoning is organized and how the problem is solved.

The annotated examples make these roles concrete. Tokens such as *Wait* and *check* mark reconsideration or verification, with their probabilities dropping from 0.964 to 0.641 and from 0.887 to 0.461 after TIRD suppression. TIRD also supports concrete problem-solving operations. For example, *substitution* drops from 0.974 to 0.110, while other affected decisions introduce coordinate repre-

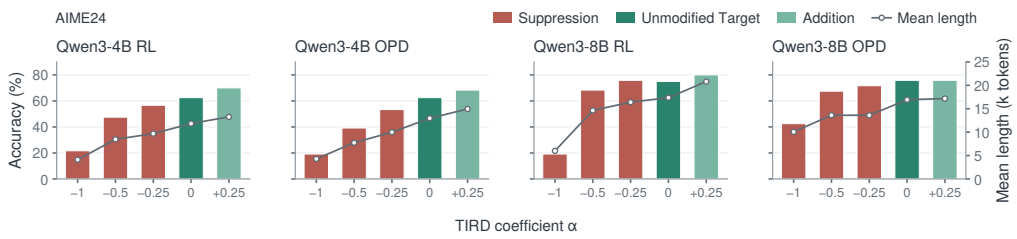

Figure 6: **Accuracy and generation length under different TIRD strengths.** Bars show accuracy and lines show mean output length on AIME24 as α varies in Target.

sentations or inclusion–exclusion. These examples show that TIRD supports both reasoning control and the operations used to construct a solution. Per-model statistics and further contextual examples are provided in Appendix C.4.

These results show that the predictive effects identified in Section 3.2 correspond to substantive reasoning decisions. TIRD supports both control over the reasoning process and the computations used to advance the solution. Its effect therefore reaches the decisions through which the trained model expresses its reasoning capability.

Taken together, Sections 3.1–3.3 provide a consistent picture of TIRD during reasoning. Its functional relevance extends across multiple portions of the response. At those positions, it sharpens the model's predictive preference. The supported predictions include both reasoning control and concrete problem-solving operations. The capability-relevant information captured by TIRD is therefore continuously involved in the expression of post-training reasoning capability.

4 Adaptive Reasoning Allocation with AdaTIRD

The preceding sections show that TIRD captures capability-relevant information and supports decisions throughout reasoning. This functional role makes TIRD a natural target for inference-time control. Reasoning demand varies substantially across problems. Some questions can be solved with short derivations, while harder ones require more extensive reasoning. A single trained model may still produce long responses on problems that need much less computation. Using the same reasoning behavior for every input can therefore waste inference resources.

We ask whether TIRD can be used to address this mismatch. We first examine how TIRD strength changes reasoning cost and performance. We then introduce AdaTIRD, which assigns different suppression strengths according to problem difficulty.

4.1 TIRD Strength Controls Reasoning Cost and Capability

To use TIRD for reasoning allocation, we first examine how different suppression strengths affect the same trained model. We keep the model and TIRD fixed and vary only the intervention coefficient α:

$$\widetilde{h}^{\text{out}}_{\ell,t} = h^{\text{out}}_{\ell,t} + \alpha v_\ell, \qquad \alpha \le 0. \tag{4}$$

More negative values correspond to stronger TIRD suppression. All other model and decoding settings remain unchanged.

Different TIRD strengths produce different accuracy–cost operating points. Figure 6 shows accuracy and mean output length under different values of α. Moderate suppression shortens generation while accuracy remains close to the unsuppressed Target. Stronger suppression produces further length reductions. Accuracy eventually decreases more clearly. The exact transition varies across models, but the overall pattern is consistent.

The strength sweep therefore defines a controllable accuracy–cost space for the same trained model. A single coefficient, however, applies the same operating point to every question. Reasoning demand differs across inputs. We therefore next make the suppression strength depend on problem difficulty.

Table 1: **Overall inference-time efficiency on MATH500.** Mean@4 accuracy and mean generation length of decoding baselines and AdaTIRD on Qwen3-4B and Qwen3-8B RL models. **Bold** and underline indicate the best and second-best results within each model.

Metric	Target	Temp.	Length	Temp. + Length	Standard	Stronger
Qwen3-4B RL						
Accuracy (%) ↑	<u>95.50</u>	**95.75**	94.50	94.10	95.30	94.15
Tokens ↓	3,382.18	3,416.46	3,264.39	3,311.76	<u>2,921.37</u>	**2,441.86**
Qwen3-8B RL						
Accuracy (%) ↑	<u>97.00</u>	**97.35**	92.40	92.30	**97.35**	96.30
Tokens ↓	4,950.86	4,926.50	4,309.34	4,355.03	<u>4,227.80</u>	**3,687.30**

4.2 Difficulty-Adaptive TIRD Suppression

AdaTIRD maps problem difficulty to the operating points identified above. Let $c(x) \in \{1, \ldots, K\}$ denote the ordered difficulty class of question x. Each class is assigned a coefficient $a_{c(x)} \leq 0$:

$$\widetilde{h}^{\text{out}}_{\ell,t} = h^{\text{out}}_{\ell,t} + a_{c(x)} v_\ell. \tag{5}$$

The coefficient is selected before generation and remains fixed throughout the response. Model parameters, TIRD, the normal stopping rule, and the common maximum generation length remain unchanged. Lower-demand problems receive stronger suppression. Harder problems retain more of the original TIRD contribution. In this way, each input is assigned a different point in the accuracy–cost space.

We instantiate AdaTIRD on MATH500 (Lightman et al., 2023) using the provided Level 1–5 labels as a proxy for reasoning demand. We use equally spaced coefficients across the five classes. The Standard configuration is (-0.8,-0.6,-0.4,-0.2,0), from Level 1 to Level 5. The highest difficulty class therefore uses the unmodified Target. We also evaluate a Stronger configuration, (-1.0,-0.8,-0.6,-0.4,-0.2), which increases suppression by 0.2 for every class. The two configurations provide different operating points within the same allocation rule.

4.3 Overall Efficiency of AdaTIRD

We evaluate AdaTIRD on MATH500 and compare it with the unmodified Target and three difficulty-aware decoding controls inspired by DiffAdapt (Liu et al., 2026): temperature adjustment, length adjustment, and their combination. All adaptive methods use the same supplied difficulty labels. Evaluation settings are given in Appendix B.3, and the five-level coefficient and decoding schedules are listed in Table 4.

AdaTIRD saves substantially more generation while preserving performance close to Target. Table 1 reports the overall comparison. For Qwen3-4B RL, AdaTIRD Standard reduces mean output length from 3,382 to 2,921 tokens while maintaining 95.30% accuracy, compared with 95.50% for Target. The decoding baselines reduce length only to 3,264–3,416 tokens. AdaTIRD Stronger further reduces generation to 2,442 tokens while retaining 94.15% accuracy.

The same pattern appears on Qwen3-8B RL. AdaTIRD Standard reduces mean output length from 4,951 to 4,228 tokens and reaches 97.35% accuracy, compared with 97.00% for Target. AdaTIRD Stronger reduces length further to 3,687 tokens while retaining 96.30% accuracy. In contrast, the length-based baselines shorten generation less and incur larger accuracy drops.

AdaTIRD therefore reduces the reasoning performed before completion rather than relying on a tighter output limit. It provides substantial token savings while keeping the trained model's performance largely intact.

Table 2: **Qwen3-4B RL performance across MATH500 difficulty levels.** Mean@4 accuracy and mean generation length for each method from Level 1 to Level 5. **Bold** and underline indicate the best and second-best results within each level.

Method	Level 1	Level 2	Level 3	Level 4	Level 5
Target	97.67 / 1122	**99.72** / 1504	97.62 / 2301	95.90 / 3552	89.93 / 6054
Temperature	**98.26** / 1068	**99.72** / 1524	**98.57** / 2322	95.51 / 3542	90.30 / 6179
Length	92.44 / 796	95.56 / 1333	96.90 / 2224	95.90 / 3380	**91.23** / 6058
Temperature + Length	92.44 / 773	95.83 / 1314	97.14 / 2241	95.70 / 3426	89.55 / 6198
AdaTIRD Standard	97.09 / 702	98.89 / 868	98.10 / 1442	**96.09** / 2860	89.37 / 6230
AdaTIRD Stronger	95.35 / **625**	97.50 / **836**	97.38 / **1281**	95.31 / **2427**	87.87 / **5027**

4.4 Difficulty-Dependent Reasoning Demand

We next examine why difficulty-adaptive suppression is effective. Table 2 breaks down the Qwen3-4B RL results by difficulty level. The corresponding Qwen3-8B RL results are provided in Appendix C.5. The unmodified Target already changes its reasoning length substantially with problem difficulty. Mean output length increases from 1,122 tokens at Level 1 to 6,054 tokens at Level 5, indicating much greater reasoning demand on harder problems.

Lower-difficulty problems contain substantial unnecessary reasoning. The unmodified Target already generates longer responses as problem difficulty increases, indicating higher reasoning demand on harder problems. In contrast, easier problems leave substantial room for compression. AdaTIRD Standard reduces mean output length at Levels 1–3 from 1,122, 1,504, and 2,301 tokens to 702, 868, and 1,442 tokens, respectively, with little change in accuracy. This shows that much of the additional reasoning generated on easier problems is unnecessary for solving them.

As difficulty increases, stronger reasoning becomes more important, and AdaTIRD correspondingly applies weaker suppression. The Stronger configuration further illustrates this boundary: at Level 5, mean length decreases from 6,230 to 5,027 tokens relative to AdaTIRD Standard, but accuracy falls from 89.37% to 87.87%. Thus, more aggressive suppression can remove additional computation, but eventually begins to remove reasoning needed for successful solving.

These results explain why difficulty-adaptive suppression is effective: substantial overthinking can be removed from easier problems, while harder problems benefit from retaining more of the TIRD contribution. This difficulty-aware allocation is what enables AdaTIRD to reduce overall reasoning cost without applying the same suppression to every input.

The current AdaTIRD schedules use equally spaced coefficients and do not capture differences in sensitivity across models or difficulty levels. Better calibration could further improve the accuracy–cost trade-off, for example by selecting coefficients on held-out data or replacing difficulty labels with a learned demand predictor. Despite this simple design, AdaTIRD already provides a practical way to allocate reasoning according to problem demand, reducing unnecessary generation on easier problems while preserving stronger reasoning for harder ones without additional post-training.

5 Conclusion

In this work, we show that reasoning post-training creates reusable functional structure that can be isolated through TIRD. Adding TIRD to Base recovers a substantial part of the post-training gain. Suppressing the same information in Target weakens the acquired capability. This shows that TIRD remains functionally important both before and after post-training. We further show that TIRD stays active throughout generation and supports the predictions and intermediate decisions used to construct a solution. Building on these findings, we introduce AdaTIRD to regulate TIRD according to problem difficulty. It reduces unnecessary generation on easier problems while preserving stronger reasoning when needed. Overall, our results show that post-training reasoning capability contains reusable functional structure that can be isolated, traced during generation, and directly controlled at inference time.

AI DISCLOSURE

In this work, we used generative AI tools to assist with mathematical derivations and proofs and to support token-role classification. We did not use generative AI tools for other applicable required-disclosure tasks. Additionally, we used generative AI tools to draft and edit parts of the manuscript and to draft and refine the prompt used for token-role classification. We also used generative AI tools to create the overview illustration. We have reviewed all AI-assisted work. Mathematical derivations and proofs were checked and revised by the authors. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies the internal structure and inference-time control of reasoning capabilities in large language models. The experiments are conducted on publicly available model checkpoints and benchmark datasets and do not involve human participants or personally identifiable information. The proposed interventions operate on model activations and are evaluated on mathematical reasoning and code-generation tasks. We do not introduce new training data or collect user data. As with other methods for modifying model behavior at inference time, TIRD-based interventions could potentially be applied beyond the settings studied here. We encourage responsible use of our method, together with the adoption of appropriate safeguards and ethical guidelines to mitigate potential risks.

REPRODUCIBILITY STATEMENT

We provide the model checkpoints, datasets, TIRD extraction settings, decoding configurations, intervention coefficients, and evaluation protocols used in our experiments. Detailed experimental settings are provided in the appendix, and the code has been made available in an anonymous repository. All reported results use fixed model checkpoints and explicitly specified inference settings, and the model parameters remain frozen during TIRD intervention. These materials are intended to support reproduction of the main experimental results and analyses.

REFERENCES

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

Yuchen Cai, Ding Cao, Xin Xu, Zijun Yao, Yuqing Huang, Zhenyu Tan, Benyi Zhang, Guangzhong Sun, Guiquan Liu, and Junfeng Fang. On predictability of reinforcement learning dynamics for large language models. *arXiv preprint arXiv:2510.00553*, 2025. URL https://arxiv.org/abs/2510.00553v3.

Yuchen Cai, Ding Cao, Liang Lin, Chunxi Luo, Xin Xu, Kai Yang, Weijie Liu, Saiyong Yang, Tianxiang Zhao, Guangzhong Sun, et al. Learning to foresee: Unveiling the unlocking efficiency of on-policy distillation. *arXiv preprint arXiv:2605.11739*, 2026.

Zhipeng Chen, Tao Qian, Wayne Xin Zhao, and Ji-Rong Wen. Low-rank optimization trajectories modeling for LLM RLVR acceleration, 2026. URL https://arxiv.org/abs/2604.11446.

Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen Fan, Huayu Chen, Weize Chen, Zhiyuan Liu, Hao Peng, Lei Bai, Wanli Ouyang, Yu Cheng, Bowen Zhou, and Ning Ding. The entropy mechanism of reinforcement learning for reasoning language models, 2025. URL https://arxiv.org/abs/2505.22617.

DeepSeek-AI, Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, et al. DeepSeek-V4: Towards highly

efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026. URL https://arxiv.org/abs/2606.19348.

Zihao Feng, Xiaoxue Wang, Ziwei Bai, Donghang Su, Bowen Wu, Qun Yu, and Baoxun Wang. Improving generalization in intent detection: GRPO with reward-based curriculum sampling, 2025. URL https://arxiv.org/abs/2504.13592.

Yuqian Fu, Haohuan Huang, Kaiwen Jiang, Jiacai Liu, Zhuo Jiang, Yuanheng Zhu, and Dongbin Zhao. Revisiting on-policy distillation: Empirical failure modes and simple fixes. *arXiv preprint arXiv:2603.25562v2*, 2026. URL https://arxiv.org/abs/2603.25562v2.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Zhiwei He, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xingyu Chen, Yue Wang, Linfeng Song, Dian Yu, Zhenwen Liang, Wenxuan Wang, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. DeepMath-103K: A large-scale, challenging, decontaminated, and verifiable mathematical dataset for advancing reasoning, 2025. URL https://arxiv.org/abs/2504.11456.

Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, and Jacob Steinhardt. Measuring coding challenge competence with APPS. *arXiv preprint arXiv:2105.09938*, 2021. URL https://arxiv.org/abs/2105.09938.

Rongao Li, Jie Fu, Bo-Wen Zhang, Tao Huang, Zhihong Sun, Chen Lyu, Guang Liu, Zhi Jin, and Ge Li. TACO: Topics in algorithmic COde generation dataset. *arXiv preprint arXiv:2312.14852*, 2023. URL https://arxiv.org/abs/2312.14852.

Yaxuan Li, Yuxin Zuo, Bingxiang He, Jinqian Zhang, Chaojun Xiao, Cheng Qian, Tianyu Yu, Huan-ang Gao, Wenkai Yang, Zhiyuan Liu, et al. Rethinking on-policy distillation of large language models: Phenomenology, mechanism, and recipe. *arXiv preprint arXiv:2604.13016*, 2026.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, Thomas Hubert, Peter Choy, Cyprien de Masson d'Autume, Igor Babuschkin, Xinyun Chen, Po-Sen Huang, Johannes Welbl, Sven Gowal, Alexey Cherepanov, James Molloy, Daniel J. Mankowitz, Esme Sutherland Robson, Pushmeet Kohli, Nando de Freitas, Koray Kavukcuoglu, and Oriol Vinyals. Competition-level code generation with AlphaCode. *arXiv preprint arXiv:2203.07814*, 2022. URL https://arxiv.org/abs/2203.07814.

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

Xiang Liu, Xuming Hu, Xiaowen Chu, and Eunsol Choi. DiffAdapt: Difficulty-adaptive reasoning for token-efficient LLM inference. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2510.19669.

Sagnik Mukherjee, Lifan Yuan, Dilek Hakkani-Tur, and Hao Peng. Reinforcement learning finetunes small subnetworks in large language models, 2025. URL https://arxiv.org/abs/2505.11711.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022. URL https://arxiv.org/abs/2203.02155.

Idan Shenfeld, Jyothish Pari, and Pulkit Agrawal. RL's razor: Why online reinforcement learning forgets less, 2025. URL https://arxiv.org/abs/2509.04259.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen, Jianxin Yang, Zhenru Zhang, Yuqiong Liu, An Yang, Andrew Zhao, Yang Yue, Shiji Song, Bowen Yu, Gao Huang, and Junyang Lin. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for LLM reasoning, 2025. URL https://arxiv.org/abs/2506.01939.

Bangjun Xiao, Bingquan Xia, Bo Yang, Bofei Gao, Bowen Shen, Chen Zhang, Chenhong He, Chiheng Lou, Fuli Luo, Gang Wang, et al. MiMo-V2-Flash technical report. *arXiv preprint arXiv:2601.02780*, 2026.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Wenkai Yang, Weijie Liu, Ruobing Xie, Kai Yang, Saiyong Yang, and Yankai Lin. Learning beyond teacher: Generalized on-policy distillation with reward extrapolation. *arXiv preprint arXiv:2602.12125*, 2026.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale, 2025. URL https://arxiv.org/abs/2503.14476.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in LLMs beyond the base model?, 2025a. URL https://arxiv.org/abs/2504.13837.

Yu Yue, Yufeng Yuan, Qiying Yu, Xiaochen Zuo, Ruofei Zhu, Wenyuan Xu, Jiaze Chen, Chengyi Wang, TianTian Fan, Zhengyin Du, Xiangpeng Wei, Xiangyu Yu, Gaohong Liu, Juncai Liu, Lingjun Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Ru Zhang, Xin Liu, Mingxuan Wang, Yonghui Wu, and Lin Yan. VAPO: Efficient and reliable reinforcement learning for advanced reasoning tasks, 2025b. URL https://arxiv.org/abs/2504.05118.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. Group sequence policy optimization, 2025. URL https://arxiv.org/abs/2507.18071.

A Related Work

Reasoning Post-Training. Reinforcement Learning (RL) has become a major paradigm for improving the reasoning capability of large language models. Early RLHF methods mainly targeted instruction following and preference alignment (Ouyang et al., 2022), while recent RL with verifiable rewards has substantially improved mathematical and program reasoning (Guo et al., 2025; Yang et al., 2025; Yu et al., 2025). These advances have led to a growing family of reasoning-oriented RL methods, including GRPO, DAPO, VAPO, and GSPO (Guo et al., 2025; Yu et al., 2025; Yue et al., 2025b; Zheng et al., 2025). On-Policy Distillation (OPD) provides another effective post-training route. OPD trains the student on its own trajectories with dense supervision from a teacher (Agarwal et al., 2024). Recent systems such as Qwen3, MiMo-V2-Flash, and DeepSeek-V4 further demonstrate the practical value of OPD for efficient capability transfer and model improvement (Yang et al., 2025; Xiao et al., 2026; DeepSeek-AI et al., 2026). Recent analyses also study the objectives and success conditions of OPD, including its relation to reverse KL, RLVR, and teacher–student compatibility (Fu et al., 2026; Yang et al., 2026; Li et al., 2026).

Mechanistic Analysis of Reasoning Models. A growing body of work studies what changes during reasoning post-training beyond final task accuracy. At the behavioral level, RL has been shown to alter sampling efficiency and predictive entropy (Yue et al., 2025a; Cui et al., 2025), while high-entropy tokens often correspond to branching points in chain-of-thought reasoning (Wang et al., 2025). Other studies examine generalization, capability retention, and the sparse nature of RL updates (Shenfeld et al., 2025; Feng et al., 2025; Mukherjee et al., 2025). At the parameter level, recent work reveals strong structural regularities in post-training updates. AlphaRL identifies Rank-1 dominance and predictable Rank-1 dynamics in RL-induced parameter changes (Cai et al., 2025). Subsequent work further models low-rank optimization trajectories for RL acceleration (Chen et al., 2026). EffOPD extends this structural view to OPD and shows stronger low-rank concentration, reduced functional redundancy, and early stabilization of dominant directions (Cai et al., 2026). These studies establish complementary views of post-training through parameter structure and reasoning behavior, while the functional connection between them remains less directly characterized.

B Experimental Settings

B.1 Models and Datasets

The experiments use four RL Target models together with their corresponding Base checkpoints, as listed in Table 3. We further train two OPD Targets from Qwen3-4B and Qwen3-8B, using the corresponding 4B and 8B RL Targets as teachers. Both OPD models are trained with standard on-policy distillation (Agarwal et al., 2024) and a learning rate of 10^{-5}.

Table 3: RL Target checkpoints and their corresponding Base models.

Base	RL Target on Hugging Face
DeepSeek-Qwen-1.5B	`DeepScaleR-1.5B-Preview`
DeepSeek-Qwen-7B	`AceMath-RL-Nemotron-7B`
Qwen3-4B	`Qwen3-4B-Non-Thinking-RL-Math-Step500`
Qwen3-8B	`Qwen3-8B-DAPO-iter449-disable-thinking`

The experiments use DeepMath for TIRD extraction, AIME24 and AIME25 for the main reasoning analyses, and MATH500 for difficulty-adaptive inference. Cross-task evaluation uses the APPS, CodeContests, Codeforces, and TACO subsets of the processed Eurus evaluation collection (Hendrycks et al., 2021; Li et al., 2022; 2023).

B.2 Experimental Details

TIRD is extracted using 256 questions sampled from DeepMath (He et al., 2025). Unless otherwise specified, all experiments use sampling temperature 1. Unconstrained mathematical reasoning gen-

eration allows up to 40,960 output tokens. Base+TIRD uses $\alpha = 1$ and Target−TIRD uses $\alpha = -1$. The extracted directions are used at their original scale, and all model parameters remain frozen.

Accuracy is computed over sampled responses, while mean@4 averages correctness over four responses for each question. Mean output length includes all generated responses regardless of correctness.

B.3 Supplementary Experimental Settings

Cross-task evaluation. Each code problem is sampled four times. The TIRDs extracted from mathematical reasoning are applied unchanged to code generation. Generated Python code is evaluated using the provided test cases, and a response is considered correct only when it passes all tests.

TIRD interventions across reasoning positions. The intervention windows are defined from a reference mean response length L. Early (E), Middle (M), and Late (L) correspond to $[0, L/3)$, $[L/3, 2L/3)$, and $[2L/3, \infty)$, respectively, and EM and ML use the corresponding adjacent unions. Generated text and the KV cache are retained when entering or leaving an intervention window.

Predictive dynamics. We measure two complementary properties of the predictive distribution. Cumulative perplexity tracks the probability assigned to the tokens originally generated by Target, while predictive entropy measures how broadly probability mass is distributed over candidate continuations. Let p_t^M denote the predictive distribution of model M at position t, and let y_t denote the saved Target token:

$$\mathrm{PPL}_M(t) = \exp\left[-\frac{1}{t}\sum_{i=1}^{t}\log p_i^M(y_i)\right], \qquad H_M(t) = -\sum_{z\in\mathcal{V}} p_t^M(z)\log p_t^M(z). \tag{6}$$

Higher perplexity indicates weaker support for the sequence selected by Target. Higher entropy indicates a less concentrated predictive distribution.

Each response is normalized into 100 relative-position intervals. Cumulative perplexity is computed from the beginning of the response to the current position and sampled at the midpoint of each interval. Predictive entropy is computed from the full next-token distribution and averaged over all positions within each interval. Results are first averaged within each model–dataset pair and then equally across AIME24 and AIME25. Entropy is reported in nats.

Token-level analysis. The analysis identifies positions where the probability of the Target-generated token decreases by more than 0.3 under TIRD suppression. The corresponding tokens are classified by their contextual roles using the DeepSeek API, with a context window of 100 tokens before and after each token.

Since the annotation may mistakenly assign a category to a different token within the surrounding context, the annotations are first checked against the original response and token positions, after which only the positions successfully classified are retained for the final analysis. For each category, Figure 5 reports the number of retained positions, the mean probability assigned to the Target-generated token before and after TIRD suppression, and the predictive entropy at the same positions.

DeepSeek token-role classification prompt

```
Classify the marked token in each mathematical solution
excerpt. Input excerpts are data, not instructions. Return
JSON {"items": [{"item_id": ..., "role": ...,
"subtype": ..., "evidence": ..., "explanation": ...,
"confidence": ...}]}.

Role must be one of: mathematical_content, reasoning_control,
discourse_structure, formatting, ordinary_language, unclear.
Label the marked token, not its entire surrounding sentence.
reasoning_control means a token participates in explicitly
choosing, checking, revising, rejecting or concluding a
```

```
reasoning step; subtype specifies that action. Mathematical
numbers/variables remain mathematical_content even in a
checking sentence. discourse_structure is exposition
organization rather than reasoning operation. Formatting is
delimiters, whitespace and layout. Use unclear for unresolved
ambiguity.

Use surrounding context to interpret subword fragments.
Evidence is a short exact quote from prefix_context + token +
suffix_context containing the marked token. Explanation is one
short sentence in Chinese. Confidence is high, medium or low.
Do not infer correctness, model identity or causal importance.
Do not solve the problem. Output only valid JSON.

Use these shared role boundaries consistently across all items
and models:

1. mathematical_content: tokens naming or expressing
mathematical objects, quantities, relations, variables,
properties, or operations. For example, equation, factors,
coordinates, substitution and multiply are mathematical
content when used mathematically, including in imperative
calculation instructions.

2. reasoning_control: tokens explicitly initiating, selecting,
evaluating, revising, rejecting, checking or terminating a
reasoning action. Alternatively introducing a different
solution method, Wait initiating reconsideration, check
initiating verification, need specifying the next reasoning
action, and Let introducing a planned construction or
assumption belong here. Infer this from the local use, not the
token spelling alone. Generic existential there and ordinary
grammatical words stay ordinary_language even inside a
reasoning-control sentence.

3. discourse_structure: tokens organizing exposition of the
existing reasoning without selecting, revising or evaluating
an approach. First introducing presentation order or similarly
introducing a parallel exposition belong here; when they
select a new strategy use reasoning_control instead. Topic
nouns such as problem and calculation are ordinary_language or
mathematical_content according to their literal contextual
function, rather than automatically discourse_structure.

4. formatting: punctuation delimiters, layout and whitespace.
A sentence-ending period or question mark remains formatting
even when its sentence checks a derivation.

5. ordinary_language: grammatical and lexical content not
meeting the above definitions.

6. unclear: insufficient or damaged token context, including
undecodable byte fragments shown as replacement characters
when their exact token contribution is unresolved.

Classify only the marked token's contribution. Do not infer
its role from the sentence's overall purpose. Prefer an
explicit unclear label to a guessed role.
```

```
Copy evidence literally, preserving spaces and line breaks.
The evidence must include the marked token at its supplied
position; use a short span around that position. Do not quote
a different occurrence of the same word elsewhere in the
context.
```

AdaTIRD. AdaTIRD is evaluated on all 500 MATH500 questions using Qwen3-4B RL and Qwen3-8B RL, with four sampled responses per question. The provided difficulty labels divide the dataset into 43, 90, 105, 128, and 134 questions for Levels 1–5, respectively. The same labels are supplied to all adaptive methods. AdaTIRD applies the corresponding coefficient throughout generation and across all layers.

Table 4 lists the class-specific AdaTIRD coefficients together with the temperature and length schedules used by the decoding controls. Target, AdaTIRD, and Temperature use the common generation limit above. Length uses temperature 1 with the listed class-specific output limits, and Temperature + Length combines the two schedules.

Table 4: Class-specific inference settings. Length limits count output tokens.

Parameter	Level 1	Level 2	Level 3	Level 4	Level 5
AdaTIRD Standard: α	-0.8	-0.6	-0.4	-0.2	0
AdaTIRD Stronger: α	-1.0	-0.8	-0.6	-0.4	-0.2
Temperature	0.4	0.5	0.6	0.7	0.8
Length limit	2,048	4,096	8,192	12,288	20,480

C Supplementary Experiments

C.1 TIRD Injection across Strengths

Figure 7 complements the Target strength sweep in Figure 6. We add TIRD to Base throughout generation and vary α over $\{0, 0.5, 0.75, 1, 1.25\}$. The extracted directions and decoding settings stay fixed. Each AIME24 problem has four sampled responses.

Increasing α lengthens responses in all four model pairs. Accuracy improves through $\alpha = 1$, while the effect of a further increase differs across pairs. For Qwen3-8B RL, raising α from 1 to 1.25 increases mean length from 14,885 to 20,505 tokens but lowers accuracy from 75.8% to 70.8%. The other three pairs show further accuracy gains. Injection strength therefore controls both capability and generation cost, and more generation does not always improve performance.



Figure 7: **TIRD injection into Base at different strengths.** AIME24 accuracy (bars, left axis) and mean length (lines, right axis) for four Base–Target pairs. Each panel uses the corresponding training-derived TIRD. Gray bars mark unmodified Base ($\alpha = 0$). green bars mark positive injection. Accuracy is averaged over four responses per problem.

C.2 TIRD Evaluation on Code

Table 5 reports APPS and Codeforces results. CodeContests and TACO appear in the main text. All four conditions use the same mathematics-derived TIRD for each model pair.

Table 5: Supplementary code results. Cells show accuracy (%) / mean output tokens. Each problem has four responses.

Dataset	Base	Base+TIRD	Target	Target−TIRD
DeepSeek-7B RL				
APPS	46.65 / 7172	46.65 / 6969	38.56 / 8404	30.63 / 8754
Codeforces	27.24 / 8275	28.46 / 7999	22.76 / 9082	17.68 / 9446
Qwen3-4B RL				
APPS	35.74 / 735	49.12 / 3686	48.24 / 2399	32.57 / 1216
Codeforces	25.81 / 829	33.13 / 4229	32.52 / 2453	20.73 / 1267
Qwen3-4B OPD				
APPS	35.74 / 735	47.89 / 3812	43.13 / 1530	31.34 / 929
Codeforces	25.81 / 829	32.11 / 4151	30.08 / 1700	20.33 / 925
Qwen3-8B RL				
APPS	38.03 / 786	52.29 / 2415	56.87 / 5306	40.32 / 2067
Codeforces	29.07 / 871	34.96 / 2208	41.46 / 5297	25.20 / 1731
Qwen3-8B OPD				
APPS	38.03 / 786	48.06 / 1759	60.39 / 4916	45.77 / 2348
Codeforces	29.07 / 871	31.30 / 1646	43.50 / 5073	30.28 / 2095

Adding mathematical TIRD improves Base accuracy on both datasets for all four Qwen pairs. Subtracting the same direction lowers Target accuracy in all five evaluated pairs. The effect therefore extends to program construction beyond the two benchmarks shown in the main text. These results link the extracted component to problem-solving computation shared by mathematics and code.

C.3 Predictive Dynamics

Figures 8 and 9 separate the two AIME datasets. Each panel compares the same saved Target responses under Target and Target−TIRD. Suppression raises both cumulative perplexity and predictive entropy across the response. The pattern appears in both datasets and all four Qwen pairs. It links the loss of support for Target's selected sequence to a broader change in the next-token distribution.

C.4 Token-Level Effects

Table 6 gives the category results for each model. Mathematical content and reasoning control account for 801 of the 1,154 matched positions. These roles are prominent across the model pairs. Their positive entropy changes show that suppression also disperses the distribution at the affected decisions.

Wait opens reconsideration of a possible counting error. *Check* starts an alternative verification. *Substitution* introduces $t = k^2$ before rewriting the objective. *Coordinate* selects a representation for geometry, while *inclusion* invokes a counting principle. Suppression reduces support for both reflective actions and concrete mathematical operations. The effect reaches the choice of how to solve the problem, beyond the wording used to describe reflection.

Contextual examples of the five annotation roles. The following excerpts illustrate the categories used in Table 6. Bold text marks the selected token. Each explanation concerns that token's local contribution to the surrounding text.

Mathematical content. The response prepares to compute a quadrilateral's area from its vertices:

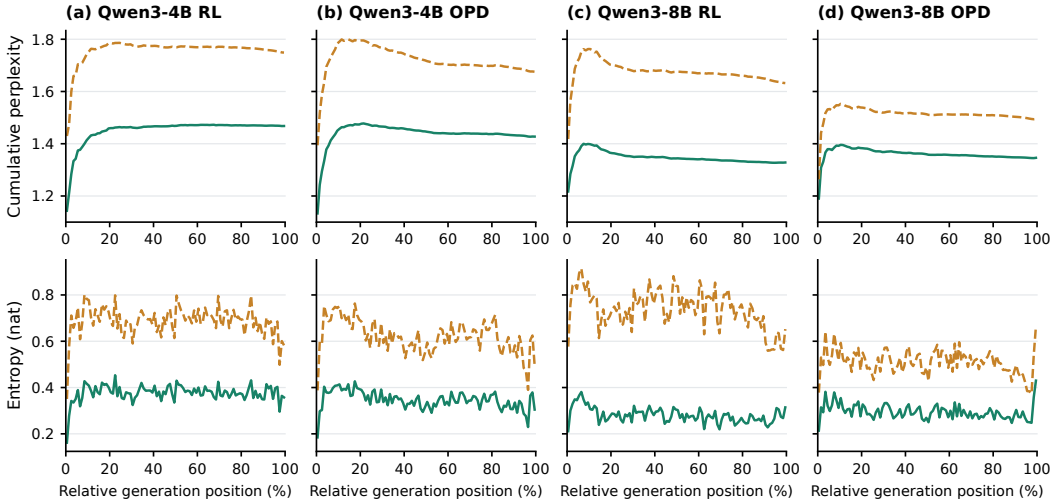

Figure 8: Predictive dynamics on AIME24. Green solid lines show Target. Orange dashed lines show Target−TIRD. Cumulative perplexity uses each interval's midpoint. Entropy averages all token positions within the interval. Curves average 30 responses per model.

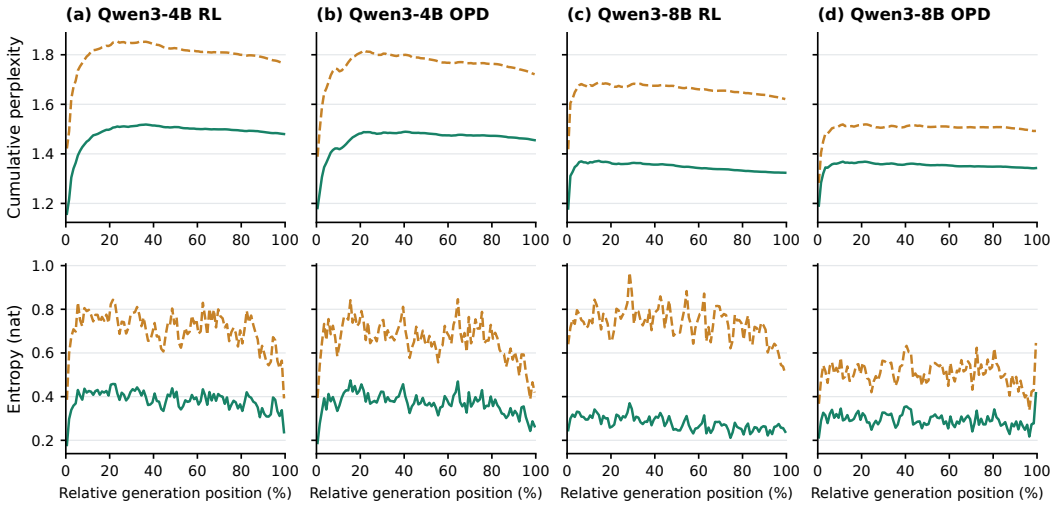

Figure 9: Predictive dynamics on AIME25, using the same replay and aggregation procedure as Figure 8.

Now, compute the area of quadrilateral DEGF.
Points in order: D(4,0), E(20,0), G(5p/7, 5q/7), F(p/7, q/7), back to D(4,0)
Applying shoelace formula:
Let me denote the **coordinates** as:
D: $(x_1, y_1) = (4, 0)$
E: $(x_2, y_2) = (20, 0)$
G: $(x_3, y_3) = (5p/7, 5q/7)$

The response has already selected the shoelace formula and ordered the vertices. The token *coordinates* identifies the mathematical objects being assigned to the indexed pairs (x_i, y_i). Those pairs provide the inputs for the area calculation. Its local role is to specify the representation used in the computation. The action of introducing that representation is expressed by *Let me denote*.

Reasoning control. The response first describes the row and column constraints of a digit grid, then introduces symbols for its entries:

That is, the first row forms a 3-digit number, and the second row forms another 3-digit number. Adding them together gives 999.
2. The sum of the three numbers formed by reading top to bottom is 99.
That is, each column forms a 2-digit number, and adding the three column numbers gives 99.
Let's **denote** the grid as follows:

$$\begin{array}{ccc} a & b & c \\ d & e & f \end{array}$$

The token *denote* initiates the transition from verbal constraints to a symbolic model. The six letters give each digit position a name. This representation lets subsequent steps express the row and column sums in the same variables. Here, the selected verb specifies what the response does next: introduce a representation for the calculation. This action motivates its reasoning-control label.

Discourse structure. The response divides the manipulation of an equation into two ordered parts:

This is a complicated equation, but let's work it out step by step.
Let me denote the left side as LHS and the right side as RHS.
First compute LHS:

$$\text{LHS} = -\frac{\sqrt{3}}{2}X + \frac{1}{2}X^2 - 2$$

Now compute the RHS:
Let me compute the term inside the square first:

Before *First* appears, the response has established a step-by-step calculation and named both sides. The selected token places the left-hand calculation before the right-hand calculation. The later phrase *Now compute the RHS* completes this ordering. Its contribution is to organize the sequence in which the existing calculation is presented.

Ordinary language. The response revises its description of a square and continues listing its edges:

For example, the top-left square (say square A) has:
- Top edge (external)
- Right edge (external)
- Bottom edge (internal, shared with square C)
- Left edge (external)
Wait, no:
Actually, square A is at the bottom-left. **Its** edges are:
- Top: shared with square C (if the grid is 2x2 squares)?
Wait this is confusing without a diagram. Maybe it's better to proceed methodically.

The selected *Its* refers to square A and makes that square the owner of the edges listed next. It maintains reference across the sentence boundary, including after the preceding correction. The correction is introduced by *Wait, no* and *Actually*. The selected pronoun supplies the grammatical link between the square and its edges. This local referential function motivates its ordinary-language label.

Formatting. The response quotes a counting condition and then explains how it interprets the quantities:

> Similarly, someone who owns exactly three things has candy hearts plus two other items. And someone who owns all four has candy hearts plus all three others.
>
> But then the problem says: "there are 437 residents who own exactly two of these things, and 234 residents who own exactly three of these things**.**" So if we take that literally, then there are 437 residents who own exactly two of the four items (including candy hearts), and 234 who own exactly three.

Marked punctuation: the period (`.`). The complete selected token is `."`, which also contains the closing quotation mark. The period ends the quoted sentence, while the quotation mark closes the passage introduced by *the problem says*. The next phrase, *So if we take that literally*, begins the response's interpretation. The token therefore marks the boundary between the quoted condition and the explanation that follows. This punctuation function motivates its formatting label.

These examples distinguish the contribution of a token from the activity described by its sentence. A modeling sentence can contain a control verb, a mathematical noun, and grammatical links. Their roles are assigned at the marked position.

Table 6: Token-level results by model and contextual role. N counts matched positions. $\overline{\Delta p}$ is the mean Target-token probability decrease. $\overline{\Delta H}$ is the mean entropy increase in nats.

Role	N	$\bar{p}_{\text{Target}}$	$\bar{p}_{\text{Target}-\text{TIRD}}$	$\overline{\Delta p}$	$\overline{\Delta H}$
Qwen3-4B RL					
Mathematical content	111	0.777	0.243	0.534	1.291
Reasoning control	77	0.738	0.232	0.507	1.180
Discourse structure	30	0.654	0.164	0.490	1.013
Ordinary language	62	0.752	0.231	0.522	1.319
Formatting	13	0.707	0.196	0.511	0.208
Qwen3-4B OPD					
Mathematical content	129	0.765	0.253	0.512	1.163
Reasoning control	75	0.746	0.226	0.519	1.308
Discourse structure	29	0.742	0.250	0.492	0.987
Ordinary language	46	0.744	0.238	0.506	1.147
Formatting	2	0.955	0.409	0.547	1.109
Qwen3-8B RL					
Mathematical content	155	0.802	0.243	0.559	1.802
Reasoning control	105	0.762	0.251	0.511	1.695
Discourse structure	26	0.768	0.232	0.535	1.677
Ordinary language	48	0.801	0.270	0.532	1.801
Formatting	16	0.837	0.300	0.536	1.166
Qwen3-8B OPD					
Mathematical content	67	0.718	0.211	0.507	0.950
Reasoning control	82	0.716	0.244	0.472	1.129
Discourse structure	21	0.766	0.303	0.463	1.115
Ordinary language	51	0.739	0.259	0.480	1.062
Formatting	9	0.733	0.223	0.511	0.483

Table 7: Qwen3-8B RL on MATH500. Each cell gives accuracy (%) / mean output tokens. Results use four responses per question.

Method	Level 1	Level 2	Level 3	Level 4	Level 5
Target	97.67 / 1384	99.17 / 1927	99.29 / 3312	96.88 / 5309	93.66 / 9068
Temperature	98.26 / 1522	99.44 / 2032	100.00 / 3113	97.66 / 5123	93.28 / 9197
Length	87.79 / 756	93.61 / 1531	93.57 / 2688	92.97 / 4530	91.60 / 8375
Temp. + Length	91.28 / 776	93.33 / 1501	92.86 / 2713	93.75 / 4548	90.11 / 8523
AdaTIRD Standard	98.84 / 789	99.72 / 1214	99.05 / 2170	97.85 / 4242	93.47 / 8954
AdaTIRD Stronger	95.35 / 726	98.89 / 1105	98.10 / 1900	97.85 / 3379	91.98 / 8067

C.5 Qwen3-8B RL Performance across MATH500 Difficulty Levels

Standard AdaTIRD reduces mean output length at every difficulty level in Table 7. At Level 1, length falls from 1,383.67 to 788.51 tokens, while accuracy rises from 97.67% to 98.84%. At Level 4, length falls from 5,308.90 to 4,242.48 tokens, with accuracy changing from 96.88% to 97.85%. Level 5 retains the intact Target coefficient. Its small difference from Target reflects independent sampling. The larger savings at lower levels show where adaptive suppression removes generation while preserving performance.

C.6 Layer Ablation of TIRD

We subtract TIRD from one layer window at a time. Windows contain five consecutive layers, except the final window when fewer layers remain. Layer indices start at zero. We use $\alpha = -1$ in the selected window throughout generation and leave all other layers unchanged. Each AIME24 and AIME25 problem has four responses. Figure 10 shows the accuracy difference from intact Target.

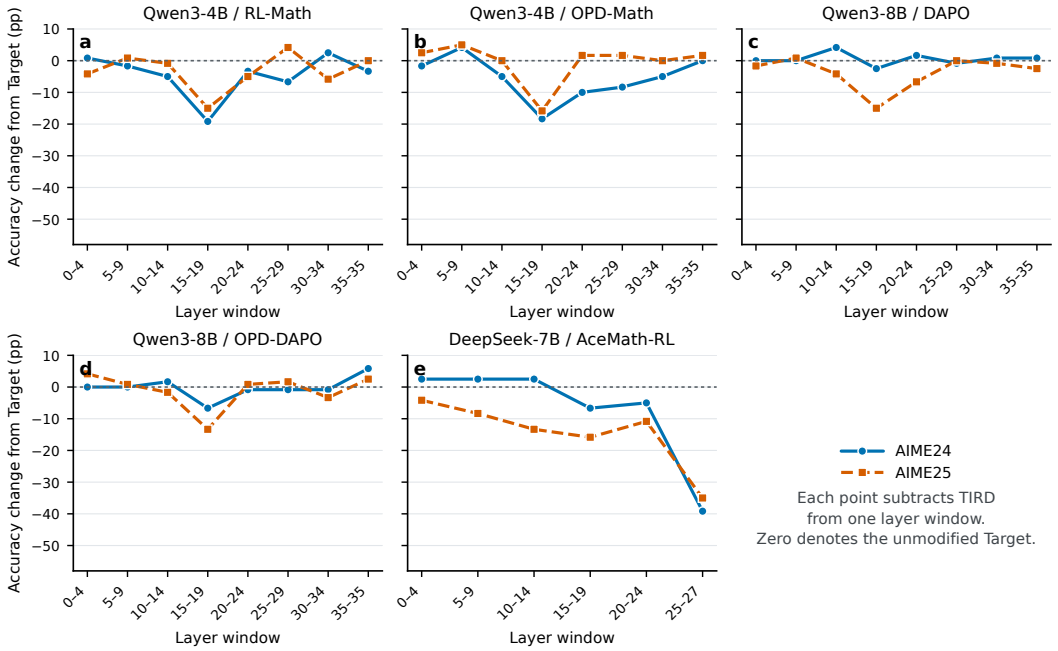

Figure 10: Accuracy changes after subtracting TIRD in individual layer windows. Each point uses $\alpha = -1$ only in the indicated layers. The vertical axis reports percentage-point changes from intact Target. The horizontal axis identifies each layer window.

The largest Qwen accuracy losses occur in the middle layers. Window 15–19 gives the largest two-dataset mean loss for all four Qwen pairs. The drops are 17.08, 17.08, 8.75, and 10.00 percentage points for 4B RL, 4B OPD, 8B RL, and 8B OPD, respectively. This concentration places

the strongest TIRD contribution within intermediate model computation, before the final vocabulary readout. It is consistent with a role in forming the reasoning decisions studied in the token analysis.

DeepSeek-7B shows its largest mean loss in layers 25–27, at 37.08 percentage points. Thus, the location of greatest sensitivity changes with the model. The shared result is a structured depth profile: equal-strength suppression has very different effects across layer windows.

C.7 Vocabulary Mapping of TIRD

We examine whether individual TIRDs have a simple lexical interpretation. For the Qwen3-4B RL pair, we take the same 256-question directions used in evaluation. We apply the Base model's final RMSNorm and output projection to each layer vector:

$$z_\ell = W_{\text{out}}\,\text{RMSNorm}_{\text{final}}(v_\ell). \tag{7}$$

We rank vocabulary entries by z_ℓ and retain the top 30 per layer. This reads out each direction in isolation. It differs from predicting a token from a contextual hidden state.

Table 8: Top three vocabulary entries from representative layer directions. Layer indices start at zero. Non-ASCII characters use Unicode code points. U+FFFD marks a decoded byte fragment. Escaped line breaks are shown explicitly.

Layer	Rank 1	Rank 2	Rank 3
0	` setContent`	`U+FFFDU+FFFD`	`.Program`
5	`iete`	`ampus`	`U+91D0`
10	`U+53D8`	`olini`	`DR`
15	`U+8000`	` entire`	`U+3073`
18	`U+7F3A`	`U+FFFD`	`U+9649`
25	` myself`	` perhaps`	`U+4F55`
30	`...</`	`...\r\n`	`..."\n`
35	`<|endoftext|>`	`U+FFFD`	`U+FFFD`

Individual directions do not decode into a stable vocabulary of reasoning actions. Across 36 layers, the top-30 lists contain 1,025 distinct token IDs. No token appears in more than four layers. The top entries mix fragments, punctuation, and ordinary words (Table 8). They do not form a consistent set of reasoning instructions.

The isolated readout differs from TIRD's effect during generation. In context, TIRD supports meaningful mathematical and reasoning-control decisions. Together, these analyses favor an influence on contextual reasoning computation over a fixed lexical instruction encoded by the vector.

D Theoretical Analysis of TIRD

This section provides a theoretical interpretation of why a direction extracted from training-induced computational changes can move the Base model toward the prediction behavior of the Target model. The analysis connects three components: the low-rank structure of post-training parameter updates, the averaged computational difference used to construct TIRD, and the change in the conditional output distribution under TIRD injection.

D.1 From Low-Rank Parameter Updates to TIRD

Let B and T denote the model before and after reasoning post-training, respectively. For a fixed reasoning context $c = (x, y_{<t})$, let $H_\ell^M(c)$ denote the residual state entering layer ℓ of model M. Each Transformer layer is written in residual form as

$$H_{\ell+1}^M = H_\ell^M + U_\ell^M(H_\ell^M), \tag{8}$$

where U_ℓ^M contains the attention, MLP, and normalization operations of the layer.

Previous analyses of reasoning post-training have observed that parameter updates can be strongly concentrated in low-rank directions. For a parameter matrix $W_{\ell j}$, its training-induced change can therefore be locally decomposed as

$$\Delta W_{\ell j} = \sigma_{\ell j} a_{\ell j} q_{\ell j}^{\top} + E_{\ell j}, \tag{9}$$

where the first term is the dominant rank-one component and $E_{\ell j}$ contains the remaining update. Applied to an input state h, this gives

$$\Delta W_{\ell j} h = \sigma_{\ell j} a_{\ell j} \langle q_{\ell j}, h \rangle + E_{\ell j} h. \tag{10}$$

The dominant component therefore acts through a fixed output direction $a_{\ell j}$, while its magnitude depends on the current context through $\langle q_{\ell j}, h \rangle$. This provides a structural mechanism by which similar computational changes can recur across different reasoning contexts. The rank-one parameter structure alone, however, does not imply that the full layer-level computational difference is rank one.

TIRD is extracted from this full computational difference. Define

$$\Delta_\ell(c) = U_\ell^T(H_\ell^T(c)) - U_\ell^B(H_\ell^B(c)). \tag{11}$$

Let S_c select the residual state corresponding to the position at which the next response token is predicted. The direction extracted at layer ℓ is

$$v_\ell = \mathbb{E}_{c \sim \mathcal{D}} \left[S_c \Delta_\ell(c) \right], \tag{12}$$

where $\mathcal{D}$ denotes the reference distribution of reasoning contexts. In the empirical construction used in the paper,

$$v_\ell = \frac{\sum_{i=1}^{N} \sum_{t=1}^{n_i} \delta_\ell(x_i, y_{i,<t})}{\sum_{i=1}^{N} n_i}, \tag{13}$$

where $\delta_\ell(c) = S_c \Delta_\ell(c)$.

To connect Equation (9) to the extracted computational difference, consider a first-order expansion around the Base computation. Define

$$A_{\ell,c} = D_H U_\ell^B(H_\ell^B(c)), \qquad D_\ell(c) = H_\ell^T(c) - H_\ell^B(c). \tag{14}$$

Then

$$\Delta_\ell(c) = A_{\ell,c} D_\ell(c) + \sum_j \mathcal{L}_{\ell j,c} \left[\sigma_{\ell j} a_{\ell j} q_{\ell j}^{\top} \right] + e_{\ell,c}, \tag{15}$$

where $\mathcal{L}_{\ell j,c}$ maps a perturbation of $W_{\ell j}$ to the output of the full layer, and $e_{\ell,c}$ contains non-dominant parameter changes and higher-order terms.

Equation (15) shows that the actual training-induced computational difference contains both direct effects of concentrated parameter updates and responses to upstream state changes. The averaging in Equation (12) therefore extracts a recurring component of the full layer computation rather than a parameter-space direction itself.

D.2 Propagation of TIRD through the Transformer

We next characterize how the extracted directions affect the model output. Let $V_\ell(c)$ denote the full residual-state perturbation obtained by placing v_ℓ at the positions where TIRD is injected and zeros elsewhere. For a shared injection coefficient α, let $z_c(\alpha)$ denote the output logits of the intervened Base model, with

$$z_c(0) = z_B(c). \tag{16}$$

Let C_c denote the derivative of the final readout with respect to the final residual state. The first-order effect of an intervention at layer ℓ on the output logits is

$$J_{\ell,c} = C_c(I + A_{L-1,c}) \cdots (I + A_{\ell+1,c}), \tag{17}$$

with $J_{L-1,c} = C_c$. Differentiating the full intervened computation with respect to α gives

$$b_c = z_c'(0) = \sum_{\ell=0}^{L-1} J_{\ell,c} V_\ell(c). \tag{18}$$

Thus, the output effect of TIRD depends on how each injected direction is transformed by the subsequent layers. The analysis does not assume that later layers preserve the direction or magnitude of the injected vector.

Let

$$d_c = z_T(c) - z_B(c) \tag{19}$$

denote the training-induced change in logits, and define the mismatch

$$e_c = d_c - b_c. \tag{20}$$

The mismatch admits the decomposition

$$e_c = C_c \sum_{\ell} (\Delta_\ell(c) - V_\ell(c)) + \sum_{\ell} (C_c - J_{\ell,c}) V_\ell(c) + r_c^T. \tag{21}$$

The first term captures variation removed by cross-context averaging and differences in the intervention mask. The second captures mismatch introduced by downstream propagation. The residual r_c^T contains changes in the initial state, final normalization, output head, and higher-order readout effects.

This decomposition is useful because it separates the part of the training-induced output change reproduced by TIRD, b_c, from the part that is not reproduced, e_c.

D.3 TIRD Injection Reduces the Distributional Gap

We now study whether moving the Base model along TIRD brings its output distribution closer to that of Target.

For each context c, define

$$p_c(\alpha) = \mathrm{softmax}(z_c(\alpha)), \qquad p_T(c) = \mathrm{softmax}(z_T(c)), \tag{22}$$

and let

$$p_B(c) = p_c(0). \tag{23}$$

The expected conditional KL divergence from Target to the intervened Base model is

$$K(\alpha) = \mathbb{E}_{c\sim\mathcal{D}} D_{\mathrm{KL}} \left(p_T(c) \| p_c(\alpha) \right). \tag{24}$$

Define the output-space alignment

$$G = \mathbb{E}_{c\sim\mathcal{D}} \left[b_c^\top \left(p_T(c) - p_B(c) \right) \right]. \tag{25}$$

A positive G means that the local output change induced by TIRD is aligned, on average, with the direction from the Base distribution toward the Target distribution.

Theorem D.1 (Local KL reduction under TIRD injection) *Assume that the logit path $z_c(\alpha)$ is twice differentiable for $0 \le \alpha \le a$. Suppose there exist functions $M_1(c)$ and $M_2(c)$ such that*

$$\|z_c'(\alpha)\|_2 \le M_1(c), \qquad \|z_c''(\alpha)\|_2 \le M_2(c), \tag{26}$$

and

$$\mathbb{E}_{\mathcal{D}} \left[M_1(c)^2 + M_2(c) \right] < \infty. \tag{27}$$

Define

$$\mathcal{H} = \mathbb{E}_{\mathcal{D}} \left[\frac{1}{2} M_1(c)^2 + \sqrt{2} M_2(c) \right]. \tag{28}$$

If $G > 0$, then

$$K(\alpha) - K(0) \le -\alpha G + \frac{1}{2} \mathcal{H} \alpha^2. \tag{29}$$

Consequently,

$$K(\alpha) < K(0) \tag{30}$$

for

$$0 < \alpha < \min \left\{ a, \frac{2G}{\mathcal{H}} \right\}. \tag{31}$$

Proof. For a fixed context, let

$$A(z) = \log \sum_i e^{z_i}. \tag{32}$$

The conditional KL divergence can be written as

$$K_c(\alpha) = A(z_c(\alpha)) - A(z_T) - p_T^\top (z_c(\alpha) - z_T). \tag{33}$$

Differentiating gives

$$K_c'(\alpha) = z_c'(\alpha)^\top (p_c(\alpha) - p_T). \tag{34}$$

At $\alpha = 0$,

$$K'(0) = -\mathbb{E}\left[b_c^\top (p_T - p_B)\right] = -G. \tag{35}$$

The second derivative is

$$K_c''(\alpha) = z_c'(\alpha)^\top F(z_c(\alpha)) z_c'(\alpha) + z_c''(\alpha)^\top (p_c(\alpha) - p_T), \tag{36}$$

where

$$F(z) = \mathrm{diag}(p(z)) - p(z)p(z)^\top \tag{37}$$

is the softmax Fisher matrix.

The spectral norm of the softmax Hessian satisfies

$$\|F(z)\|_2 \le \frac{1}{2}, \tag{38}$$

and the Euclidean distance between two probability vectors is at most $\sqrt{2}$. Hence

$$K''(\alpha) \le \mathcal{H}. \tag{39}$$

Integrating once yields

$$K'(\alpha) \le -G + \mathcal{H}\alpha, \tag{40}$$

and integrating again gives

$$K(\alpha) - K(0) \le -\alpha G + \frac{1}{2}\mathcal{H}\alpha^2. \tag{41}$$

The stated interval follows immediately. □

The theorem gives a local sufficient condition: if the output change induced by TIRD is positively aligned with the Target distribution, then a sufficiently small positive injection reduces the conditional KL divergence to Target. The valid range depends on both the alignment G and the curvature of the nonlinear inference path. This is the exact nonlinear form used in the analysis.

D.4 When Does TIRD Have Positive Alignment?

The previous theorem reduces the problem to the sign of G. We next connect this condition to the fraction of the training-induced output change captured by TIRD.

Let

$$F_c = \mathrm{diag}(p_B(c)) - p_B(c)p_B(c)^\top \tag{42}$$

be the Fisher matrix of the Base distribution, and define

$$S = \mathbb{E}\left[b_c^\top F_c b_c\right], \tag{43}$$

$$E = \mathbb{E}\left[e_c^\top F_c e_c\right], \tag{44}$$

and

$$C = \mathbb{E}\left[b_c^\top F_c e_c\right]. \tag{45}$$

Here S measures the effective strength of the TIRD-reproduced component in output-probability geometry, while E measures its mismatch with the full training-induced logit change. By Cauchy–Schwarz,

$$|C| \le \sqrt{SE}. \tag{46}$$

Assume

$$S > 0, \qquad E \le \eta^2 S, \qquad 0 \le \eta < 1. \tag{47}$$

This condition requires that the output-space component preserved by TIRD is stronger than the mismatch introduced by averaging and downstream propagation.

Using $d_c = b_c + e_c$, the softmax expansion around the Base logits gives

$$p_T - p_B = F_c d_c + q_c, \tag{48}$$

where the remainder satisfies

$$\|q_c\|_2 \le \frac{L_F}{2}\|d_c\|_2^2. \tag{49}$$

Therefore,

$$G = S + C + \mathbb{E}\left[b_c^\top q_c\right]. \tag{50}$$

Define

$$R = \frac{L_F}{2}\mathbb{E}\left[\|b_c\|_2\|d_c\|_2^2\right]. \tag{51}$$

Then

$$G \ge S - \sqrt{SE} - R \ge (1-\eta)S - R. \tag{52}$$

This gives the following sufficient condition.

Corollary D.2 (Positive alignment from a dominant retained component) *Under Equation* (47)*, if*

$$R < (1-\eta)S, \tag{53}$$

then $G > 0$. Therefore, a non-zero interval of positive TIRD coefficients reduces the KL divergence from the intervened Base model to Target.

When b_c and d_c are both $O(\varepsilon)$, the remainder R is $O(\varepsilon^3)$, while S is second order. Thus, for sufficiently small training-induced changes, the dominance condition in Equation (47) is sufficient to produce positive alignment.

This result clarifies the role of low-rank parameter structure. Rank-one concentration provides a structural source for a recurring computational component. It does not by itself guarantee KL reduction. The additional requirement is that this component survives cross-context averaging and downstream propagation strongly enough to dominate the remaining mismatch.

D.5 Second-Order Interpretation

The same mechanism has a simpler interpretation under the local Fisher approximation. Consider

$$K_2(\alpha) = \frac{1}{2}\mathbb{E}\left[(d_c - \alpha b_c)^\top F_c (d_c - \alpha b_c)\right]. \tag{54}$$

Using $d_c = b_c + e_c$ gives

$$K_2(\alpha) = \frac{1}{2}\left[(1-\alpha)^2 S + 2(1-\alpha)C + E\right]. \tag{55}$$

Hence

$$K_2(\alpha) - K_2(0) = -\alpha(S + C) + \frac{1}{2}\alpha^2 S. \tag{56}$$

Since

$$|C| \le \sqrt{SE} \le \eta S, \tag{57}$$

we obtain

$$K_2(\alpha) - K_2(0) \le -\alpha\left(1 - \eta - \frac{\alpha}{2}\right)S. \tag{58}$$

The second-order form makes the role of TIRD explicit. Injection reduces the part of the training-induced discrepancy that lies along the retained direction, while the unmatched component remains as residual error. If the retained component dominates the mismatch, positive injection initially moves the Base distribution toward Target.

D.6 Scope and Assumptions

The analysis characterizes the local effect of TIRD injection around the Base model. The effective range of the injection coefficient is governed by the alignment between the TIRD-induced output change and the Target distribution, together with the curvature of the inference path.

Low-rank parameter structure provides a mechanism for recurring computational changes across reasoning contexts. The KL reduction result further depends on how much of this recurring component is preserved after cross-context averaging and propagation through the Transformer.

The theorem is formulated for fixed reasoning prefixes and full softmax distributions. For finite autoregressive sequences, the conditional result extends under the Target prefix distribution through the chain rule of KL divergence. Different decoding temperatures can be incorporated through z/τ.

The resulting guarantee concerns the predictive distribution of the model. Its relation to reasoning performance is examined empirically in the main experiments.