A APPENDIX

This is the supplementary document for the submitted manuscript, COT VECTORS: TRANSFERRING AND PROBING THE REASONING MECHANISMS OF LLMS.

This appendix contains five parts: a statement regarding the usage of large language models (Section A.1), a detailed derivation of the CoT shift formulation introduced in the main text (Section A.2), additional implementation details (Section A.3) and complete experimental results in main text (Section A.4), and finally extended ablation studies and analyses of CoT Vectors (Section A.5).

A.1 THE USE OF LARGE LANGUAGE MODELS

In preparing this manuscript, we use Large Language Models (LLMs) solely as a writing assistant for grammar correction and minor phrasing refinement. The LLMs doesn't contribute to research ideation, experimental design, analysis, or interpretation of results. All scientific content and claims are entirely the responsibility of the authors. The LLMs are not involved in authorship and are not considered contributors to this work.

A.2 THE DERIVATION OF COT SHIFT

When the model reasons with CoT, the generation of an answer token $a \in A$ depends not only on the question tokens but also on the intermediate CoT tokens. Formally, the single-head self-attention can be expressed as

$$\begin{aligned}
\mathrm{SA}(a, [\boldsymbol{K}_Q, \boldsymbol{K}_C, \boldsymbol{K}_A], [\boldsymbol{V}_Q, \boldsymbol{V}_C, \boldsymbol{V}_A]) &= \frac{\exp(a\boldsymbol{K}_Q^\top)}{Z_{\text{total}}}\boldsymbol{V}_Q + \frac{\exp(a\boldsymbol{K}_C^\top)}{Z_{\text{total}}}\boldsymbol{V}_C + \frac{\exp(a\boldsymbol{K}_A^\top)}{Z_{\text{total}}}\boldsymbol{V}_A \\
&= \frac{Z_Q}{Z_{\text{total}}} \cdot \underbrace{\frac{\exp(a\boldsymbol{K}_Q^\top)}{Z_Q}\boldsymbol{V}_Q}_{\mathrm{SA}(a,[\boldsymbol{K}_Q],[\boldsymbol{V}_Q])} + \frac{Z_C}{Z_{\text{total}}} \cdot \underbrace{\frac{\exp(a\boldsymbol{K}_C^\top)}{Z_C}\boldsymbol{V}_C}_{\mathrm{SA}(a,[\boldsymbol{K}_C],[\boldsymbol{V}_C])} + \frac{Z_A}{Z_{\text{total}}} \cdot \underbrace{\frac{\exp(a\boldsymbol{K}_A^\top)}{Z_A}\boldsymbol{V}_A}_{\mathrm{SA}(a,[\boldsymbol{K}_A],[\boldsymbol{V}_A])}
\end{aligned}$$

where $Z_Q = \sum \exp(a\boldsymbol{K}_Q^\top)$, $Z_C = \sum \exp(a\boldsymbol{K}_C^\top)$, $Z_A = \sum \exp(a\boldsymbol{K}_A^\top)$, and $Z_{\text{total}} = Z_Q + Z_C + Z_A$.

In contrast, without CoT, the self-attention reduces to

$$\mathrm{SA}(a, [\boldsymbol{K}_Q, \boldsymbol{K}_A], [\boldsymbol{V}_Q, \boldsymbol{V}_A]) = \frac{Z_Q}{Z_Q + Z_A} \cdot \mathrm{SA}(a, [\boldsymbol{K}_Q], [\boldsymbol{V}_Q]) + \frac{Z_A}{Z_Q + Z_A} \cdot \mathrm{SA}(a, [\boldsymbol{K}_A], [\boldsymbol{V}_A])$$

Therefore, through algebraic transformations and let $\mu = \dfrac{Z_C}{Z_{\text{total}}}$, we can obtain,

$$\mathrm{SA}(a, [\boldsymbol{K}_Q, \boldsymbol{K}_C, \boldsymbol{K}_A], [\boldsymbol{V}_Q, \boldsymbol{V}_C, \boldsymbol{V}_A]) = \underbrace{\mathrm{SA}(a, [\boldsymbol{K}_Q, \boldsymbol{K}_A], [\boldsymbol{V}_Q, \boldsymbol{V}_A])}_{\text{Standard attention}} \tag{8}$$

$$+ \mu \cdot \underbrace{(\mathrm{SA}(a, [\boldsymbol{K}_C], [\boldsymbol{V}_C]) - \mathrm{SA}(a, [\boldsymbol{K}_Q, \boldsymbol{K}_A], [\boldsymbol{V}_Q, \boldsymbol{V}_A]))}_{\text{CoT shift}} \tag{9}$$

This is Equation 1 in the main text, which reveals that when LLM performs CoT inference, the actual content of CoT is a prefix for the final answer given. Its role in inference ultimately comes from an attention shift, which means that CoT information can be compressed into this shift. This discovery provides a theoretical basis for CoT Vectors.

A.3 MORE IMPLEMENTATION DETAILS

A.3.1 DATASET DETAILS

GSM8K. We randomly sample 3,000 examples from the training set to construct the support set for CoT vector extraction/learning and LoRA fine-tuning. For evaluation, we randomly selected 1,000 examples from the test set.

MATH. We have observed that the MATH dataset contains problems with five levels of difficulty, and the imbalanced distribution across these levels could lead to unstable training. To address this, we partition the dataset into two subsets based on difficulty: MATH-Easy (levels 1–3) and MATH-Hard (levels 4–5). For each subset, we combine all categories and then sample 3,000 examples from the training portion to form the support set. Similarly, 1,000 examples were sampled from the test portion for evaluation.

MMLU-Pro. Since ground-truth CoT sequences are required, we have used the official validation set as the support set. Although this set contains only 70 samples, it still provides sufficient signal for CoT vectors to capture useful reasoning patterns. For testing, we also sample 1,000 examples from the test set.

A.3.2 PROMPTS AND GENERATION CONFIGURATIONS

Prompts. We use two versions of prompts: CoT and Non-CoT, as shown in Table 3. The CoT version is specifically designed for scenarios that require step-by-step reasoning. This prompt format is consistently applied to all our baseline model tests and to the evaluation processes for both the CoT Vectors and LoRA methods. It also serves as the teacher input during the training of our Learnable CoT Vectors. Conversely, the Non-CoT version is used to directly obtain the final answer without any intermediate reasoning, serving exclusively as the student input during the training of our Learnable CoT Vectors.

Additionally, for GSM8K and MATH datasets, we instruct the LLM to directly generate the final answer, while for the multiple-choice dataset MMLU-Pro, we prompt the model to output the correct option letter. In the input construction for MMLU-Pro, the answer choices are appended immediately after the question.

Table 3: Prompt Templates.

Dataset	**Reasoning Type**	**Prompt**
GSM8K & MATH	CoT	You are a helpful and precise assistant for solving math problems. Please reason step by step, and put your final answer within \boxed{}.
	non-CoT	You are a helpful and precise assistant for solving math problems. Put your answer within \boxed{}.
MMLU-Pro	CoT	You are a helpful and precise assistant for solving problems. Please reason step by step, and put your final answer within \boxed{}. Your final output should be only the uppercase letter of the correct choice (e.g., A).
	non-CoT	You are a helpful and precise assistant for solving problems. Put your answer within \boxed{}. Your final output should be only the uppercase letter of the correct choice (e.g., A).

Generation Configurations. All tests were conducted using a consistent set of generation parameters, as detailed in Table 4.

A.3.3 HYPERPARAMETERS

We employee the AdamW optimizer (Loshchilov & Hutter, 2017) with a weight decay of 1e-3 and train using float16 precision for both Learnable CoT Vector and LoRA. Additionally, we set the accumulate gradient batches to 2 to effectively increase the batch size.

Learnable CoT Vectors. To train our Learnable CoT Vectors, we configure distinct hyperparameters for each model and dataset. All of these parameters are fine-tuned through a combination of systematic grid search and manual adjustments to ensure stable and efficient training. For the LLM, we adopt a unique tiered learning rate strategy. Using higher learning rate for the earlier layers of

Table 4: Generation Configuration Parameters

Parameter	**Value**
Number of beams	3
Maximum new tokens	512
Length penalty	0.0
Do sample	False
Temperature	1.0
Top-p	1.0

Qwen is intended to obtain vectors that better fit the task-general direction, as mentioned in Section 4.2 of the main text, where shallow layers exhibit better directional properties. In contrast, using smaller learning rates for the deeper layers of Qwen and the entire Llama model aims to obtain underfitting vectors, as they demonstrate better stability when information density is high.As detailed in Table 5.

Table 5: Hyperparameters for Learnable CoT Vector

(a) Qwen

Dataset	**Samples**	**Learning Rate**	**Warm-up**	**Note**
GSM8K	3000	5e-3 and 1e-4	0.1	LR: First 4 layers used 5e-3, others 1e-4
MMLU-P	70	5e-3 and 1e-4	0.1	LR: First 4 layers used 5e-3, others 1e-4
MATH-E	2000	5e-3 and 1e-4	0.5	LR: First 4 layers used 5e-3, others 1e-4
MATH-H	2000	5e-3 and 1e-4	0.5	LR: First 4 layers used 5e-3, others 1e-4

(b) LLaMA

Dataset	**Samples**	**Learning Rate**	**Warm-up**
GSM8K	2000	1e-4	0.5
MMLU-P	70	1e-4	0.1
MATH-E	1000	1e-4	0.5
MATH-H	1000	1e-4	0.5

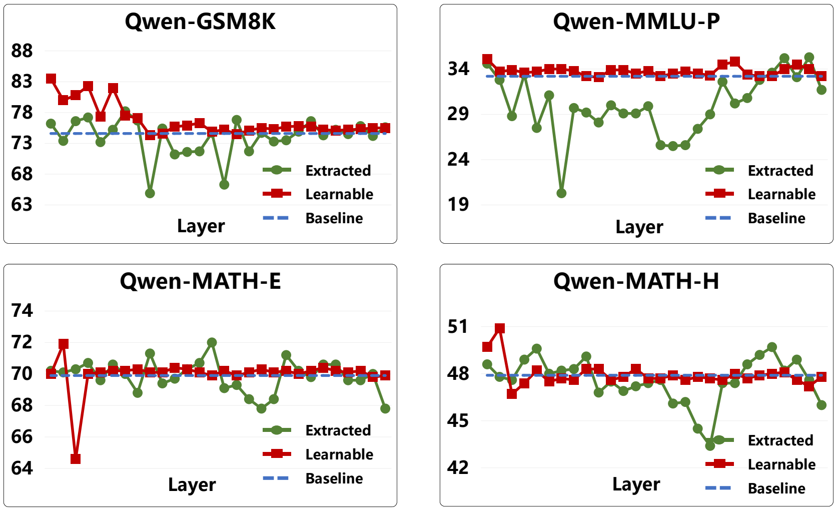
LoRA. We apply LoRA with a set of shared, general hyperparameters across all models and datasets, with training set sizes consistent with those used for CoT vector training. Interestingly, we found that when using LoRA for fine-tuning, the loss was very low from the beginning of training. This aligns with our hypothesis in Section 4.2 that for models already fine-tuned with CoT and possessing CoT capabilities, using LoRA for further specific CoT fine-tuning would yield limited gains.As detailed in Table 6.

Table 6: General Hyperparameters for LoRA.

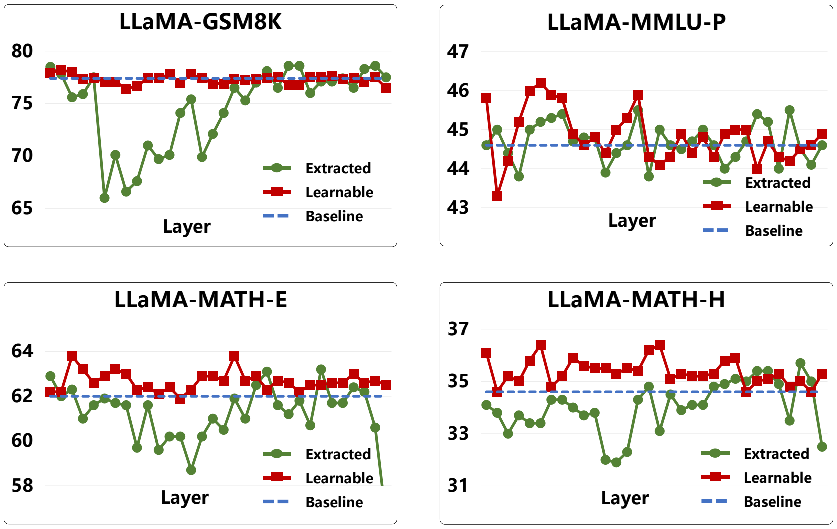
Parameter	**Value**
Learning Rate	5×10^{-5}
Warm-up	0.1
Matrix Rank	16
Dropout Rate	0.05
Target Modules	`q_proj, k_proj, o_proj, v_proj`

A.4 COMPLETE EXPERIMENTAL RESULTS

In Section 4.2 of the main text, we provide the comprehensive evaluation results for two types of CoT Vectors, baseline and LoRA cross models and benchmarks, in Table 1. Especially for CoT



(a) Layer-wise results of Qwen2.5-Math-7B



(b) Layer-wise results of LLaMA3.1-8B-Instruct

Figure 5: Layer-wise performance analysis of two LLMs and four benchmarks.

Vectors, the final result is the report of the best layer after iterating and testing the results of each layer of LLMs. Here we present all the layer-wise results of each model and dataset in Figure 5.

We can observe the following pattern. First, as mentioned in the main text, extracted layers generally exhibit performance troughs in the middle layer, with strong volatility. Relatively speaking, learnable layers usually achieve their best performance in the first few layers, while the subsequent layers are generally stable around the baseline or with a small gain around the baseline. Second, in addition, the most crucial point is that for highly volatile extracted vectors, we find that even if the same model is used, the layers that achieve the best performance on different datasets are completely different. Even a certain layer can gain a lot on one dataset, but it will damage performance on another dataset, and there is basically no regularity in this. This means that when using extracted CoT Vectors, a layer wise search must be conducted first to determine the optimal shift layer, which is almost impractical for practical applications without ground truth.

Table 7: Results of Raw Activation Extracted CoT Vectors.

Model	Method	#Params	GSM8K	MATH-E	MATH-H	MMLU-P	Avg.
Qwen2.5-Math-7B	Baseline	—	74.6	69.9	47.9	33.2	56.4
	Raw Activation	—	77.6	71.5	49.2	**35.8**	58.5
	Activation Gap	—	**78.2**	**72.0**	**49.7**	35.3	**58.8**
LLaMA3.1-8B-Instruct	Baseline	—	77.4	62.0	34.6	44.6	54.7
	Raw Activation	—	**80.4**	62.4	35.4	**45.8**	**56.0**
	Activation Gap	—	78.6	**63.2**	**35.7**	45.5	55.8

A.5 ABLATIONS AND ADDITIONAL ANALYSES

Due to space limitations in the main paper, we provide extended analyses of CoT Vectors in this section. We first present alternative acquisition methods beyond the activation-gap based approach, including raw-activation (Section A.5.1) and attention-level variants (Section A.5.2), which also improve over the baseline but yield less representative gains than the method reported in the main text. We then conduct ablation studies on key design choices, such as the scaling factor μ for extracted vectors (Section A.5.4), the loss balance for learnable vectors (Section A.5.6), and the training set size (Section A.5.7), through which we uncover additional insights into their behaviors. We further analyze robustness and transferability, considering settings such as multi-layer vector injection (Section A.5.3), varying the source of teacher CoT sequences (Section A.5.5), and cross-model or cross-dataset transfer (Section A.5.8). Finally, we complement these quantitative studies with case analyses (Section A.5.9), illustrating both successful and failure scenarios to better understand when and why CoT Vectors are effective.

A.5.1 RAW ACTIVATION EXTRACTED COT VECTOR

This method is inspired by the work in ICL (Todd et al., 2023; Hendel et al., 2023), using the direct model activations as a coarse-grained approximation of the CoT Vector. Specifically, we forward-pass the triplet (Q, CoT, A) and take the mean of the hidden states (activations) across all answer tokens A at a specific layer l.

$$\vec{v}_{\mathrm{CoT}}^{(l)} = \frac{1}{|A|} \sum_{a \in A} \boldsymbol{\alpha}_{\mathrm{CoT}}^{(l)}(a) \tag{10}$$

where $\boldsymbol{\alpha}_{\mathrm{CoT}}^{(l)}(a)$ is the hidden state (activation vector) of the answer token a at layer l under the CoT prompting input, and $|A|$ denotes the total number of answer tokens.

Raw Activation vs. Activation Gap. Comparing two extraction strategies in Table 7, the Activation-Gap vector generally outperforms the Raw-Activation vector in most datasets, as it better isolates the reasoning-induced shift. Interestingly, in some cases raw activations match or even surpass gap-based ones, suggesting that hidden states encode not only the reasoning information but also auxiliary task-relevant features.

A.5.2 ATTENTION-LEVEL COT VECTOR

From a mathematical perspective, the most precise location for the CoT Vector should be within the attention computation, as shown in Eq. 3 in main text. Therefore, in addition to the Activation-level CoT Vector presented in the main text, we have also explored an intervention at the attention level in both Extracted and Learnable CoT Vectors. This method similarly improve performance over the baseline but is slightly less effective than the activation-level approach.

Attention-level Extracted CoT Vector. We extract the attention output at a specific layer when the LLM processes the (Q, CoT, A) triplet and the (Q, A) pair separately during the forward pass, and compute the gap between them as a highly accurate attention-output-level CoT Vector. For multi-head attention, we compute a distinct vector $\vec{v}_{\text{CoT, head}}^{(i,h)}$ for each head h, enabling more fine-grained control. The final task vector is a collection of all these head-level vectors.The extracted CoT Vector

Table 8: Results of Attention-level CoT Vectors.

Model	Method	#Params	GSM8K	MATH-E	MATH-H	MMLU-P	Avg.
Qwen2.5-Math-7B	Baseline	—	74.6	69.9	47.9	33.2	56.4
	Extracted	—	76.8	71.1	49.7	35.2	58.2
	Learnable	7.2K	76.1	71.9	48.6	34.1	57.9
LLaMA3.1-8B-Instruct	Baseline	—	77.4	62.0	34.6	44.6	54.7
	Extracted	—	78.3	62.7	35.4	46.2	55.7

in activation form is inserted into the hidden states of the corresponding extraction layer, while the attention-gap form is inserted into the attention output of that same layer. During insertion, a manually set constant μ is used to scale the magnitude of the CoT shift.

Attention-level Learnable CoT Vector. Similar to the activation level learnable method, but the attention-level vector is added as a shift to the output of a specific attention module. For multi-head attention, a distinct vector is learned for each head, allowing for fine-grained control.

In practice, although this method does lead to some performance improvement against baseline, the gains are relatively modest compared to activation-level variants. The details of these experiments are listed in Table 8. This result suggests that the effectiveness of task-general CoT Vectors does not rely on fine-grained, head-specific attention patterns, but instead reflects broader representational shifts at the activation level.

A.5.3 MULTI-LAYER COT INJECTION

In our main experiments, our designed CoT Vector is exclusively obtained and inserted at a single layer of the LLM. However, we are also curious about whether such operations could be applied simultaneously across multiple layers, and whether performing shifts in more layers would lead to greater performance improvements.

Extracted CoT Vectors. As shown in Table 9, we find that simultaneously injecting the extracted CoT Vectors into multiple layers produces different effects, depending on the functional characteristics of the selected layers. When vectors are injected into multiple layers that prove beneficial for CoT integration(e.g. Layer 3, 7, 15 for GSM8K), a positive synergistic effect is observed, resulting in greater performance improvement than single-layer injection. Conversely, when vectors are simultaneously inserted into multiple layers that are sensitive to CoT but unsuitable for external intervention(e.g. Layer 8, 14), additional interference effects arise, leading to a further decline in performance,which sometimes even significantly below the baseline. This phenomenon reaffirms the significant functional specialization among different layers of the Transformer and also indicates that the injection effect of CoT Vectors can be further optimized through inter-layer coordination and combinatorial selection.

Table 9: Multi-layer Extracted CoT Injection on Qwen-GSM8K.

Model	Layers	Accuracy (%)
Baseline	–	74.6
Extracted	15	77.6
	7, 15	79.3
	3, 7, 15	**80.1**
	8, 14	45.1

Learnable CoT Vectors. We further investigate multi-layer insertion of Learnable CoT shifts. Interestingly, multi-layer insertion produces complementary effects for extracted vs. learnable vectors: extracted vectors benefit from multi-layer insertion (cumulative mild shifts yield larger gains), while learnable vectors typically perform best when injected at a single shallow layer and degrade under naive multi-layer injection as shown in Table 10. Mechanistically, this difference stems from the amplitude and directionality of the shifts. Extracted vectors produce mild, descriptive shifts that safely accumulate across layers, whereas learnable vectors induce stronger, optimization-driven shifts that can over-steer representations if applied repeatedly.

Table 10: Multi-layer Learnable CoT Injection on Qwen-GSM8K.

Method	**Layers**	**Accuracy (%)**
Baseline	–	74.6
Learnable	0	**83.5**
	0, 1	81.0
	0, 1, 2	76.1

A.5.4 ABLATION ON THE SCALING FACTOR μ OF EXTRACTED COT VECTORS

In the main experiments, the scaling factor μ for the Extracted CoT Vectors is set to a fixed value of 1.0. To further investigate the impact of this hyperparameter, we experiment with different values of μ.

For single-layer injection. In Table 11, we find that the effect of the scaling coefficient μ varies significantly across different model layers, leading to the following observations: For layers where μ = 1.0 already yields improvement (e.g., Qwen on GSM8K at layer 6), increasing μ can further enhance performance slightly, although an excessively large μ leads to over-steering and performance degradation. In contrast, for layers where μ = 1.0 causes performance degradation (e.g., Qwen on MATH-Hard at layer 17), reducing μ can mitigate the negative effect, but no value of μ can bring improvement over the baseline. This indicates that the CoT Vectors extracted from these layers are inherently low-quality, and their deficiency cannot be remedied simply by adjusting μ.

Table 11: Model Accuracy Comparison Across Different μ Values for single-layer injection

(a) Qwen-MATH-Hard-Layer 17

μ	Accuracy (%)
0.0	**47.9**
0.2	46.6
0.5	46.7
1.0	45.8
1.2	44.4

(b) Qwen-GSM8K-Layer 6

μ	Accuracy (%)
0.0	74.6
0.8	78.1
1.0	78.2
1.2	**78.4**
1.5	78.0

For multi-layer injection. For many layers of dense shift insertion, it is necessary to reduce the μ value. As shown in Table 12, when applying CoT Vector shifts to four layers of the LLM concurrently, scaling down μ to 0.5 leads to further performance improvement. When injecting CoT Vectors into all layers simultaneously, reducing μ to 0.2 is necessary to achieve optimal results.

Table 12: Model Accuracy Comparison with Different μ Values for multi-layer injection

(a) Qwen-GSM8K-all layers

Setting	Accuracy (%)
Baseline ($\mu = 0$)	74.6
$\mu = 0.05$	74.8
$\mu = 0.1$	76.3
$\mu = 0.2$	**76.7**
$\mu = 0.3$	67.1
$\mu = 0.5$	14.5
$\mu = 1.0$	15.2

(b) Qwen-GSM8K

Model	μ	Accuracy (%)
Baseline	-	74.6
Layers 3,7,15,23	1.0	79.6
Layers 3,7,15,23	0.5	**80.3**

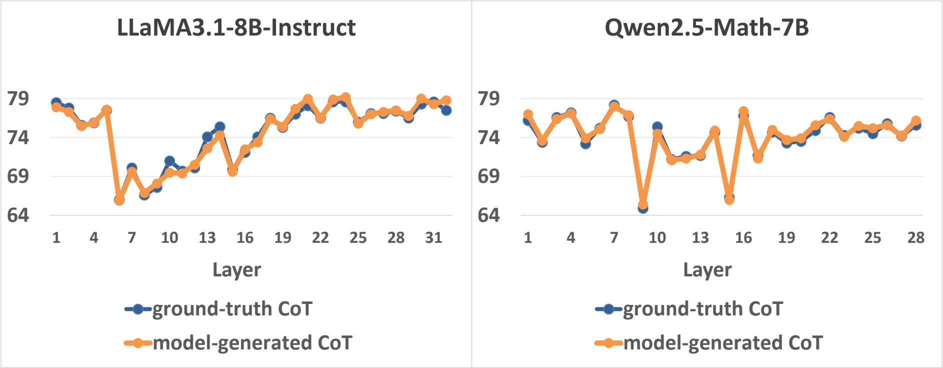


Figure 6: Performance comparison between ground-truth and model-generated CoT sequences of Extracted CoT Vectors on GSM8K.

A.5.5 SOURCES OF TEACHER COT

For the explicit CoT sequences used in extraction or as the teacher signal in the learning process, we experiment with different types of sources. These include using ground-truth CoT from the dataset (typically human-written) as well as model-generated CoT sequences, obtained from the model itself.

As shown in Figure 6,experimental results indicate that the performance difference between using ground-truth CoT and model-generated CoT is relatively minor. Both approaches yield comparable improvements in the model's reasoning capabilities, suggesting that the benefits of CoT guidance are robust to variations in the specific wording or stylistic differences of the reasoning chains.

A.5.6 ABLATION ON TRAINING OBJECTIVES FOR LEARNABLE COT VECTORS

For training the Learnable CoT Vectors, we perform an ablation study on the loss design. In addition to the hybrid loss (a combination of representation alignment loss and cross-entropy loss) adopted in the main experiments, we evaluate two simplified variants: using only the **alignment loss** or only the **cross-entropy loss**. As shown in Table 13, both simplified variants lead to clear performance degradation compared to the hybrid objective.

We attribute this to the complementary roles of the two components. The representation alignment loss is crucial for distilling task-general reasoning patterns: by enforcing the student's intermediate representations to align with the teacher's reasoning trajectory, it effectively transfers the underlying logical process. However, when used alone, it lacks explicit guidance for producing correct and coherent final answers. The cross-entropy loss complements this by anchoring the model's output to accurate reasoning traces, ensuring that the learned vector supports both faithful reasoning dynamics and correct task execution.

Table 13: Ablation Results on Training Objectives in the Qwen–GSM8K Setting

Method	Accuracy (%)
Baseline	74.6
Learnable CoT Vector	**83.5**
Learnable CoT Vector (only alignment loss)	82.9
Learnable CoT Vector (only CE loss)	78.4

A.5.7 ABLATION ON TRAINING SET SIZE FOR LEARNABLE COT VECTORS

We further conduct an ablation study on the size of the support set to compare the performance of the learnable CoT Vector and LoRA under different data regimes. As shown in Table 14, while both methods benefit from larger support sets, the Learnable CoT Vector consistently outperforms LoRA across all data scales. Notably, with a very small support set (e.g., 100 examples), the learnable CoT Vector still yields noticeable improvements over the baseline, whereas LoRA offers only marginal gains. As the support set grows, the learnable CoT Vector also demonstrates greater potential for performance improvement compared to LoRA. These phenomena all indicate that our learnable CoT Vectors are a more advantageous method for improving reasoning performance.

Table 14: Performance Comparison with Different Training Sample Sizes

Sample Size	**Baseline**	**Learnable CoT Vector**	**LoRA**
100	74.6	**78.2**	76.0
500	74.6	**79.0**	77.9
1000	74.6	**82.3**	78.5
3000	74.6	**83.5**	79.0

A.5.8 CROSS-DATASET AND CROSS-MODEL TRANSFERABILITY

Source → Target	**Baseline**	**Self**	**Transferred**
Cross-Model Transfer			
Qwen2.5-Math-7B-Instruct → Qwen2.5-Math-7B	74.6	78.2	77.5
Cross-Dataset Transfer			
GSM8K → MATH	47.9	49.7	48.6
MMLU-Pro → MATH	47.9	49.7	48.5

Table 15: Cross-model and Cross-dataset transfer results of CoT Vectors. Baseline refers to standard zero-shot CoT prompting. Self means applying the CoT Vector obtained from the same model–dataset pair (no transfer). Transferred means applying a CoT Vector obtained from a different source model or dataset.

We investigate whether CoT Vectors acquired from one source (model or dataset) can be effectively applied to another.

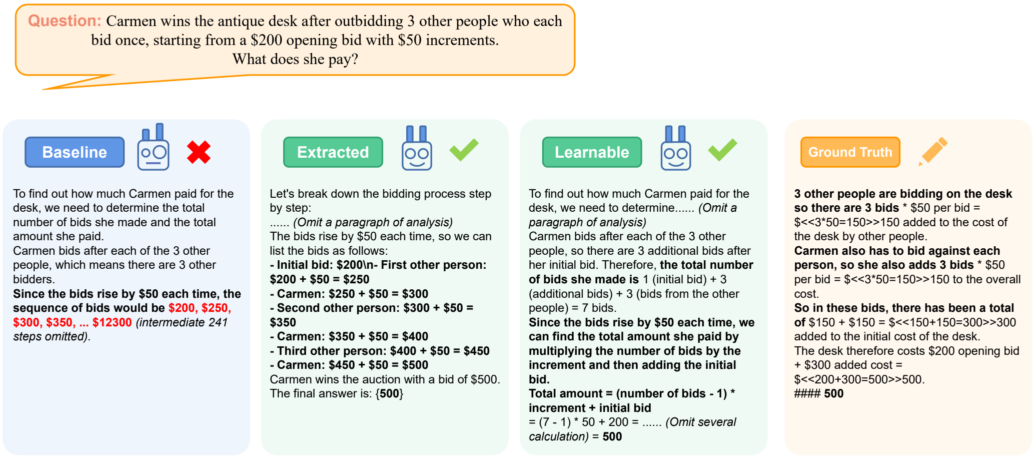
Cross-Model Transfer. As shown in Table 15, CoT Vectors gained from one model can be effectively reused in another. The vectors obtained from more powerfully instruction-tuned variant of the Qwen2.5 series (Qwen2.5-Math-7B-Instruct) consistently improve performance when applied to Qwen2.5-Math-7B.

Cross-Dataset Transfer. Results in Table 15 further demonstrate transferability across datasets. 1) In-domain: CoT Vectors obtained from the GSM8K dataset effectively enhance performance on the MATH dataset. This confirms that the vector successfully captures a generalized mathematical reasoning strategy rather than merely memorizing dataset-specific features. 2) Cross-domain: vectors obtained from MMLU-Pro yield gains on MATH. This suggests that the CoT Vector may encode a meta-reasoning capability—such as the ability to decompose problems or follow logical steps—that is beneficial across distinct task domains.

These transferability experiments underscore a central claim of our work: the CoT Vector is not merely a compressed set of features from a specific model or dataset, but a portable, generalizable representation of a reasoning process that can be effectively applied in novel contexts.

A.5.9 CASE STUDIES

To qualitatively understand the mechanistic impact of CoT Vectors, we present two representative cases. Figure 7 (a) shows a GSM8K problem where baseline (Zero-Shot CoT) fails. The model's reasoning becomes stuck in an infinite loop, repetitively counting the bids without making progress toward a solution. In contrast, both the Extracted and Learnable CoT Vector helps the model to structure the problem correctly and produce a valid solution. Notably, the learnable vector not only leads to the correct final answer but also produces a reasoning chain that was more structured and closer to the ground-truth CoT in the dataset. This highlights how CoT Vectors can reshape the model's internal reasoning trajectory, guiding it toward more faithful and efficient solutions.



Question: Carmen wins the antique desk after outbidding 3 other people who each bid once, starting from a $200 opening bid with $50 increments. What does she pay?

Baseline ✗

To find out how much Carmen paid for the desk, we need to determine the total number of bids she made and the total amount she paid.
Carmen bids after each of the 3 other people, which means there are 3 other bidders.
Since the bids rise by $50 each time, the sequence of bids would be $200, $250, $300, $350, ... $12300 *(intermediate 241 steps omitted).*

Extracted ✓

Let's break down the bidding process step by step:
...... *(Omit a paragraph of analysis)*
The bids rise by $50 each time, so we can list the bids as follows:
- Initial bid: $200\n- First other person: $200 + $50 = $250
- Carmen: $250 + $50 = $300
- Second other person: $300 + $50 = $350
- Carmen: $350 + $50 = $400
- Third other person: $400 + $50 = $450
- Carmen: $450 + $50 = $500
Carmen wins the auction with a bid of $500.
The final answer is: {**500**}

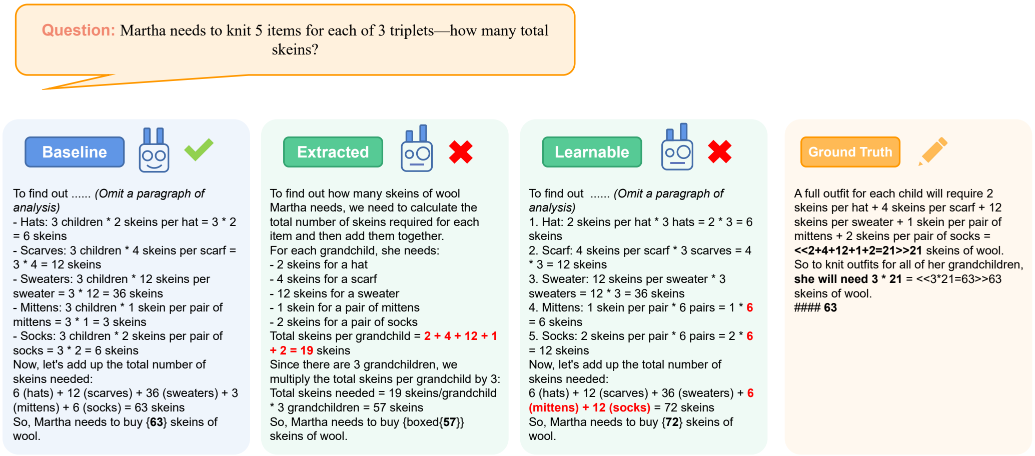
Learnable ✓

To find out how much Carmen paid for the desk, we need to determine...... *(Omit a paragraph of analysis)*
Carmen bids after each of the 3 other people, so there are 3 additional bids after her initial bid. Therefore, **the total number of bids she made is** 1 (initial bid) + 3 (additional bids) + 3 (bids from the other people) = 7 bids.
Since the bids rise by $50 each time, we can find the total amount she paid by multiplying the number of bids by the increment and then adding the initial bid.
Total amount = (number of bids - 1) * increment + initial bid
= (7 - 1) * 50 + 200 = ... *(Omit several calculation)* = **500**

Ground Truth

3 other people are bidding on the desk so there are 3 bids * $50 per bid = $<<3*50=150>>150 added to the cost of the desk by other people.
Carmen also has to bid against each person, so she also adds 3 bids * $50 per bid = $<<3*50=150>>150 to the overall cost.
So in these bids, there has been a total of $150 + $150 = $<<150+150=300>>300 added to the initial cost of the desk.
The desk therefore costs $200 opening bid + $300 added cost = $<<200+300=500>>500.
500

(a) Successful Guidance



Question: Martha needs to knit 5 items for each of 3 triplets—how many total skeins?

Baseline ✓

To find out ...... *(Omit a paragraph of analysis)*
- Hats: 3 children * 2 skeins per hat = 3 * 2 = 6 skeins
- Scarves: 3 children * 4 skeins per scarf = 3 * 4 = 12 skeins
- Sweaters: 3 children * 12 skeins per sweater = 3 * 12 = 36 skeins
- Mittens: 3 children * 1 skein per pair of mittens = 3 * 1 = 3 skeins
- Socks: 3 children * 2 skeins per pair of socks = 3 * 2 = 6 skeins

Now, let's add up the total number of skeins needed:
6 (hats) + 12 (scarves) + 36 (sweaters) + 3 (mittens) + 6 (socks) = 63 skeins
So, Martha needs to buy {**63**} skeins of wool.

Extracted ✗

To find out how many skeins of wool Martha needs, we need to calculate the total number of skeins required for each item and then add them together.
For each grandchild, she needs:
- 2 skeins for a hat
- 4 skeins for a scarf
- 12 skeins for a sweater
- 1 skein for a pair of mittens
- 2 skeins for a pair of socks

Total skeins per grandchild = **2 + 4 + 12 + 1 + 2 = 19** skeins
Since there are 3 grandchildren, we multiply the total skeins per grandchild by 3:
Total skeins needed = 19 skeins/grandchild * 3 grandchildren = 57 skeins
So, Martha needs to buy {boxed{**57**}} skeins of wool.

Learnable ✗

To find out ...... *(Omit a paragraph of analysis)*
1. Hat: 2 skeins per hat * 3 hats = 2 * 3 = 6 skeins
2. Scarf: 4 skeins per scarf * 3 scarves = 4 * 3 = 12 skeins
3. Sweater: 12 skeins per sweater * 3 sweaters = 12 * 3 = 36 skeins
4. Mittens: 1 skein per pair * 6 pairs = 1 * **6** = 6 skeins
5. Socks: 2 skeins per pair * 6 pairs = 2 * **6** = 12 skeins

Now, let's add up the total number of skeins needed:
6 (hats) + 12 (scarves) + 36 (sweaters) + **6 (mittens) + 12 (socks)** = 72 skeins
So, Martha needs to buy {**72**} skeins of wool.

Ground Truth

A full outfit for each child will require 2 skeins per hat + 4 skeins per scarf + 12 skeins per sweater + 1 skein per pair of mittens + 2 skeins per pair of socks = **<<2+4+12+1+2=21>>21** skeins of wool.
So to knit outfits for all of her grandchildren, **she will need 3 * 21** = <<3*21=63>>63 skeins of wool.
63

(b) Harmful Intervention

Figure 7: Case studies.

Conversely, we also observe cases where CoT Vectors harm performance. In Figure 7 (b), the baseline arrives at the correct answer unaided. However, when CoT Vectors are injected, the reasoning becomes more aligned with the ground-truth style (especially for extracted vectors), but both variants introduce hallucinations or arithmetic mistakes that ultimately lead to incorrect conclusions. This demonstrates that while CoT Vectors can effectively shift latent reasoning, the perturbation may sometimes interfere with correct reasoning paths, underscoring the double-edged nature of such interventions.