COT VECTORS: TRANSFERRING AND PROBING THE REASONING MECHANISMS OF LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Chain-of-Thought (CoT) prompting has emerged as a powerful approach to enhancing the reasoning capabilities of Large Language Models (LLMs). However, existing implementations, such as in-context learning and fine-tuning, remain costly and inefficient. To improve CoT reasoning at a lower cost, and inspired by the task vector paradigm, we introduce CoT Vectors, compact representations that encode task-general, multi-step reasoning knowledge. Through experiments with Extracted CoT Vectors, we observe pronounced layer-wise instability, manifesting as a U-shaped performance curve that reflects a systematic three-stage reasoning process in LLMs. To address this limitation, we propose Learnable CoT Vectors, optimized under a teacher–student framework to provide more stable and robust guidance. Extensive evaluations across diverse benchmarks and models demonstrate that CoT Vectors not only outperform existing baselines but also achieve performance comparable to parameter-efficient fine-tuning methods, while requiring fewer trainable parameters. Moreover, by treating CoT Vectors as a probe, we uncover how their effectiveness varies due to latent space structure, information density, acquisition mechanisms, and pre-training differences, offering new insights into the functional organization of multi-step reasoning in LLMs. The source code will be released.

1 INTRODUCTION

Chain-of-Thought (CoT) prompting (Wei et al., 2022) has emerged as a powerful technique to unlock the complex reasoning capabilities of Large Language Models (LLMs) (Zhao et al., 2023). By reasoning step-by-step, CoT enables models to decompose problems, mimic human-like logic, and improve performance on several challenging tasks (Imani et al., 2023; Huang & Chang, 2022). However, how to effectively harness the power of CoT in practice remains an open problem. Existing approaches generally fall into two categories: (1) In-Context Learning (ICL) (Brown et al., 2020) with few-shot CoT examples, which can enhance reasoning, but it requires longer prompts and slows inference; (2) Fine-tuning LLMs (Ziegler et al., 2019) with CoT-annotated data, which demands large amounts of high-quality reasoning traces and computational resources, while often yielding only limited improvements for models that already equipped with CoT abilities. These challenges prompt a critical question: *can we transfer the essence of CoT, i.e., the general "problem-solving mindset" of a task, into LLMs in a way that is compact, reusable, and efficient?*

Recent advances in Task Vectors (Ilharco et al., 2022) offer a promising direction. Task-specific knowledge can be distilled into a compact vector, often represented as the difference in activations (Hendel et al., 2023; Todd et al., 2023; Liu et al., 2023) or parameters (Ortiz-Jimenez et al., 2023b; Li et al., 2025) between fine-tuned and base models. Such vectors can steer model behavior toward new tasks without modifying model weights, thereby enabling parameter-efficient adaptation. However, current applications of Task Vectors have been limited to simple adaptation scenarios, leaving it unclear whether this paradigm can be extended to complex multi-step reasoning.

Through mathematical analysis, we observe that the effect of CoT can be formalized as a consistent shift in the internal activations of model (He et al., 2021), suggesting that extending task vectors to reasoning is both feasible and promising. In this work, we introduce CoT Vectors, task-general reasoning representations that adapt the task vector framework to CoT reasoning. CoT Vectors compactly encode the critical reasoning knowledge from a support set of (Question, CoT, Answer)

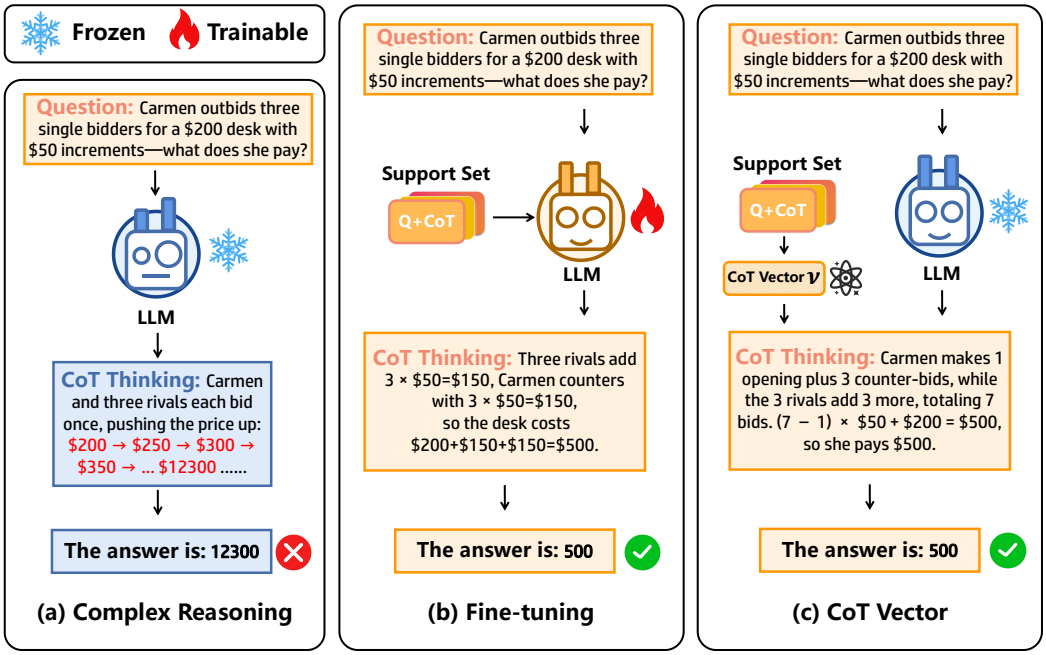

Figure 1: Overview of our approach. (a) Standard LLM may struggle to produce a correct reasoning chain for a complex problem. (b) Conventional fine-tuning adapt the model to such tasks by training on a support set, but requires updating model parameters, incurring high computational cost. (c) Our proposed CoT Vector leverages the support set to obtain a compact reasoning representation, which can be injected into the forward process of model to guide reasoning efficiently.

triplets, and can be directly injected into the forward process during inference. This approach not only enables portable reasoning enhancement without costly retraining or significant inference overhead, but also offers a new probe into how LLMs internalize and apply CoT.

Specifically, we begin with Extracted CoT Vectors, directly derived from activation differences between reasoning and non-reasoning traces, in line with the traditional task vector approach in NLP. Our studies reveal that Extracted CoT Vectors are effective but highly unstable across layers, with a striking U-shaped performance curve. This pattern suggests a systematic functional organization in LLMs, which we characterize as a three-stage reasoning process spanning perception, reasoning, and expression. Shallow and deep layers show relatively consistent representations, whereas middle layers contain highly variable, sample-specific structures that cause extracted vectors to fail.

To improve robustness, we introduce Learnable CoT Vectors, optimized via a teacher–student framework. Mathematically inspired by the additive shift formalization of CoT, our method distills a more robust and generalizable reasoning signal into a single, reusable vector. By actively learning reasoning knowledge rather than passively averaging activations, it achieves greater stability and stronger performance, overcoming the layer-wise volatility of extracted vectors. We conduct a comprehensive evaluation across various models and benchmarks, comparing extracted and learnable vectors against baselines. Our analyses further elucidate the sources of variability in CoT Vector effectiveness, highlighting how differences in acquisition mechanisms and model-specific latent structures shaped during pre-training impact reasoning performance. This perspective offers a valuable lens for understanding how LLMs organize and apply multi-step reasoning internally.

Overall, we summarize the main contributions of this work as follows:

- We introduce CoT Vectors, extending task vectors to multi-step reasoning. Experiments with traditional extracted vectors uncover their layer-wise instability, which in turn reveals a systematic three-stage reasoning process in LLMs.
- To address the limitations of extracted vectors, we propose novel Learnable CoT Vectors, optimized via a teacher–student framework, which provide more robust, stable, and task-general reasoning representations.

- We conduct a comprehensive evaluation across benchmarks and models. Using CoT Vectors as a probe, we analyze their variability from multiple perspectives, including latent space structure, information density, acquisition mechanisms, and model pre-training differences, thereby providing new insights into the mechanistic organization of reasoning in LLMs.

2 Related work

Enhancing Chain-of-Thought Reasoning in LLMs. Our work builds upon the foundational paradigm of CoT prompting (Wei et al., 2022), which significantly improves reasoning performance by eliciting step-by-step rationales. Numerous subsequent efforts have refined this idea through improved prompting strategies (Kojima et al., 2022; Khot et al., 2022), search-based reasoning frameworks (Yao et al., 2023), program-aided execution (Gao et al., 2023), and iterative self-refinement (Madaan et al., 2023). Another common strategy to improve performance is fine-tuning. Typical approaches include Supervised Fine-tuning (SFT) on (Question, CoT, Answer) triplets, as well as reinforcement learning techniques such as RLHF (Ouyang et al., 2022), PPO (Schulman et al., 2017), and GRPO (Rafailov et al., 2023). While effective, these techniques often demand substantial amounts of high-quality rationales, significant computational resources for training or alignment, making them prohibitively expensive relative to the incremental gains.

A parallel line of work seeks to compress explicit reasoning steps into fewer, or even invisible, latent representations, often referred to as Implicit CoT (Deng et al., 2023; 2024). These methods aim to internalize the reasoning process to improve efficiency and performance. Some approaches modify model architecture (Geiping et al., 2025) or use placeholder tokens in prompts (Pfau et al., 2024) to extend latent reasoning depth, effectively condensing multi-step thinking into compressed latent transitions that lead directly to answers. However, these methods typically require specialized model modifications and carefully engineered training regimes involving intensive post-training of model parameters (Hao et al., 2024; Shen et al., 2025; Cheng & Van Durme, 2024), which demand substantial resources while often delivering only modest gains. In contrast, our approach keeps the model architecture untouched: we distill reasoning into an external, plug-and-play CoT Vectors that can be acquired and applied rapidly for new tasks, combining flexibility with strong performance.

Task vectors. Task vectors (Ilharco et al., 2022) have emerged as a compact representation of task-specific knowledge, typically obtained either by computing weight differences between fine-tuned and pre-trained models (Ortiz-Jimenez et al., 2023a; Li et al., 2025), or by capturing activation differences induced by distinct input prompts (Liu et al., 2023; Todd et al., 2023; Hendel et al., 2023). These vectors not only enable parameter-efficient task transfer but have also been leveraged to provide preliminary insights into the internal mechanisms of LLMs (Yang et al., 2025). However, existing studies largely focus on relatively simple scenarios such as classifications or in-context learning, leaving the application of task vectors to complex multi-step reasoning underexplored. Several recent preliminary study (Azizi et al., 2025; Tang et al., 2025; Zhang & Viteri) have tentatively explored steering vectors in CoT, suggesting the feasibility of the paradigm. However, Azizi et al. (2025) focuses on compressing CoT chains, while Tang et al. (2025) aims to stimulate longer reasoning trajectories, both focusing on controlling CoT generation rather than capturing a task-general reasoning pattern. Meanwhile, Zhang & Viteri remains limited to conventional extraction techniques and offered only surface-level analysis. Our work moves substantially beyond this early exploration. Instead of relying solely on a basic extraction method, we introduce a novel learnable mechanism that actively optimizes CoT Vectors for better generalization and performance. Furthermore, our study provides a comprehensive analysis absent from prior work.

3 Methodology

We first formalize the concept of CoT Vectors in Section 3.1, deriving it from the mechanistic effect of CoT reasoning on the model's attention outputs. This formulation not only establishes the feasibility of our approach but also provides the guiding principle for the subsequent development. Building on this, we develop our two practical frameworks in Section 3.2 for acquiring CoT Vectors: a non-parametric extraction method and a novel parametric learning-based method, along with how to efficiently integrate the resulting vectors during inference to steer the model's reasoning.

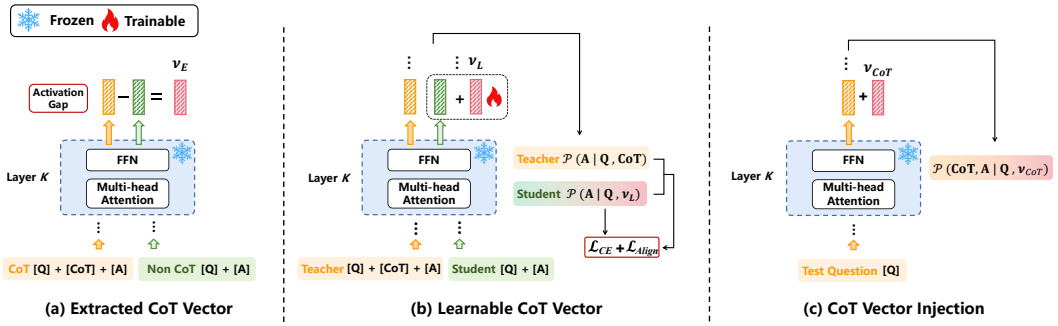

Figure 2: Methods for acquiring and applying CoT Vectors. (a) Extracted CoT Vector is obtained by recording the activation gap at the k-th layer between inputs with and without CoT. (b) Learnable CoT Vector is inserted into the k-th layer activations of a student sequence without CoT, and trained by aligning the student's final answer-token hidden states with those of a teacher sequence that includes CoT. (c) At test time, CoT Vector is added to the activations at the k-th layer during each forward pass of auto-regressive generation, guiding the reasoning process.

3.1 Conceptualization and Formalization of CoT Vector

The effectiveness of CoT prompting highlights that inserting an explicit reasoning sequence between the input question Q and the final answer A significantly enhances the model's reasoning capability. Our objective is to capture and quantify the effect of this reasoning process within the model.

In transformer-based language models, information flow is governed by self-attention. He et al. (2021) suggest that the effect of input prefixes can be understood as shifts in the attention outputs. In the context of CoT, the CoT sequence can be viewed as a specialized prefix that modulates the generation of answer tokens. When reasoning with CoT, the generation of an answer token depends not only on the question tokens but also on the intermediate CoT tokens. For each answer token $a \in A$, the single-head self-attention with and without the CoT sequence is denoted as $\mathrm{SA}(a, [K_Q, K_C, K_A], [V_Q, V_C, V_A])$ and $\mathrm{SA}(a, [K_Q, K_A], [V_Q, V_A])$ respectively, where the subscripts Q, C, and A refer to the question, CoT, and answer. We then derive the following equation:

$$\mathrm{SA}(a, [\boldsymbol{K}_Q, \boldsymbol{K}_C, \boldsymbol{K}_A], [\boldsymbol{V}_Q, \boldsymbol{V}_C, \boldsymbol{V}_A]) = \underbrace{\mathrm{SA}(a, [\boldsymbol{K}_Q, \boldsymbol{K}_A], [\boldsymbol{V}_Q, \boldsymbol{V}_A])}_{\text{Standard attention}} \tag{1}$$

$$+ \mu \cdot \underbrace{(\mathrm{SA}(a, [\boldsymbol{K}_C], [\boldsymbol{V}_C]) - \mathrm{SA}(a, [\boldsymbol{K}_Q, \boldsymbol{K}_A], [\boldsymbol{V}_Q, \boldsymbol{V}_A]))}_{\text{CoT shift}} \tag{2}$$

The introduction of CoT induces an additional term in the attention output, whose influence is quantified by a scalar coefficient μ (see the supplementary material for the full derivation). This additional contribution reflects precisely the knowledge injected by the CoT sequence. We formalize this effect as a CoT Shift, and denote the corresponding representation as the CoT Vector $\vec{v}_{\mathrm{CoT}}$. Accordingly, Equation 1 can be reformulated as

$$\mathrm{SA}(a, [\boldsymbol{K}_Q, \boldsymbol{K}_C, \boldsymbol{K}_A], [\boldsymbol{V}_Q, \boldsymbol{V}_C, \boldsymbol{V}_A]) = \mathrm{SA}(a, [\boldsymbol{K}_Q, \boldsymbol{K}_A], [\boldsymbol{V}_Q, \boldsymbol{V}_A]) + \mu \cdot \vec{v}_{\mathrm{CoT}} \tag{3}$$

The CoT Vector serves as a compact representation of the reasoning knowledge compressed from the CoT sequence. We hypothesize that, for tasks of the same type, the CoT Vectors derived from individual examples reside in a continuous semantic space. The centroid of this space, which we call the task-general CoT Vector, encodes the shared reasoning strategy for that task. For a new problem, injecting $\vec{v}_{\mathrm{CoT}}$ into the model's forward pass, which reversely applies the Equation 3, can effectively guide the model toward an appropriate reasoning trajectory and thereby improves task accuracy.

3.2 Task-General CoT Vectors

To leverage the advantages of task-general CoT Vectors, we first acquire them from a support set D. We propose two approaches for this acquisition: a traditional extraction-based method and a

novel parametric learning-based method. Once obtained, the task-general vector is injected into the model's forward pass during inference to steer its reasoning process.

3.2.1 EXTRACTED COT VECTORS

Given a support set of pairs (Q, A) and triplets (Q, CoT, A), we define the Extracted CoT Vector $\vec{v}_{\text{CoT}}^{(l)}$ as the difference in model activations of answer token a at layer l between these inputs, as shown in Figure 2 (a):

$$\vec{v}_{\text{CoT}}^{(l)} = \frac{1}{|A|} \sum_{a \in A} \left(\boldsymbol{\alpha}_{\text{CoT}}^{(l)}(a) - \boldsymbol{\alpha}_{\text{Non-CoT}}^{(l)}(a) \right) \tag{4}$$

where $\boldsymbol{\alpha}^{(l)}(a)$ is the hidden state of answer token a at layer l for the input and $|A|$ denotes the total number of answer tokens.

For each instance (Q_i, CoT_i, A_i), we compute an instance-specific CoT Vector $\vec{v}_{\text{CoT},i}$. A task-general Extracted CoT Vector $\vec{v}_E$ is then obtained by averaging across all N support instances:

$$\vec{v}_E = \frac{1}{N} \sum_{i=1}^{N} \vec{v}_{\text{CoT},i} \tag{5}$$

3.2.2 LEARNABLE COT VECTORS

Beyond extraction, we propose a novel parametric method that learns a task-general CoT Vector through gradient-based optimization. As depicted in Figure 2 (b), $\vec{v}_L$ is initialized as learnable parameters, added as a shift to the hidden state at a specific layer, and optimized on the support set $\mathcal{D}$ to encode generalized reasoning knowledge. We adopt a teacher–student framework. For each instance (Q_i, CoT_i, A_i), the teacher path processes the full triplet with frozen model parameters, providing the supervisory signal. The student path, in contrast, only processes (Q_i, A_i), while $\vec{v}_L$ is injected to compensate for the missing CoT sequence. Through this process, $\vec{v}_L$ distills essential reasoning signals from the teacher into a compact, transferable representation.

Throughout optimization, all original LLM parameters are kept frozen; only $\vec{v}_L$ are updated. The training objective combines two components: Prediction loss ($\mathcal{L}_{\text{CE}}$) is the cross-entropy loss on the student's predicted answer tokens, ensuring that the injected vector guides the model toward correct outputs. Representation alignment loss ($\mathcal{L}_{\text{Align}}$) is the mean KL loss between hidden states of teacher and student paths at the answer tokens, enforcing alignment of internal reasoning representations. The final objective is:

$$\mathcal{L} = \mathcal{L}_{\text{Align}} + \lambda \cdot \mathcal{L}_{\text{CE}} \tag{6}$$

where λ is a hyperparameter balancing the two terms.

3.2.3 INTEGRATING THE COT VECTOR TO REASONING

At inference time, given a new question, task-general CoT Vector $\vec{v}_{\text{CoT}}$ obtained from the support set at specific layer l, is then injected into the model at the same layer during every forward pass of CoT thinking, as shown in Figure 2 (c).

$$\tilde{\boldsymbol{\alpha}}^{(l)} = \boldsymbol{\alpha}^{(l)} + \mu^{(l)} \cdot \vec{v}_{\text{CoT}}^{(l)} \tag{7}$$

For Extracted CoT vectors, μ is an explicitly defined constant scaling factor. For Learnable CoT Vectors, however, μ is effectively internalized—since $\vec{v}_L$ is optimized end-to-end, the scaling factor is absorbed into the vector during training rather than being maintained as a separate constant. This integration incurs almost no additional overhead: it does not increase the input context length, and the runtime cost is negligible since the operation reduces to a simple vector addition. As a result, our approach provides an extremely efficient mechanism for enhancing reasoning in LLMs.

4 EXPERIMENTS

In this section, we first outline the setup and implementation details (Section 4.1). Next, we explore the adaptation of task vectors to multi-step reasoning in LLMs by introducing and analyzing CoT Vectors (Section 4.2). This investigation extends beyond mere performance evaluation, utilizing CoT Vectors as a tool to probe the underlying functional mechanisms of reasoning within LLMs.

4.1 Setup and Implementation Details

Models and Datasets. We conduct experiments on two open-source LLMs: Qwen2.5-Math-7B (Yang et al., 2024), a model fine-tuned for mathematical reasoning tasks, and LLaMA-3.1-8B-Instruct (Grattafiori et al., 2024), an instruction-tuned model with broad domain coverage. We evaluate our method on five benchmarks: GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2024), MMLU-Pro (Wang et al., 2024), CommonsenseQA (Talmor et al., 2019), and StrategyQA (Geva et al., 2021). GSM8K and MATH cover mathematical reasoning, MMLU-Pro spans diverse subject domains, and CommonsenseQA/StrategyQA focus on natural-language logical reasoning. For MATH, we divide the dataset into two subsets based on difficulty: MATH-Easy (levels 1–3) and MATH-Hard (levels 4–5), ensuring balanced difficulty in the support set. We sample 3,000 examples from GSM8K and the two MATH subsets as the support set. For MMLU-Pro, we use all 70 ground-truth annotated problems as the support set. For CommonsenseQA and StrategyQA, no official CoT traces are provided, so we rely on model-generated CoT sequences instead.

Implementation Details. We use standard zero-shot CoT prompting as baseline, where the model is instructed to "think step by step." For CoT Vectors, both extracted and learnable variants are implemented following the procedures described in Section 3. For extracted vectors, the scaling factor μ is fixed at 1.0. For learnable vectors, the loss balancing factor λ in Eq. 6 is set to 0.5. We select LoRA (Hu et al., 2022) as the representative parameter-efficient fine-tuning baseline, where LoRA adapters are trained on $(Q,\ \text{CoT},\ A)$ triplets. Following common practice, we apply LoRA to the projection matrices W_Q, W_K, W_V, and W_O in all attention layers.

4.2 Exploring CoT Vectors and the Mechanism of Reasoning

This section explores the adaptation of task vectors to multi-step reasoning via CoT Vectors. Our analysis begins with examining CoT Vectors obtained by the traditional extraction method, which prove effective but highly unstable performance across layers. This instability reveals consistent layer-wise patterns, uncovering a three-stage reasoning process in LLMs. Building upon these insights, we introduce Learnable CoT Vectors that achieve greater stability and stronger performance. We comprehensively evaluate both CoT Vectors against baseline and LoRA across two models and four benchmarks, and interpret performance variations through analyses of latent space structure, information density, acquisition mechanisms, and pre-training differences. Together, these studies not only extend the task vector framework to reasoning, but also reveal new perspectives on the underlying mechanisms of LLM reasoning.

4.2.1 The Three-Stage Reasoning Process

To assess whether task vectors can be extended to reasoning, we first explore the applicability of conventional extracted task vectors in CoT setting. Table 1 shows that Extracted CoT Vectors are indeed effective, improving over the baseline by an average of 2.4 and 1.1 points on the two models respectively and demonstrating the feasibility of task vectors in reasoning; however, their effectiveness is highly unstable across different layers (Figure 3 (a)) with the layer-wise average performance even falling below the baseline. Interestingly, this instability follows a non-random pattern. We observe a sawtooth U-shaped pattern: despite the fluctuations, the overall trend shows that performance enhancements when vectors are injected into either the shallow and deep layers, whereas injections into the middle layers yield minimal gains or even degrade performance. This contrasts with prior task vector research on simpler tasks (Todd et al., 2023; Hendel et al., 2023), where middle-layer interventions are typically most effective. This divergence suggests that the functional organization of complex multi-step reasoning in LLMs differs fundamentally from that of simpler tasks, highlighting the unique mechanisms involved in complex reasoning.

This layer-wise variability naturally leads us to hypothesize that the underlying reasoning process in LLMs may itself be structured in stages. Building on insights from prior work on layer specialization (Tenney et al., 2019; Chuang et al., 2023; Skean et al., 2025), we posit a three-stage organization of perception, reasoning, and expression. In this view, Shallow layers primarily perform basic feature extraction and semantic encoding, producing more linear and unified representations. Middle layers execute core reasoning process, leading to sample-specific, high-dimensional representations

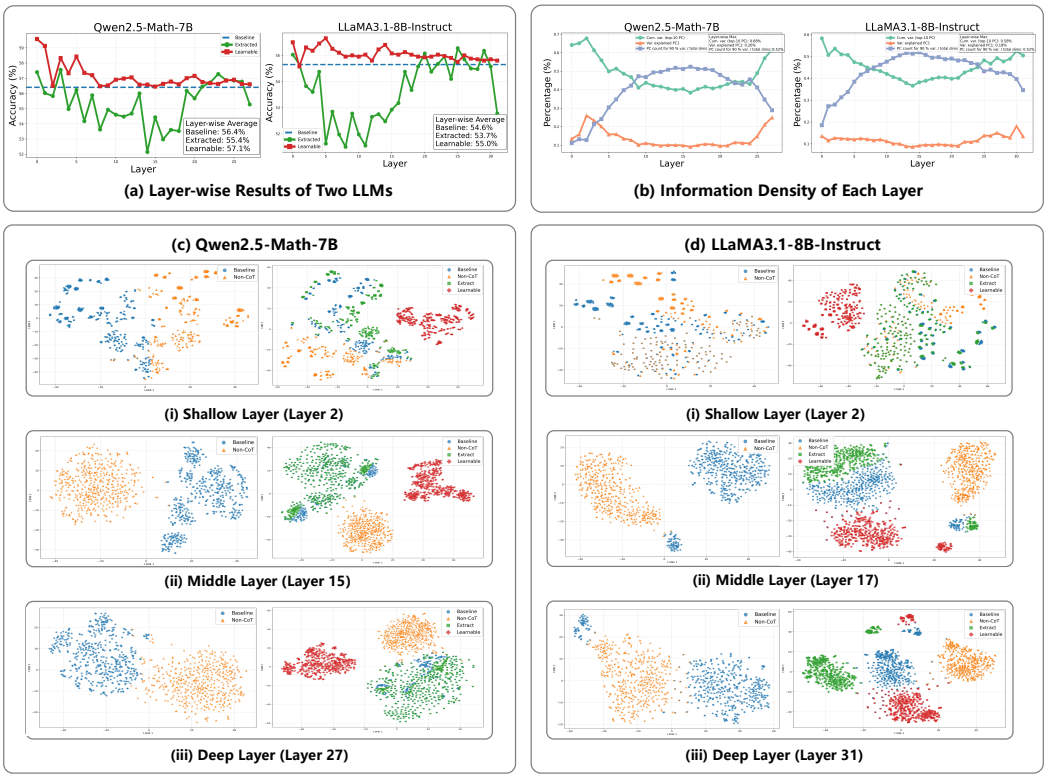

Figure 3: (a) Layer-wise performance of two LLMs with both extracted and Learnable CoT Vectors, averaged over four datasets. (b) Layer-wise information density curves of two LLMs, obtained via PCA on 500 sampled instances across four datasets. Abbreviations: PC = principal component; var. = variance; cum. = cumulative; dims = dimensions. (c–d) T-SNE visualizations of hidden states at shallow, middle, and deep layers on GSM8K (500 samples) of two LLMs. Left: sample distributions under non-CoT and baseline (with CoT) inputs. Right: same baseline with additional insertion of Extracted and Learnable CoT Vectors. Color scheme is consistent across (a, c, d): orange = non-CoT, blue = baseline, green = Extracted CoT Vector, red = Learnable CoT Vector.

with no dominant direction. Deep layers map internal reasoning states into surface-level linguistic outputs, where the representations again become more unified.

To test this hypothesis, we conduct an information density analysis via PCA on hidden states from 500 randomly sampled instances (Figure 3 (b)). We observe that mid-layers require significantly more principal components to explain the variance compared to shallow and deep layers, while the variance explained by the top components drops sharply. This indicates higher representational complexity and the absence of a dominant direction in mid-layers, ultimately delineating three distinct stages across shallow, middle, and deep layers. Visualizing the latent space with t-SNE (Figures 3 (c-d)) further reveal that middle-layer activations with CoT (baseline) form dispersed, input-specific clusters, reflecting a highly complex and non-linear structure that differs markedly from the non-CoT distribution. In contrast, shallow and deep layers exhibit more uniform activations, supporting that the middle layers serve as core stage for sample-specific reasoning. These findings explain why Extracted CoT Vectors fail in the middle layers: the mid-layer activations lack a coherent, task-general direction, making it difficult to extract a compact and reusable CoT Vector.

Further cross-layer transfer experiments support this conclusion. In Table 2, injecting mid-layer vectors into shallow layers degrades performance, whereas shallow-layer vectors improve performance when injected into middle layers. This indicates that the failure of mid-layer injection stems not from location, but from the intrinsically sample-specific and non-generalizable nature of mid-layer representations, which are ill-suited for capturing a compact, task-wide reasoning direction.

Table 1: Comprehensive evaluation results. MATH-E = MATH-Easy, MATH-H = MATH-Hard, MMLU-P = MMLU-Pro, CSQA = CommonsenseQA, SQA = StrategyQA. Reported CoT Vector results correspond to the best injection layer selected from layer-wise evaluation. We note that extraction-based vectors are particularly dependent on this choice, whereas learnable vectors maintain more consistent performance across layers.

Model	Method	#Params	GSM8K	MATH-E	MATH-H	MMLU-P	CSQA	SQA	Avg.
Qwen2.5-Math-7B	Baseline	—	74.6	69.9	47.9	33.2	53.8	23.7	50.5
	Extracted	—	78.2	**72.0**	49.7	**35.3**	57.5	29.1	53.6
	Learnable	3.6K(×1.0)	**83.5**	71.9	**50.9**	35.1	**58.2**	**31.2**	**55.1**
	LoRA	10.0M(×2777.8)	79.0	70.4	48.2	33.8	58.0	**31.2**	53.4
LLaMA3.1-8B-Instruct	Baseline	—	77.4	62.0	34.6	44.6	72.7	60.8	58.7
	Extracted	—	**78.6**	63.2	35.7	45.5	73.2	64.3	60.1
	Learnable	4.2K(×1.0)	78.2	**63.8**	**36.4**	**46.2**	**73.7**	**65.0**	**60.6**
	LoRA	13.6M(×3238.0)	**78.6**	63.5	36.3	45.5	73.6	64.8	60.4

Table 2: Cross-layer CoT Vector transfer results on Qwen-GSM8K. Performance when injecting a CoT Vector extracted from a Source Layer (column) into a different Target Layer (row). The diagonal shows baseline performance (source = target). Δ indicates the absolute change from the target layer's baseline. Green arrows (↑) indicate improvement, red arrows (↓) indicate degradation.

	Source: Shallow (L6)		Source: Middle (L14)	
Target Layer	**Accuracy**	**Δ**	**Accuracy**	**Δ**
Shallow (Layer 6)	78.2	—	63.8	↓14.4
Middle (Layer 14)	75.3	↑9.0	66.3	—

4.2.2 Learnable CoT Vectors

Learnable vs. Extracted CoT Vectors. Beyond the conventional extraction-based approach, we further introduce novel Learnable CoT Vectors, optimized via a teacher-student architecture to distill generalizable reasoning patterns. Experimental results reveal that the Learnable CoT Vector demonstrates two clear advantages over its extracted counterpart: (i) higher overall performance across benchmarks (Table 1), and (ii) significantly greater stability across layers with higher layer-wise average accuracy (Figure 3 (a)). Unlike the sawtooth U-shaped curve observed for extracted vectors, where gains concentrate in shallow and deep layers but diminish in the middle and with strong fluctuations, the learnable vector peaks at the first layers and maintains a consistent plateau across all subsequent layers. Consequently, while extracted vectors show noticeable drops at mid-layers compared to baseline, learnable vectors consistently provide improvements across nearly all layers.

We attribute this divergence to the fundamental nature of each vector type. The extracted vector is a descriptive statistic, passively recording the average activation difference between CoT and non-CoT forward passes. Its efficacy is thus constrained by the representational properties of the source layer: strong when representations have clear dominant directions (e.g., shallow and deep layers), but fragile when such structure is absent. As a consequence, it not only induces relatively mild shifts in latent space (Figure 3 (c-d)) but also retains sample-specific noise, leading to sharp volatility where even adjacent layers at similar depths behave inconsistently.

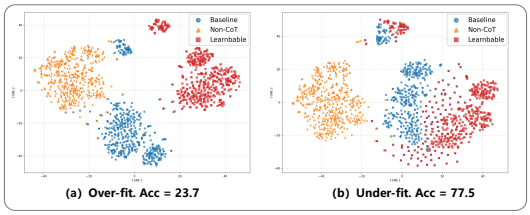

Figure 4: T-SNE visualization of over-fit and under-fit Learnable CoT Vectors (Layer 30 of LLaMA on GSM8K).

In contrast, learnable vectors are optimized via gradient descent to mimic the teacher model's reasoning. This results in a more directional and aggressive shift in the latent space (Figure 3 (c-d)), enabling it to overcome representational limitations of individual layers and avoid being intervened

by sample-specific noise. Consequently, the learnable vector achieves stronger and more stable performance across layers.

These differences have important practical implications. Extracted vectors suffer from high instability, with the optimal injection layer varying across tasks and models (see Appendix for more details). In real-world deployment, where ground truth is unavailable, such unpredictability severely limits their usability. Learnable CoT Vectors, however, produce consistent gains across all layers, with their strongest performance consistently emerging in the shallowest layers—often at the very first layer. This stability permits simple and robust application: even on unseen tasks, injecting the vector at the first layer suffices to achieve near-optimal performance.

However, the aggressive steering of Learnable CoT Vectors also brings risks. As visualized in Figure 4, vectors applied to middle or deep layers are prone to overfitting, over-steering the latent space and collapsing diverse reasoning paths, which significantly degrades accuracy. This fragility stems from the representational nature of these layers that mid-layer activations are heterogeneous and sample-specific, while deep layers are closely tied to surface outputs, where even small perturbations can destabilize generation. To mitigate this, we employ early stopping or reduced learning rates, which produce mildly under-fitted vectors that still provide modest gains without catastrophic collapse. These findings reinforce our earlier conclusion that shallow layers are the most suitable for Learnable CoT Vectors, while mid and deep layers are less amenable to strong external guidance.

Learnable CoT Vectors vs. LoRA. From parameter-efficiency perspective, our Learnable CoT Vector demonstrates advantages over LoRA fine-tuning. As illustrated in Table 1, it outperforms LoRA on most datasets while requiring orders of magnitude fewer trainable parameters. We attribute this to the fact that instruction-tuned LLMs already possess strong CoT priors, leaving limited room for LoRA to improve. In contrast, our approach adds an external guidance signal that efficiently steers the model's latent reasoning without altering the model's existing functional structure.

4.2.3 Model Differences

As shown in Table 1, the effectiveness of CoT Vectors varies across models: Qwen benefits more consistently and substantially from CoT Vector injection compared to LLaMA. For example, averaged across benchmarks, Qwen gains up to 4 points over the baseline, whereas LLaMA yields a more modest improvement of 1.5 points with Learnable CoT Vectors. We trace this discrepancy to differences in the latent space structures of the two models throughout the three-stage reasoning process. In Figure 3 (b), Qwen exhibits a more distinct three-stage reasoning pattern than LLaMA. Notably, its top principal components explain more variance than those of LLaMA, suggesting a lower information density and a more structured latent space with clearer principal directions. This facilitates both extraction and optimization in capturing high-quality, task-general signals.

We conjecture that this structural disparity stems from differences in training data and procedures. Qwen has undergone more domain-focused and standardized fine-tuning, whereas LLaMA has been trained on broader and less curated corpora. As a result, Qwen demonstrate a more distinct functional separation of layers. This structural clarity allows CoT Vectors to more easily capture task-general reasoning directions. In summary, the performance gap highlights that the efficacy of CoT Vectors is influenced by the inherent properties of the model's representations. Models with more structured latent spaces provide a more fertile ground for the CoT Vector intervention.

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 3: 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.

4.2.4 CROSS-DATASET AND CROSS-MODEL TRANSFERABILITY

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 3, 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 (74.6 $\rightarrow$ 77.5).

Cross-Dataset Transfer. Results in Table 3 further demonstrate transferability across datasets. 1) In-domain: CoT Vectors obtained from the GSM8K dataset effectively enhance performance on the MATH dataset (47.9 $\rightarrow$ 48.6). 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 (47.9 $\rightarrow$ 48.5). 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.

4.2.5 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 4, 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. This highlights the strong data efficiency of our approach. As the support set grows, Learnable CoT Vector also demonstrates greater potential for performance improvement compared to LoRA. These phenomena all indicate that our Learnable CoT Vectors provide a more effective and scalable mechanism for enhancing reasoning performance than LoRA across diverse data conditions.

Table 4: Performance Comparison with Different Training Sample Sizes on Qwen-GSM8K.

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

Overall, our results confirm that CoT Vectors are a highly efficient and effective means of enhancing reasoning capabilities. However, directly applying traditional extraction methods to CoT still presents challenges, particularly related to the internal mechanisms of LLM reasoning. Our newly introduced Learnable CoT Vectors offer significant advantages in this domain. Due to space limitations, further ablation studies and robustness analyses are provided in the supplementary material.

5 CONCLUSION

We have presented CoT Vectors, extending the task vector paradigm to multi-step reasoning in LLMs. Our analyses uncover a consistent three-stage reasoning process and show that the newly introduced Learnable CoT Vectors provide stronger and more stable gains than the traditional extraction-based approach, while also offering multiple perspectives on why their effectiveness differs. These results demonstrate both the practical utility of CoT Vectors and their value as a probe into the mechanisms and organization of multi-step reasoning in LLMs. However, performance variability in intermediate layers highlights structural limitations, suggesting that task-level vectors may not fully capture intra-task diversity. Future work could explore finer-grained or adaptive vectorization strategies to improve robustness and generalization.

Ethics Statement

This work adheres to the ICLR Code of Ethics. Our study does not involve human subjects, personally identifiable information, or proprietary data. All datasets used, including GSM8K, MATH, and MMLU-Pro, are publicly available. The proposed method, CoT Vectors, is a parameter-efficient technique for steering the reasoning process of pre-trained large language models. It does not introduce any new capabilities that could cause harm, nor does it enable misuse beyond the standard capabilities of existing large language models. We are not aware of any potential risks related to bias, fairness, or security that arise specifically from the method proposed. However, we acknowledge that the effectiveness and potential output of CoT Vectors are dependent on the base model and the support set data; as such, they may reflect or amplify biases present in these sources. No conflicts of interest, legal compliance issues, or sponsorship-related influences are present in this work.

Reproducibility Statement

We have taken multiple steps to ensure the reproducibility of our work. All datasets used in our experiments are publicly available and properly cited in the main text and appendix. Training configurations, including hyperparameters, optimizers, and evaluation settings, are described in detail in Section 4.1 and Appendix A.3. Theoretical claims, including the formalization of the CoT shift, are formally derived in Section 3.1 and Appendix A.2. Experimental results include multiple models, reasoning benchmarks, and various ablations to validate robustness in Section 4 and Appendix A.4-A.5. We will release the full source code and pre-trained vectors upon publication to further support reproducibility.

References

Seyedarmin Azizi, Erfan Baghaei Potraghloo, and Massoud Pedram. Activation steering for chain-of-thought compression. *arXiv preprint arXiv:2507.04742*, 2025.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Jeffrey Cheng and Benjamin Van Durme. Compressed chain of thought: Efficient reasoning through dense representations. *arXiv preprint arXiv:2412.13171*, 2024.

Yung-Sung Chuang, Yujia Xie, Hongyin Luo, Yoon Kim, James Glass, and Pengcheng He. Dola: Decoding by contrasting layers improves factuality in large language models. *arXiv preprint arXiv:2309.03883*, 2023.

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

Yuntian Deng, Kiran Prasad, Roland Fernandez, Paul Smolensky, Vishrav Chaudhary, and Stuart Shieber. Implicit chain of thought reasoning via knowledge distillation. *arXiv preprint arXiv:2311.01460*, 2023.

Yuntian Deng, Yejin Choi, and Stuart Shieber. From explicit cot to implicit cot: Learning to internalize cot step by step. *arXiv preprint arXiv:2405.14838*, 2024.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models. In *International Conference on Machine Learning*, pp. 10764–10799. PMLR, 2023.

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. *arXiv preprint arXiv:2502.05171*, 2025.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361, 2021.

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

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*, 2024.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. Towards a unified view of parameter-efficient transfer learning. *arXiv preprint arXiv:2110.04366*, 2021.

Roee Hendel, Mor Geva, and Amir Globerson. In-context learning creates task vectors. *arXiv preprint arXiv:2310.15916*, 2023.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset, 2021. *URL https://arxiv. org/abs/2103.03874*, 2, 2024.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

Jie Huang and Kevin Chen-Chuan Chang. Towards reasoning in large language models: A survey. *arXiv preprint arXiv:2212.10403*, 2022.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*, 2022.

Shima Imani, Liang Du, and Harsh Shrivastava. Mathprompter: Mathematical reasoning using large language models. *arXiv preprint arXiv:2303.05398*, 2023.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. Decomposed prompting: A modular approach for solving complex tasks. *arXiv preprint arXiv:2210.02406*, 2022.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.

Hongkang Li, Yihua Zhang, Shuai Zhang, Meng Wang, Sijia Liu, and Pin-Yu Chen. When is task vector provably effective for model editing? a generalization analysis of nonlinear transformers. *arXiv preprint arXiv:2504.10957*, 2025.

Sheng Liu, Haotian Ye, Lei Xing, and James Zou. In-context vectors: Making in context learning more effective and controllable through latent space steering. *arXiv preprint arXiv:2311.06668*, 2023.

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

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594, 2023.

Guillermo Ortiz-Jimenez, Alessandro Favero, and Pascal Frossard. Task arithmetic in the tangent space: Improved editing of pre-trained models. *Advances in Neural Information Processing Systems*, 36:66727–66754, 2023a.

Guillermo Ortiz-Jimenez, Alessandro Favero, and Pascal Frossard. Task arithmetic in the tangent space: Improved editing of pre-trained models. *Advances in Neural Information Processing Systems*, 36:66727–66754, 2023b.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744, 2022.

Jacob Pfau, William Merrill, and Samuel R Bowman. Let's think dot by dot: Hidden computation in transformer language models. *arXiv preprint arXiv:2404.15758*, 2024.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. Codi: Compressing chain-of-thought into continuous space via self-distillation. *arXiv preprint arXiv:2502.21074*, 2025.

Oscar Skean, Md Rifat Arefin, Dan Zhao, Niket Patel, Jalal Naghiyev, Yann LeCun, and Ravid Shwartz-Ziv. Layer by layer: Uncovering hidden representations in language models. *arXiv preprint arXiv:2502.02013*, 2025.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4149–4158, 2019.

Xinyu Tang, Xiaolei Wang, Zhihao Lv, Yingqian Min, Wayne Xin Zhao, Binbin Hu, Ziqi Liu, and Zhiqiang Zhang. Unlocking general long chain-of-thought reasoning capabilities of large language models via representation engineering. *arXiv preprint arXiv:2503.11314*, 2025.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. Bert rediscovers the classical nlp pipeline. *arXiv preprint arXiv:1905.05950*, 2019.

Eric Todd, Millicent L Li, Arnab Sen Sharma, Aaron Mueller, Byron C Wallace, and David Bau. Function vectors in large language models. *arXiv preprint arXiv:2310.15213*, 2023.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37:95266–95290, 2024.

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

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, et al. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.

Haolin Yang, Hakaze Cho, Yiqiao Zhong, and Naoya Inoue. Unifying attention heads and task vectors via hidden state geometry in in-context learning. *arXiv preprint arXiv:2505.18752*, 2025.

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

Jason Zhang and Scott W Viteri. Uncovering latent chain of thought vectors in large language models. In *Workshop on Neural Network Weights as a New Data Modality*.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 1(2), 2023.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.