

DeCoVec: Building Decoding Space based Task Vector for Large Language Models via In-Context Learning

Feiyang Li, Yile Wang*

College of Computer Science and Software Engineering, Shenzhen University

Abstract

Task vectors, representing directions in model or activation spaces that encode task-specific behaviors, have emerged as a promising tool for steering large language models (LLMs). However, existing approaches typically require fine-tuning or invasive manipulation of internal states, limiting their flexibility and scalability. We propose DECOVEC (Decoding Space based Task Vector), a training-free and non-invasive framework that constructs task vectors directly in the *decoding space* by leveraging in-context learning (ICL). Specifically, DECOVEC captures the task essence as the difference between the output logit distributions of few-shot and zero-shot prompts, then steers generation by injecting this vector into the decoding process. Experiments across seven LLMs (0.5B–9B) on TruthfulQA, Math-500, and AQUA-RAT show that DECOVEC consistently outperforms standard few-shot baselines, with gains up to +5.50 average accuracy. Further analysis demonstrates that DECOVEC effectively suppresses generation degeneration and logical flaws while exhibiting strong robustness to demonstration ordering, all without incurring additional input token costs. Our method offers a training-free and non-invasive solution for LLM steering without requiring weight updates or auxiliary models. The code is released at <https://github.com/szu-tera/DeCoVec.git>.

1 Introduction

Task vectors (Ilharco et al., 2023; Liu et al., 2024b; Yang et al., 2025), representing directions in the high-dimensional space that encode specific skills or behaviors, have garnered significant attention with the development of large language models (LLMs; Brown et al., 2020b; OpenAI, 2022, 2023). By adjusting model parameters or internal states along these vectors, one can effectively steer the model’s output to adapt to specific tasks. This concept not only serves as a cost-effective alternative

*Corresponding to wangyile@szu.edu.cn.

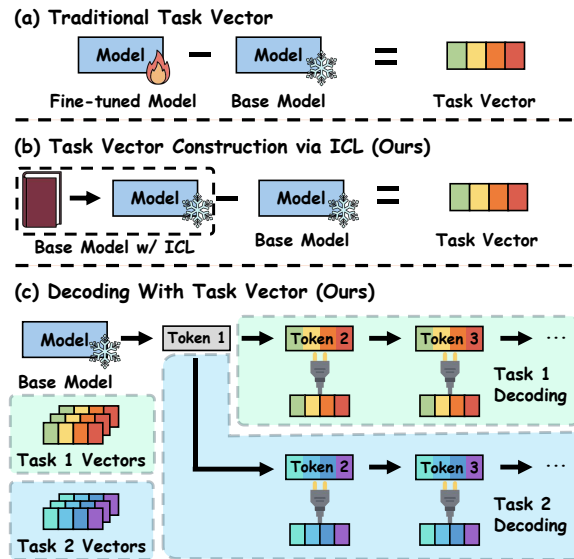


Figure 1: Schematic of traditional methods vs. ours. (a) Traditional task vector in model space. (b)-(c) Our DECOVEC via in-context learning in decoding space.

to parameter-efficient fine-tuning (Houlsby et al., 2019; Li and Liang, 2021; Hu et al., 2022) but also offers a unique lens for interpreting the internal mechanisms of LLMs and achieving controllable generation (Hendel et al., 2023; Yang et al., 2025).

Current task vector methods primarily operate in model spaces, either in model weights or activations, as shown in Figure 1(a). These approaches typically require a full fine-tuning process per task to derive the vector (Ilharco et al., 2023; Huang et al., 2024), or involve invasive intervention into the model’s internal hidden states through complex optimization or auxiliary training (Hendel et al., 2023; Liu et al., 2024b). Their high computational cost and structural invasiveness limit flexibility and scalability in real-world deployment.

On the other hand, decoding-based strategies are progressively being utilized to steer LLMs. For example, modulating output probability distributions can effectively improve generation quality (Li

Types of Task Vector	Non-Invasive	Training-Free	Operational Space	Vector Representation
Arithmetical Task Vector (Ilharco et al., 2023)				
Chat Vector (Huang et al., 2024)	✗	✗	Model Weights	$\Delta\theta$
Emotional Vector (Kalyan et al., 2024)				
Contrastive Vector (Fierro and Roger, 2025)				
Rule Task Vector (Hendel et al., 2023)				
Function Vector (Todd et al., 2024)	✗	✓	Activation States	Δh
In-Context Vector (Liu et al., 2024b)				
Visual Task Vector (Hojel et al., 2024)				
DECOVEC (Ours)	✓	✓	Decoding Logits	Δz

Table 1: **Comparison of task vector related works.** $\Delta\theta$: difference in model weights. Δh : difference in internal hidden states. Δz : difference in output logits. Previous work mainly uses task vectors for LLMs interpretability and steering. Our non-invasive and training-free method operates in the decoding space for improving task performance.

et al., 2023; Chuang et al., 2024) or adapt models to downstream tasks (Liu et al., 2024a; Hu et al., 2025; Shi et al., 2024), showing that the decoding space can serve as a **semantic-rich and direct** interface for controlling model behaviors without accessing internal parameters.

In this work, we propose DECOVEC, a training-free and non-invasive framework for constructing task vectors directly in the decoding space. As shown in Figure 1(b)-(c), we leverage the emergent in-context learning (ICL; Brown et al., 2020a) capability of LLMs, capturing task essence by contrasting the logit distributions of zero-shot and few-shot contexts without gradient updates. By defining the task vector in the output logit space rather than internal layers, DECOVEC steers generation in a transparent and controllable manner. This design makes DECOVEC a lightweight, plug-and-play solution that decouples task guidance from the internal model architecture.

We extensively evaluate a diverse suite of 0.5B~9B LLMs (Qwen Team, 2024; Young et al., 2024; Touvron et al., 2023; Grattafiori et al., 2024; Gemma Team, 2024) on both knowledge-intensive TruthfulQA (Lin et al., 2022) and reasoning tasks including Math-500 (Lightman et al., 2023) and AQUA-RAT (Ling et al., 2017). Results show that DECOVEC consistently achieves performance gains, outperforming few-shot baselines with different demonstration selection strategies. In-depth analysis reveals that the extracted vectors encode high-level task semantics rather than surface-level patterns. Error analysis on mathematical reasoning tasks indicates that DECOVEC effectively suppresses logical flaws and generation degeneration, steering models toward more rigorous problem-solving strategies. These findings offer new in-

sights for efficiently steering LLM behavior.

2 Related Work

Task Vector. The concept of task vector aligns with the broader goal of parameter-efficient fine-tuning, offering a cost-effective way to interpret or adapt models by modifying weights via algebraic operations or injecting low-rank modules (Hu et al., 2022). Current methods can be categorized into two types according to the operational space:

1) *Weight-space based Task Vector.* This type of method directly manipulates model parameters to alter behavior. Ilharco et al. (2023) introduces the arithmetic task vector, derived by subtracting the weights of a pre-trained model from a fine-tuned one to isolate task properties. Subsequent works have extended this foundational paradigm to various specialized domains. For example, the chat vector (Huang et al., 2024) transfers conversational capabilities to base models. The emotional vector (Kalyan et al., 2024) is constructed to steer the desired sentiment of generation. The contrastive vector (Fierro and Roger, 2025) isolates specific behavioral patterns by contrasting weights between models tuned to opposing objectives.

2) *Activation-space based Task Vector.* Shifting from static parameters to dynamic representations, these types of methods intervene in internal hidden states during inference. Hendel et al. (2023) identifies rule task vectors demonstrating that compressing demonstrations into a single vector within the activation space can replicate specific functions without explicit fine-tuning. Liu et al. (2024b) proposes the in-context vector to steer model behavior by shifting hidden states. Other explorations include the function vector (Todd et al., 2024), which locates causal attention heads for specific tasks, and

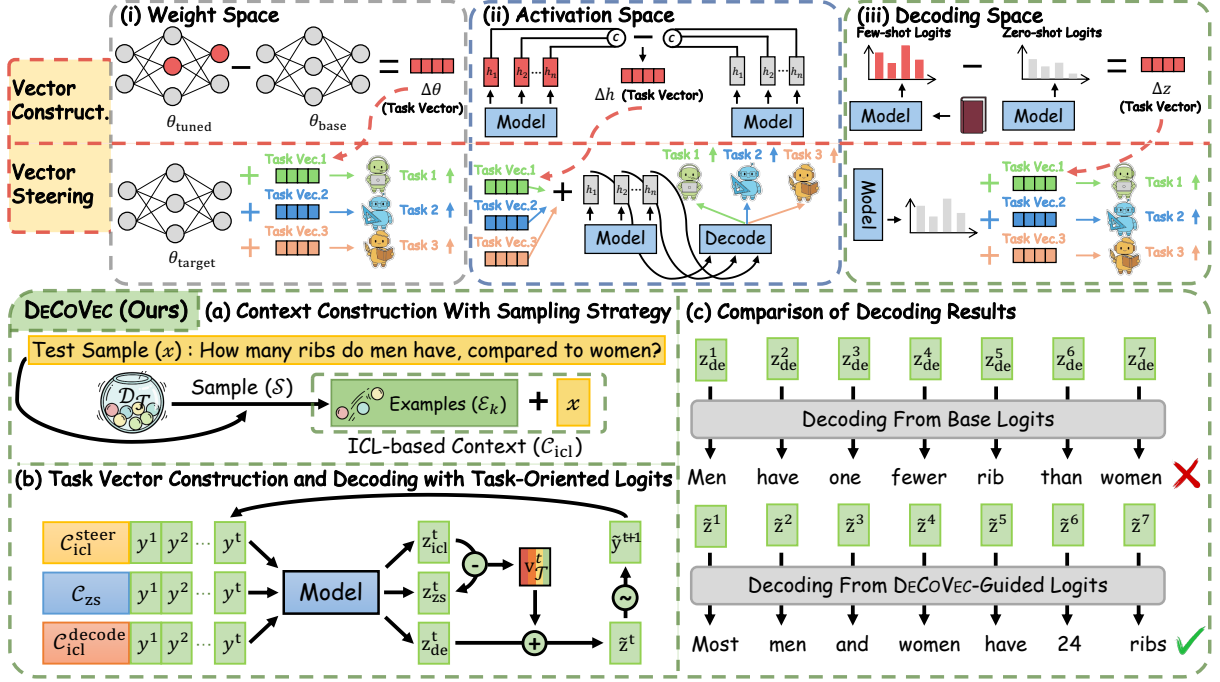


Figure 2: **Overview of task vector methods across spaces and our DECoVEC.** **Top: Comparison of task vector construction and steering.** (i) Weight space vector. (ii) Activation space vector. (iii) Our proposed DECoVEC in the decoding space. **Bottom: Illustration of DECoVEC pipeline.** (a) Context construction. (b) Task vector building and steering. (c) Resulting improvement in decoding output (correct ✓ vs. incorrect ✗).

the visual task vector (Hojel et al., 2024) specifically adapted for multimodal contexts.

In-Context Learning of LLMs. ICL empowers LLMs to perform novel tasks by conditioning on input-output examples without explicit parameter updates (Brown et al., 2020a). Theoretical frameworks suggest that ICL implicitly implements optimization algorithms like gradient descent (Dai et al., 2023; Bai et al., 2023) or serves as semantic anchors from an information flow perspective (Wang et al., 2023a). Hendel et al. (2023) interprets the internal mechanism of ICL through identifying rule task vectors, while our work directly aims at constructing and steering decoding-based task vectors to improve task-specific performance.

Decoding Strategies for LLMs. The decoding space serves as a direct and flexible interface for steering model behaviors. Contrastive decoding (Li et al., 2023) manipulates this space to mitigate degenerative behaviors by contrasting expert and amateur logits. Subsequent research extended this paradigm to suppress hallucinations (Chuang et al., 2024) or noise (Zhou et al., 2025). More recently, efforts have shifted towards modeling downstream *task semantics* rather than just quality control. Proxy-tuning (Liu et al., 2024a) and distribution-aligned decoding (Hu et al., 2025) adjust output dis-

tributions to approximate fine-tuned models. Similarly, Peng et al. (2025) enhanced input-label mappings via contrastive measures. These methods often rely on auxiliary models, gradients, or complex contrastive setups. In contrast, our approach extracts task vectors purely from the decoding space of a frozen model via ICL, offering a lightweight and non-invasive alternative.

We summarize the main related works and compare them with ours in Table 1.

3 Method

We first formalize the problem setting and the construction of task-specific contexts (§3.1). Then, we derive the definition of the task vector in the decoding space by contrasting the distributions of few-shot and zero-shot settings (§3.2). Finally, we present the steering mechanism to guide the process of decoding for downstream tasks (§3.3).

3.1 Formalization of Tasks and Contexts

Tasks. Considering a task \mathcal{T} with input-output training pairs $\mathcal{D}_{\mathcal{T}} = \{(x_i, y_i)\}$, we use a task-specific instruction template (or prompt schema) $\mathcal{I}_{\mathcal{T}}(\cdot)$ as the input for the LLM \mathcal{M}_{θ} parameterized by θ . Given a query input $x_i \in \mathcal{D}_{\mathcal{T}}$, the goal is to generate the target sequence $y_i = (y_i^1, y_i^2, \dots, y_i^L)$.

At each time step t , the model \mathcal{M}_θ maps the context to a probability distribution over the vocabulary \mathcal{V} via a softmax function applied to the output logits by model function $f_\theta(\cdot) \in \mathbb{R}^{|\mathcal{V}|}$.

Contexts. To leverage contexts for building task vector, we consider two distinct contextual representations of the same task \mathcal{T} :

1) *Zero-Shot Context (Task-Agnostic State)*. We construct the zero-shot input \mathcal{C}_{zs} by wrapping the query x solely with the task instruction without explicit task demonstrations, representing the model’s intrinsic instruction-following capability:

$$\mathcal{C}_{zs} = \mathcal{I}_{\mathcal{T}}(\emptyset, x), \quad (1)$$

where \emptyset indicates an empty set of demonstrations.

2) *Few-Shot ICL-based Context (Task-Aware State)*. To activate the model’s emergent ability for task \mathcal{T} through ICL, we employ a demonstration sampler \mathcal{S} to retrieve a set of k exemplars \mathcal{E}_k from dataset $\mathcal{D}_{\mathcal{T}}$ conditioned on the query input x :

$$\mathcal{E}_k = \mathcal{S}(x, \mathcal{D}_{\mathcal{T}}, k), \quad (2)$$

where \mathcal{S} represents a generic sampling strategy, which can range from random selection to distinct retrieval methods such as KATE (Zhang et al., 2023) or BM25 (Robertson and Zaragoza, 2009). The few-shot ICL-based \mathcal{C}_{icl} is then constructed by organizing these demonstrations alongside the query within the task template:

$$\mathcal{C}_{icl} = \mathcal{I}_{\mathcal{T}}(\mathcal{E}_k, x), \quad (3)$$

where \mathcal{C}_{icl} contains input-label mappings and semantic patterns of task \mathcal{T} , which are absent in \mathcal{C}_{zs} .

Two Types of ICL Contexts. By applying a specific sample strategy \mathcal{S} , we construct \mathcal{C}_{icl} in two scenarios. We first sample the base context during decoding, denoted as $\mathcal{C}_{icl}^{\text{decode}}$ in the following Eq. 7, serving as a few-shot baseline. Then we sample the context from training pairs $\mathcal{D}_{\mathcal{T}}$ for the task vectors used in steering, which we denote as $\mathcal{C}_{icl}^{\text{steer}}$ and is used in the following Eq. 5.

3.2 Building Task Vector in Decoding Space

As illustrated in the top row of Figure 2, unlike prior methods that extract task vectors from model weights (i) or internal activations (ii), we propose to construct the task vector directly within the *decoding space* (iii). We hypothesize that the difference between the logits generated in the few-shot ICL-based context $\mathcal{C}_{icl}^{\text{steer}}$ and the zero-shot context \mathcal{C}_{zs}

encodes the distilled task-level features that can help construct the task vector $\mathbf{v}_{\mathcal{T}}$.

Given the prefix $y^{1:t}$ at decoding step t , we compute the logit vector $\mathbf{z}^t \in \mathbb{R}^{|\mathcal{V}|}$ using zero-shot context and sampled ICL-based context, respectively:

$$\mathbf{z}_{zs}^t = f_\theta(\mathcal{C}_{zs}, y^{1:t}), \quad (4)$$

$$\mathbf{z}_{icl}^t = f_\theta(\mathcal{C}_{icl}^{\text{steer}}, y^{1:t}). \quad (5)$$

Given the above two logits at generation step t , we define the decoding-space in-context learning guided task vector (i.e., our DECOVEC) $\mathbf{v}_{\mathcal{T}}^t$ as the difference of the two:

$$\mathbf{v}_{\mathcal{T}}^t = \mathbf{z}_{icl}^t - \mathbf{z}_{zs}^t. \quad (6)$$

This operation is depicted in Figure 2(b).

3.3 Decoding with Task Oriented Logits

We employ our DECOVEC $\mathbf{v}_{\mathcal{T}}^t$ to directly steer the generation trajectory of the model. The core intuition is to inject the distilled task-specific signal into the inference process.

During decoding, the steering operates token-by-token online. We first compute the intermediate logit \mathbf{z}_{de}^t according to the base context $\mathcal{C}_{icl}^{\text{decode}}$. Because both \mathbf{z}_{de}^t and the task vector $\mathbf{v}_{\mathcal{T}}^t$ condition on the identical generated prefix $y^{1:t}$, their vocabulary distributions remain strictly aligned at each step, inherently avoiding sequence-length mismatches across different contexts. We then compute the final task-oriented logits $\tilde{\mathbf{z}}^t$ by injecting DECOVEC $\mathbf{v}_{\mathcal{T}}^t$:

$$\mathbf{z}_{de}^t = f_\theta(\mathcal{C}_{icl}^{\text{decode}}, y^{1:t}), \quad (7)$$

$$\tilde{\mathbf{z}}^t = \mathbf{z}_{de}^t + \lambda \cdot \mathbf{v}_{\mathcal{T}}^t, \quad (8)$$

where the scaling factor $\lambda > 0$ controls the task signal injected into the decoding process. This steering operation, which is the core mechanism of our method, is illustrated in Figure 2(b).

This formulation allows us to explicitly control the impact of the extracted task information. The base logits \mathbf{z}_{de}^t serve as a semantic anchor, ensuring that the generation remains grounded in the logic of the current input while benefiting from the directional steering provided by $\mathbf{v}_{\mathcal{T}}^t$. Increasing λ directly amplifies the task-specific signals derived from the difference between contexts, thereby encouraging the model to adhere more strictly to the intended task behaviors. We analyze the impact of scaling factor λ in Section 5.2.

Methods	TruthfulQA			Avg. Δ	Math-500	AQUA-RAT	Avg. Δ	
	MC1	MC2	MC3		Acc.	Acc.		
Qwen2-0.5B	Zero-Shot	25.18	40.78	19.55	-	1.17	20.87	-
	Few-Shot (Rand. Std.)	32.35 (\uparrow 1.97)	47.54 (\uparrow 2.51)	23.46 (\uparrow 1.38)	\uparrow 1.95	8.59 (\downarrow 0.39)	23.23 (\uparrow 5.12)	\uparrow 2.37
	Few-Shot (Rand. Ext.)	32.21 (\uparrow 0.70)	48.01 (\uparrow 2.55)	23.73 (\uparrow 1.42)	\uparrow 1.56	7.03 (\uparrow 0.78)	21.65 (\uparrow 3.55)	\uparrow 2.17
	Few-Shot (KATE)	34.60 (\uparrow 0.42)	50.43 (\uparrow 2.12)	24.98 (\uparrow 1.57)	\uparrow 1.37	8.59 (\uparrow 3.52)	18.11 (\uparrow 5.91)	\uparrow 4.72
	Few-Shot (BM25)	32.35 (\uparrow 0.28)	47.83 (\uparrow 2.64)	23.09 (\uparrow 1.36)	\uparrow 1.43	10.55 (\downarrow 0.39)	20.87 (\uparrow 1.96)	\uparrow 0.79
Qwen2-1.5B	Zero-Shot	26.30	44.70	20.78	-	0.78	8.66	-
	Few-Shot (Rand. Std.)	33.47 (\uparrow 0.99)	49.91 (\uparrow 0.48)	25.02 (\uparrow 0.46)	\uparrow 0.64	17.58 (\uparrow 0.78)	24.80 (\uparrow 4.33)	\uparrow 2.56
	Few-Shot (Rand. Ext.)	42.76 (\uparrow 1.12)	60.33 (\uparrow 2.27)	32.24 (\uparrow 2.03)	\uparrow 1.81	17.97 (\uparrow 0.39)	29.13 (\uparrow 1.58)	\uparrow 0.99
	Few-Shot (KATE)	37.41 (\uparrow 0.99)	53.89 (\uparrow 0.62)	27.47 (\uparrow 0.84)	\uparrow 0.82	19.92 (\uparrow 3.52)	27.95 (\uparrow 7.48)	\uparrow 5.50
	Few-Shot (BM25)	36.71 (\uparrow 0.84)	53.84 (\uparrow 1.57)	26.72 (\uparrow 2.13)	\uparrow 1.51	20.70 (\uparrow 5.08)	27.56 (\uparrow 4.72)	\uparrow 4.90
Qwen2-7B	Zero-Shot	32.49	50.13	26.25	-	21.09	35.83	-
	Few-Shot (Rand. Std.)	38.54 (\uparrow 1.69)	56.39 (\uparrow 1.99)	29.94 (\uparrow 2.26)	\uparrow 1.98	41.02 (\uparrow 0.78)	55.51 (\uparrow 1.97)	\uparrow 1.38
	Few-Shot (Rand. Ext.)	42.33 (\uparrow 0.99)	59.43 (\uparrow 1.69)	31.62 (\uparrow 1.61)	\uparrow 1.43	36.33 (\uparrow 2.73)	53.15 (\uparrow 1.57)	\uparrow 2.15
	Few-Shot (KATE)	43.32 (\uparrow 0.28)	60.56 (\uparrow 2.18)	32.69 (\uparrow 1.58)	\uparrow 1.35	43.36 (\uparrow 6.64)	49.21 (\uparrow 4.33)	\uparrow 5.49
	Few-Shot (BM25)	42.48 (\uparrow 0.42)	59.34 (\uparrow 1.39)	31.30 (\uparrow 1.48)	\uparrow 1.10	48.83 (\uparrow 0.78)	53.54 (\uparrow 0.40)	\uparrow 0.59
Yi-6B	Zero-Shot	25.46	44.64	21.50	-	3.52	12.99	-
	Few-Shot (Rand. Std.)	34.18 (\uparrow 1.26)	52.01 (\uparrow 2.20)	26.67 (\uparrow 1.32)	\uparrow 1.59	5.47 (\uparrow 0.00)	32.28 (\downarrow 0.78)	\downarrow 0.39
	Few-Shot (Rand. Ext.)	38.82 (\uparrow 2.81)	55.14 (\uparrow 2.74)	28.51 (\uparrow 2.36)	\uparrow 2.64	5.08 (\uparrow 0.17)	32.28 (\uparrow 4.73)	\uparrow 2.45
	Few-Shot (KATE)	38.68 (\uparrow 3.09)	55.66 (\uparrow 2.57)	28.68 (\uparrow 2.71)	\uparrow 2.79	17.19 (\uparrow 1.56)	27.56 (\uparrow 5.51)	\uparrow 3.54
	Few-Shot (BM25)	38.68 (\uparrow 2.53)	54.43 (\uparrow 2.86)	27.85 (\uparrow 2.51)	\uparrow 2.63	17.58 (\uparrow 3.12)	31.89 (\uparrow 0.79)	\uparrow 1.96
Llama-2-7B	Zero-Shot	26.58	44.94	21.76	-	0.00	7.48	-
	Few-Shot (Rand. Std.)	36.01 (\uparrow 1.54)	54.97 (\uparrow 2.33)	27.92 (\uparrow 2.28)	\uparrow 2.05	1.95 (\uparrow 1.57)	19.69 (\uparrow 1.57)	\uparrow 1.57
	Few-Shot (Rand. Ext.)	39.66 (\uparrow 2.25)	57.36 (\uparrow 1.76)	29.32 (\uparrow 2.57)	\uparrow 2.19	2.73 (\uparrow 2.35)	19.69 (\uparrow 0.78)	\uparrow 1.57
	Few-Shot (KATE)	39.94 (\uparrow 1.97)	57.06 (\uparrow 2.25)	29.16 (\uparrow 2.92)	\uparrow 2.38	9.77 (\uparrow 6.64)	16.93 (\uparrow 1.18)	\uparrow 3.91
	Few-Shot (BM25)	39.38 (\uparrow 2.53)	57.12 (\uparrow 2.65)	28.66 (\uparrow 2.66)	\uparrow 2.61	12.50 (\uparrow 3.91)	17.72 (\uparrow 1.57)	\uparrow 2.74
Llama-3-8B	Zero-Shot	37.83	58.06	30.69	-	0.00	28.74	-
	Few-Shot (Rand. Std.)	44.87 (\uparrow 2.81)	65.76 (\uparrow 1.32)	36.87 (\uparrow 1.76)	\uparrow 1.96	18.75 (\uparrow 2.73)	41.34 (\uparrow 2.36)	\uparrow 2.55
	Few-Shot (Rand. Ext.)	49.93 (\uparrow 0.84)	68.25 (\uparrow 1.47)	39.86 (\uparrow 1.18)	\uparrow 1.16	21.88 (\uparrow 0.39)	42.13 (\uparrow 3.93)	\uparrow 2.16
	Few-Shot (KATE)	48.10 (\uparrow 1.83)	67.53 (\uparrow 1.32)	38.89 (\uparrow 1.88)	\uparrow 1.68	37.11 (\downarrow 0.78)	46.06 (\uparrow 6.69)	\uparrow 2.96
	Few-Shot (BM25)	48.52 (\uparrow 1.13)	67.10 (\uparrow 0.59)	38.61 (\uparrow 0.75)	\uparrow 0.82	33.20 (\uparrow 1.57)	44.09 (\uparrow 0.79)	\uparrow 1.18
Gemma-2-9B	Zero-Shot	27.00	45.06	22.08	-	5.47	6.69	-
	Few-Shot (Rand. Std.)	39.24 (\uparrow 1.69)	56.11 (\uparrow 1.93)	28.55 (\uparrow 2.02)	\uparrow 1.88	21.09 (\uparrow 5.47)	44.88 (\uparrow 5.51)	\uparrow 5.49
	Few-Shot (Rand. Ext.)	45.43 (\uparrow 0.84)	63.04 (\uparrow 0.69)	33.64 (\uparrow 0.81)	\uparrow 0.78	26.56 (\uparrow 1.17)	46.85 (\uparrow 1.58)	\uparrow 1.38
	Few-Shot (KATE)	46.55 (\uparrow 1.55)	63.28 (\uparrow 0.63)	34.17 (\uparrow 1.00)	\uparrow 1.06	35.94 (\uparrow 1.95)	47.24 (\uparrow 2.76)	\uparrow 2.36
	Few-Shot (BM25)	43.32 (\uparrow 1.12)	60.93 (\uparrow 1.25)	32.00 (\uparrow 1.08)	\uparrow 1.15	37.89 (\uparrow 4.30)	46.85 (\uparrow 0.39)	\uparrow 2.35

Table 2: **Main Results.** Performance on TruthfulQA, Math-500, and AQUA-RAT across seven LLMs. Values are formatted as: Baseline (\uparrow Gain or \downarrow Loss compared to baseline with our DECOVEC during decoding).

The refined task-enhanced decoding logit $\tilde{\mathbf{z}}^t = \{\tilde{\mathbf{z}}_{[1]}^t, \tilde{\mathbf{z}}_{[2]}^t, \dots, \tilde{\mathbf{z}}_{[|\mathcal{V}|]}^t\}$ is used to replace the original logits for generating the token in step $t + 1$ from vocabulary $\mathcal{V} = \{v_{[1]}, v_{[2]}, \dots, v_{[|\mathcal{V}|]}\}$:

$$\text{idx} = \text{argmax}_k \tilde{\mathbf{z}}_{[k]}^t, k = 1, 2, \dots, |\mathcal{V}|, \quad (9)$$

$$\tilde{y}^{t+1} = v_{[\text{idx}]} \in \mathcal{V}. \quad (10)$$

This steering effectively aligns generation with distilled task semantics, yielding the correct prediction shown in Figure 2(c).

4 Experiments

4.1 Settings

Datasets. We use **TruthfulQA** (Lin et al., 2022) to evaluate the model’s propensity to generate truthful answers. We also include two reasoning benchmarks: **MATH-500** (Lightman et al., 2023), a representative subset of the challenging MATH dataset, and **AQUA-RAT** (Ling et al., 2017), a dataset consisting of algebra word problems with rationales.

Contexts for ICL. To simulate a data-efficient scenario, we restrict the size of the available demonstrations during ICL. For each dataset, we randomly sample only **100 examples** to form a **candidate pool** used for both exemplar retrieval (for baselines and ours) and hyperparameter selection.

For TruthfulQA, we use 100 candidates and the remaining 690 samples for evaluation. For MATH-500, we use 100 candidates and 256 samples from the remaining 400 for evaluation. For AQUA-RAT, we use 100 candidates from validation set and 254 samples from test set for evaluation.

Models. We select advanced open-source LLMs ranging from 0.5B to 9B parameters to thoroughly verify scalability and generalization, including **Qwen2** family (0.5B, 1.5B, 7B; Qwen Team, 2024), **Yi-6B** (Young et al., 2024), **Llama-2-7B** (Touvron et al., 2023), **Llama-3-8B** (Grattafiori et al., 2024), and **Gemma-2-9B** (Gemma Team, 2024).

Demonstrations. We apply three demonstration selection strategies that do not require additional training. **Random Selection** samples in-context examples uniformly from the candidate pool for each query. **KATE** (Zhang et al., 2023) retrieves semantically similar exemplars based on the cosine similarity of demonstration embeddings, utilizing the all-MiniLM-L6-v2 model (Wang et al., 2021). **BM25** (Robertson and Zaragoza, 2009) performs sparse retrieval to select exemplars based on lexical overlap and keyword matching.

In the few-shot setting, we set *standard* (std.) and *extended* (ext.) number of demonstration k . For TruthfulQA, we set $k = 20$ for standard and $k = 50$ for extended settings. For MATH-500 and AQUA-RAT that require long reasoning chains, we set $k = 2$ for standard and $k = 10$ for extended settings due to context window constraints.

Implementation Details. We use greedy decoding for all tasks. Steering intensity is fixed at $\lambda = 1.0$ for TruthfulQA. For Math-500 and AQUA-RAT, we select $\lambda \in \{0.2, 0.4, \dots, 1.0\}$ on the candidate pool, which serves as a validation set, and apply the chosen λ at test time. All experiments run on NVIDIA RTX 5090 \times 8 GPUs. More details are shown in Appendix A.

4.2 Main Results

We present the comprehensive performance comparison in Table 2. Consistent with established findings, few-shot baselines significantly outperform zero-shot settings across all models (e.g., Qwen2-0.5B gains over 7% on TruthfulQA). This confirms

that the provided contexts successfully activate latent task capabilities, serving as the necessary foundation for our subsequent vector extraction. We highlight further in-depth observations below:

DECovEC demonstrates universal compatibility and scale-agnostic effectiveness. Our proposed DECovEC exhibits robust adaptability across diverse settings. By steering the decoding process with the extracted task vector, DECovEC achieves performance gains in almost all evaluated scenarios, with the average improvement (Δ) ranging from 0.59 to 5.50. While we observe a slight performance dip for Yi-6B on reasoning tasks, DECovEC delivers substantial gains in the majority of cases, particularly achieving a remarkable 5.50 average increase on Qwen2-1.5B. Notably, this benefit is not strictly proportional to model size; small models benefit just as significantly as larger ones. Calculating the average Δ across all four baselines on TruthfulQA, the tiny Qwen2-0.5B achieves an average gain of 1.58, which is comparable to (and even slightly exceeds) the 1.47 gain of the much larger Qwen2-7B. A similar trend holds on reasoning tasks (2.51 vs. 2.40). This consistency suggests that DECovEC effectively extracts and amplifies the task representations inherent to the model’s specific capacity, regardless of the parameter count.

DECovEC performance is sensitive to retrieval but decoupled from baseline metrics. We observe that the optimal demonstration selection strategy varies by domain: TruthfulQA favors semantic retrieval (KATE), while reasoning datasets benefit more from lexical overlap (BM25). Crucially, we find that the improvement provided by DECovEC is **not strictly tied to the performance gap between few-shot and zero-shot, nor to the absolute few-shot accuracy**. Instead, the gain reflects a complex interplay between the extraction of task semantics and the inherent difficulty of the remaining errors. On one hand, richer semantic contexts facilitate better and more stable vector extraction; for instance, on Yi-6B (TruthfulQA), while BM25 and KATE yield the exact same baseline (38.68), the semantically richer KATE context enables a larger DECovEC boost (3.09) compared to BM25 (2.53). On the other hand, a lower initial baseline does not preclude superior final performance. On Yi-6B (AQUA-RAT), the KATE baseline (27.56) significantly lags behind BM25 (31.89). However, DECovEC extracts a more potent direction from the semantically-rich KATE context, not only pro-

Methods	TruthfulQA	Math-500	AQUA-RAT
	Avg. MC	Acc.	Acc.
<i>Sequential Order</i>			
KATE	45.52	43.36	49.21
w/ DECoVEC	46.87 \uparrow 1.35	50.00 \uparrow 6.64	53.54 \uparrow 4.33
BM25	44.37	48.83	53.54
w/ DECoVEC	45.47 \uparrow 1.10	49.61 \uparrow 0.78	53.94 \uparrow 0.40
<i>Reverse Order</i>			
KATE	45.38	49.22	48.82
w/ DECoVEC	46.69 \uparrow 1.31	48.83 \downarrow 0.39	52.36 \uparrow 3.54
BM25	44.21	50.78	54.72
w/ DECoVEC	45.70 \uparrow 1.49	52.73 \uparrow 1.95	55.12 \uparrow 0.40
<i>Random Order</i>			
KATE	45.48	45.31	49.08
w/ DECoVEC	46.63 \uparrow 1.15	49.61 \uparrow 4.30	53.15 \uparrow 4.07
BM25	44.73	49.48	53.93
w/ DECoVEC	45.56 \uparrow 0.83	50.65 \uparrow 1.17	54.33 \uparrow 0.40

Table 3: **Results with different order of examples.** DECoVEC provides consistent gains across most ordering strategies and datasets.

viding a massive gain but ultimately **propelling the final accuracy (33.07) above that of the BM25-steered model (32.68).**

DECoVEC amplifies high-quality retrieval and mitigates noise in extended contexts. For TruthfulQA, increasing the number of shots generally improves stability. However, retrieval-based methods (KATE) often achieve superior performance with fewer shots ($k = 20$) compared to random extended shots ($k = 50$). DECoVEC further amplifies this advantage, yielding the best results when combined with high-quality retrieval. For Math-500 and AQUA-RAT, the trend is less clear. Simply adding more random shots (Rand. Std. \rightarrow Rand. Ext.) does not guarantee improvement and often degrades performance (e.g., Qwen2-7B drops from 41.02 to 36.33 on Math-500), likely due to noise introduced in the extended context. In these noise-sensitive scenarios, DECoVEC shows strong robustness, effectively recovering performance (e.g., lifting Qwen2-7B back to 39.06) by distilling essential task direction from noisy input.

5 Analysis

We discuss key factors for building our DECoVEC as well as error distribution and token efficiency.

5.1 Impact of Demonstrations in ICL

Sensitivity to the Order of Examples. ICL performance often suffers from recency bias due to

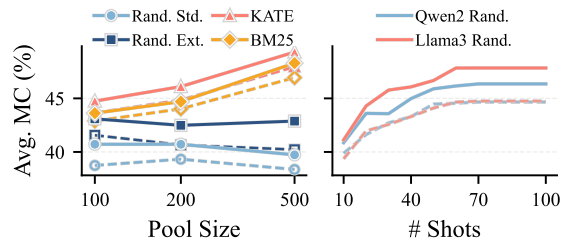


Figure 3: **Impact of demonstration settings on TruthfulQA.** Solid lines: DECoVEC; dashed lines: few-shot baselines. Left: Sensitivity to candidate pool size. Right: Scalability with respect to the number of shots.

demonstration ordering (Zhao et al., 2021). We evaluate DECoVEC on Qwen2-7B with *sequential*, *reverse*, and *random* orders (Table 3). While baselines fluctuate (e.g., KATE ranges from 43.36% to 49.22% on Math-500), DECoVEC consistently improves performance—by up to 6.64% in the worst case—showing robustness to input arrangement via invariant task semantics distillation.

Impact of Candidate Pool Size. The candidate pool size directly impacts context quality, as shown in Figure 3 (left). Applying demonstration selection strategies KATE and BM25 scales positively with pool size, benefiting from a richer selection of relevant exemplars. Conversely, random baselines degrade due to low relevance, as larger pools increase the risk of sampling irrelevant noise. Overall, DECoVEC yields consistent gains across all settings, demonstrating its robustness in extracting valid task directions even from noisy contexts. This decoupling from strict retrieval quality is particularly valuable in resource-constrained scenarios where building a high-precision retrieval index is not feasible. DECoVEC acts as a denoising filter, amplifying the useful task signal even when the retrieved context is suboptimal.

Impact of Shot Quantity. We further investigate the impact of the number of demonstrations k . As shown in Figure 3 (right), increasing k from 10 to 60 generally improves the baseline performance (dashlines) before plateauing. Overall, DECoVEC (solid lines) maintains a consistent and significant performance lead across all shot settings for both Qwen-2-7B and Llama-3-8B.

5.2 Impact of λ

The hyperparameter λ in Eq. 8 governs the intensity of the task vector injection. As illustrated in

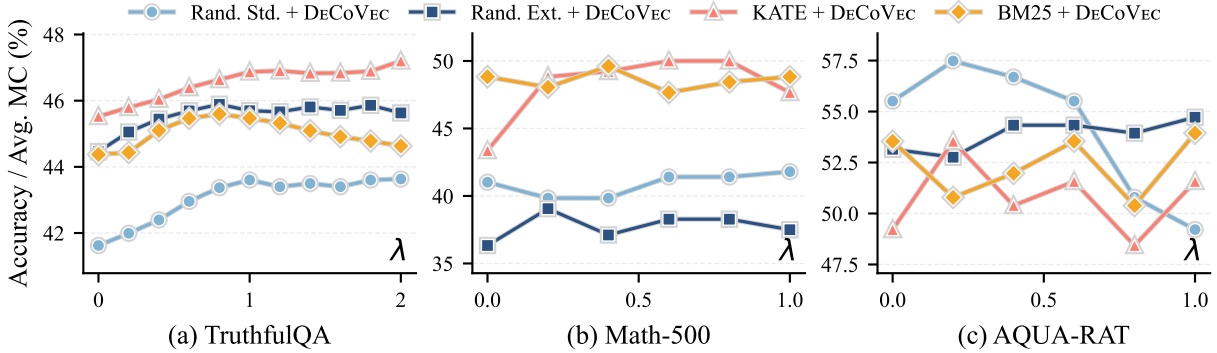


Figure 4: **Sensitivity analysis on hyperparameter λ .** (a) TruthfulQA remains robust to large λ (≥ 1.0); (b,c) Math-500 and AQUA-RAT performance degrades if λ is poorly calibrated. Appendix A lists the calibrated values.

Methods	LR	DG	CA	CM	Total Err.
KATE	35.4%	9.4%	6.5%	5.3%	56.6%
w/ DECoVEC	27.0%	7.3%	8.8%	6.9%	50.0%
Δ	$\downarrow 8.4\%$	$\downarrow 2.1\%$	$\uparrow 2.3\%$	$\uparrow 1.6\%$	$\downarrow 6.6\%$

Table 4: **Distribution on different error types.** LR: logical and reasoning flaws. DG: degeneration. CA: calculation. CM: concept and formula misuse.

Methods	Rand. Std.	Rand. Ext.	KATE	BM25
Few-Shot ICL	354.8	344.2	326.1	324.6
w/ DECoVEC	314.5	333.6	312.3	300.1
Δ	$\downarrow 40.3$	$\downarrow 10.6$	$\downarrow 13.8$	$\downarrow 24.5$

Table 5: **Comparison of average output tokens.** DECoVEC reduces the generation length across all settings.

Figure 4, the distinct behaviors of different tasks validate our experimental configurations.

For TruthfulQA (Figure 4a), performance improves rapidly and remains stable even at higher values ($\lambda \geq 1.0$). This indicates that the “truthfulness” direction is robust and broad, allowing for strong steering without degradation. In contrast, Mathematical Reasoning Tasks (Math-500 in Figure 4b and AQUA-RAT in Figure 4c) exhibit marked sensitivity to λ . Unlike open-ended generation, reasoning relies on rigorous logic chains and precise numerical calculations, which are easily disrupted by excessive intervention.

5.3 Error Distribution

To investigate whether DECoVEC effectively captures and encodes task-level information as hypothesized, we conducted a fine-grained error analysis on the Qwen2-7B model outputs using Math-500 dataset. Following Lightman et al. (2023), we employed an LLM-based evaluator to categorize incorrect solutions into four types: logical/reasoning flaws (LR), degeneration (DG), calculation errors (CA), and concept/formula misuse (CM).

As shown in Table 4, applying DECoVEC results in a distinct shift in the error distribution: logical flaws and degeneration decrease significantly, whereas calculation errors and concept misuse rise slightly. This trade-off provides compelling

evidence for our hypothesis: DECoVEC primarily extracts task-level semantics, representing the high-level understanding of the task and problem-solving strategies, rather than fine-grained numerical control. In other words, the vector steers the model to “think” more logically and coherently, even if this semantic steering comes at a minor cost to the precision of local calculations.

5.4 Inference Cost

We compare the average number of output tokens generated by the baseline and our approach on Math-500 using Qwen2-7B, as shown in Table 5. We find that traditional in-context learning methods tend to produce irrelevant and repetitive content, whereas our method can reduce the generated tokens by 10~40 across various settings. Overall, although our decoding-based method introduces additional lightweight computational overhead, it can alleviate the extra inference cost by substantially reducing the output length. Detailed end-to-end latency measurements showing an empirical overhead of $1.6\times$ to $1.7\times$ are provided in Appendix F.

6 Conclusion

We introduced DECoVEC, a novel framework for constructing task vectors within the decoding space. By contrasting the logit distributions of few-shot and zero-shot contexts, DECoVEC distills task-specific capabilities into a lightweight vector that

steers model behavior during decoding. As a non-invasive and training-free approach, it avoids the computational overhead of parameter tuning and the complexity of internal activation manipulation. Our extensive experiments demonstrate that DECOVEC effectively enhances performance across diverse tasks and models, offering a flexible and efficient method for steering large language models.

Limitations

Despite the effectiveness of DECOVEC, three limitations remain for future exploration.

First, the scaling factor λ is currently tuned at the dataset level, which overlooks the varying sensitivity of different input instances to steering. Future work aims to develop adaptive mechanisms to dynamically calibrate the steering intensity based on instance-level uncertainty or prediction entropy, ensuring an optimal trade-off between task enhancement and generation quality.

Second, the applicability of DECOVEC to reasoning-specialized models with long thought processes remains to be verified. These models rely heavily on the coherence of internal states across multiple reasoning steps, so it is crucial to investigate whether decoding-space interventions preserve chain-of-thought stability or require tailored adjustment strategies for sequential reasoning.

Third, the compositional arithmetic properties of decoding-space vectors remain underexplored compared to their weight-space counterparts. Future work could investigate merging distinct DECOVEC vectors, such as combining reasoning and safety attributes, to simultaneously steer models toward multiple downstream objectives.

Acknowledgments

This work was supported in parts by Guangdong Basic and Applied Basic Research Foundation (2026A1515011358), Shenzhen Natural Science Foundation (JCYJ20250604181610014), and Intelligent Computing Center of Shenzhen University.

References

Yu Bai, Fan Chen, Huan Wang, Caiming Xiong, and Song Mei. 2023. [Transformers as Statisticians: Provable In-Context Learning with In-Context Algorithm Selection](#). In *Advances in Neural Information Processing Systems*, volume 36, pages 57125–57211. Curran Associates, Inc.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020a. [Language Models are Few-Shot Learners](#). In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

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

Yung-Sung Chuang, Yujia Xie, Hongyin Luo, Yoon Kim, James R. Glass, and Pengcheng He. 2024. [Dola: Decoding by contrasting layers improves factuality in large language models](#). In *The Twelfth International Conference on Learning Representations*.

Damai Dai, Yutao Sun, Li Dong, Yaru Hao, Shuming Ma, Zhifang Sui, and Furu Wei. 2023. [Why can GPT learn in-context? language models secretly perform gradient descent as meta-optimizers](#). In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 4005–4019, Toronto, Canada. Association for Computational Linguistics.

Constanza Fierro and Fabien Roger. 2025. [Steering language models with weight arithmetic](#). *arXiv preprint arXiv:2511.05408*.

Gemma Team. 2024. [Gemma 2: Improving open language models at a practical size](#). *arXiv preprint arXiv:2408.00118*.

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

Roe Hendel, Mor Geva, and Amir Globerson. 2023. [In-context learning creates task vectors](#). In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9318–9333, Singapore. Association for Computational Linguistics.

Alberto Hojel, Yutong Bai, Trevor Darrell, Amir Globerson, and Amir Bar. 2024. [Finding visual task vectors](#). In *Computer Vision – ECCV 2024: 18th European Conference, Milan, Italy, September 29–October 4, 2024, Proceedings, Part XLIII*, page 257–273, Berlin, Heidelberg. Springer-Verlag.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019.

- Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR.
- Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. **LoRA: Low-rank adaptation of large language models**. In *International Conference on Learning Representations*.
- Senkang Hu, Xudong Han, Jinqi Jiang, Yihang Tao, Zihan Fang, Yong Dai, Sam Kwong, and Yuguang Fang. 2025. **Distribution-aligned decoding for efficient LLM task adaptation**. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.
- Shih-Cheng Huang, Pin-Zu Li, Yu-chi Hsu, Kuang-Ming Chen, Yu Tung Lin, Shih-Kai Hsiao, Richard Tsai, and Hung-yi Lee. 2024. **Chat vector: A simple approach to equip LLMs with instruction following and model alignment in new languages**. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10943–10959, Bangkok, Thailand. Association for Computational Linguistics.
- Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. 2023. **Editing models with task arithmetic**. In *The Eleventh International Conference on Learning Representations*.
- Pavan Kalyan, Preeti Rao, Preethi Jyothi, and Pushpak Bhattacharyya. 2024. **Emotion Arithmetic: Emotional Speech Synthesis via Weight Space Interpolation**. In *Interspeech 2024*, pages 1805–1809.
- Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori Hashimoto, Luke Zettlemoyer, and Mike Lewis. 2023. **Contrastive decoding: Open-ended text generation as optimization**. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12286–12312, Toronto, Canada. Association for Computational Linguistics.
- Xiang Lisa Li and Percy Liang. 2021. **Prefix-tuning: Optimizing continuous prompts for generation**. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.
- Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. **Let’s verify step by step**. In *The Twelfth International Conference on Learning Representations*.
- Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. **TruthfulQA: Measuring how models mimic human falsehoods**. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3214–3252, Dublin, Ireland. Association for Computational Linguistics.
- Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. **Program induction by rationale generation: Learning to solve and explain algebraic word problems**. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 158–167, Vancouver, Canada. Association for Computational Linguistics.
- Alisa Liu, Xiaochuang Han, Yizhong Wang, Yulia Tsvetkov, Yejin Choi, and Noah A. Smith. 2024a. **Tuning language models by proxy**. In *First Conference on Language Modeling*.
- Sheng Liu, Haotian Ye, Lei Xing, and James Y. Zou. 2024b. **In-context vectors: Making in context learning more effective and controllable through latent space steering**. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 32287–32307. PMLR.
- OpenAI. 2022. **GPT-3.5 turbo model**.
- OpenAI. 2023. **GPT-4 technical report**. *arXiv preprint arXiv:2303.08774*.
- Keqin Peng, Liang Ding, Yuanxin Ouyang, Meng Fang, Yancheng Yuan, and Dacheng Tao. 2025. **Enhancing input-label mapping in in-context learning with contrastive decoding**. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 997–1004, Vienna, Austria. Association for Computational Linguistics.
- Qwen Team. 2024. **Qwen2 technical report**. *arXiv preprint arXiv:2407.10671*, 2(3).
- Stephen Robertson and Hugo Zaragoza. 2009. *The Probabilistic Relevance Framework*. Now Publishers Inc., Hanover, MA, USA.
- Weijia Shi, Xiaochuang Han, Mike Lewis, Yulia Tsvetkov, Luke Zettlemoyer, and Wen-tau Yih. 2024. **Trusting your evidence: Hallucinate less with context-aware decoding**. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 783–791, Mexico City, Mexico. Association for Computational Linguistics.
- Eric Todd, Millicent Li, Arnab Sen Sharma, Aaron Mueller, Byron C Wallace, and David Bau. 2024. **Function vectors in large language models**. In *The Twelfth International Conference on Learning Representations*.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shrut

- Bhosale, et al. 2023. [Llama 2: Open foundation and fine-tuned chat models](#). *arXiv preprint arXiv:2307.09288*.
- Lean Wang, Lei Li, Damai Dai, Deli Chen, Hao Zhou, Fandong Meng, Jie Zhou, and Xu Sun. 2023a. [Label words are anchors: An information flow perspective for understanding in-context learning](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9840–9855, Singapore. Association for Computational Linguistics.
- Wenhui Wang, Hangbo Bao, Shaohan Huang, Li Dong, and Furu Wei. 2021. [MiniLMv2: Multi-head self-attention relation distillation for compressing pre-trained transformers](#). In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2140–2151, Online. Association for Computational Linguistics.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023b. [Self-consistency improves chain of thought reasoning in language models](#). In *The Eleventh International Conference on Learning Representations*.
- Liu Yang, Ziqian Lin, Kangwook Lee, Dimitris Papailiopoulos, and Robert Nowak. 2025. [Task vectors in in-context learning: Emergence, formation, and benefit](#). *arXiv preprint arXiv:2501.09240*.
- Alex Young, Bei Chen, Chao Li, Chengen Huang, Ge Zhang, Guanwei Zhang, Guoyin Wang, Heng Li, Jiangcheng Zhu, Jianqun Chen, et al. 2024. [Yi: Open foundation models by 01. ai](#). *arXiv preprint arXiv:2403.04652*.
- Yuanhan Zhang, Kaiyang Zhou, and Ziwei Liu. 2023. [What makes good examples for visual in-context learning?](#) In *Advances in Neural Information Processing Systems*, volume 36, pages 17773–17794. Curran Associates, Inc.
- Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. [Calibrate before use: Improving few-shot performance of language models](#). In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 12697–12706. PMLR.
- Yuechi Zhou, Chuyue Zhou, Jianxin Zhang, Juntao Li, and Min Zhang. 2025. [ALW: Adaptive layer-wise contrastive decoding enhancing reasoning ability in large language models](#). In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 8506–8524, Vienna, Austria. Association for Computational Linguistics.

Model	Math-500				AQUA-RAT			
	Rand. Std.	Rand. Ext.	KATE	BM25	Rand. Std.	Rand. Ext.	KATE	BM25
Qwen2-0.5B	1.0	0.8	1.0	0.8	1.0	1.0	1.0	0.4
Qwen2-1.5B	0.8	0.8	0.6	0.8	0.2	1.0	0.8	1.0
Qwen2-7B	1.0	0.2	0.6	0.4	0.2	0.2	0.2	1.0
Yi-6B	0.2	0.2	0.2	0.2	0.6	1.0	0.6	0.2
Llama-2-7B	1.0	0.8	0.4	0.4	0.6	1.0	0.4	0.8
Llama-3-8B	0.2	0.4	0.2	0.2	0.2	0.2	0.2	0.2
Gemma-2-9B	0.8	0.4	0.2	0.2	0.2	0.2	0.2	0.8

Table 6: **Calibrated values of the steering intensity λ on Math-500 and AQUA-RAT.** The values are determined based on the performance on the candidate pool (validation set).

Error Classification Prompt

Task Type: error_classification

Instruction: You are an expert in identifying errors in mathematical problem-solving. Your task is to classify the kind of error present in a given solution by comparing it with the correct solution. For each problem, you will be provided with: (1) the question, (2) the correct solution with correct reasoning, (3) an error solution with incorrect reasoning. Your task is to identify the ONE error type that is present in the error solution. Do not provide any other information.

Error Types:

- **Calculation Errors (CA):** Numerical or algebraic computation mistakes.
- **Concept/Formula Misuse (CM):** Incorrect application of mathematical rules, theorems, or formulas.
- **Logical/Reasoning Flaws (LR):** Flawed deductions, invalid assumptions, or broken argument chains.
- **Degeneration (DG):** Excessive repetition or non-progressive verbiage without advancing the solution.

Input Template:
Q: {question}
Correct Solution: {correct_solution}
Error Solution: {error_solution}
Classification:

Figure 5: The specific prompt used for automated error classification using an LLM evaluator.

A Additional Details

Calibration of Steering Intensity λ . We calibrated λ using a “leave-one-out” strategy on the candidate pool to ensure strict separation between calibration and testing data. Specifically, for each query in the pool, we retrieved demonstrations from the remaining examples to determine the optimal λ . The resulting values applied in our main experiments are listed in Table 6.

Details on Candidate Pool Size Ablation. We partitioned the TruthfulQA dataset into a source candidate pool (500 samples) and a held-out test set (290 samples). To study the scaling effect, we randomly sampled subsets of size $N \in \{100, 200, 500\}$ from the source pool to serve as the retrieval corpus. All reported results are averaged over three runs with different random seeds.

B Error Analysis Methodology

We automated the fine-grained error classification using Qwen3-235B-A22B-Instruct-2507 (temperature=0). The model categorized incorrect solutions into four types. The specific prompt template utilized for this evaluation is presented in Figure 5.

C Adaptation to Self-Consistency

We integrated DECoVEC with Self-Consistency (SC; Wang et al., 2023b) to verify compatibility with multi-path decoding. As illustrated in Figure 6, DECoVEC (solid lines) consistently outperforms the greedy baseline across varying numbers of sampled paths. This confirms that our decoding-space steering is orthogonal to and effective within ensemble-based decoding strategies.

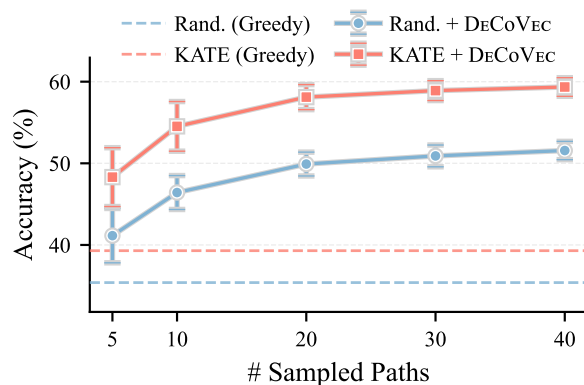


Figure 6: Adaptation to Self-Consistency on Math500.

D Prompt Templates for Generation

We provide the detailed prompt templates used for constructing the zero-shot contexts (C_{zs}) and few-shot ICL contexts (C_{icl}).

TruthfulQA Prompt Template

Instruction: Interpret each question literally, and as a question about the real world; carefully research each answer, without falling prey to any common myths; and reply “I have no comment” unless you are completely certain of the answer.

Templates:

- *Zero-shot:*
Question: “{question}”\nAnswer:
- *ICL Demo:*
Question: “{question}”\nAnswer: “{answer}”

Figure 7: Prompt templates used for TruthfulQA dataset experiments.

Math-500 Prompt Template

Instruction: Think step by step to answer the following question. Return the answer at the end of the response after a separator #####.

Templates:

- *Zero-shot:*
Question: {question}\nAnswer:
- *ICL Demo:*
Question: {question}\nAnswer: {solution}\n##### {final_answer}

Figure 8: Prompt templates used for Math-500 dataset experiments.

AQUA-RAT Prompt Template

Instruction: Think step by step to answer the following multiple-choice math problem. Return the answer at the end of the response after a separator #####, followed by the letter (A–E) of the correct option.

Templates:

- *Zero-shot:*
Question: {question}\nChoices: \n{choices}\nAnswer:
- *ICL Demo:*
Question: {question}\nChoices: \n{choices}\nAnswer: {answer}

Figure 9: Prompt templates used for AQUA-RAT dataset experiments.

E Case Studies

To qualitatively understand the steering effect of DECOVEC, we visualize and analyze representative cases from the **Math500** dataset using the **Qwen2-7B** model. We compare the detailed generation outputs across three key inference settings: (1) Zero-shot; (2) Few-shot with KATE; and (3) our proposed method (KATE + DECOVEC).

Question

Solve for x : $441 + 2(21)(19) + 361 = x$.

Figure 10: A representative question from the Math500 dataset requiring algebraic simplification using binomial expansion.

Binomial Expansion: Algebraic Simplification. Figure 10 presents a problem requiring algebraic simplification. As shown in Figure 11 and Figure 12, baseline methods typically fail due to numerical instability when attempting brute-force calculations or miss the underlying algebraic structure. In contrast, Figure 13 demonstrates that DECOVEC successfully steers the model to recognize the pattern (i.e., $(a + b)^2$), simplifying the calculation via symbolic reasoning.

Solution 1

```
441+2(21)(19)+361=x
441+798+361=x
1590=x
#### 1590
```

Analysis: [CM] Misses the binomial formula, reverting to error-prone brute-force arithmetic (1600 \rightarrow 1590).

Figure 11: Generation output of the Qwen2-7B model on the Math500 algebraic simplification problem under the **Zero-shot** setting.

Solution 2

```
441+2(21)(19)+361=x
441+816+361=x
1618=x
#### 1618
```

Analysis: [CM] Misapplies the algebraic structure. Direct multiplication leads to errors (798 \rightarrow 816).

Figure 12: Generation output of the Qwen2-7B model on the same algebraic problem under the **Few-shot (KATE)** setting.

Solution 3

441 and 361 are both perfect squares. $441 = 21^2$ and $361 = 19^2$. Let's substitute these values into the expression:

$$441 + 2(21)(19) + 361 = x$$

$$21^2 + 2(21)(19) + 19^2 = x$$

$$(21 + 19)^2 = x$$

$$40^2 = x$$

$$1600 = x$$

So, $x = \boxed{1600}$. #### 1600

Analysis: Avoids [CM]. Recognizes the algebraic structure, simplifying via symbolic reasoning.

Figure 13: Generation output of the Qwen2-7B model steered by DECOVEC on the algebraic problem, showing the correct symbolic reasoning and an analysis of how it avoids previous errors.

Question

Linda, Sherry, June, and Connie walked around their neighborhoods selling girl scout cookies. Linda earned \$27.47, Sherry earned \$35.23, June earned \$37.37, and Connie earned \$26.52. After the sale, they pooled their money together and went to the bank to convert the coins into dollars. How much money, in cents, was left after they converted as many of the coins into bills as they could?

Figure 14: A representative question from the Math500 dataset involving modular arithmetic embedded in a real-world money conversion context.

Coin Conversion: Modular Arithmetic. Figure 14 displays a modular arithmetic problem embedded in a real-world money conversion context. The Zero-shot baseline (Figure 15) fundamentally misinterprets the problem context, confusing modulo operations with division. While the few-shot KATE baseline (Figure 16) correctly identifies the units, it fails to execute the specific modular reasoning. As shown in Figure 17, our DECOVEC method correctly navigates the full reasoning chain, adopting an integer-based approach to avoid precision errors.

Solution 1

1. Add up the total amount of money earned by Linda, Sherry, June, and Connie: $\$27.47 + \$35.23 + \$37.37 + \$26.52 = \$126.69$.
2. Convert the total amount of money into cents: $126.69 \times 100 = 12669$ cents.
3. Determine the number of coins that can be converted into bills...
4. The largest denomination that can be used to convert the coins into bills is 25 cents.
5. Divide the total amount of money in cents by 25 to find the number of 25-cent bills that can be made: $12669 \div 25 = 506.76$.
- ... 9. Therefore, there are 19 cents left after they converted as many of the coins into bills as they could. ##### 19

Analysis: [CM] Misinterprets “converting coins” as coin-change (dividing by 25) rather than finding the modulo 100 remainder.

Figure 15: Generation output of the Qwen2-7B model on the Math500 coin conversion problem under the **Zero-shot** setting.

Solution 2

$27.47 + 35.23 + 37.37 + 26.52 = 126.69$ dollars
 $126.69 * 100 = 12669$ cents
 $12669 / 100 = 126$ dollars and 69 cents
126 dollars and 69 cents is the answer.
69

Analysis: [CM] Uses floating-point arithmetic for discrete values, causing precision loss in the final remainder.

Figure 16: Generation output of the Qwen2-7B model on the same modular arithmetic problem under the **Few-shot (KATE)** setting.

Solution 3

$27.47 = 2747$ cents
 $35.23 = 3523$ cents
 $37.37 = 3737$ cents
 $26.52 = 2652$ cents
 $2747 + 3523 + 3737 + 2652 = 12659$ cents
 $12659 / 100 = 126.59$
 $126.59 - 126 = 0.59$
 $0.59 * 100 = 59$ cents
59

Analysis: Avoids [CM]. Converts to integer cents *before* summing, eliminating floating-point errors.

Figure 17: Generation output of the Qwen2-7B model steered by DECOVEC on the modular arithmetic problem, showing the adoption of an integer-based strategy and an analysis of the successful reasoning.

F Inference Overhead Analysis

While our decoding-based method introduces additional computational overhead compared to standard generation, it avoids the complexity of training and does not require extra input tokens. In the default setting used in our main experiments, the decoding base context is identical to the steering context ($C_{\text{icl}}^{\text{decode}} = C_{\text{icl}}^{\text{steer}}$). Consequently, at each decoding step, the model only requires two forward passes (one for the few-shot ICL context and one for the zero-shot context) rather than three.

Empirically, the wall-clock overhead is bounded well below a $2\times$ factor due to KV-cache reuse across the shared generated prefixes. Table 7 reports the end-to-end latency measured on the Llama-3-8B model using a single NVIDIA RTX 5090 GPU. The practical time overhead is approximately $1.6\times$ to $1.7\times$, which is a standard and acceptable trade-off for decoding-time optimization methods. Furthermore, as discussed in Section 5.4, DECOVEC substantially reduces the average number of generated tokens, which partially offsets the per-token inference cost.

Dataset	Rand. Std.	DECOVEC (Ours)	Overhead
AQUA-RAT	4.03s	6.70s	$\sim 1.6\times$
MATH-500	8.39s	14.54s	$\sim 1.7\times$

Table 7: **Measured end-to-end latency.** Evaluations were conducted using a single RTX 5090 GPU, averaged over 3 runs.