

I³CL: Restoring Interactive Capabilities to Implicit In-Context Learning

Hao Wu^{1,2}, Bin Shi^{1,2}, Kai Sun^{1,2}, Aoxiang Zhou³, Ziqi Liu⁵,
Zhiqiang Zhang⁵, Jun Zhou⁵, Wenhan Zeng⁴, Peng Liu³, Bo Dong^{1,2*}

¹Shaanxi Provincial Key Laboratory of Big Data Knowledge Engineering, XJTU

²School of Computer Science and Technology, Xi'an Jiaotong University

³School of Computer Science and Engineering, Guangxi Normal University

⁴XJTU-POLIMI Joint School, Xi'an Jiaotong University ⁵Ant Group

Correspondence: dong.bo@xjtu.edu.cn

Abstract

To address the limitations of conventional ICL, recent task-vector-based methods, such as Implicit ICL (I²CL), encode demonstrations into a unified task vector that is injected during inference, yielding promising improvements in efficiency and stability. However, existing methods encode each demonstration in isolation, neglecting the rich interactions between demonstrations and between demonstrations and queries—a core capability of standard ICL. This results in task vectors with limited expressiveness. To bridge this gap, we propose Interactive Implicit In-Context Learning (I³CL), a simple yet effective framework that restores these essential interactive capabilities. I³CL introduces only two lightweight attention-based modules, preserving the efficiency advantages of I²CL while delivering significant performance improvements. Experiments across nine diverse tasks using four LLMs show that I³CL achieves an average performance improvement of 3.54% over prior task-vector-based baselines. Comprehensive analyses on efficiency, scalability and task vector properties further confirm the superiority of our approach.

1 Introduction

In-context learning (ICL) (Brown et al., 2020) enables large language models (LLMs) to learn tasks using only a few input examples, eliminating the need for parameter updates or explicit fine-tuning. This approach has become a key technique for efficient LLM deployment (Luo et al., 2024). Despite promising results across diverse reasoning tasks (Li et al., 2023; Liu et al.; Zhang et al., 2025), ICL suffers from significant limitations: sensitivity to demonstration format and order (Zhao et al., 2021; Lu et al., 2022), and quadratic computational and memory overhead as demonstration count increases (Wang et al., 2023; Li et al., 2025). These constraints hinder real-world scalability.

* Corresponding author

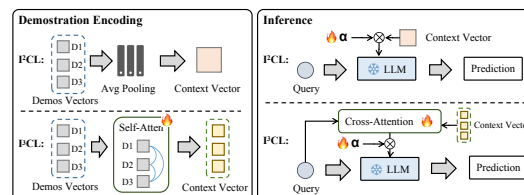


Figure 1: Key differences between I²CL and our I³CL.

To overcome these issues, recent work has turned to task-vector-based methods (TV-based) (Hendel, 2023; Wang et al., 2023; Liu et al., 2024; Li et al., 2025). For instance, unlike standard ICL, which concatenates demonstration tokens into the input prompt, implicit ICL (I²CL) (Li et al., 2025) encodes each demonstration into a dense vector. These vectors are then aggregated into a single compact task vector. During inference, this task vector is injected directly into the LLM’s latent space to guide the model’s generation. This approach introduces minimal computational overhead at inference time—requiring only lightweight vector operations—and thus achieves inference speeds close to zero-shot. Moreover, by merging demonstrations through order-invariant aggregation, it eliminates the order sensitivity inherent in standard ICL.

While demonstrating significant gains in efficiency, constructing a highly expressive task vector remains a challenge. ICL-TV (Dong et al., 2026) reveals that a single global task vector struggles to represent high-rank task mappings. To address this, it extracts and injects multiple task vectors from demonstrations, thereby preserving richer example-level information. However, we contend that a more fundamental cause of this limitation lies in the fact that, in pursuit of faster inference, existing TV-based methods often compromise two core mechanisms essential to ICL’s effectiveness.

First, prior methods encode each demonstration in isolation, failing to capture the rich inter-demonstration interactions inherent to standard

ICL. In standard ICL, self-attention mechanisms explicitly enable information flow across demonstrations—a process crucial for constructing a powerful and adaptable task vector (Hendel, 2023; Saglam et al.). By omitting this process, existing methods restrict the expressiveness of the resulting task vector.

Second, to distill a task vector, prior methods typically employ simple average pooling over the encoded demonstration vectors. While computationally efficient, this uniform averaging strategy treats all demonstrations as equally informative. In contrast, standard ICL leverages attention mechanisms to reweight demonstrations according to the query, enabling the model to focus on the most relevant information (Yousefi et al., 2023; Chen et al., 2024). This query-aware, selective attention is crucial for distilling and preserving the most valuable information for downstream reasoning.

To bridge these gaps, we build upon recent state-of-the-art I²CL (Li et al., 2025) and introduce Interactive Implicit In-Context Learning (I³CL), a framework that restores these essential capabilities through two simple yet effective interaction modules. As illustrated in Figure 1, which highlights the key distinctions from I²CL, our I³CL framework incorporates two levels of interaction: (i) **Inter-demonstration interaction (encoding)**: Demonstrations exchange information during encoding to form a more expressive task vector; and (ii) **Demonstration-to-query interaction (inference)**: The task vector is dynamically conditioned on the query prior to injection, enabling the model to focus on the most relevant demonstrations. Both interactions are implemented via lightweight attention mechanisms, ensuring that inference speed remains on par with I²CL.

While simple, I³CL represents a significant conceptual shift: it reframes the TV-based methods from one that trades performance for efficiency to one that restores essential capabilities while preserving efficiency. This allows us to directly address the core weaknesses of current methods, delivering substantial performance gains with minimal computational overhead.

We evaluate I³CL across a diverse range of LLMs and tasks. I³CL consistently outperforms prior baselines, achieving an average accuracy improvement of 3.54%. Comprehensive analyses on efficiency, scalability, and task vector properties further demonstrate the superiority of our approach.

2 Related Work

Conventional ICL remains unstable, with performance highly sensitive to the content and format of demonstrations. Moreover, its computational and memory costs grow quadratically with sequence length, rendering larger demonstration sets especially expensive. Current work addressing these limitations falls mainly into three categories: task-vector-based methods, prompt-based methods, and prompt compression.

2.1 Task-Vector-based Methods

The core of these methods is to extract a unified “task vector” from a few demonstrations, and inject it back into the LLM to guide reasoning of test samples (Hendel, 2023; Wang et al., 2023; Liu et al., 2024; Li et al., 2025). For example, (Hendel, 2023) interpreted ICL as compressing demonstrations into a task vector that conditions transformer behavior, providing a novel theoretical perspective. (Liu et al., 2024) introduced a context-vector framework for open-ended generation that encodes task information into a single vector, reducing prompt length and improving efficiency. More recently, (Li et al., 2025) introduced I²CL, which injects task vectors into the residual stream through learnable parameters, achieving promising performance on the few-shot tasks. Recent work has also begun to explore the underlying mechanisms of task vectors. Dong et al. (2026) show that single-vector representations are insufficient for modeling high-rank task mappings.

Our approach enhances the expressiveness of the task vector from a novel perspective by restoring two essential capabilities inherent in standard ICL: inter-demonstration interaction and demonstration-to-query interaction. Together, these complementary interaction mechanisms enable our method to generate more compact and discriminative task vectors for in-context learning.

2.2 Prompt-based Methods

Prompt-based methods analyze how demonstration factors influence ICL performance, guiding the design of prompt templates and selection strategies (Rubin et al., 2022; Yin et al., 2023; Wan et al., 2023; Fan et al., 2024; Peng et al., 2024; Jin et al., 2025). For instance, (Rubin et al., 2022; Yin et al., 2023) focused on optimizing and retrieving prompt templates to reduce the cost of manual prompt engineering. (Fan et al., 2024) used contrastive samples

based on the minimum-edit principle to address information bias, while (Peng et al., 2024) proposed “TopK+ConE”, a selection strategy combining data features and model behavior.

2.3 Prompt Compression Methods

Prompt compression methods (Chang et al., 2024) share the objective of improving inference efficiency in LLMs through mitigating the redundant context in the prompt (Chevalier et al., 2023; Ge et al., 2023; Yen et al., 2024). For instance, AutoCompressors (Chevalier et al., 2023) recursively compressed long contexts into summary vectors, which are injected as soft prompts to extend context capacity. CEPE (Yen et al., 2024) employed a lightweight encoder and cross-attention to process long texts in chunks, feeding them into a frozen decoder-only model to extend context length with minimal overhead.

3 Methodology

The proposed Interactive Implicit In-Context Learning (I³CL) enhances I²CL (Li et al., 2025) with two core components: (i) an **inter-demonstration interaction** module that enables demonstrations to exchange information; and (ii) a **demonstration-to-query interaction** module that dynamically weights demonstrations based on the test query. We first formalize the ICL task objective, then detail our approach in subsequent sections. The overview of I³CL is presented in Figure 2.

3.1 Task Definition for ICL

Given a task, ICL assumes a demonstration set $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$, where each pair (x_i, y_i) consists of an input x_i and its corresponding label y_i . During inference, the task instruction I , the demonstration set $\mathcal{D}$, and the test query x_t are combined to form a prompt $\mathcal{P}_t = \{I, \mathcal{D}, x_t\}$, which is fed into the LLM to generate the prediction $\hat{y}_t$ from the label space $\mathcal{Y}$.

3.2 Inter-Demonstration Interaction

To encode each demonstration $(x_i, y_i) \in \mathcal{D}$, we first form $\mathcal{I}_i = \{I, x_i, y_i\}$ by concatenating the task instruction I , input x_i and label y_i . $\mathcal{I}_i$ is then passed through the LLM, and we extract the hidden representations from both the multi-head attention (MHA) module and the feedforward network (MLP) at each layer of the transformer model $\mathcal{M}$:

$$\langle a_i^l, m_i^l \rangle = \mathcal{M}_\theta^l(\mathcal{I}_i), \quad (1)$$

where l indexes the transformer layer; $a_i^l \in \mathbb{R}^{s \times d}$ denotes the MHA module output for layer l ; $m_i^l \in \mathbb{R}^{s \times d}$ denotes the MLP output for layer l ; s and d are the sequence length and representation dimension, respectively.

Prior approaches (Liu et al., 2024; Li et al., 2025) typically represent each demonstration using only the Label Last Token (LLT), i.e., the final token in sequence $\mathcal{I}_i$. However, causal language models (CLMs) exhibit positional bias that causes the LLT’s attention distribution to concentrate excessively on its own position under autoregressive objectives (Wang et al.). This self-referential bias limits the LLT’s capacity to integrate information from task instructions (I) and input contexts (x_i). To mitigate this, we introduce the Prompt Last Token (PLT), defined as the last token of the $\{I, x_i\}$, as a complementary signal. We construct a compressive demonstration representation by averaging the LLT and PLT embeddings at each layer l :

$$\bar{a}_i^l = \frac{1}{2}(\bar{a}_{i,\text{PLT}}^l + \bar{a}_{i,\text{LLT}}^l) \in \mathbb{R}^d, \quad (2)$$

$$\bar{m}_i^l = \frac{1}{2}(\bar{m}_{i,\text{PLT}}^l + \bar{m}_{i,\text{LLT}}^l) \in \mathbb{R}^d, \quad (3)$$

where $\bar{a}_{i,\text{PLT}}^l \in \mathbb{R}^d$ and $\bar{m}_{i,\text{PLT}}^l \in \mathbb{R}^d$: PLT embeddings from MHA/MLP outputs; $\bar{a}_{i,\text{LLT}}^l \in \mathbb{R}^d$ and $\bar{m}_{i,\text{LLT}}^l \in \mathbb{R}^d$: LLT embeddings from MHA/MLP outputs.

We concatenate $\bar{a}_i^l$ and $\bar{m}_i^l$ to form layer- l context vector for the i -th demonstration $c_i^l = [\bar{a}_i^l; \bar{m}_i^l] \in \mathbb{R}^{2d}$. Stacking all demonstration vectors yields the demonstration set’s layer- l context matrix:

$$C^l = [c_1^l; c_2^l; \dots; c_{|\mathcal{D}|}^l] \in \mathbb{R}^{|\mathcal{D}| \times 2d}. \quad (4)$$

Subsequently, we design a lightweight interaction module based on the self-attention mechanism to model the dependencies among demonstrations. Specifically, we adopt the standard scaled dot-product self-attention (Vaswani et al., 2017) with trainable parameters. For the l -th layer’s context vector $C^l \in \mathbb{R}^{|\mathcal{D}| \times 2d}$, we project it into Query (Q), Key (K), and Value (V) vectors using trainable weight matrices $W_Q, W_K, W_V \in \mathbb{R}^{|\mathcal{D}| \times |\mathcal{D}|}$. The output is then computed as:

$$C'^l = \text{Softmax} \left(\frac{QK^T}{\sqrt{d}} \right) V, \quad (5)$$

where $Q = C^l W_Q$, $K = C^l W_K$, and $V = C^l W_V$.

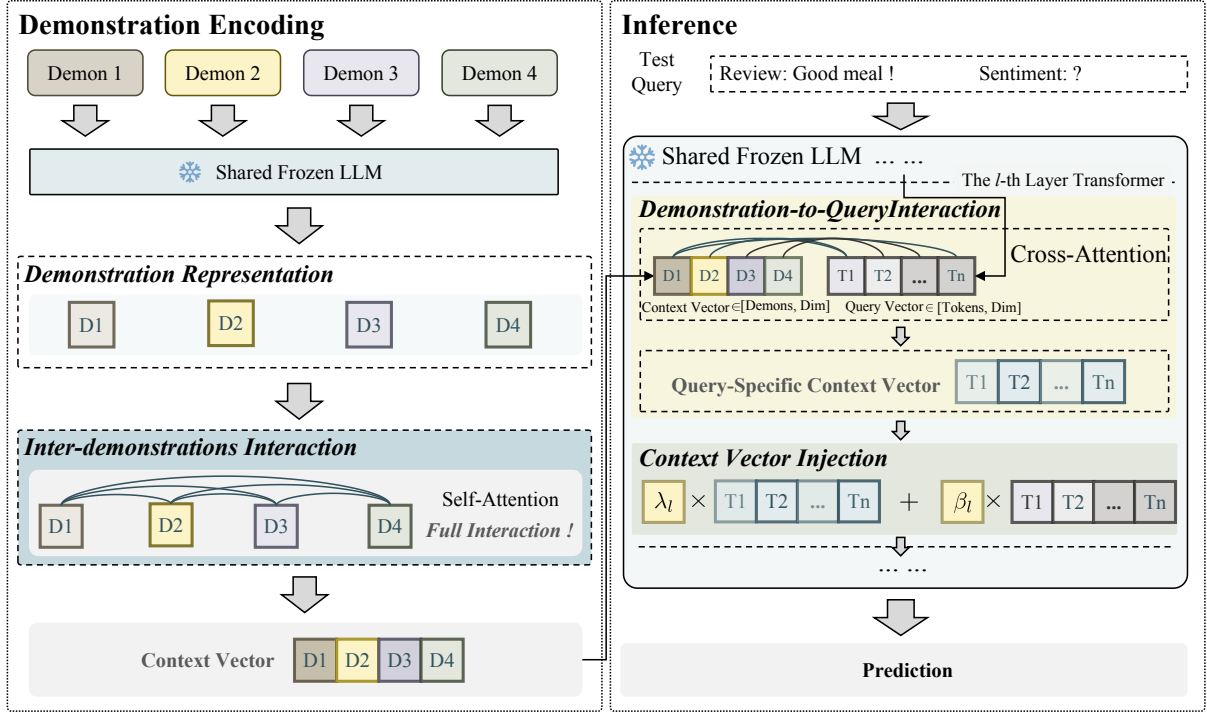


Figure 2: Overview of our Interactive Implicit In-context Learning (I³CL) framework.

The motivation for inter-demonstration interaction stems from the observation that task regularities cannot be fully revealed by a single demonstration; instead, they emerge from the joint implications of multiple demonstrations (Xie et al., 2022; Akyürek et al., 2022). When demonstrations are encoded independently, the model tends to capture only local individual regularities, but fails to establish global collective regularities, i.e., a compressive and discriminative task vector. A detailed analysis of this module is in Section 4.5.2.

3.3 Demonstration-to-Query Interaction

To this end, we employ a cross-attention mechanism to assess the importance of each demonstration $(x_i, y_i) \in \mathcal{D}$ to the test query x_t . Specifically, taking the output of the l -th layer as example, the importance scores from MHA and MLP are computed as follows:

$$\text{score}_{a,x_t}^l = \text{softmax}(a_{x_t}^l \cdot c_a^{l\top}), \quad (6)$$

$$\text{score}_{m,x_t}^l = \text{softmax}(m_{x_t}^l \cdot c_m^{l\top}), \quad (7)$$

where matrices $c_a^l, c_m^l \in \mathbb{R}^{|\mathcal{D}| \times d}$ represent the corresponding MHA and MLP parts extracted from the demonstration set’s context matrix C^l ; $a_{x_t}^l, m_{x_t}^l \in \mathbb{R}^{s \times d}$ denote the MHA and MLP outputs of the test query x_t .

Overall, this equation derives normalized importance scores that quantify the relevance of each demonstration relative to each token in the test query. For example, score_{a,x_t}^l is derived as follows: The dot product $a_{x_t}^l \cdot c_a^{l\top}$ computes pairwise token-demonstration similarities. These raw similarities are normalized via softmax applied along the demonstration axis ($|\mathcal{D}|$), yielding the matrix $\text{score}_{a,x_t}^l \in \mathbb{R}^{s \times |\mathcal{D}|}$. Each element $\text{score}_{a,x_t}^l[i, j]$ quantifies the relevance of the j -th demonstration to the i -th token in x_t .

Based on these importance scores, we construct query-specific context vectors as follows:

$$c_{a,x_t}^{*l} = \text{score}_{a,x_t}^l \cdot c_a^l, \quad (8)$$

$$c_{m,x_t}^{*l} = \text{score}_{m,x_t}^l \cdot c_m^l, \quad (9)$$

where c_{a,x_t}^{*l} and $c_{m,x_t}^{*l} \in \mathbb{R}^{s \times d}$ denote the query-specific context vectors for MHA and MLP, respectively.

These vectors are then injected as follows:

$$\begin{aligned} o_{x_t}^l &= o_{x_t}^{l-1} \\ &+ (\lambda_a^l \cdot c_{a,x_t}^{*l} + \beta_a^l \cdot a_{x_t}^l) \\ &+ (\lambda_m^l \cdot c_{m,x_t}^{*l} + \beta_m^l \cdot m_{x_t}^l) \end{aligned} \quad (10)$$

where $\lambda_a^l, \beta_a^l, \lambda_m^l, \beta_m^l$ are learnable coefficients weighting the query-specific context vectors and

the original query vectors at the l -th layer; $o_{x_t}^l \in \mathbb{R}^{s \times d}$ is the layer’s final output for the query, formed by adding the previous layer’s output $o_{x_t}^{l-1}$ via a residual connection.

3.4 Coefficient Calibration

Scale mismatches between the fused query vectors and the LLM’s original distribution can cause inference difficulties. To address this, the coefficient calibration serves to adapt the fused vectors for compatibility with the LLM’s pre-trained parameters. Following I²CL (Li et al., 2025), we employ a parameter-efficient fine-tuning (PEFT) strategy to optimize these coefficients. Specifically, we freeze all pre-trained parameters of the LLM and update only the following two sets of parameters via gradient descent: (1) the parameters of the interaction modules (e.g., the weight matrices $W_Q, W_K, W_V \in \mathbb{R}^{|\mathcal{D}| \times |\mathcal{D}|}$ in self-attention), and (2) the coefficients λ and β . Therefore, the total number of trainable parameters is $3|\mathcal{D}|^2 + 4L$. The optimization objective is to minimize the perplexity of the label tokens over the demonstration set $\mathcal{D}$, with the loss function defined as follows:

$$\mathcal{L} = -\frac{1}{|\mathcal{D}|} \sum_{(x,y) \in \mathcal{D}} \log P(y|x, \lambda, \beta, O_x^*) \quad (11)$$

where $P(y|x, \lambda, \beta, O_x^*)$ denotes the LLM’s predictive probability of label y for the test query x ; O_x^* denotes the last layer’s output of the query; λ and β denotes the learnable coefficients.

This optimization constitutes an offline calibration process: the fusion coefficients λ and β are calibrated via this task-specific demonstration set $\mathcal{D}$ in a bootstrapping training stage. Once calibrated through this procedure, these coefficients remain fixed during inference. Importantly, for tasks that require the model to generate multiple tokens, our approach injects contextual vectors at each decoding step. This ensures every generated token benefits from the in-context guidance, preserving coherence throughout long-form generation.

4 Experiments

4.1 Setup

4.1.1 Datasets

Following (Li et al., 2025), we evaluate our method on eight classification datasets. Additionally, we include a text generation task: text style transfer. Following the work of ICV (Liu et al., 2024), we

test on the ParaDetox dataset (Logacheva et al., 2022), where the goal is to rewrite toxic sentences into neutral expressions while preserving meaning. This task demands precise context understanding and controlled generation capabilities. Detailed descriptions and statistics of the datasets are provided in Appendix A.1.

4.1.2 Compared Baselines

Baselines are categorized into three groups (from top to bottom in Table 1): basic LLM methods, task-vector-based methods and prompt compression methods. Since I²CL uses random sampling on the test set without releasing the specific sampled instances, we replicate their sampling strategy to reconstruct the test set and reproduce all baselines under this setup. Descriptions and implementation details of the baselines are provided in Appendix A.2.

4.1.3 LLM Backbones

We evaluate I³CL on: LLaMA2-7B (Touvron et al., 2023), LLaMA3-8B (Dubey et al., 2024), GLM4-9B (Zeng et al., 2024) and DeepSeek-R1 (Guo et al., 2025). These four models are selected because of their prevalent use in prior works (Hendel, 2023; Liu et al., 2024; Li et al., 2025).

4.2 Implementation details

In the few-shot setting, we follow (Li et al., 2025) and randomly sample five demonstrations per class to mitigate majority-label bias and obtain more robust few-shot performance. During evaluation, we report the average accuracy across eight tasks under five different random seeds. For calibration, we optimize the coefficients for 100 epochs using the AdamW optimizer (Loshchilov and Hutter, 2017) on the same demonstration set. Hyperparameter settings, task instructions and manually designed prompt templates are provided in Appendix A.3.

4.3 Main Result

The performance comparison across different classification tasks is summarized in Table 1. Among the basic LLM methods, LoRA fine-tuning generally achieves the best results, but at the cost of significant computational overhead due to parameter updates. Among task-vector-based methods, I²CL delivers strong performance across these datasets. However, its reliance on independent demonstration encoding and average pooling limits its ability to capture the interactive mechanisms that drive the

Method	SST-2	SST-5	TREC	AGNews	HS18	DBPedia	MR	Subj
Zero-Shot	89.16	41.04	41.96	45.60	76.44	69.44	76.88	52.52
ICL	91.00	44.32	60.08	85.00	77.92	71.48	89.56	56.24
LoRA-FT	92.15 $\pm$ 1.16	<u>48.44</u> $\pm$ 1.10	<u>85.36</u> $\pm$ 1.11	86.04 $\pm$ 1.30	82.48 $\pm$ 0.99	<u>95.36</u> $\pm$ 0.78	85.72 $\pm$ 1.23	<u>87.48</u> $\pm$ 1.39
Noise-Vector	51.80 $\pm$ 2.04	23.04 $\pm$ 2.90	51.28 $\pm$ 2.89	26.80 $\pm$ 1.33	54.92 $\pm$ 3.17	8.64 $\pm$ 0.90	52.44 $\pm$ 1.40	17.88 $\pm$ 1.33
Label-Anchor	86.00 $\pm$ 0.84	38.96 $\pm$ 1.43	80.36 $\pm$ 1.43	83.52 $\pm$ 1.74	58.72 $\pm$ 2.10	73.68 $\pm$ 2.39	71.32 $\pm$ 1.85	56.44 $\pm$ 1.79
Task-Vector	84.92 $\pm$ 0.91	36.48 $\pm$ 2.04	71.92 $\pm$ 2.32	73.52 $\pm$ 2.18	65.00 $\pm$ 1.66	76.48 $\pm$ 1.90	84.80 $\pm$ 1.07	56.04 $\pm$ 1.30
ICV	90.16 $\pm$ 1.04	35.36 $\pm$ 2.19	55.52 $\pm$ 1.97	73.28 $\pm$ 1.95	73.68 $\pm$ 2.04	90.08 $\pm$ 1.99	85.12 $\pm$ 1.27	59.08 $\pm$ 2.21
I ² CL	90.28 $\pm$ 1.32	43.76 $\pm$ 1.73	75.44 $\pm$ 1.92	84.24 $\pm$ 1.27	76.40 $\pm$ 2.04	91.87 $\pm$ 1.05	85.52 $\pm$ 1.56	76.36 $\pm$ 1.32
ICL-TV	<u>92.60</u> $\pm$ 1.64	44.58 $\pm$ 1.25	82.69 $\pm$ 1.84	85.04 $\pm$ 1.71	<u>83.25</u> $\pm$ 1.32	93.04 $\pm$ 1.41	87.23 $\pm$ 1.67	85.21 $\pm$ 1.29
AutoComp	91.11	27.35	59.15	85.74	58.84	91.18	84.33	61.51
ICAE	89.79	40.25	51.04	80.47	67.73	59.93	86.77	50.44
CEPE	75.83	35.35	55.36	79.44	60.83	85.56	82.75	59.34
I ³ CL (Ours)	92.66 $\pm$ 0.90	51.16 $\pm$ 1.97	87.36 $\pm$ 1.14	87.07 $\pm$ 1.33	86.28 $\pm$ 1.65	95.88 $\pm$ 0.57	90.67 $\pm$ 0.93	90.84 $\pm$ 1.88

Table 1: Accuracy comparison between our approach and baseline methods on LLaMA2-7B across eight classification tasks under the 5-shot few-shot setting. The best results are highlighted in bold, and the second-best results are underlined.

Method	Tox. (%)	R-1	BERT
Original Test Set	84.58	-	-
Gold-standard Reference	16.23	-	-
Zero-Shot	71.60	73.15	93.32
ICL	66.81	74.19	93.11
LoRA FT	48.94	57.32	89.34
ICV	39.54	65.97	92.73
I ² CL	57.56	70.55	90.79
ICL-TV	55.75	71.34	90.86
I ³ CL (Ours)	24.80	69.19	91.74

Table 2: Performance comparison on ParaDetox using LLaMA2-7B with 5-shot demonstrations. Tox., R-1, and BERT denote toxicity score, ROUGE-1 and BERT-Score respectively.

success of standard ICL. This limitation is particularly evident in I²CL’s underperformance relative to standard ICL on datasets such as SST-2, SST-5 and MR, suggesting that its task vectors lack sufficient expressiveness for these datasets.

ICL-TV improves expressiveness by extracting and injecting multiple task vectors. It consistently outperforms I²CL across all eight classification datasets, confirming that increasing the representational capacity of task vectors is beneficial. Nevertheless, ICL-TV lags behind I³CL on all evaluated tasks, particularly on SST-5, TREC and Subj. These results indicate that simply increasing the number of task vectors is insufficient; restoring the interactive mechanisms of standard ICL is crucial for constructing more task-adaptive vectors.

Specifically, I³CL outperforms all prior baselines across all eight evaluated datasets. Notably, I³CL even surpasses the performance of LoRA fine-

tuning (LoRA-FT) on several datasets. These results underscore the superiority of I³CL. We also computed the average standard deviation across all eight datasets for the task-vector-based baselines. I³CL achieves the lowest average standard deviation of 1.30, compared with 1.51 (ICL-TV), 1.53 (I²CL), 1.83 (ICV), 1.68 (Task-Vector), 1.70 (Label-Anchor), and 2.00 (Noise-Vector), indicating that our method yields not only stronger but also more stable performance. Results on other backbones are provided in Appendix E.

Results on the text generation task, evaluated using the ParaDetox dataset, are presented in Table 2. I³CL achieves significantly lower toxicity (24.80%) than all baselines, outperforming the current state-of-the-art method ICV (39.54%). It also maintains strong performance in fluency (ROUGE-1) and semantic preservation (BERT-Score), matching or exceeding previous baselines. Moreover, I³CL’s improvement over I²CL highlights that the latter’s static, coarse-grained context aggregation is insufficient for this generative task. This limitation is further evidenced by I²CL’s high average training loss (2.70). ICL-TV achieves performance comparable to that of I²CL. In contrast, I³CL’s dynamic, interaction-aware mechanism results in a lower average loss of 0.67, enabling superior performance on this generative task. These results demonstrate the effectiveness of I³CL on more complex, non-classification tasks.

4.4 Ablation Study

To evaluate the contribution of each component, we extend I²CL with three key modules as ablated

Method / Dataset (Avg Len)	SST-2 246	SST-5 370	TREC 653	AGNews 343	HS18 512	DBPedia 475	MR 366	Subj 637
I ² CL	90.28	43.76	75.44	84.24	76.40	91.87	85.52	76.36
+PLT	91.16	44.08	75.64	84.64	85.36	92.08	86.44	78.60
+Inter-Demo	92.56	44.56	77.16	84.76	82.96	94.48	86.64	77.52
+Demo2Query	92.32	45.32	86.56	86.44	82.28	93.52	87.00	87.20
+ All (I ³ CL)	92.66	51.16	87.36	87.07	86.28	95.88	90.67	90.84
I ³ CL w/o FC	91.40	44.20	77.60	85.27	85.15	92.67	86.27	79.96

Table 3: Accuracy performance across different ablation variants of I³CL, illustrating the impact of each component.

Token Selection			
Strategy	SST-5	TREC	Subj
Last-Token	51.16	87.36	90.84
Random-Token	48.88	85.40	88.80
Max-Pooling	24.56	25.88	51.32
Avg-Pooling	44.08	85.85	80.07
Demonstration Interaction			
Strategy	SST-5	TREC	Subj
Standard Attention	51.16	87.36	90.84
Multi-Head Attention	52.73	87.93	92.53

Table 4: Impact of different strategies for demonstration encoding.

variants: (1) “+PLT” introduces the prompt last token representation during demonstration encoding, (2) “+Inter-Demo” adds inter-demonstration interaction, and (3) “+Demo2Query” incorporates demonstration-to-query interaction. As shown in Table 3, all variants individually outperform the I²CL baseline. Notably, “+PLT” yields consistent gains across all datasets, with substantial improvements on tasks involving longer sequences (e.g., TREC: +4.20, HS18: +6.96, Subj: +4.24), demonstrating their effectiveness in addressing the limited context captured by a single last label token. Among all modules, “+Demo2Query” brings the largest overall improvement, while combining it with “+Inter-Demo” forms the complete I³CL framework, which achieves the best performance. Even when removing the additional fully connected layers used for the q , k , and v projections (“I³CL w/o FC”), the model still surpasses the baseline, confirming that the structural design of I³CL is intrinsically strong, with the FC layers further enhancing generalization.

4.5 Further Analysis

Due to space constraints, analyses on “Necessity of Task Instruction for Demonstration Encoding”, “Representation Robustness” and “Case Study” are

provided in Appendix B-D.

4.5.1 Efficiency Comparison

We evaluate our method and baselines—including zero-shot, conventional ICL and I²CL—under identical settings using the TREC dataset and LLaMA2-7B, tracking calibration time, inference overhead, and peak memory usage. As shown in Table 5, we make three key observations: **(1) Calibration efficiency.** In standard few-shot settings (5-30 shots), I³CL increases $\leq 7\%$ overhead compared to I²CL, demonstrating comparable efficiency. While 100-shot calibration increases by 34% (reflecting interactive complexity), this one-time offline cost is justified by significant accuracy gains (+11.92%). **(2) Inference Speed.** In critical few-shot scenarios (5-30 shots) show $\leq 1.3\%$ increase in inference latency. Although overhead rises at 100-shot (4.91x vs. I²CL’s 4.08x), it remains dramatically lower than standard ICL. **(3) Memory Efficiency.** I³CL’s peak memory usage closely matches I²CL’s. Crucially, I³CL avoids Out-Of-Memory (OOM) errors in all tested scenarios, unlike standard ICL which fails at 100-shot. This demonstrates I³CL’s superior scalability.

4.5.2 I²CL vs. I³CL Task Vectors

We sample distinct demonstration sets from two separate datasets (A and B) and compute the cosine similarity for every pair of task vectors. Figure 3 shows the results for I²CL (left side) and I³CL (right side); bold diagonal values indicate intra-task similarity.

We observe that I²CL exhibits high inter-task similarity (typically exceeding 85%) alongside its high intra-task consistency ($>98\%$ diagonal). This suggests contamination by task-irrelevant features, hindering clear task separation. In contrast, I³CL captures more distinct task-specific patterns, maintaining strong intra-task consistency ($>95\%$ diagonal) while achieving sharp inter-task discrimination. For unrelated tasks (e.g., SST-2 vs. TREC), simi-

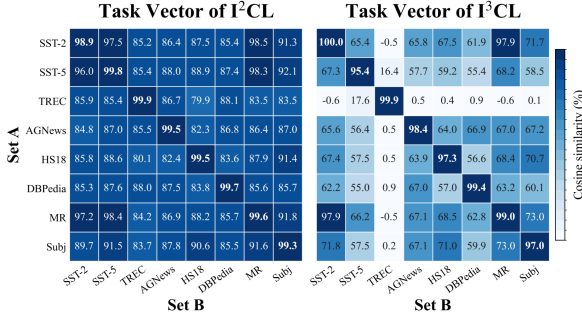


Figure 3: Pairwise cosine similarity (%) between task vectors (Left: I^2CL , Right: I^3CL).

larity drops to near-zero levels. For related tasks (e.g., SST-2 vs. SST-5) with differing classification goal, similarity remains moderated (65.40%), significantly lower than I^2CL 's 97.54%. These results highlight I^3CL 's ability to encode more nuanced, task-adaptive vectors.

Method	Acc (5-Shot)	5-Shot	10-Shot	30-Shot	100-Shot
Calibration Time Comparison					
I^2CL	75.44	1x	1x	1x	1x
ICL-TV	82.69	1.04x	1.06x	1.07x	1.10x
I^3CL (Ours)	87.36	1.02x	1.03x	1.07x	1.34x
Inference Time Comparison					
Zero-Shot	41.96	1x	1x	1x	1x
ICL	60.08	5.84x	9.93x	36.66x	— (OOM)
I^2CL	75.44	4.08x	4.08x	4.08x	4.08x
ICL-TV	82.69	4.08x	4.08x	4.08x	4.08x
I^3CL (Ours)	87.36	4.10x	4.13x	4.27x	4.91x
Peak Memory Usage					
Zero-Shot	41.96	1x	1x	1x	1x
ICL	60.08	1.59x	2.15x	5.84x	— (OOM)
I^2CL	75.44	2.04x	2.04x	2.04x	2.04x
ICL-TV	82.69	2.13x	2.13x	2.14x	2.18x
I^3CL (Ours)	87.36	2.15x	2.25x	2.45x	2.83x

Table 5: Calibration time, Inference Time and Peak memory usage comparison under different shots.

4.5.3 Scalability Across Shot Numbers

To investigate the scalability of I^3CL , we increase the number of shots and compare its performance with the baselines I^2CL , ICL-TV and standard ICL. The results are shown in Figure 4. We observe that although I^3CL is designed for few-shot settings, its performance continues to improve with more demonstrations and peaks around 30 shots, indicating that it not only excels in low-data regimes but also scales effectively. In contrast, I^2CL shows marginal gains as the shot number increases. Stan-

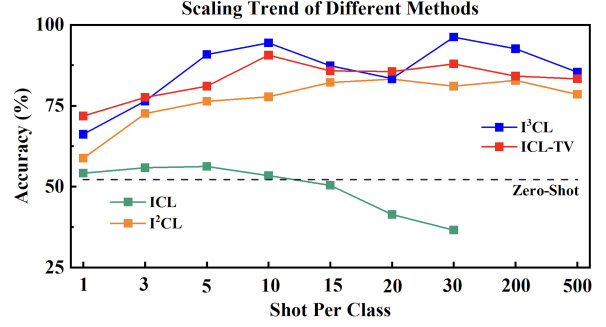


Figure 4: Scalability across shot numbers for different methods, tested on the Subj dataset using LLaMA2-7B.

dard ICL performs even worse and often degrades with too many demonstrations, revealing both inefficiency and instability. ICL-TV outperforms I^2CL , underscoring the importance of enhancing task-vector expressiveness. Nevertheless, I^3CL generally surpasses ICL-TV across different shot numbers, demonstrating its superior ability to integrate information across multiple demonstrations.

4.5.4 Cross-Dataset Transferability of Calibrated Parameters

As shown in Table 6, we study whether the calibrated parameters transfer across datasets by calibrating I^2CL and I^3CL on SST-5 and then freezing all calibrated parameters before evaluating on SST-2, TREC, and Subj. The subscript denotes the absolute accuracy drop relative to calibration directly on the corresponding target dataset. I^3CL transfers relatively well to SST-2 and Subj compared with I^2CL , but suffers a larger degradation on TREC. These results indicate that the calibrated parameters remain task-dependent: related tasks may share some transferable structure, whereas tasks with substantially different semantics generally require recalibration. This observation is consistent with the task-vector similarity analysis in the Section 4.5.2, which shows that I^3CL produces more task-specific and discriminative representations.

4.5.5 Impact of Different Strategies for Demonstration Encoding

We explore alternative strategies for token selection and demonstration interaction modeling during demonstration encoding. As illustrating in Table 4, for the token selection, we experimented with the following strategies: (1) keeping only the last token from both the prompt and label segments (“Last-Token”), (2) randomly selecting a token from each segment (“Random-Token”), (3)

Method	SST-2	TREC	Subj
I ² CL	87.36 _{↓2.92}	61.84 _{↓13.60}	54.32 _{↓22.04}
I ³ CL	90.96 _{↓1.70}	60.91 _{↓26.45}	79.36 _{↓11.48}

Table 6: Cross-dataset transfer performance when calibrated on SST-5. All calibrated parameters are fixed and directly transferred to the target datasets.

selecting the token with the highest weight from each segment (Max-Pooling), and (4) averaging all tokens into a single representation (Avg-Pooling). The results show that the “Last-Token” strategy performs best, significantly outperforming “Random-Token”, “Max-Pooling”, and “Avg-Pooling”. This suggests that the final token position in both segments carries more discriminative task-related information in the context. For the interaction modeling strategy, Multi-Head Attention clearly outperforms the standard attention mechanism, achieving consistent improvements on the three tasks. This demonstrates the superior expressive power of the multi-head mechanism in integrating information across demonstrations. Together, these results validate our design choices.

5 Conclusion

In this paper, we build upon implicit ICL (I²CL) and introduce Interactive Implicit In-Context Learning (I³CL), a simple yet effective framework that incorporates two lightweight interaction modules to restore the core capabilities of standard ICL—namely, rich interactions between demonstrations and between demonstrations and queries. Experiments across diverse LLMs and datasets demonstrate that our method delivers substantial improvements over I²CL with minimal computational overhead. Extensive ablation study validates the effectiveness of our key designs. Further analysis confirms that I³CL encodes more discriminative, task-adaptive vectors and scales effectively across varying numbers of shots.

6 Limitations

Similar to existing task-vector-based methods, I³CL still has several limitations. First, it is less flexible in dynamic retrieval-based scenarios, where the demonstration set may vary across test queries. In such cases, the task vectors and the associated interaction representations may need to be recomputed frequently, which may reduce inference efficiency. Second, task-vector injection can

introduce instability in generation tasks. We observe that injection-based methods, including ICV, I²CL, and I³CL, may occasionally produce repetitive, incoherent, or malformed outputs. A possible reason is that externally injected vectors perturb the LLM’s original hidden states and generation distribution. Although calibration can alleviate this issue, it does not eliminate it completely.

7 Acknowledgments

This research was partially supported by the Key Research and Development Project in Shaanxi Province No. 2024PT-ZCK-89, the National Natural Science Foundation of China Nos. 62406242, 62672397 and 62476215, and Ant Group.

References

- Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. 2022. What learning algorithm is in-context learning? investigations with linear models. *arXiv preprint arXiv:2211.15661*.
- Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.
- Kaiyan Chang, Songcheng Xu, Chenglong Wang, Yingfeng Luo, Xiaoqian Liu, Tong Xiao, and Jingbo Zhu. 2024. Efficient prompting methods for large language models: A survey. *arXiv preprint arXiv:2404.01077*.
- Brian K Chen, Tianyang Hu, Hui Jin, Hwee Kuan Lee, and Kenji Kawaguchi. 2024. Exact conversion of in-context learning to model weights in linearized-attention transformers. In *International Conference on Machine Learning*, pages 6833–6846. PMLR.
- Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. 2023. Adapting language models to compress contexts. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3829–3846.
- Ona de Gibert, Naiara Pérez, Aitor García-Pablos, and Montse Cuadros. 2018. Hate speech dataset from a white supremacy forum. In *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, pages 11–20.
- Yuxin Dong, Jiachen Jiang, Zhihui Zhu, and Xia Ning. 2026. [Understanding task vectors in in-context learning: Emergence, functionality, and limitations](#). In *The Fourteenth International Conference on Learning Representations*.

- Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *CoRR*.
- Caoyun Fan, Jidong Tian, Yitian Li, Hao He, and Yaohui Jin. 2024. Comparable demonstrations are important in in-context learning: A novel perspective on demonstration selection. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 10436–10440. IEEE.
- Tao Ge, Jing Hu, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. 2023. In-context autoencoder for context compression in a large language model. *arXiv preprint arXiv:2307.06945*.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shitong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.
- Amir Hendel, Roei. 2023. In-context learning creates task vectors. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9318–9333.
- Jiarui Jin, Yuwei Wu, Haoxuan Li, Xiaoting He, Weinan Zhang, Yiming Yang, Yong Yu, Jun Wang, and Mengyue Yang. 2025. Large language models are demonstration pre-selectors for themselves. *arXiv e-prints*, pages arXiv–2506.
- Yingcong Li, Kartik Sreenivasan, Angeliki Gianou, Dimitris Papailiopoulos, and Samet Oymak. 2023. Dissecting chain-of-thought: Compositionality through in-context filtering and learning. *Advances in Neural Information Processing Systems*, 36:22021–22046.
- Zhuowei Li, Zihao Xu, Ligong Han, Yunhe Gao, Song Wen, Di Liu, Hao Wang, and Dimitris N Metaxas. 2025. Implicit in-context learning.
- Jiayu Liu, Zhenya Huang, Chaokun Wang, Xunpeng Huang, ChengXiang Zhai, and Enhong Chen. What makes in-context learning effective for mathematical reasoning. In *Forty-second International Conference on Machine Learning*.
- Sheng Liu, Haotian Ye, Lei Xing, and James Y Zou. 2024. In-context vectors: Making in context learning more effective and controllable through latent space steering. In *International Conference on Machine Learning*, pages 32287–32307. PMLR.
- Varvara Logacheva, Daryna Dementieva, Sergey Ustyantsev, Daniil Moskovskiy, David Dale, Irina Krotova, Nikita Semenov, and Alexander Panchenko. 2022. Paradetox: Detoxification with parallel data. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6804–6818.
- Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.
- Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8086–8098.
- Man Luo, Xin Xu, Yue Liu, Panupong Pasupat, and Mehran Kazemi. 2024. In-context learning with retrieved demonstrations for language models: A survey. *arXiv preprint arXiv:2401.11624*.
- B PANG. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL)*, 2004.
- Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL’05)*, pages 115–124.
- Keqin Peng, Liang Ding, Yancheng Yuan, Xuebo Liu, Min Zhang, Yuanxin Ouyang, and Dacheng Tao. 2024. Revisiting demonstration selection strategies in in-context learning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9090–9101.
- Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2022. Learning to retrieve prompts for in-context learning. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2655–2671.
- Baturay Saglam, Zhuoran Yang, Dionysis Kalogerias, and Amin Karbasi. Learning task representations from in-context learning. In *ICML 2024 Workshop on In-Context Learning*.
- Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and 1 others. 2022. Lora: Low-rank adaptation of large language models.
- Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Ellen M Voorhees and Dawn M Tice. 2000. Building a question answering test collection. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 200–207.

Xingchen Wan, Ruoxi Sun, Hanjun Dai, Sercan Arik, and Tomas Pfister. 2023. Better zero-shot reasoning with self-adaptive prompting. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 3493–3514.

Lean Wang, Lei Li, Damai Dai, Deli Chen, Hao Zhou, Fandong Meng, Jie Zhou, and Xu Sun. 2023. 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.

Ziqi Wang, Hanlin Zhang, Xiner Li, Kuan-Hao Huang, Chi Han, Shuiwang Ji, Sham M Kakade, Hao Peng, and Heng Ji. Eliminating position bias of language models: A mechanistic approach. In *NeurIPS 2024 Workshop on Scientific Methods for Understanding Deep Learning*.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. 2022. An explanation of in-context learning as implicit bayesian inference. In *ICLR*.

Howard Yen, Tianyu Gao, and Danqi Chen. 2024. Long-context language modeling with parallel context encoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2588–2610.

Fan Yin, Jesse Vig, Philippe Laban, Shafiq Joty, Caiming Xiong, and Chien-Sheng Wu. 2023. Did you read the instructions? rethinking the effectiveness of task definitions in instruction learning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3063–3079.

Safoora Yousefi, Leo Betthauser, Hosein Hasanbeig, Raphaël Millièvre, and Ida Momennejad. 2023. Decoding in-context learning: Neuroscience-inspired analysis of representations in large language models. *arXiv preprint arXiv:2310.00313*.

Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Diego Rojas, Guanyu Feng, Hanlin Zhao, Hanyu Lai, Hao Yu, and 1 others. 2024. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *CoRR*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28.

Yufeng Zhang, Xuepeng Wang, Lingxiang Wu, and Jinqiao Wang. 2025. Enhancing chain of thought prompting in large language models via reasoning patterns. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 25985–25993.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *International conference on machine learning*, pages 12697–12706. PMLR.

Dataset	#Classes	Train	Test
SST-2	2	67,349	1,821
SST-5	5	8,544	2,210
TREC	6	5,452	500
AGNews	4	120,000	7,600
HS18	2	8,755	2,189
DBPedia	14	560,000	70,000
MR	2	9,596	1,066
Subj	2	8,000	2,000
ParaDetox	–	18,777	671

Table 7: Statistics of the datasets used in the main experiments.

Model	Batch Size	Learning Rate
LLaMA2-7B	2	$2e^{-4}$
LLaMA3-8B	2	$2e^{-5}$
GLM4-9B	2	$2e^{-6}$
DeepSeek-R1	2	$2e^{-4}$

Table 8: The hyperparameter configurations for LLMs. “Batch Size” refers to the training batch size, while “Learning Rate” denotes the initial learning rates for LLMs, which is decayed gradually during training.

A Supplementary Experimental Setup

A.1 Datasets

We first adopt the eight classification tasks used in Li et al. (2025), covering a diverse range of benchmarks: sentiment analysis on SST-2 and SST-5 (Socher et al., 2013), question classification on TREC (Voorhees and Tice, 2000), topic classification on AGNews and DBPedia (Zhang et al., 2015), movie review classification on MR (Pang and Lee, 2005), subjectivity classification on Subj (PANG, 2004), and hate speech detection on hate.speech18 (HS18) (de Gibert et al., 2018). For fair comparison, we follow the setting of I²CL: except for DBPedia, five demonstrations are randomly sampled per class from the official training split and are used for both context representation construction and parameter calibration. This results in 10

Method	SST-2	SST-5	TREC	AGNews	HS18	DBPedia	MR	Subj
I ³ CL	92.66	51.16	87.36	87.07	86.28	95.88	90.67	90.84
w/o DI	91.20	49.00	86.40	85.68	84.20	93.40	87.82	82.68

Table 9: Ablation study on the effect of encoding task instructions for each demonstration. “w/o DI” indicates that task instructions are removed from the demonstrations.

Method	SST-2	SST-5	TREC	AGNews	HS18	DBPedia	MR	Subj
Zero-Shot	88.40	47.84	67.84	70.60	63.00	78.60	82.62	71.96
ICL	92.49	50.76	72.24	74.80	52.86	87.67	87.27	74.53
LoRA-FT	94.26	<u>52.48</u>	<u>85.64</u>	87.63	86.01	96.83	86.21	<u>91.22</u>
Noise-Vector	51.83	29.23	24.47	27.28	57.81	7.87	52.20	47.42
Label-Anchor	92.45	45.86	34.62	69.84	73.93	84.28	73.75	54.83
Task-Vector	91.85	45.65	54.27	85.55	80.60	67.66	82.47	53.82
ICV	88.70	40.68	58.62	84.58	90.15	92.83	81.27	51.43
I ² CL	89.63	48.64	76.00	85.83	90.60	93.47	82.85	77.12
ICL-TV	91.24	49.27	77.51	85.26	<u>90.79</u>	93.89	83.42	80.26
I ³ CL (Ours)	<u>93.43</u>	53.60	88.28	<u>85.89</u>	91.60	<u>94.16</u>	88.20	92.08

Table 10: Performance comparison on LLaMA3-8B.

calibration examples for SST-2, HS18, MR, and Subj, 25 for SST-5, 30 for TREC, and 20 for AGNews. Since DBPedia contains 14 classes, we sample one demonstration per class, for a total of 14 examples. Different random seeds correspond to different demonstration samples.

To further extend our evaluation beyond classification, we include ParaDetox (Logacheva et al., 2022), a generation-style task used in Liu et al. (2024), where comments flagged for toxicity are paired with non-toxic paraphrases that preserve the original meaning in a more neutral manner. Table 7 summarizes the dataset statistics used in our experiments.

For classification tasks, we follow the evaluation protocol of Li et al. (2025). Specifically, I²CL (Li et al., 2025) constructs each task-specific test set by randomly sampling 500 instances from the original test split. However, since neither the sampled instances nor the random seed are publicly released, we resample 500 test instances for each task following the same protocol. To ensure fair comparison, all baselines are reproduced and evaluated on the same resampled test sets. We will release the sampled test data together with our source code to facilitate reproducibility. For the generation-style task ParaDetox, we use the same experimental setting as Liu et al. (2024).

A.2 Reproduction of Baseline Methods

We categorize the baseline methods into three groups: basic LLM methods, including Zero-Shot,

standard ICL, and LoRA-FT; task-vector-based methods (TV-based Methods), including Noise-Vector, Label-Anchor, Task-Vector, ICV, I²CL, and ICL-TV; and prompt compression methods, including AutoComp, ICAE, and CEPE. The reproduction details are as follows.

Zero-Shot (Li et al., 2025): We directly perform inference on the test samples using the LLM without any task-specific demonstrations or fine-tuning.

ICL (Li et al., 2025): For each class, we randomly sample five demonstrations from the training set and prepend them to the prompt to condition the LLM for inference on the test samples.

LoRA-FT (Shen et al., 2022): Following the same demonstration setting as ICL, we fine-tune the LLM with LoRA using the sampled demonstrations. In this baseline, we use LLaMA2-7B, LLaMA3-8B, GLM-4-9B, and DeepSeek-R1-8B as backbone models and fine-tune them with LoRA implemented in the PEFT framework. AdamW is used as the optimizer. For all models, both the LoRA rank and α are set to 8, bias parameters are not trained, and the weight decay is set to 1×10^{-3} . The batch size and learning rate are adjusted for each backbone model on the validation set, as reported in Table 8.

Noise-Vector (Li et al., 2025): Following I²CL (Li et al., 2025), we replace the learned context vectors with random noise vectors while keeping all other settings identical to I²CL.

Label-Anchor (Wang et al., 2023): We evaluate the official implementation released at <https://github.com/label-anchoring/label-anchoring>.

Method	SST-2	SST-5	TREC	AGNews	HS18	DBPedia	MR	Subj
Zero-Shot	94.20	50.84	69.60	86.21	83.43	88.83	89.28	54.04
ICL	95.45	51.64	77.88	86.67	84.21	97.40	89.41	61.16
LoRA-FT	<u>95.61</u>	<u>51.65</u>	93.09	91.19	<u>85.18</u>	97.85	91.40	<u>82.21</u>
Noise-Vector	48.27	23.64	18.88	25.81	47.82	7.81	54.62	55.18
Label-Anchor	90.88	49.46	53.43	48.41	71.64	83.29	81.04	53.80
Task-Vector	95.85	48.42	51.27	68.64	74.41	70.87	82.25	62.80
ICV	94.36	49.68	69.20	84.46	80.81	96.21	86.83	63.27
I ² CL	94.63	51.00	77.84	88.60	81.49	97.86	88.10	59.60
ICL-TV	95.49	51.51	78.89	88.83	84.78	<u>97.94</u>	89.42	59.83
I ³ CL (Ours)	95.92	52.36	<u>92.76</u>	<u>89.66</u>	87.81	98.24	<u>90.04</u>	82.40

Table 11: Performance comparison on GLM4-9B.

Method	SST-2	SST-5	TREC	AGNews	HS18	DBPedia	MR	Subj
Zero-Shot	86.63	30.80	45.40	80.21	80.27	87.31	73.81	53.64
ICL	91.10	39.28	68.40	81.22	82.26	90.80	85.25	55.56
LoRA-FT	<u>91.86</u>	<u>47.51</u>	90.39	<u>88.20</u>	89.30	97.82	86.85	64.83
Noise-Vector	51.80	19.83	19.41	<u>26.86</u>	51.23	8.28	52.37	52.65
Label-Anchor	86.46	45.84	54.29	66.49	56.48	75.65	85.87	50.51
Task-Vector	86.53	29.86	66.80	83.45	83.68	64.97	83.46	47.35
ICV	87.88	28.02	41.87	80.87	85.04	81.87	81.82	63.27
I ² CL	81.84	33.00	73.26	84.63	86.26	88.64	80.82	61.64
ICL-TV	86.83	38.64	78.84	85.62	<u>89.77</u>	92.25	83.84	<u>67.04</u>
I ³ CL (Ours)	91.91	49.55	<u>89.60</u>	89.69	90.74	<u>96.45</u>	<u>86.46</u>	76.00

Table 12: Performance comparison on DeepSeek-R1-8B.

[//github.com/lancopku/label-words-are-anchors](https://github.com/lancopku/label-words-are-anchors) on our tasks.

Task-Vector (Hendel, 2023): We reproduce the task-vector method using the implementation provided in the I²CL repository: <https://github.com/LzVv123456/I2CL>.

ICV (Liu et al., 2024): We reproduce ICV using the implementation provided in the I²CL repository: <https://github.com/LzVv123456/I2CL>.

I²CL (Li et al., 2025): We evaluate the official implementation released at <https://github.com/LzVv123456/I2CL> on our tasks.

ICL-TV (Dong et al., 2026): ICL-TV introduces the idea of constructing multiple task vectors. We refer to its open-source implementation at <https://github.com/Yuxin-Dong/ICL-TaskVector> and reproduce its core procedure within the I²CL framework for a fair comparison with task-vector-based methods.

AutoComp (Chevalier et al., 2023): We evaluate the official implementation released at <https://github.com/princeton-nlp/AutoCompressors> on our tasks.

ICAE (Ge et al., 2023): We evaluate the official implementation released at <https://github.com/getao/icae> on our tasks.

CEPE (Yen et al., 2024): We evaluate the offi-

cial implementation released at <https://github.com/princeton-nlp/CEPE> on our tasks.

A.3 Implementation Details

We implement all experiments using PyTorch and HuggingFace Transformers, and run them on the NVIDIA A800 80GB GPU. For classification datasets, we follow Li et al. (2025) and adopt the manually designed task instructions and prompt templates shown in Figure 5. For ParaDetox, we follow Liu et al. (2024) and sample five toxic-neutral sentence pairs as in-context demonstrations.

During calibration, we freeze the backbone LLM and update only the lightweight parameters introduced by I³CL, including the projection matrices in the interaction modules and the injection coefficients. The backbone-specific batch sizes and learning rates used in the main experiments are reported in Table 8.

B Necessity of Task Instruction for Demonstration Encoding

To preserve the semantic completeness of each demonstration, I³CL encodes each example in isolation, requiring the task instruction I to supply essential contextual grounding. Without I , the LLM often struggles to relate the demonstration’s con-

Test Samples	Per	Num	Desc	Loc	Abbrev	Entity
Q: What state has the least amount of rain per year? (Location)	0.136	0.170	0.165	0.168	0.154	0.167
Q: Who was the first African American to play for the Brooklyn Dodgers? (Person)	0.181	0.166	0.162	0.164	0.171	0.165
Q: What is pilates? (Description)	0.166	0.159	0.184	0.160	0.168	0.182
Q: When was the Boston tea party? (Number)	0.139	0.188	0.169	0.182	0.155	0.167
Q: What is a mirror made out of? (Entity)	0.165	0.167	0.167	0.166	0.165	0.169
Q: What is TUJ? (Abbreviation)	0.169	0.166	0.165	0.162	0.172	0.166

Table 13: Case Study: Attention Weights (Post-Softmax) for Demonstrations Across Test Samples

Task	Prompt Template
SST-2, SST-5, MR	<Start of Instruction> You are given a list of sentiment labels: {label_list}. \n Your task is to analyze the given sentence and determine the most appropriate sentiment label from the list based on its context. <End of Instruction> \n \n <Query> Given Sentence: {text} \n Please determine which sentiment of {label_list} based on the context. \n ASSISTANT: The sentiment of '{text}' is
AGNews	<Start of Instruction> You are a news classifier. Given a news, classify it into one of the following types: {label_list}. <End of Instruction> \n \n <Query> Given News: {text} \n Please determine which type of {label_list} for the given news based on the context. \n ASSISTANT: The type of '{text}' is
DBpedia	<Start of Instruction> You are an AI language model trained to determine the semantic type of the entity mentioned in the sentence. Classify the entity into one of 14 types: {label_list} <End of Instruction> \n \n <Query> Given Sentence: {text} \n Entity: {aspect} \n Please determine which type of {label_list} for the entity based on the context. \n ASSISTANT: The type of '{aspect}' is
HateSpeech	<Start of Instruction> You are an AI language model trained to detect hate speech language. Classify the following sentence into one of two categories: {label_list}. <End of Instruction> \n \n <Query> Given Sentence: {text} \n Please determine which category of {label_list} based on the context. \n ASSISTANT: The category of '{text}' is
Subj	<Start of Instruction> You are an AI model trained to complete the subjectivity classification. Classify the following sentence into one of two subjectivity types: {label_list}. <End of Instruction> \n \n <Query> Given Sentence: {text} \n Please determine which subjectivity type of {label_list} for the given sentence based on the context. \n ASSISTANT: The subjectivity type of '{text}' is
TREC	<Start of Instruction> You are a question classifier. Given a question, classify it into one of the following types: {label_list}. <End of Instruction> \n \n <Query> Given Question: {text} \n Please determine which type of {label_list} for the given question based on the context. \n ASSISTANT: The type of the question is

Figure 5: Prompt templates specific to each task are employed consistently across all experiments.

tent to the intended task objective. As shown in Table 9, removing per-demonstration instructions (w/o DI) consistently leads to performance degradation, with the average accuracy dropping from 85.24% to 82.55%. This confirms that the instruction I plays a crucial role in enabling the model to extract task-relevant and high-quality representations from demonstrations.

C Representation Robustness

Figure 6 compares the representations of I^2 CL and our proposed I^3 CL on the Subj and TREC tasks under 0% and 50% noise. Here, “noise” refers to standard Gaussian noise vectors injected into the context vectors at varying magnitudes to simulate irrelevant information in the demonstration set relative to the current query, allowing us to evaluate the robustness of the methods. I^2 CL produces en-

tangled clusters and degrades significantly under noisy conditions, whereas I^3 CL consistently yields more compact and well-separated clusters, maintaining clear decision boundaries even with high noise. This demonstrates that I^3 CL achieves better generalization and robustness, as it leverages attention mechanisms to adaptively select the most suitable demonstration vectors for prediction.

D Case Study

To illustrate how the softmax-based attention mechanism in I^3 CL guides the selection of relevant demonstrations for a given test sample, we conducted a focused case study on six test questions from the TREC dataset. The demonstration set was carefully constructed to cover six distinct question categories, including Person, Number, Description, Location, Abbreviation, and Entity, with one repre-

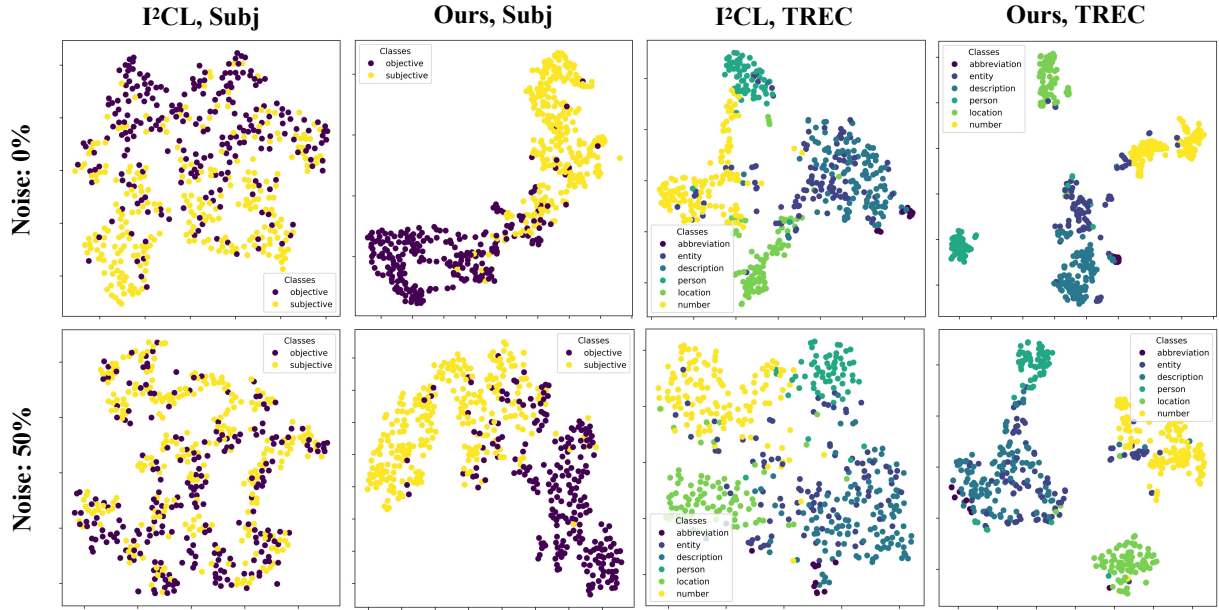


Figure 6: t-SNE visualizations on the Subj and TREC tasks. We compare I^2CL with our method under different noise levels.

sentative example per category:

- **Person:** What famous husband-and-wife team did radio ads for Blue Nun wine? → Label: Person.
- **Number:** What’s men’s par on a 455-yard golf hole? → Label: Number
- **Description:** What function does a community’s water tower serve? → Label: Description
- **Location:** Where is Belize located? → Label: Location
- **Abbreviation:** What does R.E.M. stand for, as in the rock group R.E.M.? → Label: Abbreviation
- **Entity:** What was the name of Roy Rogers’s dog? → Label: Entity

For each test question, we visualized the post-softmax attention weights that I^3CL assigned to all six demonstrations. As shown in Table 13, the results demonstrate that I^3CL dynamically adjusts attention weights based on the input question. In each case, the demonstration most semantically aligned with the test question receives the highest weight. For example, the “Location” demonstration has the highest weight for the question about the state with the least rainfall, while the “Person” demonstration is most emphasized for the question about the first

African American baseball player. This case study highlights the effectiveness of I^3CL ’s importance-weighting mechanism, showing that the model can selectively prioritize the most relevant demonstrations from a small, diverse set, thereby improving its ability to classify questions accurately.

E Performance Comparison on Other LLMs

We provide comprehensive experimental results on three widely used model backbones—LLaMA3-8B (Dubey et al., 2024), GLM4-9B (Zeng et al., 2024), and DeepSeek-R1-8B (Guo et al., 2025). As reported in Tables 10, 11, and 12, all results are averaged over five independent runs to ensure reliability.

Across all backbones and tasks, our method (I^3CL) consistently achieves the best performance, substantially outperforming strong baselines such as Zero-Shot, ICL, and I^2CL , demonstrating robust generalization and strong cross-model transferability. For example, on LLaMA3-8B, I^3CL surpasses I^2CL by 4.96%, 12.28%, and 14.96% on SST-5, TREC, and Subj, respectively, with similar improvements observed on GLM4-9B and DeepSeek-R1. These results confirm the effectiveness of our approach in leveraging demonstration information across diverse model architectures.

Interaction Module	SST-5	TREC	Subj
MLP	47.50 $\pm$ 2.56	77.88 $\pm$ 2.82	88.02 $\pm$ 2.31
Gate	49.48 $\pm$ 1.77	85.70 $\pm$ 3.26	88.12 $\pm$ 1.09
Self-Attention	51.16 $\pm$ 1.97	87.36 $\pm$ 1.14	90.84 $\pm$ 1.88

Table 14: Effect of different inter-demonstration interaction modules.

Method	SST-5	TREC	Subj
I ² CL	43.76 $\pm$ 1.73	75.44 $\pm$ 1.92	76.36 $\pm$ 1.32
I ³ CL (additive injection)	51.16 $\pm$ 1.97	87.36 $\pm$ 1.14	90.84 $\pm$ 1.88
I ³ CL (soft prompt)	46.36 $\pm$ 1.79	83.56 $\pm$ 1.22	85.67 $\pm$ 1.45

Table 15: Effect of different context-vector injection strategies on I³CL.

Method	SST-5	TREC	Subj
I ³ CL _{SimCSE-D2Q}	41.96	75.68	82.64
I ³ CL _{Attn-D2Q}	51.16	87.36	90.84

Table 16: Comparison between SimCSE-based relevance modeling and the attention-based demonstration-to-query interaction.

Strategy	SST-5	TREC	Subj
Average	51.16 $\pm$ 1.97	87.36 $\pm$ 1.14	90.84 $\pm$ 1.88
MLP	50.90 $\pm$ 1.94	87.24 $\pm$ 1.63	90.04 $\pm$ 1.35
Attention	50.84 $\pm$ 1.34	87.90 $\pm$ 1.87	91.72 $\pm$ 0.59

Table 17: Performance comparison of different aggregation strategies for the Prompt-Last and Label-Last token representations.

F Additional Ablation Studies

F.1 Alternative Inter-Demonstration Interaction Modules

To investigate whether more lightweight modules can achieve comparable results, we compare the default interaction module with two lightweight alternatives: an MLP-based module and a gating-based module. Specifically: (1) MLP-based method: Demonstration representations are concatenated and fed into an MLP for interaction. (2) Gate-based method: Demonstration representations are first averaged to form a global vector. This global vector is then combined with each individual demonstration representation, and an MLP with a sigmoid function computes a gating score. The global vector and each demonstration representation are subsequently combined via the learned gating weights.

As shown in Table 14, while both lightweight variants outperform I²CL, they fall short of the self-attention variant. The possible reasons are twofold: (1) self-attention aligns with the operations of the Transformer and may better adapt to and utilize the pretrained knowledge encoded in the model; and (2) the stronger expressive capacity of self-attention enables more thorough fitting—we find its final converged training loss is roughly an order of magnitude lower than that of MLP and Gate. These results support the rationale for using self-attention to model inter-demonstration interactions.

Method	Pass@1 $\uparrow$
Zero-Shot	42.60
ICL	47.20
LoRA-FT	68.02
I ² CL	53.65
ICL-TV	55.40
ICV	59.11
I ³ CL (Ours)	62.08

Table 18: Pass@1 on MBPP+ using LLaMA2-7B.

F.2 Prompt-Last and Label-Last Aggregation

The default I³CL representation averages the Prompt-Last Token (PLT) and Label-Last Token (LLT) embeddings with equal weights. To test whether a learned combination is beneficial, we compare equal averaging with two learned alternatives. The MLP variant concatenates the PLT and LLT embeddings and feeds them into an MLP. The attention variant uses the PLT embedding as the query and the LLT embedding as both key and value to obtain a learned weighted representation. Table 17 shows that the three aggregation strategies yield similar performance across the evaluated datasets. Because equal averaging introduces no additional trainable parameters or computation, we retain it as the default design.

F.3 Context-Vector Injection Strategies

To isolate the contribution of the interaction modules from that of the layer-wise additive injection

Method	SST-2	SST-5	TREC	AGNews	HS18	DBPedia	MR	Subj
Zero-Shot _{GPT-5.5}	90.47	48.76	81.46	83.44	82.04	87.39	88.59	76.94
ICL _{GPT-5.5}	94.79	56.20	90.27	89.22	85.08	96.85	91.18	86.41
I ² CL _{LLaMA2-7B}	90.28 ± 1.32	43.76 ± 1.73	75.44 ± 1.92	84.24 ± 1.27	76.40 ± 2.04	91.87 ± 1.05	85.52 ± 1.56	76.36 ± 1.32
I ³ CL _{LLaMA2-7B}	92.66 ± 0.90	51.16 ± 1.97	87.36 ± 1.14	87.07 ± 1.33	86.28 ± 1.65	95.88 ± 0.57	90.67 ± 0.93	90.84 ± 1.88

Table 19: Accuracy comparison between GPT-5.5 prompting baselines and LLaMA2-7B TV-based methods across eight classification datasets.

mechanism, we constructed a soft-prompt variant of I³CL by replacing additive injection with prepending the context vectors produced by the interaction modules as soft tokens to the input sequence. As shown in Table 15, with soft-prompt injection, I³CL still outperforms I²CL by 2.60, 8.12, and 9.31 points on SST-5, TREC, and Subj, respectively. This indicates that the gains do not depend solely on the additive injection mechanism; the interaction modules themselves produce more informative context representations. Meanwhile, additive injection further improves performance over the soft-prompt variant by 4.80, 3.80, and 5.17 points, respectively, demonstrating that the injection strategy also affects how effectively the context information is utilized. Overall, the gains of I³CL arise from both improved context representation quality delivered by the interaction modules and more effective context integration.

F.4 Alternative Demonstration-to-Query Relevance Modeling

To test whether a standard semantic-relevance mechanism can replace the learnable demonstration-to-query interaction, we construct a SimCSE-based variant. This baseline measures semantic relevance between each demonstration and the test query using SimCSE and uses the resulting relevance scores in place of the learnable attention mechanism. As shown in Table 16, the attention-based interaction outperforms the SimCSE-based variant by 9.20, 11.68, and 8.20 percentage points on SST-5, TREC, and Subj, respectively. This indicates that general semantic similarity alone is insufficient to capture the task-specific utility of individual demonstrations, whereas the learnable attention mechanism provides more effective query-conditioned contextual guidance.

G Code Generation on MBPP+

To further evaluate whether the benefits of I³CL extend beyond classification and text-style transfer, we conduct experiments on the MBPP+ code-

generation benchmark. We use LLaMA2-7B as the backbone and evaluate all methods on the official test set. Under the 5-shot setting, five demonstrations are sampled from the official training split, and all methods use the same demonstration-sampling and generation configuration.

As shown in Table 18, I³CL achieves a Pass@1 score of 62.08, outperforming standard ICL, I²CL, ICL-TV, and ICV by 14.88, 8.43, 6.68, and 2.97 points, respectively. These results provide complementary evidence that our method generalizes beyond classification to effectively extract task patterns from code demonstrations for long-form generation. LoRA-FT remains the strongest method in this experiment, reflecting the expected advantage of task-specific parameter adaptation when additional training is allowed.

H Comparison with a Strong Large-Model ICL Baseline

To contextualize the performance of I³CL with a 7B backbone, we additionally compare against Zero-Shot and standard ICL using GPT-5.5 on the eight classification datasets. As shown in Table 19, GPT-5.5 with standard ICL achieves an average accuracy of 86.25 across the eight datasets, while I³CL with LLaMA2-7B reaches 85.24—a gap of only 1.01 percentage points. Moreover, I³CL outperforms ICL (GPT-5.5) by 1.20 and 4.43 points on HS18 and Subj, respectively, and surpasses GPT-5.5 Zero-Shot (average accuracy: 79.89) by 5.35 points. These results demonstrate that I³CL achieves competitive performance while offering promising efficiency due to its substantially smaller model size.