Beyond Task-Level Context: Class-Conditional Context Vectors for Implicit In-Context Learning

Jianxin Zhang[1]*, Yilu Hu[1]*, Teng Liu[1], Pei Guo[1], Juntao Li[1,2]†
[1]School of Computer Science and Technology, Soochow University
[2]Key Laboratory of Data Intelligence and Advanced Computing, Soochow University
{jxzhang2020, ylhuhyl}@stu.suda.edu.cn
ljt@suda.edu.cn

Abstract

Implicit In-Context Learning compresses demonstration examples into a single context vector and injects it into the model's activation space, achieving few-shot-level performance at near zero-shot inference cost. However, existing approaches typically aggregate demonstrations from all classes into a shared, task-level context vector, capturing global task information but without explicitly preserving fine-grained, class-conditional semantic distinctions. In this work, we propose **Class-Conditional Context Vectors** (**C^{3}V**), a simple yet effective extension to implicit in-context learning that explicitly models class-specific contextual information by constructing separate context vectors for each class, without relying on explicit prompts. These class-conditional context vectors are additively injected into the model's residual streams in a single forward pass, enabling contextual contributions to be modulated in a class-aware manner while keeping the backbone frozen. We evaluate C^{3}V on multiple text classification benchmarks across several families of large language models. Experimental results demonstrate that C^{3}V consistently outperforms task-level context vector baselines, and achieves higher average accuracy than conventional few-shot learning on most models.

1 Introduction

In-context learning (ICL) enables large language models (LLMs) to adapt to new tasks by conditioning on a small set of labeled demonstrations without updating model parameters (Honda and Oka, 2025; Saglam et al., 2025; Zhang et al., 2025b). However, explicitly concatenating demonstrations with each query incurs non-trivial inference overhead (Qin et al., 2025), motivating recent work on implicit in-context learning, which compresses demonstrations into latent context vectors and injects them directly into the model's activation space (Li et al., 2025; Todd et al., 2024; Liu et al., 2024; Hu et al., 2025; Wang et al., 2025a; Hendel et al., 2023; Rimsky et al., 2024). This formulation achieves performance comparable to few-shot learning while retaining near zero-shot inference efficiency.

Among existing implicit in-context learning approaches (Hu et al., 2025; Li et al., 2025; Todd et al., 2024; Liu et al., 2024; Wang et al., 2025a; Hendel et al., 2023; Rimsky et al., 2024), I2CL (Li et al., 2025) adopts a simple and effective formulation that blends context vectors with query activations via linear combination. In doing so, I2CL constructs a task-level context vector by aggregating demonstrations from all classes into a single shared representation. This design is effective for capturing global task information and achieves strong empirical performance across a range of benchmarks while retaining high inference efficiency (Li et al., 2025). However, by treating all demonstrations uniformly, it does not explicitly preserve class-specific structure. In classification tasks, demonstrations from different classes often exhibit distinct semantic characteristics, as shown in Section 4.3 and Appendix D, raising the question of whether a single task-level context vector is sufficient to represent such heterogeneous information. Given its simplicity and empirical success, we take I2CL as a natural foundation and investigate how its task-level context vectorization can be extended to better model class-specific structure.

Motivated by this consideration, we propose a simple class-conditional extension to implicit in-context learning. Specifically, we group demonstrations by class, construct a separate context vector for each label, and jointly inject all class-conditional context vectors into the model's residual streams via learnable linear coefficients, which are trained using a small labeled support set, enabling class-aware conditioning without multiple

*Equal contribution
†Corresponding author

Findings of the Association for Computational Linguistics: ACL 2026, pages 30543–30566

inference passes or explicit prompting during inference.

We refer to this approach as Class-Conditional Context Vectors (C^3V). C^3V provides a more fine-grained formulation than task-level context vectorization by retaining class-specific contextual information in the latent space. Experiments on multiple text classification benchmarks and 6 LLMs show that this class-conditional formulation improves classification accuracy over task-level context vector baselines in most cases. We further analyze the resulting representations to better understand how class-conditional context vectors influence latent activation structure. Our main contributions are summarized as follows:

- We highlight a potential limitation of task-level context vectorization in implicit in-context learning, which aggregates demonstrations from different classes into a single shared representation without explicitly modeling class-specific structure.
- We propose C^3V, a lightweight and general class-conditional extension that introduces class-conditional context vectors at the cost of a modest increase in inference time.
- We demonstrate the effectiveness of C^3V on multiple classification benchmarks and provide representation-level analyses to better understand its behavior.

2 Related Work

In-Context Learning. In-context learning enables LLMs to perform new tasks by conditioning on a small set of demonstrations provided in the input, without parameter updates (Honda and Oka, 2025). Prior work has studied various aspects of ICL, including prompt design, demonstration selection, and ordering (Liu et al., 2025; Sun et al., 2025; Kothapalli et al., 2025). Despite its effectiveness, explicit demonstration concatenation incurs non-trivial inference overhead, motivating more efficient alternatives (Qin et al., 2025).

Implicit In-Context Learning and Context Vectors. To reduce inference cost, recent approaches propose implicit in-context learning methods that compress demonstration information into latent context vectors and inject them directly into the model's internal activations (Zhang et al., 2025a; Wang et al., 2025b; Nguyen et al., 2025; Yousefpour et al., 2025; Su et al., 2025; Konen et al., 2024; Scalena et al., 2024; Zhao et al., 2025; Yang et al., 2025; Stolfo et al., 2025; Li et al., 2025; Todd et al., 2024; Liu et al., 2024; Hu et al., 2025; Wang et al., 2025a; Hendel et al., 2023; Rimsky et al., 2024). These methods achieve strong performance while avoiding explicit prompt-based conditioning. However, most existing context-vector-based approaches operate at the task level by aggregating demonstrations into a single shared context vector (Li et al., 2025; Todd et al., 2024; Liu et al., 2024; Hu et al., 2025; Wang et al., 2025a; Hendel et al., 2023; Rimsky et al., 2024). While effective for capturing global task structure, such aggregation may obscure fine-grained, class-conditional semantic information. In contrast to these approaches, our method introduces class-conditional structure directly into implicit context representations, without requiring explicit prompts.

Class-Aware Conditioning and Representation Analysis. Prior work has shown that incorporating label or class information into model inputs can play an important role in guiding model behavior (Yao et al., 2024; Wang et al., 2023), particularly in classification settings. These approaches suggest that explicitly exposing models to class-related cues can help align internal representations with task-relevant semantics.

Our work builds on this insight by introducing class-conditional structure directly into implicit context representations. We further analyze the representational effects of class-conditional context vectors through intra-class and inter-class distance measurements, complementing prior representation analyses of implicit in-context learning.

3 Method

We propose C^3V, a simple extension to implicit ICL that captures fine-grained class-level semantics. As illustrated in Figure 1, instead of constructing a single task-level context vector, C^3V builds separate context vectors for each class from labeled demonstrations and jointly injects them into the model via linear interventions. Algorithm 1 provides the pseudocode of the C^3V algorithm.

3.1 Preliminaries: I2CL

We briefly review I2CL, which forms the basis of our approach. Given a classification task with a small set of labeled demonstrations

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}, \quad (1)$$

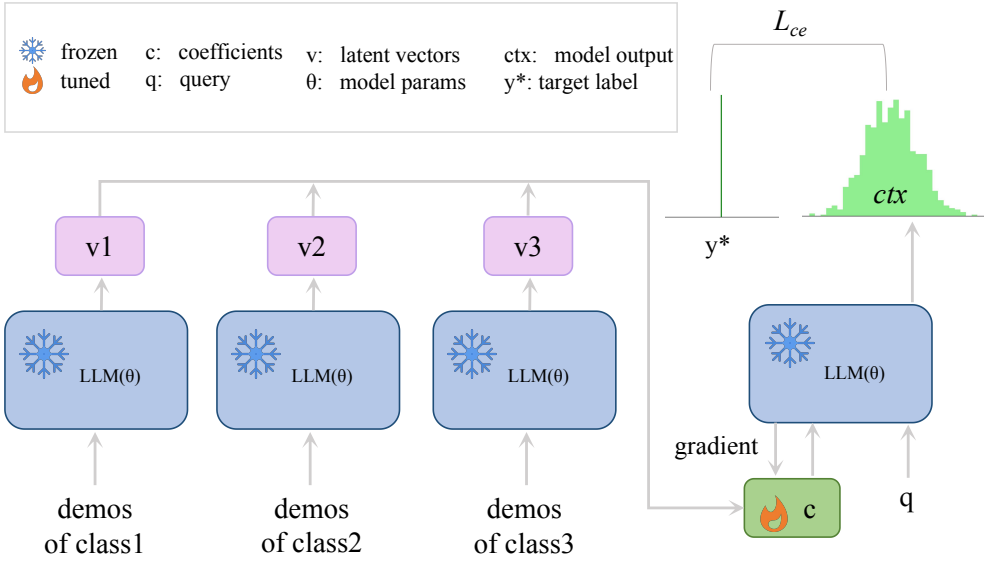


Figure 1: C^{3}V overview. Demonstrations are grouped by class and encoded by a frozen model into class-conditional latent vectors. These vectors are jointly injected with the query using learnable coefficients in a single forward pass, which are optimized with cross-entropy while the backbone remains frozen.

For each demonstration (x_i, y_i), I2CL extracts attention and MLP outputs at selected token positions (e.g., the end-of-sequence token e) across all layers $l = 1, \ldots, L$, denoted as $\{a^e_{i,l}, m^e_{i,l}\}^L_{l=1}$. These activations are averaged across all demonstrations to obtain a task-level context vector $\{\bar{a}^e_l, \bar{m}^e_l\}^L_{l=1}$, which is linearly injected into the residual stream during inference to bias the model toward task-relevant directions while preserving near zero-shot inference efficiency.

3.2 Class-Conditional Context Vector Construction

To capture class-specific semantics, we construct *class-conditional context vectors* by grouping demonstrations according to their labels. Let $\mathcal{C}$ denote the set of class labels, and $\mathcal{D}_c = \{(x_i, y_i) \in \mathcal{D} \mid y_i = c\}$ denote the demonstration subset for class c. For each $(x_i, y_i) \in \mathcal{D}_c$, we extract the same attention and MLP activations as in I2CL:

$$d_i = \{a^e_{i,l}, m^e_{i,l}\}^L_{l=1}. \tag{2}$$

Aggregating these activations within each class yields the class-conditional context vector

$$\bar{d}_c = \{\bar{a}^e_{c,l}, \bar{m}^e_{c,l}\}^L_{l=1}, \tag{3}$$

where

$$\bar{a}^e_{c,l} = \frac{1}{|\mathcal{D}_c|} \sum_{i \in \mathcal{D}_c} a^e_{i,l},$$

$$\bar{m}^e_{c,l} = \frac{1}{|\mathcal{D}_c|} \sum_{i \in \mathcal{D}_c} m^e_{i,l}.$$

3.3 Class-Conditional Context Injection

Let $\mathbf{r}^t_l$ denote the residual representation at layer l and token position t. We inject all class-conditional context vectors into the residual stream via a linear intervention:

$$\begin{aligned} \mathbf{r}^t_l = \mathbf{r}^t_{l-1} &+ \sum_{c \in \mathcal{C}} \lambda^a_{c,l}\, \bar{a}^e_{c,l} + \beta^a_l\, a^t_l \\ &+ \sum_{c \in \mathcal{C}} \lambda^m_{c,l}\, \bar{m}^e_{c,l} + \beta^m_l\, m^t_l. \end{aligned} \tag{4}$$

Here, $\lambda^a_{c,l}$ and $\lambda^m_{c,l}$ control the contribution of class-conditional attention and MLP components, while β^a_l and β^m_l preserve the original residual pathways of the backbone model. All class-conditional vectors are jointly injected within a single forward pass, resulting in a superposition of class-level semantic biases in the residual stream.

Rather than treating these coefficients as fixed hyperparameters, we learn

$$\mathbf{c} = \{\lambda^a_{c,l}, \lambda^m_{c,l}, \beta^a_l, \beta^m_l\}^L_{c \in \mathcal{C}, l=1} \tag{5}$$

via gradient-based optimization on the same few-shot demonstrations used to construct the class-conditional context vectors. In I2CL, λ is initialized to 0.1. Since our formulation injects class-conditional context vectors additively over classes, we scale the initialization by the number of classes and set it to $0.1/|\mathcal{C}|$ to keep the overall injection magnitude comparable. Following I2CL, we initialize the residual scaling coefficients β to 1. All

coefficients are optimized by minimizing the negative log-likelihood of the ground-truth labels:

$$\mathcal{L} = -\frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} \log P\big(y \mid x, \{\bar{d}_c\}_{c\in\mathcal{C}}, \mathbf{c}\big)\,, \quad (6)$$

while keeping the backbone model parameters fixed.

3.4 Inference and Prediction

During inference, prediction is performed with a single forward pass in which all class-conditional context vectors are jointly injected into the model. Given a query input x, the model produces a unified hidden representation that integrates class-specific semantic information through the learned coefficients. Class scores are computed from the predicted probabilities associated with each candidate class label, and the predicted class is selected as

$$\hat{y} = \arg\max_{c\in\mathcal{C}} P\big(y_c \mid x, \{\bar{d}_c\}_{c\in\mathcal{C}}, \mathbf{c}\big)\,, \quad (7)$$

where y_c denotes the label token corresponding to class c.

4 Experiments

4.1 Experimental Setup

Implementation Details. We outline the experimental configuration and setup used for all experiments unless otherwise specified. We construct the class-conditional context vectors by sampling five demonstration examples per class for each task. Demonstration examples are sampled from the training set and fixed across methods to ensure fair comparison. Input sequences are constructed using simple yet effective manually designed templates, which are summarized in Table 11 in the Appendix H. Each task is evaluated on a test set consisting of 500 data points, consistent with the baseline setup. For the training process, we employ the AdamW (Loshchilov and Hutter, 2019) optimizer with a learning rate of 1×10^{-2} and weight decay of 1×10^{-3}. The learning rate is adjusted using a cosine learning rate scheduler. The training is performed over 100 epochs with a batch size of 2. To ensure statistical reliability, each experiment is independently run five times under the same configuration with different random seeds, each corresponding to a different set of sampled demonstrations. All experiments were conducted on NVIDIA A5000 GPUs.

Datasets. We evaluate our method and baseline approaches on nine widely used text classification benchmarks. Following prior work (e.g., I2CL), we adopt the same benchmark suite to ensure fair comparison. Specifically, the evaluated datasets include SST-2 and SST-5 (Socher et al., 2013) for binary and fine-grained sentiment analysis, MR (Pang and Lee, 2005) for movie review sentiment classification, TREC (Voorhees and Tice, 2000) for question type classification, AGNews and DBPedia (Zhang et al., 2015) for topic and multi-class document classification, Subj (Pang and Lee, 2004) for subjectivity detection, EmoC (Chatterjee et al., 2019) for emotion classification, and HateSpeech18 (de Gibert et al., 2018) for hate speech detection.

Models. We select three famous model families: LLaMA, GPT and Qwen. Specifically, we choose LLaMA2-7B (Touvron et al., 2023), LLaMA3-8B (AI@Meta, 2024), LLaMA3.2-1B (AI@Meta, 2024), GPT-J-6B (Wang and Komatsuzaki, 2021), GPT2-XL (Radford et al., 2019) and Qwen3-8B (Team, 2025). The models were chosen based on their alignment with the I2CL and the availability of our computational resources. We only provide a detailed report of the results for LLaMA2-7B, with **the results for other LLMs available in Appendix C**.

Baselines. We compare our method against the following baselines: (1) Label Anchor (Wang et al., 2023), which uses anchor tokens in the input for task adaptation without fine-tuning model parameters. (2) Task Vector (Hendel et al., 2023), which injects a task-specific vector derived from demonstration examples into the model's optimal replacement layer to guide task-specific behavior. (3) ICV (Liu et al., 2024), which leverages positive and negative demonstration pairs to generate the steering vector, mainly for open-ended generation tasks. (4) I2CL (Li et al., 2025), which condenses demonstration examples into a context vector and uses linear operations to fuse information. In addition to these, several tuning-based approaches are also included, such as (5) AutoCompressors (Chevalier et al., 2023), which finetunes the model to compress knowledge representations. (6) ICAE (Ge et al., 2024), which uses a LoRA-adapted encoder to encode context into memory slots. (7) CEPE (Yen et al., 2024), which utilizes demonstrations from the encoder context to improve performance. (8) Prompt-tuning (Lester et al., 2021), which adds learnable prompt tokens to adapt the model without modifying its parameters. (9) LoRA (Hu et al.,

Method	SST-2	SST-5	TREC	AGNews	Subj	HateS	DBPedia	EmoC	MR	Avg.	# cached P.
Zero-shot	83.00	27.00	50.00	70.20	51.40	54.20	72.00	41.80	73.60	58.13	0
Few-shot	$94.44_{1.44}$	$41.72_{3.68}$	$77.32_{4.41}$	$85.68_{2.00}$	$52.56_{3.09}$	$70.24_{5.80}$	$96.64_{0.48}$	$75.48_{1.63}$	$93.24_{0.50}$	76.37	$2MDL$
Label-anchor	$83.32_{5.95}$	$27.68_{4.21}$	$77.48_{3.49}$	$83.72_{1.04}$	$53.00_{2.95}$	$64.52_{8.09}$	$81.40_{3.67}$	$59.12_{10.60}$	$84.40_{5.89}$	68.29	$2(M/K)DL$
Task-vector	$81.44_{4.73}$	$25.96_{0.59}$	$65.68_{1.93}$	$79.68_{4.07}$	$58.56_{4.91}$	$67.68_{3.70}$	$89.48_{2.58}$	$44.64_{3.53}$	$82.32_{5.37}$	66.16	D
ICV	$86.28_{0.55}$	$33.48_{0.65}$	$63.84_{0.15}$	$72.40_{0.37}$	$56.56_{0.70}$	$60.56_{1.50}$	$73.64_{0.88}$	$49.16_{1.24}$	$84.04_{1.10}$	64.44	DL
I2CL	$\mathbf{87.68}_{2.47}$	$39.12_{2.69}$	$78.56_{5.32}$	$85.48_{1.16}$	$\mathbf{73.84}_{3.84}$	$69.88_{5.67}$	$90.16_{1.86}$	$63.72_{1.37}$	$87.68_{2.26}$	75.12	$2DL$
C^3V	$87.06_{2.01}$	$\mathbf{44.64}_{2.65}$	$\mathbf{78.86}_{4.56}$	$\mathbf{85.52}_{2.49}$	$70.40_{7.06}$	$\mathbf{74.80}_{1.64}$	$\mathbf{93.16}_{0.97}$	$\mathbf{63.94}_{2.97}$	$\mathbf{88.48}_{3.12}$	**76.32**	$2\lvert\mathcal{C}\rvert DL$
AutoComp.	$92.44_{3.29}$	$25.80_{4.80}$	$62.52_{9.34}$	$86.36_{1.03}$	$60.16_{0.32}$	$53.20_{6.10}$	$92.68_{2.86}$	$29.56_{5.07}$	$82.76_{7.34}$	63.94	$2(M/K)DL$
ICAE	$91.64_{1.69}$	$38.80_{1.56}$	$50.92_{8.38}$	$80.48_{2.35}$	$50.52_{9.17}$	$65.48_{7.18}$	$62.08_{1.86}$	$54.04_{4.69}$	$89.48_{1.45}$	64.83	$2(M/K)DL$
CEPE	$74.28_{3.90}$	$36.20_{0.56}$	$55.48_{3.42}$	$78.00_{3.49}$	$59.12_{1.60}$	$61.72_{5.26}$	$87.24_{1.20}$	$42.28_{3.31}$	$82.36_{1.61}$	64.08	Md_{enc}

Table 1: Comparison of methods on various classification benchmarks using LLaMA2-7B. The best results are highlighted in **bold**. In addition to evaluating the memory consumption, we also conduct an analysis of the cached parameters. Specifically, M, D, and L denote the number of demonstration tokens, the model hidden dimensionality, and the number of architectural layers, respectively. Furthermore, $1/K$ represents the compression ratio of the respective context-compression technique, $|\mathcal{C}|$ denotes the number of classes, and d_{enc} denotes the hidden dimension of the auxiliary encoder (when applicable). A detailed explanation of "# cached P." can be found in Appendix A. Due to space constraints, the signs for the standard deviation were omitted, but it should be expressed as mean ± standard deviation.

Method	# trainable params. (K)	SST-2	SST-5	TREC	AGNews	Subj	HateS	DBPedia	EmoC	MR	Avg.
Prompt-tuning	4.10	$56.24_{6.99}$	$24.24_{2.96}$	$55.20_{4.14}$	$78.00_{7.60}$	$57.40_{4.93}$	$49.56_{6.96}$	$74.40_{6.43}$	$35.08_{5.29}$	$54.32_{1.76}$	54.94
LoRA (r=8)	4194.30	$84.80_{6.59}$	$39.87_{4.33}$	$75.97_{10.77}$	$83.80_{2.32}$	$70.47_{10.68}$	$75.32_{2.88}$	$91.40_{3.54}$	$53.67_{16.27}$	$83.07_{0.25}$	73.15
LoRA (r=1)	524.29	$82.44_{0.77}$	$38.88_{3.47}$	$73.08_{6.56}$	$83.32_{2.76}$	$60.68_{0.68}$	$63.32_{2.83}$	$84.52_{4.14}$	$57.88_{2.66}$	$87.96_{1.68}$	70.23
IA3	262.14	$89.40_{2.08}$	$46.93_{0.81}$	$75.41_{4.94}$	$84.43_{1.45}$	$56.67_{3.07}$	$62.54_{5.58}$	$93.91_{0.49}$	$59.75_{3.67}$	$88.00_{1.88}$	73.00
C^3V	$0.064(\lvert\mathcal{C}\rvert+1)$	$87.06_{2.01}$	$44.64_{2.65}$	$78.86_{4.56}$	$85.52_{2.49}$	$70.40_{7.06}$	$74.80_{1.64}$	$93.16_{0.97}$	$63.94_{2.97}$	$88.48_{3.12}$	76.32

Table 2: Comparison of different PEFT methods on various datasets. Due to space constraints, the signs for the standard deviation were omitted, but it should be expressed as mean ± standard deviation. Here, $|\mathcal{C}|$ denotes the number of class labels in the corresponding dataset.

2022) and (10) IA3 (Liu et al., 2022), which are PEFT methods from HuggingFace PEFT library[1]. All baseline configurations refer to the settings in I2CL.

Evaluation Metrics. We report classification accuracy as the primary evaluation metric. For multi-class datasets, we report macro-averaged accuracy. All results are averaged over five random seeds. All results are reported in percentage (%).

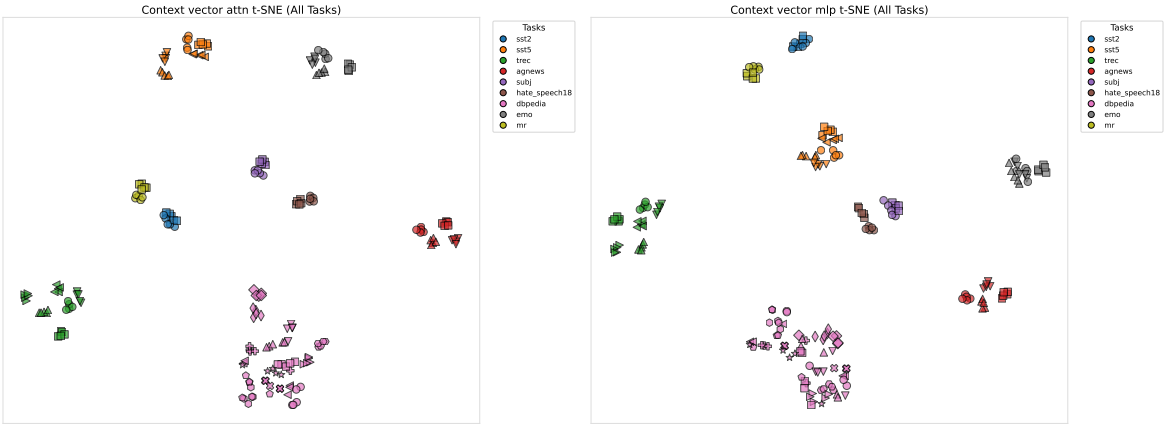
4.2 Main Results

C^3V Achieves Few-shot Level Accuracy with Moderate Parameter Caching. Table 1 reports results on LLaMA2-7B. C^3V achieves performance comparable to few-shot inference while requiring only moderate additional cached parameters. It attains 76.32% average accuracy, outperforming the zero-shot baseline by 18.19 points. **Notably, as shown in Appendix C, C^3V even surpasses few-shot performance on average on the remaining 5 LLMs.**

[1] https://github.com/huggingface/peft

Comparison with Methods without Explicit Demonstrations. Label Anchor, Task Vector, ICV, and I2CL perform inference without relying on explicit demonstration examples. As shown in Table 1, C^3V achieves the best overall performance among these approaches, reaching the highest average accuracy. In particular, C^3V outperforms the strongest baseline I2CL by 1.2 points on average, with gains on challenging datasets such as SST-5, HateS, and DBPedia. These results suggest that explicitly modeling class-conditional contextual information can offer a more effective alternative to task-level context representations, while avoiding any reliance on demonstration examples during inference.

Comparison with Methods for Prompt Compression during Inference. We compare C^3V with representative prompt compression methods, including AutoCompressors (Chevalier et al., 2023), ICAE (Ge et al., 2024), and CEPE (Yen et al., 2024). As shown in Table 1, these approaches yield mixed improvements over zero-shot baselines and exhibit inconsistent performance across datasets.



(a) Attention (b) MLP

Figure 2: t-SNE visualization of class-conditional latent vectors on LLaMA-2-7B. Left: attention-based latent vectors; right: MLP-based latent vectors. Points are colored by dataset and shaped by class label. Latent vectors from five random seeds are included for each class.

In contrast, C^3V does not rely on a pre-trained universal compression model; instead, it constructs class-conditional context vectors and learns task-specific injection coefficients, enabling more targeted adaptation. We note that prompt compression methods primarily aim to reduce input length for long-context inference, making them complementary to C^3V, which focuses on improving task adaptation with moderate parameter caching.

Comparison with Test-time PEFT Methods. We compare C^3V with widely used test-time PEFT methods on LLaMA2-7B, including Prompt Tuning (Lester et al., 2021), LoRA (Hu et al., 2022), and IA3 (Liu et al., 2022). As shown in Table 2, Prompt Tuning underperforms the zero-shot baseline under data-scarce conditions, while LoRA and IA3 achieve stronger performance but require substantially more trainable parameters. In contrast, C^3V attains competitive adaptation performance with a number of learnable parameters that scales linearly with the number of classes.

4.3 Representation Analysis

To better understand the representational effects of class-conditional context vectors, we analyze the structure and separability of the latent representations using both visualization and distance-based metrics based on cosine similarity.

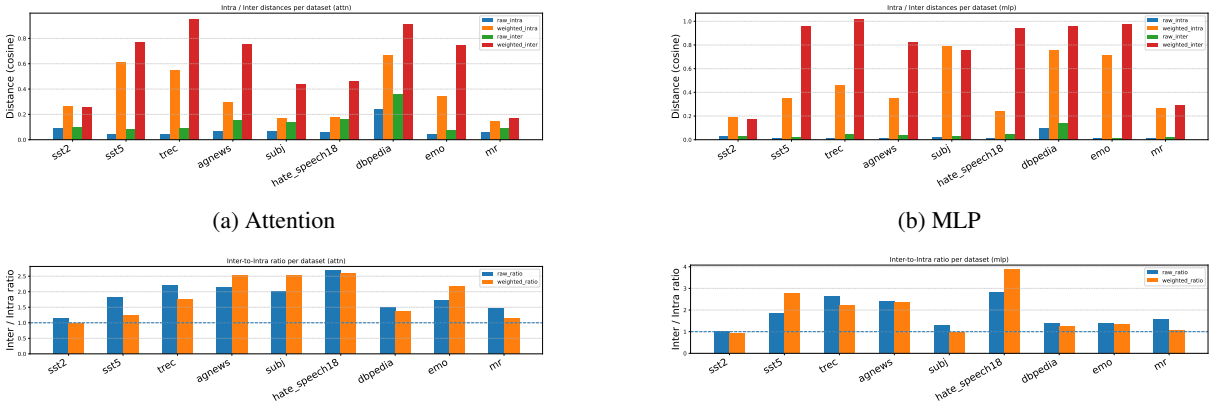
4.3.1 Visualization of Class-Conditional Latent Vectors

For each dataset, we construct five sets of class-conditional latent vectors, each obtained from a run with a different random seed, corresponding to different sampled demonstrations. For visualization, we concatenate the layer-wise attention or MLP vectors of each class-conditional context into a single high-dimensional representation and project all such representations into two dimensions using t-SNE. As shown in Figure 2, for LLaMA2-7B, the visualization reveals a hierarchical organization: latent vectors from different datasets form well-separated global clusters, while vectors corresponding to different class labels further organize into distinct sub-clusters within each dataset. Despite this overall hierarchical structure, the visualization also exhibits cases with less clear class separation. For example, in Figure 2a, the two classes in the SST-2 dataset are relatively difficult to distinguish, while in Figure 2b, classes in the EmoC dataset (e.g., circles and inverted triangles) show substantial overlap. **Class-conditional latent vectors obtained from other models are presented in Appendix D.**

These patterns suggest that class-conditional context vectors capture generally stable and discriminative class-level semantics that are robust to randomness in demonstration sampling.

4.3.2 Inter- and Intra-Class Distance Analysis

We further quantify representation separability by computing inter- and intra-class distances within each dataset. Intra-class distance measures the variation among latent vectors of the same class across different random seeds, while inter-class distance measures the separation between latent vectors of different classes. All distances are computed us-



(a) Attention

(b) MLP

(c) Attention (Inter/Intra Ratio)

(d) MLP (Inter/Intra Ratio)

Figure 3: Inter- and intra-class distance analysis of latent representations across datasets on LLaMA-2-7B. (a,b) Inter- and intra-class distances for attention-based and MLP-based representations, where each panel includes distances computed both without coefficient scaling (bars labeled with the raw- prefix in the legend) and after multiplication by trained coefficients (bars labeled with the weight- prefix). (c,d) Corresponding inter/intra ratios for attention and MLP representations, comparing the settings without (raw-) and with (weight-) trained coefficient scaling.

ing cosine distance. We perform this analysis for both attention- and MLP-based representations, before and after modulation with the learned coefficients. The detailed formulations of the distance metric, as well as the definitions of intra-class and inter-class distances and the inter/intra ratio, are provided in Appendix B. As shown in Figure 3, we report the inter- and intra-class distance analysis for LLaMA2-7B; **corresponding results for other models are provided in Appendix E**.

Across most datasets, both before and after modulation with the learned coefficients, inter-class distances are larger than intra-class distances (Figures 3a–3b), indicating inherent class separability. After modulation, both inter-class and intra-class distances generally increase, indicating that, under cosine distance, pairwise distances between latent representations exhibit an overall consistent increase. To assess relative separability, we compute the inter/intra ratio (Figures 3c–3d), where values greater than 1 indicate favorable class discrimination. In most cases, the ratio remains above 1. Notably, for a small number of datasets (e.g., SST-2 and Subj under MLP-based modulated representations), the ratio falls below 1, which aligns with cases where C^3V underperforms I2CL in the main results. These analyses suggest that class-conditional context vectors induce relatively stable and discriminative representations, while their effectiveness varies across different datasets.

Finally, we visualize the learned layer-wise coefficients to qualitatively analyze how context injection is modulated across layers (see Appendix F).

4.4 Ablation

We analyze the impact of each module in our framework using LLaMA2-7B (unless otherwise specified), with the reported results averaged across five random seeds and all nine datasets. **Due to space constraints, detailed ablation studies on target modules, target layers, injection positions, extraction positions, and context vector aggregation strategies are deferred to Appendix G.**

Training and Inference Efficiency. We compare the training and inference efficiency of C^3V with relevant baselines on LLaMA3.2-1B, as summarized in Table 3. While both C^3V and I2CL introduce learnable scaling coefficients, C^3V incurs higher training time due to the introduction of additional parameters that scale with the number of classes. This effect is particularly pronounced on datasets with more classes, such as DBPedia, SST-5, TREC, and AGNews, where modeling class-conditional contextual information leads to a noticeable increase in training cost. On average, the training time increases from 104.60 seconds for I2CL to 146.93 seconds for C^3V. At inference time, zero-shot and cached few-shot baselines achieve the lowest latency, while uncached few-shot inference is substantially more expensive. Compared to I2CL, C^3V introduces moderate additional inference cost on most datasets, resulting in an average inference time of 22.11 seconds versus 13.60 sec-

Method	SST-2	SST-5	TREC	AGNews	Subj	HateS	DBPedia	EmoC	MR	Avg.
Training Time										
I2CL	$55.27_{1.74}$	$146.17_{1.01}$	$177.07_{1.24}$	$134.32_{4.15}$	$66.03_{0.99}$	$64.47_{0.85}$	$99.93_{5.46}$	$134.59_{0.53}$	$63.59_{2.44}$	104.60
C^3V	$65.49_{1.91}$	$201.37_{4.79}$	$265.27_{2.92}$	$175.95_{26.25}$	$66.26_{3.10}$	$63.13_{3.01}$	$273.45_{40.80}$	$146.19_{4.03}$	$65.28_{1.19}$	146.93
Inference Time										
ZS	6.42	6.43	4.66	9.26	6.84	6.70	12.11	5.54	6.61	7.17
FS (Cache)	$6.87_{0.05}$	$7.53_{0.04}$	$4.99_{0.01}$	$11.24_{0.11}$	$7.48_{0.07}$	$7.24_{0.06}$	$14.92_{0.21}$	$6.32_{0.05}$	$7.21_{0.05}$	8.20
FS (No Cache)	$26.09_{3.46}$	$85.46_{2.70}$	$64.67_{2.53}$	$141.98_{6.18}$	$45.97_{6.48}$	$34.96_{5.13}$	$131.87_{14.67}$	$58.09_{4.56}$	$36.93_{2.30}$	69.56
I2CL	$11.96_{1.27}$	$12.16_{1.33}$	$9.75_{0.14}$	$15.95_{3.45}$	$12.08_{0.10}$	$11.75_{0.15}$	$24.86_{1.87}$	$11.89_{0.75}$	$12.03_{0.55}$	13.60
C^3V	$14.02_{1.10}$	$19.69_{1.38}$	$19.05_{1.63}$	$22.60_{9.58}$	$11.51_{2.66}$	$8.58_{0.03}$	$86.22_{22.11}$	$8.98_{0.06}$	$8.37_{0.11}$	22.11

Table 3: Training and Inference Time (seconds) on LLaMA3.2-1B. Values are reported as mean with standard deviation shown in subscript. Due to space constraints, the $\pm$ symbol is omitted.

Direction	Accuracy
Zero-shot	58.13
+ Noise Vector (Repl.)	69.73
+ Class-Conditional (No-Train, init = $0.1/\lvert\mathcal{C}\rvert$)	60.98
+ Class-Conditional (Train, init = 0.1)	71.94
Train, init = $0.1/\lvert\mathcal{C}\rvert$ (Ours)	76.32

Table 4: Noise Injection, Training, and Coefficient Initialization

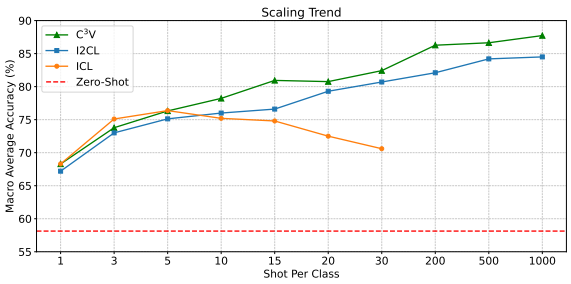


Figure 4: Scaling trend of C^3V.

onds for I2CL. In particular, datasets with a larger number of classes, such as DBPedia (14 classes), exhibit a more pronounced increase in inference time. These results highlight a clear trade-off between efficiency and performance: C^3V requires modestly higher training and inference time than task-level context vector methods, but achieves improved classification performance by explicitly modeling class-conditional structure, while remaining substantially more efficient than uncached few-shot inference.

Scaling with the Number of Demonstrations. We analyze how model performance scales with the number of demonstrations per class. As shown in Figure 4, C^3V exhibits a clear and consistent improvement as more demonstrations are provided. Notably, across all shot settings shown in the figure, C^3V consistently achieves higher accuracy than I2CL. In contrast, standard ICL shows limited gains beyond a small number of demonstrations and degrades as the context length grows. These results suggest that C^3V scales more favorably with increasing demonstration size, avoiding the degradation observed in standard ICL at larger shot counts.

Noise Injection, Training, and Coefficient Initialization. We conduct an ablation study to disentangle the effects of noise-based injection, coefficient training, and initialization strategies, as summarized in Table 4. Replacing class-conditional context vectors with noise yields only limited improvement over the zero-shot baseline, indicating that the performance gains cannot be attributed solely to the class-conditional context vectors themselves and that the trainable coefficients play an important role. Injecting class-conditional context with fixed (non-trainable) coefficients yields only modest gains, suggesting that class-aware initialization ($0.1/|\mathcal{C}|$) can introduce class-conditional signals into the model, but remains insufficient without learning. Training the coefficients with a uniform initialization (0.1) leads to substantial performance improvements; however, it remains inferior to the $0.1/|\mathcal{C}|$ initialization, underscoring the importance of appropriate initialization of the injection parameters. Our full method (**Ours**: combining class-conditional context vectors with trainable coefficients and coefficient initialization of $0.1/|\mathcal{C}|$) achieves the best performance, demonstrating that both coefficient training and initialization are crucial for fully leveraging class-conditional context vectors in C^3V.

5 Conclusion

We introduce C^3V, a lightweight extension to implicit ICL that explicitly models class-specific con-

textual information via class-conditional context injection. Experiments and representation analyses show that C^{3}V consistently improves classification performance over task-level context vectors by enhancing class discriminability, while preserving the efficiency of implicit ICL. Future work will explore extending C^{3}V to broader task settings and multimodal scenarios.

Limitations

While our method, C^{3}V, demonstrates promising results in improving ICL, there are several limitations that warrant further investigation.

1. **Task-Specific Applicability.** Currently, C^{3}V is evaluated primarily on text classification tasks. Although we show significant improvements in this domain, the effectiveness of class-conditional context vectors in more complex tasks, such as open-ended generation or multi-label classification, remains to be explored. Tasks involving long-range dependencies, multi-step reasoning, or generation may require further adaptations to the method or a more sophisticated use of context vectors.

2. **Parameter Growth with the Number of Classes.** C^{3}V employs class-conditional context vectors whose number increases with the size of the label set. As the label space grows, the number of class-related parameters to be learned or stored correspondingly increases, which may introduce additional memory and computational overhead. While this design facilitates modeling class-level semantic information, its scalability in settings with a large number of classes warrants further investigation. Future work may explore parameter-sharing mechanisms or more compact representations to improve scalability.

Acknowledgments

We want to thank all the anonymous reviewers for their valuable comments. This work was supported by the National Science Foundation of China (NSFC No. 62576232), Key Laboratory of Data Intelligence and Advanced Computing, Soochow University.

References

AI@Meta. 2024. The llama 3 herd of models.

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019. Semeval-2019 task 3: Emocontext contextual emotion detection in text. In *Proceedings of the 13th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2019, Minneapolis, MN, USA, June 6-7, 2019*, pages 39–48. Association for Computational Linguistics.

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, EMNLP 2023, Singapore, December 6-10, 2023*, pages 3829–3846. Association for Computational Linguistics.

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, ALW@EMNLP 2018, Brussels, Belgium, October 31, 2018*, pages 11–20. Association for Computational Linguistics.

Tao Ge, Jing Hu, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. 2024. In-context autoencoder for context compression in a large language model. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Roee Hendel, Mor Geva, and Amir Globerson. 2023. In-context learning creates task vectors. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 9318–9333. Association for Computational Linguistics.

Ukyo Honda and Tatsushi Oka. 2025. Exploring explanations improves the robustness of in-context learning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 23693–23714. Association for Computational Linguistics.

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 *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Xinyan Hu, Kayo Yin, Michael I. Jordan, Jacob Steinhardt, and Lijie Chen. 2025. Understanding in-context learning of addition via activation subspaces. *ArXiv*, abs/2505.05145v3.

Kai Konen, Sophie Jentzsch, Diaoulé Diallo, Peer Schütt, Oliver Bensch, Roxanne El Baff, Dominik Opitz, and Tobias Hecking. 2024. Style vectors for steering generative large language models. In *Findings of the Association for Computational Linguistics: EACL 2024*, pages 782–802, St. Julian's, Malta. Association for Computational Linguistics.

Vignesh Kothapalli, Hamed Firooz, and Maziar Sanjabi. 2025. Cot-icl lab: A synthetic framework for studying chain-of-thought learning from in-context demonstrations. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 14620–14642. Association for Computational Linguistics.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 3045–3059. Association for Computational Linguistics.

Zhuowei Li, Zihao Xu, Ligong Han, Yunhe Gao, Song Wen, Di Liu, Hao Wang, and Dimitris N. Metaxas. 2025. Implicit in-context learning. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. 2022. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Hui Liu, Wenya Wang, Hao Sun, Chris Xing Tian, Chenqi Kong, Xin Dong, and Haoliang Li. 2025. Unraveling the mechanics of learning-based demonstration selection for in-context learning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 2623–2641. Association for Computational Linguistics.

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 *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Duy Nguyen, Archiki Prasad, Elias Stengel-Eskin, and Mohit Bansal. 2025. Multi-attribute steering of language models via targeted intervention. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 20619–20634. Association for Computational Linguistics.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics, 21-26 July, 2004, Barcelona, Spain*, pages 271–278. ACL.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *ACL 2005, 43rd Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, 25-30 June 2005, University of Michigan, USA*, pages 115–124. The Association for Computer Linguistics.

Chengwei Qin, Wenhan Xia, Fangkai Jiao, Chen Chen, Yuchen Hu, Bosheng Ding, Ruirui Chen, and Shafiq Joty. 2025. Beyond output matching: Bidirectional alignment for enhanced in-context learning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 32732–32758. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, and 1 others. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Matt Turner. 2024. Steering llama 2 via contrastive activation addition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 15504–15522. Association for Computational Linguistics.

Baturay Saglam, Xinyang Hu, Zhuoran Yang, Dionysis Kalogerias, and Amin Karbasi. 2025. Learning task representations from in-context learning. In *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 6634–6663. Association for Computational Linguistics.

Daniel Scalena, Gabriele Sarti, and Malvina Nissim. 2024. Multi-property steering of large language models with dynamic activation composition. In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 577–603, Miami, Florida, US. Association for Computational Linguistics.

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, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1631–1642. ACL.

Alessandro Stolfo, Vidhisha Balachandran, Safoora Yousefi, Eric Horvitz, and Besmira Nushi. 2025. Improving instruction-following in language models through activation steering. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Jingran Su, Jingfan Chen, Hongxin Li, Yuntao Chen, Li Qing, and Zhaoxiang Zhang. 2025. Activation steering decoding: Mitigating hallucination in large vision-language models through bidirectional hidden state intervention. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12964–12974, Vienna, Austria. Association for Computational Linguistics.

Chenghao Sun, Zhen Huang, Yonggang Zhang, Le Lu, Houqiang Li, Xinmei Tian, Xu Shen, and Jieping Ye. 2025. Interpret and improve in-context learning via the lens of input-label mappings. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 3873–3895. Association for Computational Linguistics.

Qwen Team. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Eric Todd, Millicent L. 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, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and 49 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *ArXiv*, abs/2307.09288v2.

Ellen M. Voorhees and Dawn M. Tice. 2000. Building a question answering test collection. In *SIGIR 2000: Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, July 24-28, 2000, Athens, Greece*, pages 200–207. ACM.

Ben Wang and Aran Komatsuzaki. 2021. Gpt-j-6b: A 6 billion parameter autoregressive language model.

Futing Wang, Jianhao Yan, Yue Zhang, and Tao Lin. 2025a. ELICIT: LLM augmentation via external in-context capability. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

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, EMNLP 2023, Singapore, December 6-10, 2023*, pages 9840–9855. Association for Computational Linguistics.

Mengru Wang, Ziwen Xu, Shengyu Mao, Shumin Deng, Zhaopeng Tu, Huajun Chen, and Ningyu Zhang. 2025b. Beyond prompt engineering: Robust behavior control in llms via steering target atoms. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 23381–23399. Association for Computational Linguistics.

Tianyun Yang, Ziniu Li, Juan Cao, and Chang Xu. 2025. Understanding and mitigating hallucination in large vision-language models via modular attribution and intervention. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Hantao Yao, Rui Zhang, and Changsheng Xu. 2024. TCP: textual-based class-aware prompt tuning for visual-language model. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pages 23438–23448. IEEE.

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), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 2588–2610. Association for Computational Linguistics.

Ashkan Yousefpour, Taeheon Kim, Ryan Sungmo Kwon, Seungbeen Lee, Wonje Jeung, Seungju Han, Alvin Wan, Harrison Ngan, Youngjae Yu, and Jonghyun Choi. 2025. Representation bending for large language model safety. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 24073–24098. Association for Computational Linguistics.

Jinghao Zhang, Yuting Liu, Wenjie Wang, Qiang Liu, Shu Wu, Liang Wang, and Tat-Seng Chua. 2025a. Personalized text generation with contrastive activation steering. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 7128–7141. Association for Computational Linguistics.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 649–657.

Zheng Zhang, Shaocheng Lan, Lei Song, Jiang Bian, Yexin Li, and Kan Ren. 2025b. Learning to select in-context demonstration preferred by large language model. In *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 11345–11360. Association for Computational Linguistics.

Yu Zhao, Alessio Devoto, Giwon Hong, Xiaotang Du, Aryo Pradipta Gema, Hongru Wang, Xuanli He, Kam-Fai Wong, and Pasquale Minervini. 2025. Steering knowledge selection behaviours in llms via sae-based representation engineering. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2025 - Volume 1: Long Papers, Albuquerque, New Mexico, USA, April 29 - May 4, 2025*, pages 5117–5136. Association for Computational Linguistics.

A # Cached Parameters During Inference

- M: number of demonstration (or context) tokens,
- D: model hidden dimension,
- L: number of transformer layers,
- $1/K$: compression ratio.
- d_{enc}: hidden dimension of the auxiliary encoder (if used).
- $|\mathcal{C}|$: the number of classes (if used).

Specifically, the term "$2MDL$" refers to the size of the standard key-value (KV) cache used in autoregressive inference. In the KV caching mechanism, for each of the M input tokens and across all L Transformer layers, the model stores:

- a D-dimensional Key (K) vector, and
- a D-dimensional Value (V) vector.

This results in a total of $2 \times M \times D \times L = 2MDL$ cached parameters. The KV cache is maintained to avoid re-computing attention over the context at every generation step, but it directly contributes to memory footprint.

Few-shot method follows standard autoregressive inference. When the input prompt contains M demonstration tokens, the model caches their key-value representations across all L Transformer layers, resulting in a KV cache of size $2MDL$.

In contrast, the I2CL framework do *not* cache the full KV states of all M demonstration tokens, thereby avoiding the $2MDL$ cost. Instead, it operates as a latent ICL approach:

- During a pre-processing phase, the demonstrations $d_1, \ldots, d_N$ are passed through the frozen LLM, and for each demonstration, we extract the multi-head attention output $a^e_{i,l}$ and MLP output $m^e_{i,l}$ at the final token position e of each layer l.
- These activations are then averaged across demonstrations to form a compact context vector.
- The resulting context representation is $\{\bar{a}^e_l, \bar{m}^e_l\}_{l=1}^L$, which contains two D-dimensional vectors per layer (one from attention, one from MLP), totaling $2DL$ parameters.

During inference, no demonstration tokens are included in the input prompt. Instead, the precomputed context vector $\{\bar{a}^e_l, \bar{m}^e_l\}_{l=1}^L$ is directly injected into the residual stream of the query. This eliminates the need for a large KV cache proportional to M, making the method significantly more memory-efficient—especially as the number of demonstrations grows.

Building upon the latent ICL formulation of I2CL, our **C^{3}V** framework further refines the context representation by modeling class-conditional semantics explicitly. Instead of aggregating all demonstrations into a single task-level context vector, C^{3}V maintains a separate context vector for each class.

Concretely, for a classification task with $|\mathcal{C}|$ classes, demonstrations belonging to the same class $c \in \mathcal{C}$ are first processed by the frozen LLM to extract the attention and MLP activations at the final token position of each layer. These activations are averaged *within each class*, yielding a set of class-conditional context vectors

$$\left\{\bar{a}^e_{c,l}, \bar{m}^e_{c,l}\right\}_{l=1}^L, \quad c \in \mathcal{C}.$$

As a result, C^{3}V caches $|\mathcal{C}|$ such context representations, each consisting of two D-dimensional vectors per layer. The total number of cached parameters therefore scales as

$$2|\mathcal{C}|DL,$$

which is linear in the number of classes but remains independent of the number of demonstration tokens M.

During inference, no demonstration tokens are appended to the query. Instead, all class-conditional context vectors are jointly injected into the residual stream through a linear intervention, enabling fine-grained class-aware modulation of

the model's internal representations. Compared to I2CL, C^{3}V incurs a modest increase in cached parameters proportional to $|\mathcal{C}|$, while preserving the same zero-shot inference cost and avoiding any KV-cache growth with respect to M.

For Label Anchor method, the demonstrations are compressed into $\frac{M}{K}$ context tokens, including both label and formatting tokens. During inference, the model still performs token-level attention over these compressed tokens. As a result, the cached parameters correspond to the KV cache of the compressed context, yielding a memory cost of $2 \times \frac{M}{K} \times D \times L$.

For Task Vector method, the demonstrations are compressed into a single task vector $\theta(S) \in \mathbb{R}^D$. During inference, no demonstration tokens are included in the input; instead, $\theta(S)$ is patched into the residual stream at an intermediate layer. Therefore, the cached parameters consist of a single activation vector of dimension D, independent of the number of demonstrations.

For ICV method, demonstrations are summarized into a set of layer-wise activation vectors $\{h^l_{\text{ICV}}\}_{l=1}^{L}$ derived from the MLP outputs. During inference, no demonstration tokens are included; instead, one D-dimensional activation vector is added at each layer. Therefore, the cached parameters scale as DL, independent of the number of demonstrations.

ICAE uses a LoRA-fine-tuned encoder to compress the original long context M into a small number of "Memory Slots". The compression ratio is $1/K$, so during inference, the effective sequence length entering the model is no longer M, but the reduced $\frac{M}{K}$. The KV cache parameter calculation is $2 \times \frac{M}{K} \times D \times L$, where the number 2 represents the Key and Value matrices.

In AutoCompressor, the context is compressed into $\frac{M}{K}$ summary vectors, which are used as soft prompts and injected into the input of every layer of the model. During inference, the Key and Value representations of these summary tokens are cached at each layer, forming the Compressed KV Cache. Therefore, the size of the cached parameters is: $2 \times \frac{M}{K} \times D \times L$.

Ignoring the caching overhead from the query during inference, and considering only the cached parameters for the additional context, CEPE stores $M \cdot d_{\text{enc}}$ parameters. Specifically, it caches the **last-layer hidden representations** from the small bidirectional encoder $Model_{\text{enc}}$, where $d_{\text{enc}} = 1024$ (matching the hidden dimension of RoBERTa-large, as used in the paper). These d_{enc}-dimensional vectors are later **up-projected to the decoder's hidden dimension d_{dec}** (e.g., 4096 for LLaMA2-7b) **within the cross-attention layers** at each decoder block, enabling the decoder to attend to the encoded context without caching d_{dec}-dimensional keys and values for the additional tokens.

B Definition of Intra-Class and Inter-Class Distances

In this appendix, we formally define the intra-class and inter-class distance metrics used in our representation analysis.

B.1 Distance Metric

All distances are computed in the original high-dimensional representation space using *cosine distance*, defined as

$$d(\mathbf{x}, \mathbf{z}) = 1 - \frac{\mathbf{x}^\top \mathbf{z}}{\|\mathbf{x}\| \, \|\mathbf{z}\|}. \tag{8}$$

Cosine distance captures the angular relationship between representations and is widely used in representation learning due to its ability to emphasize semantic direction while remaining invariant to vector magnitude.

B.2 Intra-Class Distance

Let $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ denote the set of context representations, where each representation $\mathbf{x}_i$ is associated with a class label $y_i \in \{1, \ldots, C\}$. For each class c, we define the subset of representations belonging to that class as

$$\mathcal{X}_c = \{\mathbf{x}_i \mid y_i = c\}, \tag{9}$$

with cardinality $n_c = |\mathcal{X}_c|$. In our experiments, $\mathcal{X}_c$ consists of the class-conditional context vectors for class c obtained from multiple independent runs with different random seeds.

The *intra-class distance* is computed as the average cosine distance over all unordered pairs of representations belonging to the same class, aggregated across all classes:

$$\text{Intra} = \frac{\sum_{c=1}^{C} \sum\limits_{\substack{i<j \\ \mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}_c}} d(\mathbf{x}_i, \mathbf{x}_j)}{\sum_{c=1}^{C} \binom{n_c}{2}}. \tag{10}$$

This metric reflects the degree of dispersion of representations within each class.

B.3 Inter-Class Distance

The *inter-class distance* is defined as the average cosine distance between representations drawn from different classes:

$$\text{Inter} = \frac{\sum_{c_1<c_2} \sum_{\mathbf{x}_i \in \mathcal{X}_{c_1}} \sum_{\mathbf{x}_j \in \mathcal{X}_{c_2}} d(\mathbf{x}_i, \mathbf{x}_j)}{\sum_{c_1<c_2} n_{c_1} n_{c_2}}. \tag{11}$$

This quantity captures the overall semantic separation between classes in the representation space.

B.4 Inter-to-Intra Distance Ratio

To summarize class separability with a single scalar metric, we further report the *inter-to-intra distance ratio*:

$$\text{Ratio} = \frac{\text{Inter}}{\text{Intra}}. \tag{12}$$

A higher ratio indicates that inter-class distances increase more substantially than intra-class distances, suggesting improved discriminability of the learned representations.

We emphasize that all distances are computed directly in the original representation space, without relying on dimensionality reduction techniques such as t-SNE, which may distort pairwise distances. Moreover, this analysis does not depend on training an auxiliary classifier, allowing us to assess representation geometry in a model-agnostic manner.

C C^{3}V on Other Large Language Models

In this section, we evaluate the performance of C^{3}V on a diverse set of large language models beyond LLaMA2-7B, including models from the LLaMA, GPT, and Qwen families. All experiments follow the same experimental protocol as described in the main text, including dataset splits, demonstration construction, coefficient training, and evaluation metrics. For brevity, we focus on reporting the final performance results and key trends, while detailed analyses are deferred to the main results on LLaMA2-7B. Due to the limited context window of GPT2-XL, the AGNews and DBPedia datasets were not evaluated under the I2CL setting on GPT2-XL. For the same reason, our method, C^{3}V, was also not evaluated on these datasets when using GPT2-XL. As shown in Table 6–10, overall, C^{3}V generally outperforms the baselines across all evaluated models, with a single exception: on LLaMA3.2-1B, the Soft Prompt baseline achieves better performance than C^{3}V. This behavior may be related to the limited capacity of LLaMA3.2-1B, under which prompt-tuning-based adaptation methods such as Soft Prompt can sometimes be more effective. It is worth noting that, compared to C^{3}V, Soft Prompt involves a larger number of trainable parameters. Notably, even on the strong Qwen3-8B model, our method achieves an average improvement of nearly 0.8 points over the strongest baseline across all datasets.

D Attention- and MLP-Based Class-Conditional Latent Vector Visualizations for Other Models

Appendix D provides additional visualizations of class-conditional latent vectors obtained from other language models. Following the same visualization procedure as in Section 4.3.1, we concatenate layer-wise attention or MLP representations for each class-conditional context and project them into two dimensions using t-SNE. These results complement the main analysis and demonstrate that the observed representational patterns generalize across different models. The corresponding visualizations are shown in Figures 5–9.

E Inter- and Intra-Class Distance Analysis for Other Models

This appendix presents the inter- and intra-class distance analysis for additional language models beyond LLaMA2-7B. The analysis follows the same experimental protocol and distance definitions as described in Appendix B, using cosine distance for all computations. These results complement the main analysis and further assess the generality of the observed representational patterns across different models. The corresponding visualizations are shown in Figures 10–14.

F Layer-wise Coefficients of LLaMA2-7B

We present the layer-wise coefficients of LLaMA2-7B across all datasets for completeness and visualization purposes. As shown in Figure 15, C^{3}V provides finer control over information injection by learning layer-wise coefficients that are not strictly positive. The coefficients fluctuate across layers, implying that contextual information may be reduced rather than consistently amplified at different depths.

Name	Accuracy(%)
Zero-shot	58.13
MHA	68.90
MLP	73.13
Hidden state	59.92
MHA+MLP	76.32

(a) Target Modules

Name	Accuracy(%)
Zero-shot	58.13
Early	59.37
Middle	70.40
Late	67.54
All(ours)	76.32

(b) Target Layers

Name	Accuracy(%)
Zero-shot	58.13
First	67.78
Last	71.09
All(ours)	76.32

(c) Injection Pos

Name	Accuracy(%)
Zero-shot	58.13
First	34.12
Random	69.29
Last (ours)	76.32

(d) Extraction Pos

Name	Accuracy(%)
Zero-shot	58.13
PCA	75.08
Mean(ours)	76.32

(e) Post Fuse Method

Table 5: Comparison of Different Settings.

G Additional Ablation Studies

Target Module. We study different target modules for activation extraction and injection, including MHA, MLP, hidden states, and their combination. As shown in Table 5a, intervening at either MHA or MLP improves performance over the baseline, with the combined MHA+MLP setting achieving the best results. In contrast, using hidden states provides only marginal gains, likely due to their mixed and less targeted representations.

Target Layer. We examine where to inject context by grouping model layers into early, middle, and late stages, each containing one-third of the total layers. As shown in Table 5b, injecting context into middle or late layers yields larger gains than early-layer injection, while applying context across all layers achieves the best performance. This suggests that injecting contextual information across layers helps refine representations throughout the network.

Injection Position. By default, C^3V injects context across all token positions. We compare this strategy with alternatives that inject context at a single position, including the first token or the last token. As shown in Table 5c, injecting at the last token outperforms the first setting, while injecting across all positions achieves the best results.

Extraction Position. We study the effect of extracting class-conditional context vectors from different token positions, including the first token, a randomly selected token, and the last token in the sequence. As shown in Table 5d, extracting context vectors from the last token yields the best performance, substantially outperforming other positions. In contrast, using the first token leads to a significant performance drop. These findings suggest that extracting context vectors from the last token yields the most effective performance, while using the first token leads to a significant drop, and random extraction performs worse than extraction from the last token.

Context Vector Calculation. We analyze how C^3V aggregates class-conditional context vectors across layers, comparing mean pooling and PCA-based fusion. Mean fusion averages layer-wise context vectors, whereas PCA fusion performs dimensionality reduction before aggregation. As shown in Table 5e, mean fusion outperforms PCA, indicating that simple averaging better preserves class-relevant information for effective context injection in C^3V.

H Dataset Templates and Labels

The dataset templates and label spaces used in the classification tasks for our method and baselines are shown in Table 11.

I C^3V Algorithm Pseudocode

Algorithm 1 provides the pseudocode of the C^3V algorithm. The pseudocode summarizes the overall training procedure, including the construction of class-conditional context vectors, coefficient initialization, and coefficient optimization.

Method	SST-2	SST-5	TREC	AGNews	Subj	HateS	DBPedia	EmoC	MR	Avg.
Zero-shot	91.00	30.60	53.80	66.60	45.00	51.00	63.40	59.40	89.00	61.09
Few-shot	$94.00_{0.40}$	$46.72_{1.31}$	$84.96_{3.05}$	$85.48_{1.57}$	$50.04_{0.08}$	$52.72_{3.18}$	$87.20_{0.78}$	$66.36_{3.96}$	$92.48_{0.55}$	73.33
Soft Prompt	$50.20_{0.08}$	$23.93_{0.66}$	$80.00_{1.80}$	$82.27_{8.13}$	$59.80_{0.40}$	$52.40_{0.96}$	$34.80_{4.57}$	$54.30_{9.84}$	$51.00_{0.40}$	54.30
Task-vector	$94.00_{0.85}$	$35.44_{2.06}$	$57.48_{1.27}$	$83.44_{4.56}$	$45.76_{2.05}$	$51.56_{0.64}$	$69.52_{1.62}$	$62.28_{0.52}$	$87.60_{0.33}$	65.23
Label-anchor	$89.88_{1.65}$	$34.12_{2.05}$	$79.64_{2.05}$	$70.28_{2.82}$	$46.96_{14.34}$	$52.40_{3.44}$	$54.36_{19.58}$	$68.12_{3.95}$	$87.56_{0.74}$	64.81
I2CL	$90.36_{2.44}$	$35.92_{5.99}$	$81.68_{3.69}$	$84.00_{2.25}$	$85.00_{3.34}$	$72.40_{2.44}$	$86.80_{3.08}$	$68.04_{3.30}$	$86.92_{1.52}$	76.79
C^3V	$89.08_{2.40}$	$38.56_{4.21}$	$86.80_{3.53}$	$85.20_{2.19}$	$82.96_{2.54}$	$70.52_{3.73}$	$91.40_{0.03}$	$67.52_{5.13}$	$86.32_{0.60}$	77.60

Table 6: Classification accuracy (in %) on nine datasets using Qwen3-8B. Results are reported as acc_{std}, where std denotes the standard deviation over multiple runs.

Method	SST-2	SST-5	TREC	AGNews	Subj	HateS	DBPedia	EmoC	MR	Avg.
Zero-shot	60.60	28.00	46.80	79.80	55.60	50.80	49.20	39.20	56.20	51.80
Few-shot	$64.32_{9.69}$	$20.56_{0.93}$	$71.16_{3.19}$	$86.00_{1.23}$	$50.00_{0.00}$	$50.00_{0.00}$	$83.84_{2.21}$	$25.44_{0.78}$	$60.84_{5.81}$	56.91
Soft Prompt	$58.60_{4.66}$	$34.60_{1.59}$	$65.84_{6.75}$	$83.04_{3.71}$	$58.32_{4.21}$	$59.64_{4.07}$	$82.60_{1.78}$	$50.28_{8.61}$	$67.16_{7.90}$	62.23
Task-vector	$69.88_{2.83}$	$24.52_{2.92}$	$65.76_{1.63}$	$86.08_{2.52}$	$51.36_{0.74}$	$49.28_{4.96}$	$75.32_{3.72}$	$48.08_{0.90}$	$66.00_{3.37}$	59.59
Label-anchor	$58.92_{1.57}$	$28.80_{4.76}$	$51.76_{1.42}$	$76.36_{3.04}$	$50.72_{0.45}$	$50.00_{0.13}$	$62.00_{5.04}$	$25.84_{0.50}$	$58.64_{1.27}$	51.45
I2CL	$54.96_{2.80}$	$26.20_{2.73}$	$50.00_{8.57}$	$81.52_{3.26}$	$63.04_{3.93}$	$54.48_{7.47}$	$66.52_{4.73}$	$45.36_{2.92}$	$70.48_{4.21}$	56.95
C^3V	$57.80_{6.86}$	$26.56_{2.50}$	$63.88_{9.61}$	$83.60_{2.69}$	$60.32_{8.13}$	$55.48_{7.97}$	$87.12_{2.33}$	$50.64_{4.17}$	$69.08_{5.74}$	61.61

Table 7: Classification accuracy (in %) on nine datasets using LLaMA3.2-1B. Results are reported as acc_{std}, where std denotes the standard deviation over multiple runs.

Method	SST-2	SST-5	TREC	AGNews	Subj	HateS	DBPedia	EmoC	MR	Avg.
Zero-shot	55.60	31.40	66.40	73.60	51.00	50.40	56.20	41.00	55.60	53.47
Few-shot	$93.00_{0.62}$	$39.40_{3.21}$	$78.20_{5.94}$	$84.80_{1.30}$	$62.47_{12.44}$	$64.93_{2.65}$	$82.33_{3.64}$	$52.20_{3.92}$	$92.73_{0.50}$	72.23
Noise vector	$68.48_{6.60}$	$38.60_{3.03}$	$66.88_{9.34}$	$82.16_{2.78}$	$69.96_{5.73}$	$56.04_{5.09}$	$80.44_{4.29}$	$43.84_{7.59}$	$77.76_{8.82}$	64.91
Soft Prompt	$58.96_{12.03}$	$29.56_{7.61}$	$74.24_{10.38}$	$85.64_{2.77}$	$69.00_{9.78}$	$62.68_{5.03}$	$86.40_{0.00}$	$54.32_{11.88}$	$60.32_{11.85}$	64.57
Task-vector	$69.28_{8.96}$	$35.20_{2.83}$	$64.80_{1.50}$	$84.00_{3.74}$	$50.40_{0.00}$	$55.60_{3.41}$	$73.28_{1.27}$	$41.64_{0.99}$	$66.28_{4.70}$	60.05
Label-anchor	$52.32_{1.84}$	$29.00_{4.99}$	$43.40_{0.54}$	$74.64_{0.85}$	$50.20_{0.70}$	$66.24_{8.73}$	$72.04_{2.00}$	$56.20_{3.01}$	$58.28_{3.27}$	55.81
I2CL	$91.40_{2.26}$	$32.60_{1.25}$	$80.27_{2.58}$	$82.56_{1.57}$	$64.67_{3.52}$	$75.80_{1.02}$	$85.00_{0.33}$	$52.92_{5.28}$	$84.27_{2.07}$	72.17
C^3V	$89.16_{2.33}$	$43.60_{3.86}$	$78.24_{3.95}$	$85.04_{2.57}$	$64.96_{5.84}$	$62.72_{4.80}$	$91.56_{1.37}$	$64.80_{4.64}$	$88.52_{3.73}$	74.29

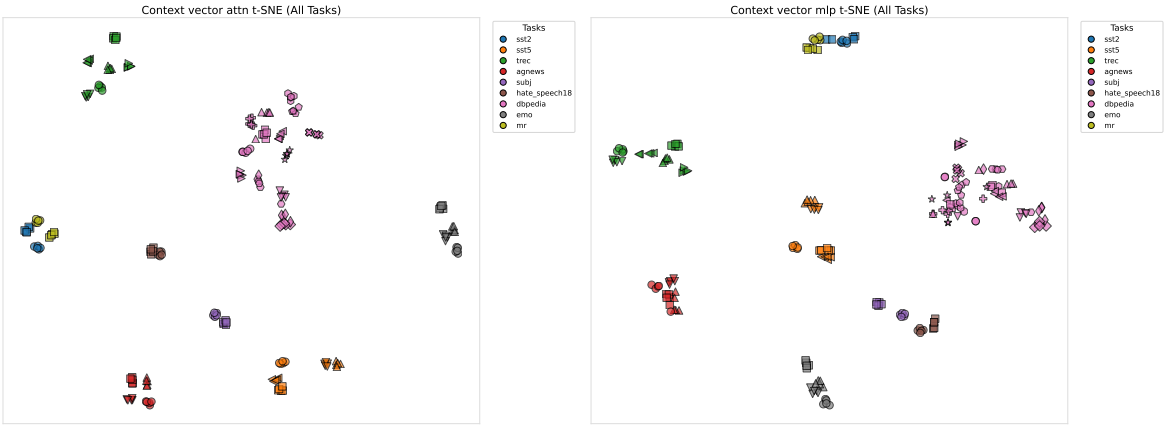
Table 8: Classification accuracy (in %) on nine datasets using LLaMA3-8B. Results are reported as acc_{std}, where std denotes the standard deviation over multiple runs.

Method	SST-2	SST-5	TREC	AGNews	Subj	HateS	DBPedia	EmoC	MR	Avg.
Zero-shot	77.76	25.60	68.20	71.60	62.40	59.92	65.56	44.68	76.88	61.40
Few-shot	$89.44_{2.60}$	$39.65_{4.57}$	$67.76_{2.11}$	$83.18_{2.03}$	$50.20_{0.22}$	$53.44_{6.84}$	$93.30_{1.19}$	$47.62_{8.62}$	$88.66_{1.23}$	68.14
Soft Prompt	$68.32_{11.99}$	$37.68_{2.91}$	$68.24_{3.83}$	$82.28_{6.22}$	$63.72_{7.50}$	$63.48_{5.93}$	$86.24_{2.50}$	$49.24_{7.64}$	$73.84_{9.23}$	65.89
Task-vector	$59.80_{4.47}$	$31.20_{2.82}$	$67.32_{0.32}$	$80.12_{2.23}$	$66.32_{2.31}$	$70.12_{2.22}$	$77.64_{4.63}$	$45.00_{4.20}$	$81.36_{3.58}$	64.32
Label-anchor	$86.12_{3.95}$	$27.44_{4.21}$	$51.68_{4.87}$	$77.52_{1.53}$	$52.64_{2.99}$	$54.32_{7.27}$	$90.16_{1.13}$	$42.16_{10.04}$	$87.56_{1.38}$	63.29
I2CL	$85.48_{1.18}$	$37.32_{3.11}$	$63.84_{7.58}$	$81.56_{3.13}$	$65.56_{8.33}$	$62.32_{5.76}$	$81.84_{4.50}$	$50.32_{4.68}$	$84.40_{2.45}$	68.07
C^3V	$84.72_{4.18}$	$38.20_{2.02}$	$68.44_{7.68}$	$82.08_{3.48}$	$66.24_{7.87}$	$63.88_{4.50}$	$90.36_{1.67}$	$57.16_{3.74}$	$83.64_{3.17}$	70.52

Table 9: Classification accuracy (in %) on nine datasets using GPT-J-6B. Results are reported as acc_{std}, where std denotes the standard deviation over multiple runs.

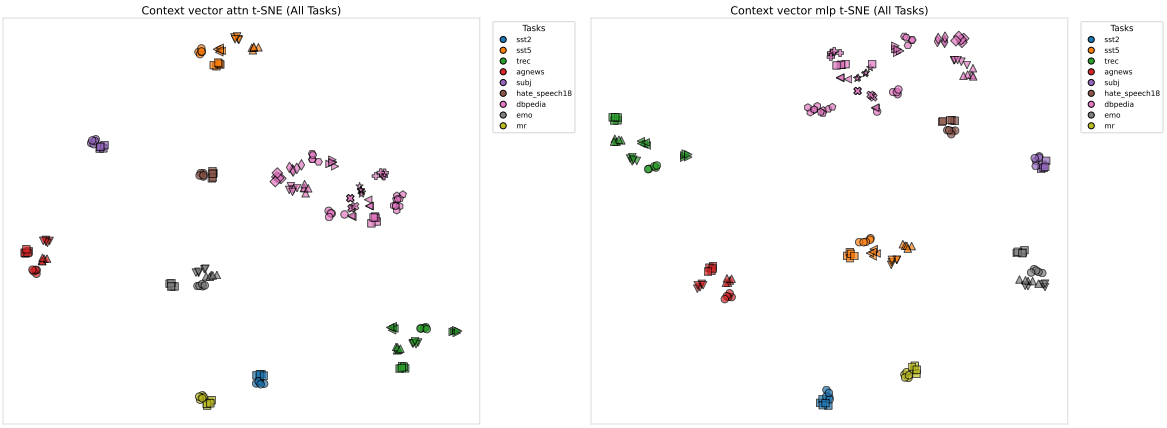
Method	SST-2	SST-5	TREC	Subj	HateS	EmoC	MR	Avg.
Zero-shot	74.60	31.40	35.40	64.40	71.80	38.60	70.40	55.23
Few-shot	$73.44_{8.97}$	$35.92_{2.41}$	$60.64_{5.00}$	$64.28_{10.89}$	$51.76_{3.06}$	$38.64_{7.46}$	$75.76_{9.49}$	57.21
Soft Prompt	$59.24_{5.12}$	$23.48_{2.16}$	$46.04_{8.46}$	$52.76_{3.74}$	$62.72_{7.50}$	$36.08_{3.28}$	$55.84_{6.06}$	48.02
Task-vector	$81.08_{4.87}$	$28.52_{1.37}$	$41.40_{5.35}$	$71.80_{1.86}$	$62.48_{2.83}$	$37.60_{2.48}$	$78.40_{2.36}$	57.33
Label-anchor	$62.36_{8.35}$	$24.28_{2.10}$	$68.60_{7.99}$	$55.36_{8.09}$	$52.56_{4.20}$	$36.20_{4.00}$	$61.72_{9.30}$	51.58
I2CL	$80.16_{3.98}$	$33.84_{2.60}$	$51.48_{5.26}$	$65.96_{4.83}$	$68.32_{4.76}$	$47.92_{1.84}$	$83.20_{3.29}$	61.55
C^3V	$83.60_{1.55}$	$37.24_{4.82}$	$58.40_{8.09}$	$69.32_{7.62}$	$65.24_{5.83}$	$55.95_{4.79}$	$82.16_{4.89}$	64.56

Table 10: Classification accuracy (in %) on seven datasets using GPT-XL. Results are reported as acc_{std}, where std denotes the standard deviation over multiple runs.



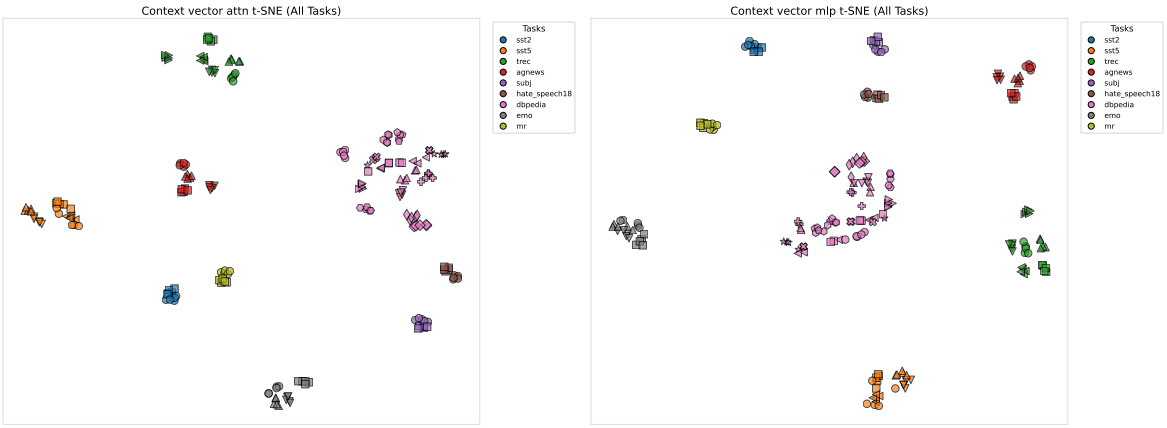
(a) Attention (b) MLP

Figure 5: t-SNE visualization of class-conditional latent vectors on Qwen3-8B. Left: attention-based latent vectors; right: MLP-based latent vectors. Points are colored by dataset and shaped by class label. Latent vectors from five random seeds are included for each class.



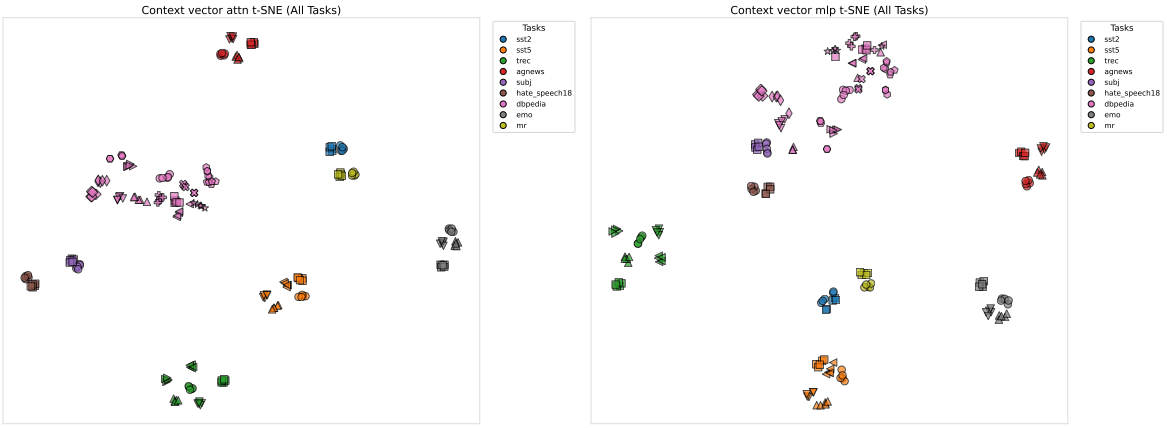
(a) Attention (b) MLP

Figure 6: t-SNE visualization of class-conditional latent vectors on LLaMA3.2-1B. Left: attention-based latent vectors; right: MLP-based latent vectors. Points are colored by dataset and shaped by class label. Latent vectors from five random seeds are included for each class.



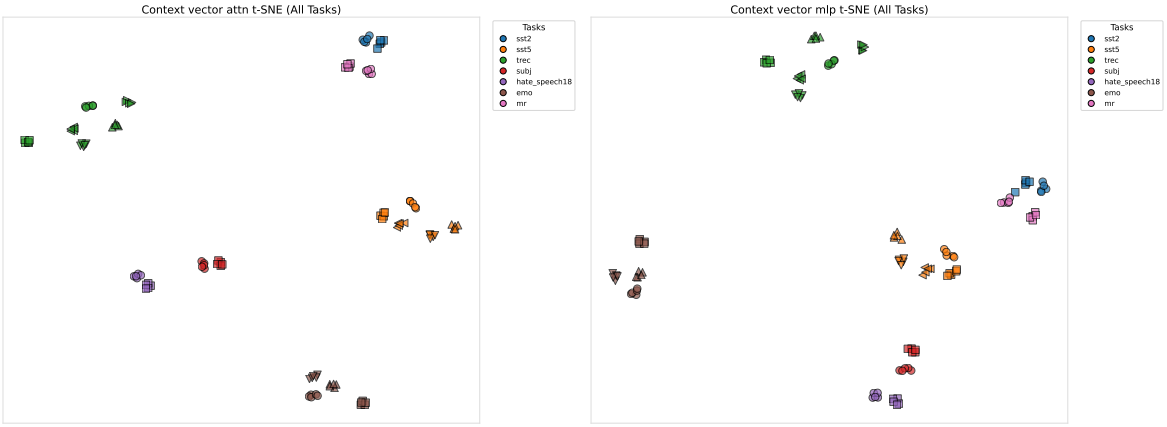
(a) Attention (b) MLP

Figure 7: t-SNE visualization of class-conditional latent vectors on LLaMA3-8B. Left: attention-based latent vectors; right: MLP-based latent vectors. Points are colored by dataset and shaped by class label. Latent vectors from five random seeds are included for each class.



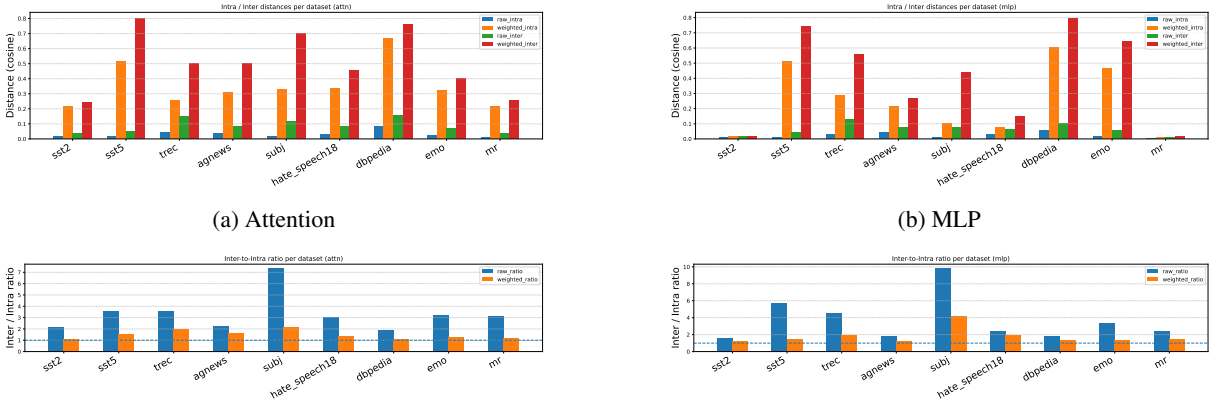
(a) Attention (b) MLP

Figure 8: t-SNE visualization of class-conditional latent vectors on GPT-J-6B. Left: attention-based latent vectors; right: MLP-based latent vectors. Points are colored by dataset and shaped by class label. Latent vectors from five random seeds are included for each class.



(a) Attention (b) MLP

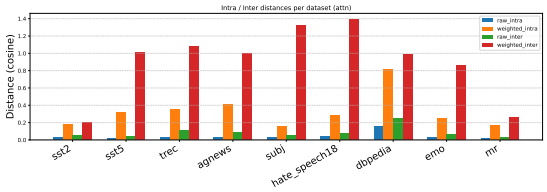
Figure 9: t-SNE visualization of class-conditional latent vectors on GPT2-XL. Left: attention-based latent vectors; right: MLP-based latent vectors. Points are colored by dataset and shaped by class label. Latent vectors from five random seeds are included for each class.



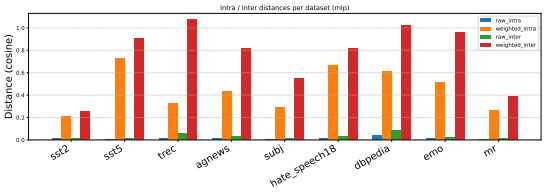
(a) Attention (b) MLP

(c) Attention (Inter/Intra Ratio) (d) MLP (Inter/Intra Ratio)

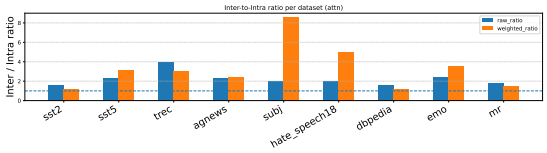
Figure 10: Inter- and intra-class distance analysis of latent representations across datasets on Qwen3-8b. (a,b) Inter- and intra-class distances for attention-based and MLP-based representations, where each panel includes distances computed both without coefficient scaling and after multiplication by trained coefficients. (c,d) Corresponding inter/intra ratios for attention and MLP representations, comparing the settings without and with trained coefficient scaling.



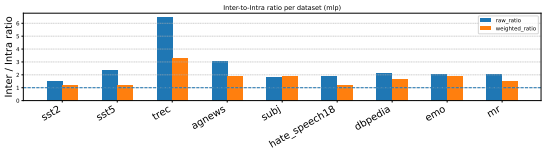
(a) Attention



(b) MLP

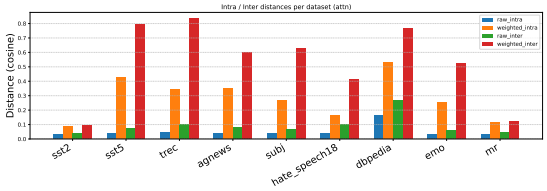


(c) Attention (Inter/Intra Ratio)

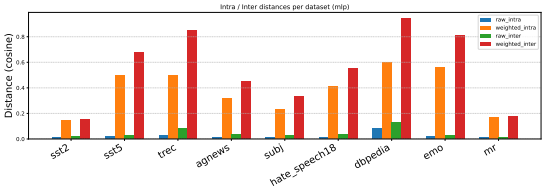


(d) MLP (Inter/Intra Ratio)

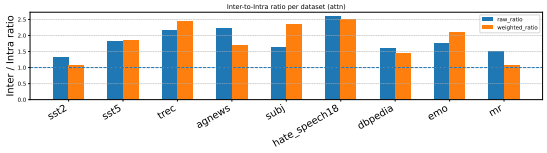
Figure 11: Inter- and intra-class distance analysis of latent representations across datasets on LLaMA3.2-1b. (a,b) Inter- and intra-class distances for attention-based and MLP-based representations, where each panel includes distances computed both without coefficient scaling and after multiplication by trained coefficients. (c,d) Corresponding inter/intra ratios for attention and MLP representations, comparing the settings without and with trained coefficient scaling.



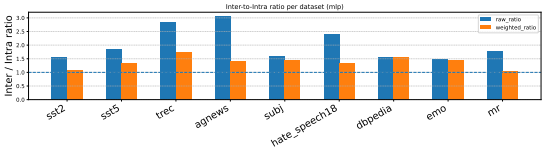
(a) Attention



(b) MLP

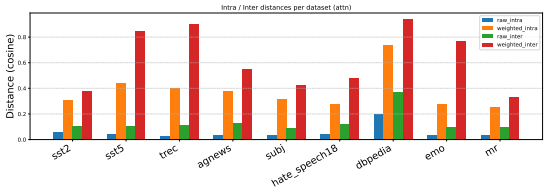


(c) Attention (Inter/Intra Ratio)

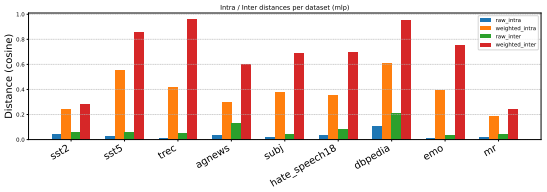


(d) MLP (Inter/Intra Ratio)

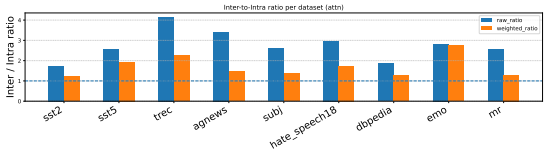
Figure 12: Inter- and intra-class distance analysis of latent representations across datasets on LLaMA3-8b. (a,b) Inter- and intra-class distances for attention-based and MLP-based representations, where each panel includes distances computed both without coefficient scaling and after multiplication by trained coefficients. (c,d) Corresponding inter/intra ratios for attention and MLP representations, comparing the settings without and with trained coefficient scaling.



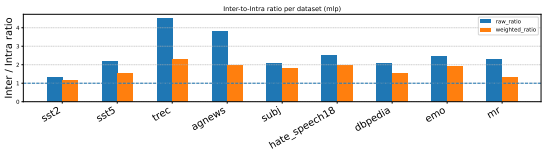
(a) Attention



(b) MLP

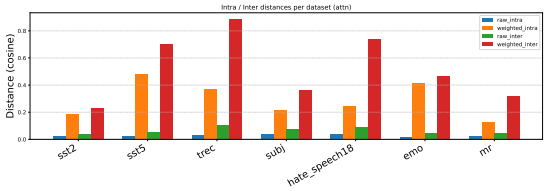


(c) Attention (Inter/Intra Ratio)

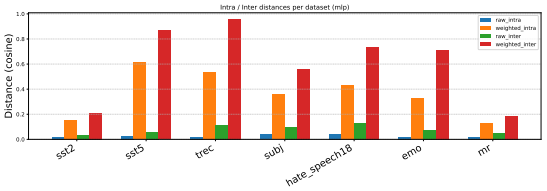


(d) MLP (Inter/Intra Ratio)

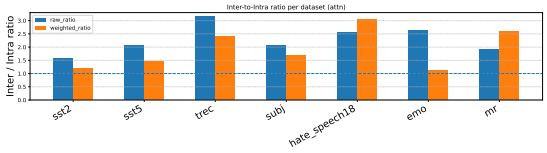
Figure 13: Inter- and intra-class distance analysis of latent representations across datasets on GPT-J-6B. (a,b) Inter- and intra-class distances for attention-based and MLP-based representations, where each panel includes distances computed both without coefficient scaling and after multiplication by trained coefficients. (c,d) Corresponding inter/intra ratios for attention and MLP representations, comparing the settings without and with trained coefficient scaling.



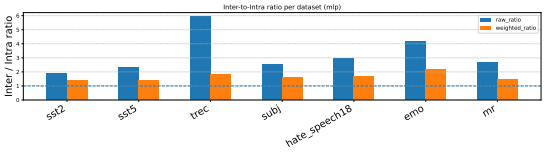
(a) Attention



(b) MLP

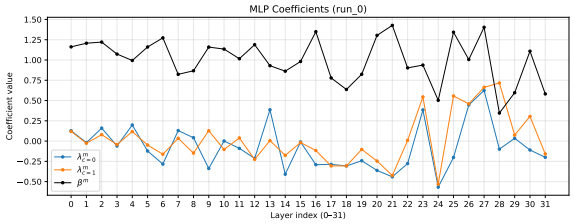


(c) Attention (Inter/Intra Ratio)

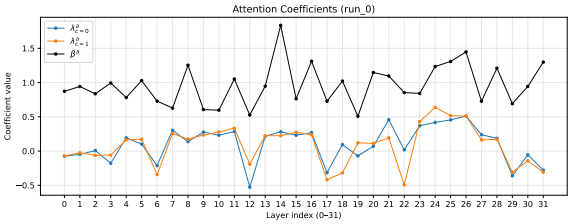


(d) MLP (Inter/Intra Ratio)

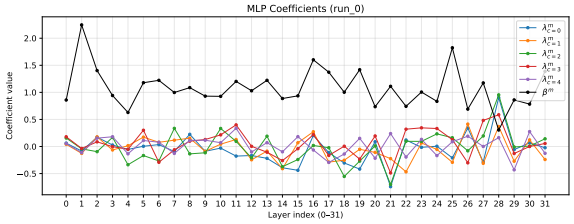
Figure 14: Inter- and intra-class distance analysis of latent representations across datasets on GPT2-XL. (a,b) Inter- and intra-class distances for attention-based and MLP-based representations, where each panel includes distances computed both without coefficient scaling and after multiplication by trained coefficients. (c,d) Corresponding inter/intra ratios for attention and MLP representations, comparing the settings without and with trained coefficient scaling.



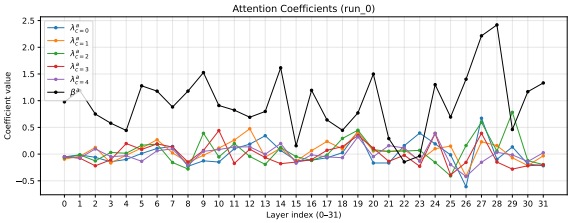
(a) sst2: MLP



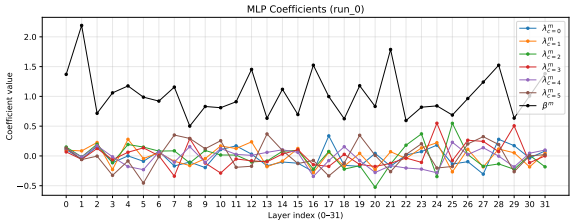
(b) sst2: Attention



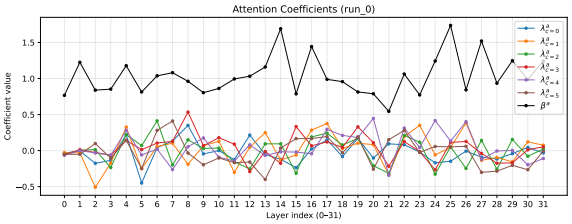
(c) sst5: MLP



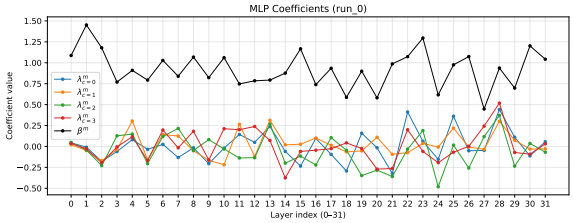
(d) sst5: Attention



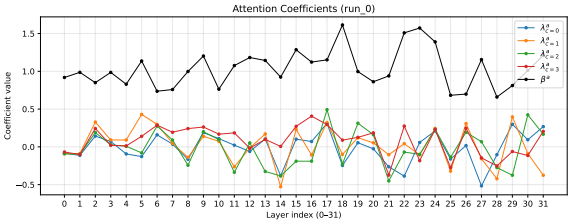
(e) trec: MLP



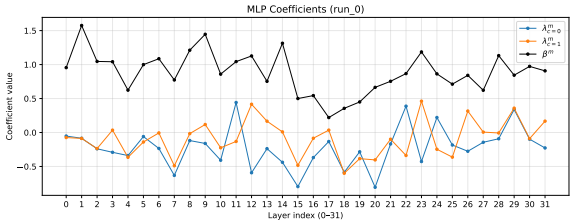
(f) trec: Attention



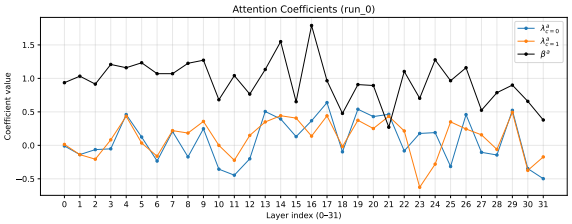
(g) agnews: MLP



(h) agnews: Attention

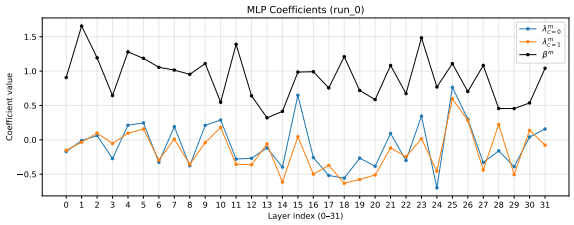


(i) subj: MLP

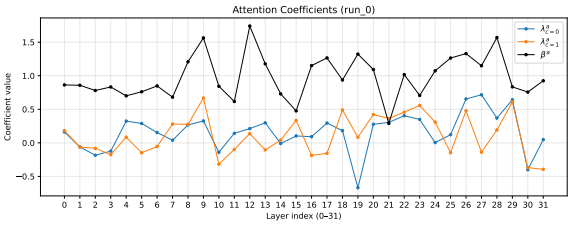


(j) subj: Attention

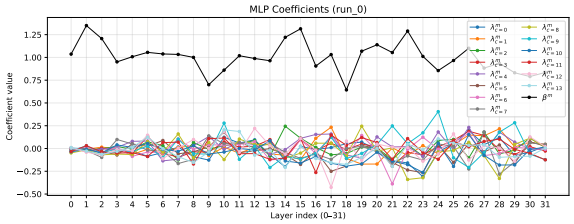
Figure 15: Layer-wise learned injection coefficients on LLaMA2-7B. Each row corresponds to a dataset; the left plot shows MLP coefficients, and the right plot shows attention coefficients. For each plot, the x-axis denotes the layer index, and the y-axis represents the learned coefficient values.



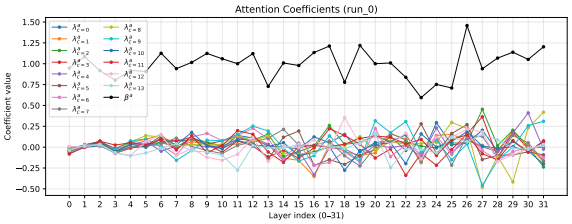
(k) hate_speech18: MLP



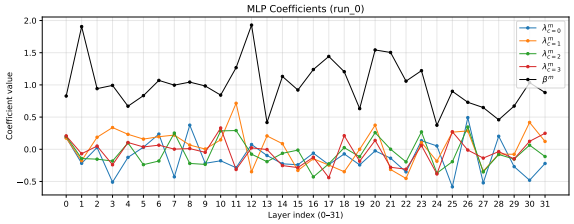
(l) hate_speech18: Attention



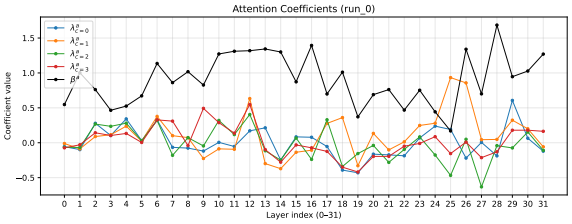
(m) dbpedia: MLP



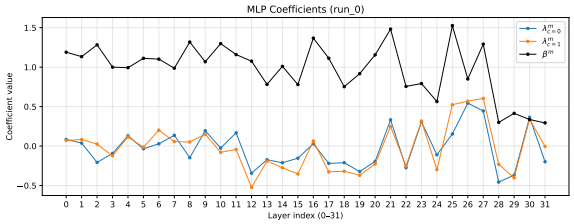
(n) dbpedia: Attention



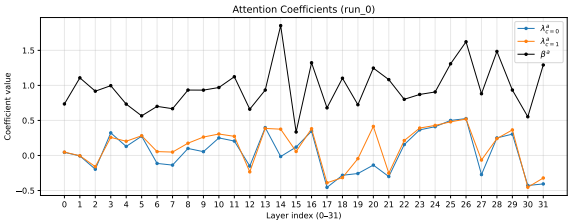
(o) emo: MLP



(p) emo: Attention



(q) mr: MLP



(r) mr: Attention

Figure 15: (continued)

Dataset	Template	Label Space
SST-2	Review: {Sentence} Sentiment: {Label}	negative / positive
SST-5	Sentence: {Sentence} Sentiment: {Label}	terrible / negative / neutral / positive / great
MR	Review: {Sentence} Sentiment: {Label}	negative / positive
Subj	Sentence: {Sentence} Label: {Label}	objective / subjective
DBPedia	Input: {Sentence} Label: {Label}	company / school / artist / athlete / politics / transportation / building / nature / village / animal / plant / album / film / book
AGNews	News: {Sentence} Type: {Label}	World / Sports / Business / Technology
TREC	Question: {Sentence} Answer Type: {Label}	Abbreviation / Entity / Person / Location / Number / Description
HateSpeech18	Text: {Sentence} Label: {Label}	neutral / hate
EmoC	Dialogue: {Sentence} Emotion: {Label}	others / happy / sad / angry

Table 11: Dataset templates and label spaces used in classification tasks.

Algorithm 1: Class-Conditional Context Vectors (C^3V)

Input: Frozen LM f_θ with L layers; demonstrations $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$; class set $\mathcal{C}$; token position for extraction e; training steps TS; learning rate η

Output: Class-conditional context vectors $\{\bar{d}_c\}_{c\in\mathcal{C}}$ and learned coefficients $\mathbf{c}$

Stage 1: Build class-conditional context vectors

foreach $c \in \mathcal{C}$ **do**
- $\mathcal{D}_c \leftarrow \{(x_i, y_i) \in \mathcal{D} \mid y_i = c\}$;
- **foreach** $(x_i, y_i) \in \mathcal{D}_c$ **do**
  - Run f_θ on (x_i, y_i) and extract activations at position e;
  - $\{a^e_{i,l}, m^e_{i,l}\}_{l=1}^{L}$;
- **for** $l = 1$ **to** L **do**
  - $\bar{a}^e_{c,l} \leftarrow \frac{1}{|\mathcal{D}_c|} \sum_{(x_i,y_i)\in\mathcal{D}_c} a^e_{i,l}$;
  - $\bar{m}^e_{c,l} \leftarrow \frac{1}{|\mathcal{D}_c|} \sum_{(x_i,y_i)\in\mathcal{D}_c} m^e_{i,l}$;
- $\bar{d}_c \leftarrow \{\bar{a}^e_{c,l}, \bar{m}^e_{c,l}\}_{l=1}^{L}$;

Stage 2: Initialize learnable coefficients

Initialize $\lambda^a_{c,l}, \lambda^m_{c,l} \leftarrow 0.1/|\mathcal{C}|$;

Initialize $\beta^a_l, \beta^m_l \leftarrow 1$;

$\mathbf{c} \leftarrow \{\lambda^a_{c,l}, \lambda^m_{c,l}, \beta^a_l, \beta^m_l\}^{L}_{c\in\mathcal{C},\, l=1}$;

Stage 3: Learn coefficients on demonstrations (backbone frozen)

for $ts = 1$ **to** TS **do**
- Sample mini-batch $\mathcal{B} \subseteq \mathcal{D}$;
- $\mathcal{L} \leftarrow 0$;
- **foreach** $(x, y) \in \mathcal{B}$ **do**
  - Run a *single* forward pass with class-conditional injection;
  - **for** $l = 1$ **to** L **do**
    - **foreach** *token position* t *in the sequence* **do**
      - $\mathbf{r}^t_l \leftarrow \mathbf{r}^t_{l-1} + \sum_{c\in\mathcal{C}} \lambda^a_{c,l} \bar{a}^e_{c,l} + \beta^a_l a^t_l + \sum_{c\in\mathcal{C}} \lambda^m_{c,l} \bar{m}^e_{c,l} + \beta^m_l m^t_l$;
  - Compute $P(y \mid x, \{\bar{d}_c\}, \mathbf{c})$ from the resulting logits;
  - $\mathcal{L} \leftarrow \mathcal{L} - \log P(y \mid x, \{\bar{d}_c\}, \mathbf{c})$;
- $\mathcal{L} \leftarrow \mathcal{L}/|\mathcal{B}|$;
- Update coefficients: $\mathbf{c} \leftarrow \mathbf{c} - \eta \nabla_{\mathbf{c}} \mathcal{L}$;

Stage 4: Inference

Given query x, run one injected forward pass using $\{\bar{d}_c\}$ and $\mathbf{c}$;

Compute class probabilities from label-token logits and output $\hat{y} = \arg\max_{c\in\mathcal{C}} P(y_c \mid x, \{\bar{d}_c\}, \mathbf{c})$;