Where and What Matters: Sensitivity-Aware Task Vectors for Many-Shot Multimodal In-Context Learning

Ziyu Ma[1*], **Chenhui Gou**[2*], **Yiming Hu**[1†], **Yong Wang**[1†], **Bohan Zhuang**[3], **Jianfei Cai**[2]
[1]AMAP, Alibaba Group, China
[2]Data Science & AI Department, Faculty of IT, Monash University, Australia
[3]ZIP Lab, Zhejiang University, China
maziyu0@gmail.com, {luxiao.hym, wangyong.lz}@alibaba-inc.com, {chenhui.gou, bohan.zhuang, jianfei.cai}@monash.edu

Abstract

Large Multimodal Models (LMMs) have shown promising in-context learning (ICL) capabilities, but scaling to many-shot settings remains difficult due to limited context length and high inference cost. To address these challenges, task-vector-based methods have been explored by inserting compact representations of many-shot in-context demonstrations into model activations. However, existing task-vector-based methods either overlook the importance of where to insert task vectors or struggle to determine suitable values for each location. To this end, we propose a novel Sensitivity-aware Task Vector insertion framework (STV) to figure out *where and what* to insert. Our key insight is that activation deltas across query-context pairs exhibit consistent structural patterns, providing a reliable cue for insertion. Based on the identified sensitive-aware locations, we construct a pre-clustered activation bank for each location by clustering the activation values, and then apply reinforcement learning to choose the most suitable one to insert. We evaluate STV across a range of multimodal models (e.g., Qwen-VL, Idefics-2) and tasks (e.g., VizWiz, OK-VQA), demonstrating its effectiveness and showing consistent improvements over previous task-vector-based methods with strong generalization.

Code — https://github.com/AMAP-ML/STV

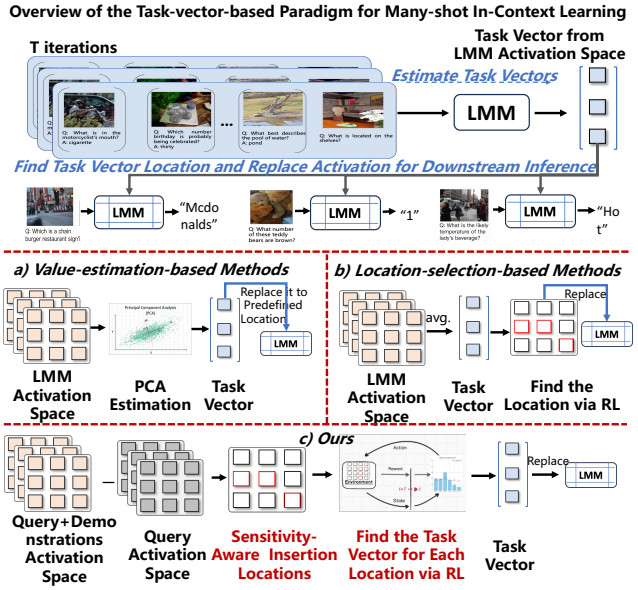

Figure 1: Comparison of previous task-vector-based methods and our sensitivity-aware method that systematically determines both insertion locations and task vector values.

1 Introduction

Large Multimodal Models (LMMs), such as GPT-4V (OpenAI 2023), LLaVA (Li et al. 2024a; Liu et al. 2024), and Qwen-VL (Wang et al. 2024; Bai et al. 2025), have demonstrated strong in-context learning (ICL) capabilities for vision-language tasks, particularly in few-shot scenarios (Bai et al. 2023; Laurençon et al. 2024b). Recent findings from commercial LMMs including Gemini (Comanici et al. 2025) and GPT-4o (Hurst et al. 2024), further highlight the potential of many-shot ICL, where increasing the number of in-context examples markedly enhances performance on multimodal tasks (Agarwal et al. 2024; Jiang et al. 2024). However, leveraging many-shot ICL in open-source LMMs remains challenging due to two key limitations: (1) limited context length (e.g., Qwen-VL (Bai et al. 2023) supports a maximum of 8192 tokens, but encoding a single image already consumes up to 256 tokens), and (2) substantial inference overhead, as incorporating more examples increases both memory usage and inference latency (Ma et al. 2025a).

The paradigm based on task vectors has recently emerged as a promising solution, attracting increasing attention. Unlike conventional methods that concatenate all in-context demonstrations into the input sequence, task-vector techniques (Hendel, Geva, and Globerson 2023; Liu et al. 2023; Todd et al. 2023; Huang et al. 2024) compress many-shot demonstrations into compact, semantically rich representations (i.e., task vectors). They are then integrated into the model's attention heads to address downstream tasks. By reframing the problem, this paradigm shifts focus from managing long input sequences to addressing two fundamental challenges: *(1) how to construct semantically meaningful task vectors, and (2) where within the model architecture*

*These authors contributed equally.
†Project leads and corresponding authors.

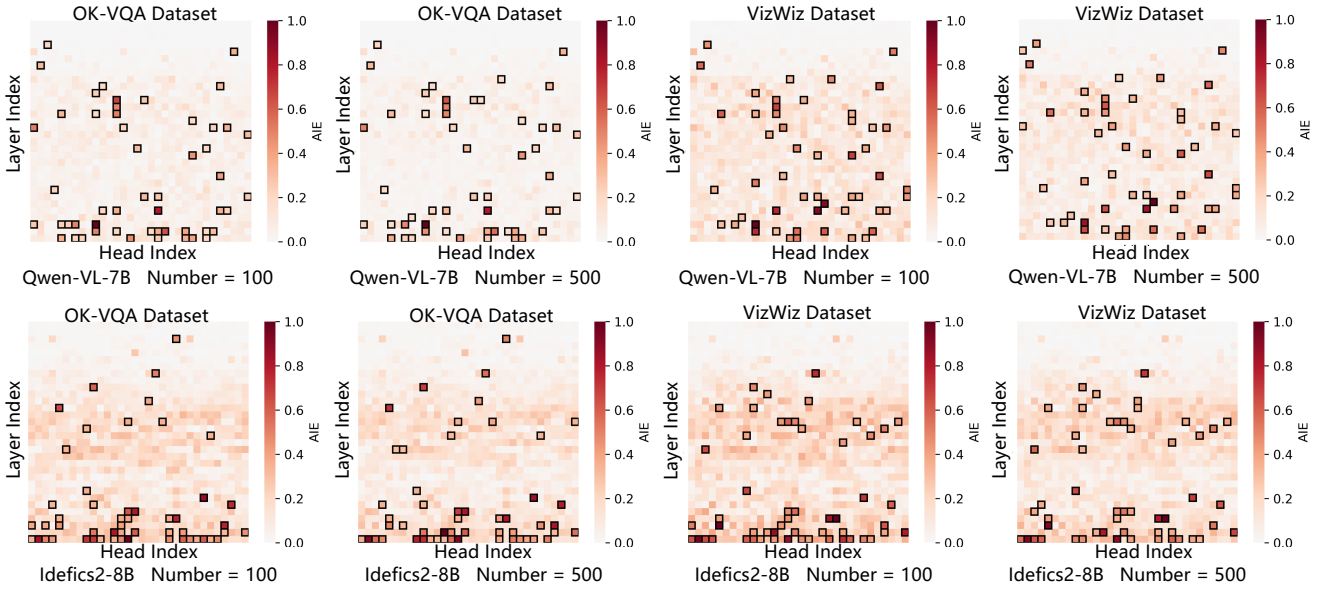

Figure 2: Attention head sensitivity is analyzed across datasets (VizWiz and OK-VQA), models (Qwen-VL-7B and Idefics2-8B), and sample sizes (100 vs. 500). Context-sensitive locations (black boxes) emerge within each task, confirming the stability of activation-delta patterns. These deltas are computed by contrasting query–context activations with query-only activations.

to insert them for optimal effectiveness.

Previous studies (Hendel, Geva, and Globerson 2023; Liu et al. 2023; Todd et al. 2023) have introduced the *value-estimation-based* methods (see Fig. 1(a)), which primarily focus on deriving task vectors from demonstrations and inserting them into predefined locations within the model. A representative approach (Liu et al. 2023) employs Principal Component Analysis (PCA) (Abdi and Williams 2010) on hidden states from multiple demonstrations and a dummy query, producing compact vectors. Such vectors are then used to replace the representation of the real query at fixed attention layers during inference. While this strategy mitigates the challenge of processing long input sequences and demonstrates efficacy for simple tasks, it struggles to generalize to complex, multimodal tasks (Peng et al. 2024). Furthermore, it does not account for the critical influence of insertion locations on task performance. In contrast, recent approaches (Huang et al. 2024) shift focus to optimizing insertion locations while using fixed task vectors, a framework known as the *location-selection-based* method (see Fig. 1(b)). They typically compute the mean activation across demonstrations to create a task vector, which is then used by a policy network to identify the insertion point for maximizing performance. However, these methods often harm the representation richness of task information and lead to the loss of task-specific details due to averaging multiple in-context demonstrations into a single fixed vector. Moreover, insertion typically relies on sampling-based policies, and we observe that repeated runs yield inconsistent locations, and some locations have little or no impact on model predictions, indicating suboptimal insertion.

The inherent limitations of existing methods raise a fundamental question: *Is it possible to determine where and what to insert in a more systematic and reliable manner?*

To address this issue, we propose a novel framework, Sensitivity-aware Task Vector insertion (STV), which determines *where* to insert and *what* to insert when incorporating task vectors. This framework is inspired by our observation that the effective insertion locations exhibit consistent, task-dependent patterns (see Section 2.1 for details). Based on this, STV consists of two stages: (1) identifying locations that are sensitive to contextual information, and (2) selecting optimal task vectors for insertion. In the first stage, we quantify contextual sensitivity by computing activation deltas between queries with and without demonstrations. Specifically, averaging these deltas across multiple samples highlights stable patterns, and the top-k locations with the highest sensitivity scores are chosen as insertion points. In the second stage, we first construct a pre-clustered activation bank for each location by clustering activation values derived from multiple query-context forward passes. For each identified sensitive location, a candidate task vector is then sampled from this activation bank by a learnable discrete distribution. And this distribution is iteratively updated through the REINFORCE algorithm (Williams 1992).

We systematically evaluate our method across multiple LMM families, including Qwen-VL (Bai et al. 2023), and Idefics-2 (Laurençon et al. 2024a), as well as a diverse set of multimodal benchmarks. The results demonstrate that our approach consistently outperforms the previous state-of-the-art task-vector-based method, MTV (Huang et al. 2024), while reducing location-searching time by over 98%. Besides, compared to traditional fine-tuning approaches, such as LoRA (Hu et al. 2022), our method achieves superior ac-

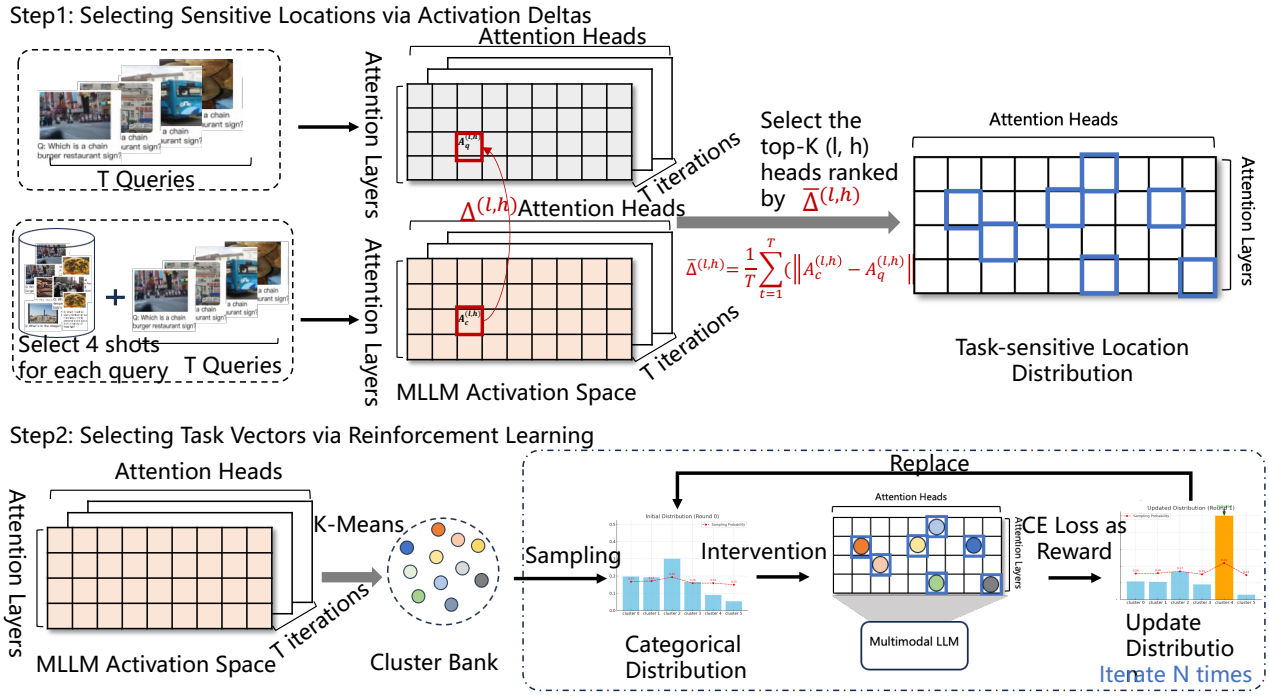

Figure 3: Overview of the STV framework. It has two stages: (1) identifying context-sensitive heads via activation deltas between query–context and query-only inputs; and (2) selecting task vectors from a pre-computed activation bank using reinforcement learning for those locations.

curacy with significantly lower computational resource requirements. Notably, it operates in an end-to-end manner on a single GPU with less than 20GB of memory, outperforming both zero-shot and few-shot ICL baselines (Brown et al. 2020) without requiring any model finetuning.

Our contributions can be summarized as follows:

- We observe that activation deltas exhibit consistent, task-dependent patterns within a given model, revealing structurally stable sensitivity to contextual information. Building on this insight, we propose a novel and efficient sensitivity-aware strategy to identify insertion locations.
- In contrast to previous methods that rely on PCA estimation or mean activations, we propose a policy-driven strategy that progressively selects the most effective task vector for each insertion location through RL.
- Empirical results across five vision-language tasks and two LMMs, consistently show that our method achieves superior performance, outperforming existing task-vector-based methods.

2 Method

We propose a Sensitivity-aware Task Vector insertion framework (STV) to enable efficient many-shot in-context learning in LMMs without extending context length. We begin by describing the background on multimodal ICL and task vectors, followed by our observations (Section 2.1). Then our two-step method is introduced: (1) identifying where to insert task vectors, and (2) selecting what to insert. The overall framework is illustrated in Fig. 3.

2.1 Preliminary Study

In the multimodal in-context learning (ICL) setting, a large multimodal model (LMM) learns to perform a new task based on a few interleaved image-text examples without parameter updates. The input typically takes the form:

$$I_{\text{few}} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n), q\}, \quad (1)$$

where (x_i, y_i) denotes the multimodal input-output pairs and q is the test query. Instead of concatenating all examples into the input, task-vector-based methods aim to represent these demonstrations more compactly. Specifically, the task vector is defined as a tuple (μ_j, λ_j), where μ_j is a set of activation vectors capturing task-level semantics. λ_j is a set of attention head locations (indexed by layer and head) where such vectors are to be inserted during inference.

Given a model F, let $\lambda = \{(l, h)\}$ denotes the full set of attention heads, where l is the layer index and h is the head index. For each head (l, h), the output before linear projection is denoted as $A^{(l,h)} \in \mathbb{R}^d$, which we refer to as the *activation*. Previous works directly insert μ_j into λ_j to modulate model behavior during inference.

However, we observe that the effective locations λ_j for inserting task vectors vary significantly across *tasks* and *models*. By computing the *activation delta* (i.e., the difference between a query's activation with and without demonstrations), we find that locations with large deltas are consistently distributed within the same model (e.g., Qwen-VL

(Bai et al. 2023)) and task (e.g., OK-VQA (Marino et al. 2019)), as shown in Fig. 2, indicating high sensitivity to contextual information and revealing structural patterns in how in-context is processed. On the other hand, these sensitive locations vary notably across models (e.g., Qwen-VL vs. idefics-2 (Laurençon et al. 2024a)) and tasks (e.g., VizWiz (Gurari et al. 2018) vs. OK-VQA). The above observations form the core motivation of our method, which leverages activation deltas to identify context-sensitive locations.

2.2 Activation Delta for Location Selection

Given a query q, its activation at each attention head (l, h) is denoted as $A_q^{(l,h)} \in \mathbb{R}^{1\times d}$. A context-augmented input is constructed by concatenating S in-context examples with the query, as defined in Eq. (1), i.e., $c = I_{\text{few}}$, which produces activations $A_c^{(l,h)}$.

To estimate the sensitivity of each location, we measure the L2 activation difference between the query with and without context. Specifically, it is computed as:

$$\bar{\Delta}^{(l,h)} = \frac{1}{T}\sum_{t=1}^{T}\left\| A_{c_t}^{(l,h)} - A_{q_t}^{(l,h)} \right\|_2, \tag{2}$$

where $(A_{c_t}^{(l,h)}, A_{q_t}^{(l,h)})$ are activations from the t-th sampled query-context pair. Here, averaging over T iterations ensures stable estimation (He et al. 2023).

The resulting matrix $\bar{\Delta} \in \mathbb{R}^{L\times H}$ captures the context sensitivity distribution across all attention heads. Then the top-K heads with the highest values in $\bar{\Delta}$ is selected as the sensitive locations:

$$\Lambda = \text{TopK}(\bar{\Delta}, K), \tag{3}$$

which serves as the basis for the second stage, where appropriate task vectors are determined and inserted.

2.3 Selecting Task Vectors via RL

Given the set of sensitivity-informed insertion locations $\mathcal{L} = \{(l_k, h_k)\}_{k=1}^{K}$, the next step is to determine appropriate activation values to insert at each location. A pre-clustered activation bank is constructed by collecting context-enhanced activations across multiple examples and applying offline clustering (Krishna and Murty 1999).

For each location (l_k, h_k), M cluster centers are computed, forming a discrete candidate set:

$$\texttt{ClusterBank}[(l_k, h_k)] = \{\mathbf{v}_1^{(k)}, \ldots, \mathbf{v}_M^{(k)}\}, \quad \mathbf{v}_i^{(k)} \in \mathbb{R}^d. \tag{4}$$

Selecting task vectors from this bank is framed as a discrete optimization problem. For each location, a categorical distribution is defined over the M candidates, parameterized by learnable logits $\boldsymbol{\alpha}^{(k)} \in \mathbb{R}^M$:

$$\mathbf{p}^{(k)} = \text{softmax}(\boldsymbol{\alpha}^{(k)}). \tag{5}$$

During training, cluster indices $\{i_k\}$ are sampled from the distributions $\mathbf{p}^{(k)}$, forming a full vector set $\mathcal{V}_i = \{\mathbf{v}_{i_k}^{(k)}\}_{k=1}^{K}$. Then they are inserted into the model activations at the corresponding locations Λ during inference. The resulting task loss $\mathcal{L}_{\text{task}}$ on sample (x_i, y_i) is used to compute a reward:

$$r_i = -\mathcal{L}_{\text{task}}(F(x_i; \Lambda, \mathcal{V}_i), y_i). \tag{6}$$

Policy learning is performed using the REINFORCE algorithm (Williams 1992). The policy loss is defined as:

$$\mathcal{L}_{\text{policy}} = -\sum_{i=1}^{N}\sum_{k=1}^{K} \log p_{i_k}^{(k)} \cdot \frac{r_i - \bar{r}}{\sigma_r + \epsilon}, \tag{7}$$

where $\bar{r}$ and σ_r are moving averages of reward mean and standard deviation for variance reduction.

The optimization is conducted via gradient descent on $\{\boldsymbol{\alpha}^{(k)}\}$ to encourage sampling high-reward clusters. After convergence, the final task vectors are determined by choosing the highest-probability cluster at each location:

$$\hat{\mathbf{v}}^{(k)} = \mathbf{v}_{i_k^*}^{(k)}, \quad i_k^* = \arg\max_i \alpha_i^{(k)}. \tag{8}$$

In this way, this RL-based vector selection enables STV to flexibly adapt to downstream tasks while avoiding exhaustive enumeration over the candidate space.

2.4 Inference with Task Vector Modification

Given the selected insertion locations $\mathcal{L} = \{(l_k, h_k)\}_{k=1}^{K}$ and task vectors $\hat{\mathcal{V}} = \{\hat{\mathbf{v}}^{(k)}\}_{k=1}^{K}$, inference is conducted by modifying intermediate activations at the designated locations during the forward pass.

For a test input x_{test}, the original model activations at each (l_k, h_k) are replaced with the corresponding task vector $\hat{\mathbf{v}}^{(k)}$. Let $F^\dagger$ denote the modified model under task vector intervention, the prediction is computed as:

$$\hat{y}_{\text{test}} = F^\dagger(x_{\text{test}}; \mathcal{L}, \hat{\mathcal{V}}). \tag{9}$$

This enables many-shot conditioning at the activation level without increasing the input length or introducing additional parameters. Only a single forward pass is required per test instance, allowing efficient and scalable inference under various input settings.

3 Experiments

3.1 Evaluation Setup

We evaluate our method on five vision-language benchmarks: VizWiz (Gurari et al. 2018), OK-VQA (Marino et al. 2019), DTD (Cimpoi et al. 2014), Flowers (Nilsback and Zisserman 2008), and CUB (Wah et al. 2011), covering diverse reasoning types and image domains, including real-world user-captured photos (VizWiz), knowledge-intensive visual questions (OK-VQA), and fine-grained classification tasks across textures, flowers, and bird species (DTD, Flowers, CUB). More details are reported in the Appendix.

To assess cross-model generalization, we evaluate on Qwen-VL-7B (Bai et al. 2023) (8192-token window, 256 tokens/image) and Idefics2-8B (Laurençon et al. 2024a) (same window, 64 tokens/image), both using interleaved multimodal inputs.

All evaluations are conducted using consistent in-context prompting without gradient updates. We compare our method with two categories of baselines: (1) standard In-Context Learning (ICL) methods (Brown et al. 2020), which concatenate multimodal exemplars with the query into the model input sequence; and (2) task-vector-based approaches

Category	Method	VizWiz	OK-VQA	DTD	Flowers	CUB	Avg.	Avg. Gain
				Qwen-VL-7B Models				
Standard ICL	zero-shot ICL	35.21	57.76	55.07	55.24	56.50	51.96	–
	4-shot ICL	42.00	54.62	55.50	54.67	56.16	52.59	↑0.63
Task Vector Based Methods	TV	36.71	40.06	50.93	49.78	52.89	46.07	↓5.89
	FV	40.51	52.78	53.23	52.95	53.56	50.61	↓1.35
	ICV	37.24	55.89	49.56	50.04	51.64	48.87	↓3.09
	I2CL	39.39	56.67	67.71	59.63	62.53	57.99	↑6.03
	MTV	<u>45.60</u>	<u>60.51</u>	<u>76.50</u>	<u>78.10</u>	<u>80.00</u>	<u>68.14</u>	↑16.18
	STV (Ours)	**58.30**	**61.94**	**80.45**	**81.51**	**82.33**	**72.91**	↑**20.95**
				Idefics2-8B Models				
Standard ICL	zero-shot ICL	31.30	52.40	88.73	82.80	88.70	68.79	–
	4-shot ICL	40.80	51.50	89.13	<u>84.32</u>	87.26	70.60	↑1.81
Task Vector Based Methods	TV	28.82	41.67	62.33	50.97	62.16	49.59	↓19.20
	FV	26.79	44.51	56.53	54.36	67.73	49.18	↓19.61
	ICV	25.59	38.62	52.37	48.64	60.46	45.94	↓22.85
	I2CL	29.03	47.13	73.63	62.17	74.50	57.29	↓11.50
	MTV	<u>52.50</u>	<u>53.00</u>	<u>89.14</u>	83.80	<u>89.80</u>	<u>73.65</u>	↑4.86
	STV (Ours)	**60.61**	**54.14**	**92.47**	**86.73**	**90.23**	**76.84**	↑**8.05**

Table 1: Performance comparison across five datasets, with best results in **bold** and second best underlined. **Average Gain** measures improvement over zero-shot ICL. The first block reports ICL baselines, the second block task-vector methods, and the final row shows our STV with the highest overall performance and gains.

Methods	MTV	Ours	Change
Location Search Time (s)	6000	88	↓ 98.53%
GPU Memory (GB)	19.8	19.8	-
Inference Time (s)	0.49	0.49	-
Performance	45.6	**58.3**	↑ 12.7

Table 2: Comparison between MTV and our method on task vector location search efficiency, resource consumption, and performance on VizWiz dataset.

that compress demonstrations into activation space for efficient adaptation, including TV (Hendel, Geva, and Globerson 2023), FV (Todd et al. 2023), ICV (Liu et al. 2023), I2CL (Li et al. 2024c), and MTV (Huang et al. 2024).

3.2 Implementation Details

All experiments are implemented in PyTorch using official model checkpoints for Qwen-VL-7B and Idefic-2-8B. All evaluations are performed on a single NVIDIA H20 GPU unless otherwise specified. For each dataset, we use 100 in-context examples under a 4-shot setting. During inference, we apply zero-shot prompting to isolate the effect of task vector insertion. Unless noted, the maximum generation length is capped at 20 tokens. To identify sensitive locations, we precompute activation deltas and select 64 attention head locations. For each location, we construct a task vector bank by clustering the corresponding activations from many-shot forward passes using k-means with 16 clusters.

3.3 Main Results

Table 1 summarizes the performance of our STV compared to a range of strong baselines across 5 benchmarks and 2 representative LMM families. Several key observations emerge:

Performance across VQA Datasets. On VizWiz, STV achieves 58.3% accuracy with Qwen-VL-7B and 60.6% with Idefics2-8B, surpassing the strongest baseline MTV by +12.7% and +8.1%, respectively. On OK-VQA, STV obtains 61.9% (Qwen-VL-7B) and 54.1% (Idefics2-8B), exceeding 4-shot ICL by +7.3% and +2.6%. On fine-grained classification datasets, STV yields 80.5% on DTD, 81.5% on Flowers, and 82.3% on CUB with Qwen-VL-7B, consistently improving over 4-shot ICL by +25.0%, +26.8%, and +26.2%. These results indicate that STV significantly enhances robustness on noise-rich real-world datasets such as VizWiz, while also excelling on fine-grained recognition benchmarks by leveraging activation sensitivity to capture discriminative cues.

Generalization across Diverse Model Architectures. With Qwen-VL-7B, STV improves the average accuracy from 68.1% (MTV) to 72.9% (+4.8%). With Idefics2-8B, STV improves the average accuracy from 73.7% (MTV) to 76.8% (+3.1%). Unlike ICV and FV, whose performance fluctuates depending on the underlying architecture, STV maintains leading accuracy across distinct backbone designs (i.e., Qwen-VL-7B and Idefics2-8B). This suggests that STV's

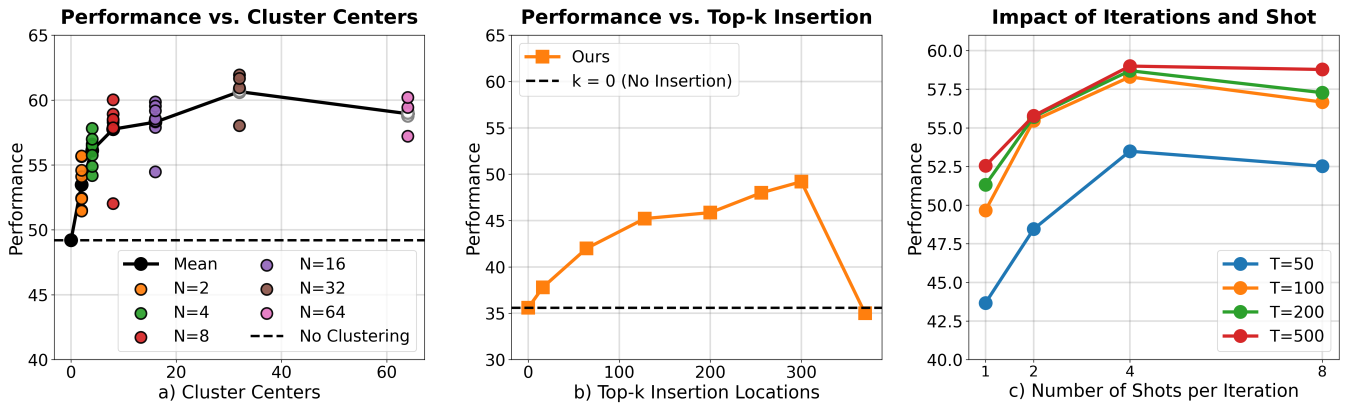

Figure 4: a) Effect of Cluster Granularity on Task Vector Selection using Qwen-VL on VizWiz Dataset. b) Impact of Top-k Location Selection on Model Performance using Qwen-VL on VizWiz Dataset. c) Impact of Iterations T and the Number Shot per Iteration on Model Performance using Qwen-VL on VizWiz Dataset.

sensitive-aware location and RL-based task vector selection mechanism captures transferable structural cues, ensuring stable adaptation across heterogeneous model families.

Efficiency and Comparative Advantage over Baselines. Compared with MTV, which requires exhaustive location search, STV achieves comparable or better performance while reducing computational overhead to only 1.5% of MTV's search cost (Table 2). On Qwen-VL-7B, STV improves the average accuracy by +3.2%, and on Idefics2-8B by +2.4%, while also providing more stable gains across datasets. These results indicate that STV improves both performance and efficiency compared to existing task vector methods, offering a balanced and practical solution for multimodal in-context adaptation.

3.4 Ablation Study

Effect of Task Vector Construction and Insertion Locations. We study STV's two key components: *where and what to insert*, on VizWiz with Qwen-VL-7B. As shown in Fig. 4(b), performance rises from 35.2% to 49.2% as more sensitive locations are used, confirming the effectiveness of our sensitivity-aware selection; yet overly large k (> 300) sharply degrades accuracy, showing that untargeted intervention disrupts model reasoning. Fig. 4(a) further fixes $k = 300$ and varies the number of cluster centers N: performance improves steadily with larger N and saturates around 32, indicating that clustering yields richer semantic task vectors and mitigates information loss. Together, location selection contributes +14.0% and task vector construction adds +9.1%, validating both components as essential to STV.

Scaling In-Context Samples. We further examine the scalability of STV with respect to the number of iterations T and the number of shots per iteration. In Fig. 4(c), increasing either T or the number of shots consistently improves performance, demonstrating two key points: (i) STV can effectively process a large number of in-context examples, highlighting the advantage of task vector-based methods over direct concatenation; (ii) larger context provides richer task information, from which STV is able to extract useful cues. However, when T or the shot size grows excessively, accuracy begins to drop, suggesting that redundant information introduces noise and interferes with reasoning. These trends confirm that STV achieves robust adaptation under larger contexts while benefiting from principled sample scaling.

Generalization Comparison with Parameter-Tuning Adaptation Methods. We compare STV with two common adaptation paradigms: LoRA-based parameter-efficient tuning and full supervised fine-tuning (SFT). As shown in Table 3, STV consistently outperforms LoRA on both VizWiz and OK-VQA, while requiring no parameter updates or task-specific optimization. Although SFT achieves strong performance on VizWiz (+26.8), it suffers a severe drop on OK-VQA (-32.7), revealing overfitting and poor transferability. This highlights a key limitation of parameter tuning: it yields task-specific gains at the cost of generalization. In contrast, STV achieves robust improvements across domains (+23.1 on VizWiz, +4.1 on OK-VQA), demonstrating the effectiveness of activation-level adaptation.

Efficiency and Effectiveness Compared with In-Context Learning. To validate our method, STV is compared with few-shot ICL baselines on VizWiz using Qwen-VL-7B. As shown in Figure 5, STV achieves superior accuracy with negligible FLOPs and runtime overhead—remaining on par with zero-shot inference. In contrast, scaling ICL from 4-shot to 32-shot increases FLOPs up to 25$\times$ and runtime up to 8$\times$, highlighting the efficiency advantage of our insertion strategy under long-context constraints. Table 4 further benchmarks performance under varying shot counts. STV reaches 58.3% accuracy, outperforming 32-shot ICL by +8.5% while incurring no memory or token overhead. Moreover, 64-shot ICL fails due to out-of-memory (OOM), underscoring the limitation of naïve scaling. Overall, STV conveys task information more effectively than token-level prompting, with better efficiency and scalability.

Effect of Exemplar Quality and Robustness. Table 5 shows that high-quality exemplars selected by Facility Location (Schreiber, Bilmes, and Noble 2020) further improve STV (+3.6%), demonstrating the benefit of exemplar quality

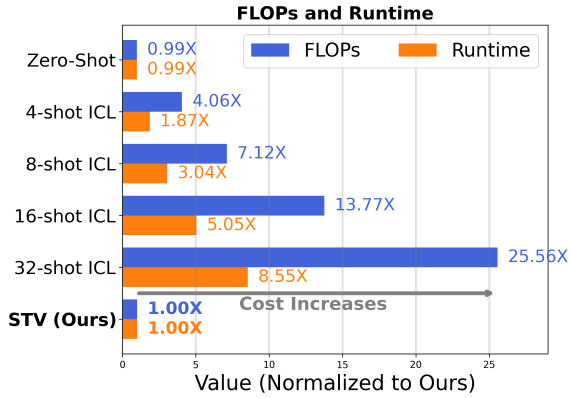

Figure 5: FLOPs and Runtime Comparison between STV and Few-Shot ICL.

Model	VizWiz	OK-VQA
Qwen-VL-7B	35.2	57.8
+ **SFT**	62.0 (+26.8)	25.1 (−32.7)
+ LoRA	44.3 (+9.1)	57.7 (−0.1)
STV (Ours)	**58.3** (+23.1)	**61.9** (+4.1)

Table 3: Accuracy on VizWiz and OK-VQA using Qwen-VL with different adaptation strategies. The values in parentheses indicate the change compared to the base model.

in activation-space augmentation. Moreover, when inserting noisy cross-domain exemplars, STV exhibits smaller degradation (−0.7) than ICL (−1.0), indicating stronger robustness and stability.

4 Related Work

Scaling In-Context Learning. Early work (Brown et al. 2020; Bertsch et al. 2024; Li et al. 2023, 2024b; Chowdhery et al. 2023) show that more in-context examples improve language model performance, but gains were constrained by short context windows (e.g., 2048 in GPT-3) and typically fewer than 100 demonstrations. Recent studies scale ICL to 1,000+ demonstrations (Li et al. 2023; Agarwal et al. 2024; Chen et al. 2023; Peng et al. 2023), achieving steady improvements across NLP tasks, though restricted to text-only settings without cross-model generalization (Ma et al. 2024).

Multimodal In-Context Learning. Research on multimodal ICL remains nascent. Closed-source LMMs (e.g., GPT-4V, Gemini) benefit from demonstrations in diverse vision-language tasks (Zhang et al. 2024; Han et al. 2023), and complex prompts improve relational reasoning (Zhao et al. 2023). A recent benchmark (Jiang et al. 2024) evaluates GPT-4o (Hurst et al. 2024) and Gemini 1.5 Pro (Team et al. 2024) with up to 2,000 multimodal shots, showing substantial gains across classification, VQA, and localization. However, these rely on full-sequence prompting with high inference cost, leaving efficiency questions open (Ma et al. 2025b; Chen et al. 2025; Jiang et al. 2025; Li et al. 2025).

Method	Num.	OOM	Acc. (%)	Change
Standard ICL	0	×	35.2	–
	4	×	42.0	+6.8
	8	×	44.3	+9.1
	16	×	46.9	+11.7
	32	×	49.8	+14.6
	64	✓	–	–
STV (Ours)	400	×	**58.3**	**+23.1**

Table 4: Comparison of accuracy across different few-shot ICL settings and our proposed method (STV). Here "OOM" is the abbreviation for "out of memory".

Model	VizWiz Dataset
(a) STV with High-Quality Shots	
Qwen-VL-7B	35.2
+ STV	58.3 (+23.1)
+ STV + F.L. Shots	61.9 (+26.7)
(b) Stability of ICL vs. STV	
4-shot ICL	41.0 (-1.0)
STV	57.6 (-0.7)

Table 5: (a) STV with high-quality shots and (b) Stability of ICL vs STV using Qwen-VL on VizWiz.

Task-Vector-based Methods. Approaches based on task vectors (Hendel, Geva, and Globerson 2023; Liu et al. 2023; Todd et al. 2023; Huang et al. 2024; Hojel et al. 2024; Yang et al. 2025) insert compact representations of demonstrations into activations, avoiding full-sequence prompts. Value Estimation methods (Hendel, Geva, and Globerson 2023; Liu et al. 2023; Todd et al. 2023) compute vectors (e.g., via PCA) and insert them at fixed locations, reducing cost but generalizing poorly to multimodal tasks. Location Selection methods (Huang et al. 2024) instead fix the vector and find insertion sites via policy networks, improving efficiency but suffering from task information loss and unstable selection.

5 Conclusion

This paper presents STV, a simple yet effective framework for enabling many-shot multimodal in-context learning without increasing input length or altering model weights. By locating context-sensitive attention heads and selecting task vectors via reinforcement learning, STV introduces task-specific knowledge through activation-level modulation. Extensive experiments across five benchmarks and two LMM families confirm consistent gains over previous ICL and task vector methods, achieving strong generalization and inference-time efficiency. These results underscore the value of sensitivity-aware activation control as a scalable adaptation strategy for large multimodal models.

References

Abdi, H.; and Williams, L. J. 2010. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4): 433–459.

Agarwal, R.; Singh, A.; Zhang, L.; Bohnet, B.; Rosias, L.; Chan, S.; Zhang, B.; Anand, A.; Abbas, Z.; Nova, A.; et al. 2024. Many-shot in-context learning. *Adv. Neural Inform. Process. Syst.*, 37: 76930–76966.

Bai, J.; Bai, S.; Yang, S.; Wang, S.; Tan, S.; Wang, P.; Lin, J.; Zhou, C.; and Zhou, J. 2023. Qwen-vl: A frontier large vision-language model with versatile abilities. *arXiv preprint arXiv:2308.12966*.

Bai, S.; Chen, K.; Liu, X.; Wang, J.; Ge, W.; Song, S.; Dang, K.; Wang, P.; Wang, S.; Tang, J.; et al. 2025. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*.

Bertsch, A.; Ivgi, M.; Xiao, E.; Alon, U.; Berant, J.; Gormley, M. R.; and Neubig, G. 2024. In-context learning with long-context models: An in-depth exploration. *arXiv preprint arXiv:2405.00200*.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *Adv. Neural Inform. Process. Syst.*, 33: 1877–1901.

Chen, S.; Wong, S.; Chen, L.; and Tian, Y. 2023. Extending context window of large language models via positional interpolation. *arXiv preprint arXiv:2306.15595*.

Chen, Z.; Wang, S.; Tan, Z.; Li, J.; and Shen, C. 2025. MAPLE: Many-Shot Adaptive Pseudo-Labeling for In-Context Learning. *arXiv preprint arXiv:2505.16225*.

Chowdhery, A.; Narang, S.; Devlin, J.; Bosma, M.; Mishra, G.; Roberts, A.; Barham, P.; Chung, H. W.; Sutton, C.; Gehrmann, S.; et al. 2023. Palm: Scaling language modeling with pathways. *J. Mach. Learn. Res.*, 24(240): 1–113.

Cimpoi, M.; Maji, S.; Kokkinos, I.; Mohamed, S.; ; and Vedaldi, A. 2014. Describing Textures in the Wild. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*

Comanici, G.; Bieber, E.; Schaekermann, M.; Pasupat, I.; Sachdeva, N.; Dhillon, I.; Blistein, M.; Ram, O.; Zhang, D.; Rosen, E.; et al. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

Gurari, D.; Li, Q.; Stangl, A. J.; Guo, A.; Lin, C.; Grauman, K.; Luo, J.; and Bigham, J. P. 2018. Vizwiz grand challenge: Answering visual questions from blind people. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, 3608–3617.

Han, Z.; Zhou, G.; He, R.; Wang, J.; Wu, T.; Yin, Y.; Khan, S.; Yao, L.; Liu, T.; and Zhang, K. 2023. How well does gpt-4v (ision) adapt to distribution shifts? a preliminary investigation. *arXiv preprint arXiv:2312.07424*.

He, H.; Cai, J.; Zhang, J.; Tao, D.; and Zhuang, B. 2023. Sensitivity-aware visual parameter-efficient fine-tuning. In *Int. Conf. Comput. Vis.*, 11825–11835.

Hendel, R.; Geva, M.; and Globerson, A. 2023. In-context learning creates task vectors. *arXiv preprint arXiv:2310.15916*.

Hojel, A.; Bai, Y.; Darrell, T.; Globerson, A.; and Bar, A. 2024. Finding visual task vectors. In *Eur. Conf. Comput. Vis.*, 257–273. Springer.

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; and Chen, W. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *Int. Conf. Learn. Represent.*

Huang, B.; Mitra, C.; Karlinsky, L.; Arbelle, A.; Darrell, T.; and Herzig, R. 2024. Multimodal task vectors enable many-shot multimodal in-context learning. *Adv. Neural Inform. Process. Syst.*, 37: 22124–22153.

Hurst, A.; Lerer, A.; Goucher, A. P.; Perelman, A.; Ramesh, A.; Clark, A.; Ostrow, A.; Welihinda, A.; Hayes, A.; Radford, A.; et al. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Jiang, Y.; Fu, J.; Hao, C.; Hu, X.; Peng, Y.; Geng, X.; and Yang, X. 2025. Mimic In-Context Learning for Multimodal Tasks. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, 29825–29835.

Jiang, Y.; Irvin, J.; Wang, J. H.; Chaudhry, M. A.; Chen, J. H.; and Ng, A. Y. 2024. Many-shot in-context learning in multimodal foundation models. *arXiv preprint arXiv:2405.09798*.

Krishna, K.; and Murty, M. N. 1999. Genetic K-means algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 29(3): 433–439.

Laurençon, H.; Marafioti, A.; Sanh, V.; and Tronchon, L. 2024a. Building and better understanding vision-language models: insights and future directions. *arXiv preprint arXiv:2408.12637*.

Laurençon, H.; Tronchon, L.; Cord, M.; and Sanh, V. 2024b. What matters when building vision-language models? *Adv. Neural Inform. Process. Syst.*, 37: 87874–87907.

Li, F.; Zhang, R.; Zhang, H.; Zhang, Y.; Li, B.; Li, W.; Ma, Z.; and Li, C. 2024a. Llava-next-interleave: Tackling multi-image, video, and 3d in large multimodal models. *arXiv preprint arXiv:2407.07895*.

Li, M.; Gong, S.; Feng, J.; Xu, Y.; Zhang, J.; Wu, Z.; and Kong, L. 2023. In-context learning with many demonstration examples. *arXiv preprint arXiv:2302.04931*.

Li, T.; Zhang, G.; Do, Q. D.; Yue, X.; and Chen, W. 2024b. Long-context llms struggle with long in-context learning. *arXiv preprint arXiv:2404.02060*.

Li, Z.; Xu, Z.; Han, L.; Gao, Y.; Wen, S.; Liu, D.; Wang, H.; and Metaxas, D. N. 2024c. Implicit in-context learning. *arXiv preprint arXiv:2405.14660*.

Li, Z.; Xu, Z.; Han, L.; Gao, Y.; Wen, S.; Liu, D.; Wang, H.; and Metaxas, D. N. 2025. Implicit In-context Learning. In *Int. Conf. Learn. Represent.*

Liu, H.; Li, C.; Li, Y.; and Lee, Y. J. 2024. Improved baselines with visual instruction tuning. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, 26296–26306.

Liu, S.; Ye, H.; Xing, L.; and Zou, J. 2023. In-context vectors: Making in context learning more effective and controllable through latent space steering. *arXiv preprint arXiv:2311.06668*.

Ma, Z.; Gou, C.; Shi, H.; Sun, B.; Li, S.; Rezatofighi, H.; and Cai, J. 2025a. Drvideo: Document retrieval based long video understanding. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, 18936–18946.

Ma, Z.; Li, S.; Sun, B.; Cai, J.; Long, Z.; and Ma, F. 2024. GeReA: Question-Aware Prompt Captions for Knowledge-based Visual Question Answering. *arXiv preprint arXiv:2402.02503*.

Ma, Z.; Zhang, S.; Wei, L.; and Tian, Q. 2025b. Efficient Multi-modal Long Context Learning for Training-free Adaptation. *arXiv preprint arXiv:2505.19812*.

Marino, K.; Rastegari, M.; Farhadi, A.; and Mottaghi, R. 2019. Ok-vqa: A visual question answering benchmark requiring external knowledge. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, 3195–3204.

Nilsback, M.-E.; and Zisserman, A. 2008. Automated flower classification over a large number of classes. In *Sixth Indian conference on computer vision, graphics & image processing*, 722–729. IEEE.

OpenAI, R. 2023. Gpt-4 technical report. arxiv 2303.08774. Accessed: 2025-07-20. *View in Article*, 2(5): 1.

Peng, B.; Quesnelle, J.; Fan, H.; and Shippole, E. 2023. Yarn: Efficient context window extension of large language models. *arXiv preprint arXiv:2309.00071*.

Peng, Y.; Hao, C.; Hu, X.; Peng, J.; Geng, X.; and Yang, X. 2024. LIVE: Learnable In-Context Vector for Visual Question Answering. In *Adv. Neural Inform. Process. Syst.*, volume 37, 9773–9800.

Schreiber, J.; Bilmes, J.; and Noble, W. S. 2020. apricot: Submodular selection for data summarization in Python. *J. Mach. Learn. Res.*, 21(161): 1–6.

Team, G.; Georgiev, P.; Lei, V. I.; Burnell, R.; Bai, L.; Gulati, A.; Tanzer, G.; Vincent, D.; Pan, Z.; Wang, S.; et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

Todd, E.; Li, M. L.; Sharma, A. S.; Mueller, A.; Wallace, B. C.; and Bau, D. 2023. Function vectors in large language models. *arXiv preprint arXiv:2310.15213*.

Wah, C.; Branson, S.; Welinder, P.; Perona, P.; and Belongie, S. 2011. The caltech-ucsd birds-200-2011 dataset. *Dataset report*.

Wang, P.; Bai, S.; Tan, S.; Wang, S.; Fan, Z.; Bai, J.; Chen, K.; Liu, X.; Wang, J.; Ge, W.; et al. 2024. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3): 229–256.

Yang, L.; Lin, Z.; Lee, K.; Papailiopoulos, D.; and Nowak, R. 2025. Task vectors in in-context learning: Emergence, formation, and benefit. *arXiv preprint arXiv:2501.09240*.

Zhang, X.; Li, J.; Chu, W.; Hai, J.; Xu, R.; Yang, Y.; Guan, S.; Xu, J.; and Cui, P. 2024. On the out-of-distribution generalization of multimodal large language models. *arXiv preprint arXiv:2402.06599*.

Zhao, H.; Cai, Z.; Si, S.; Ma, X.; An, K.; Chen, L.; Liu, Z.; Wang, S.; Han, W.; and Chang, B. 2023. Mmicl: Empowering vision-language model with multi-modal in-context learning. *arXiv preprint arXiv:2309.07915*.