Revisiting Task Vectors: A Structural Analysis of Their Limits

Leekyeung Han[1] **Hyeseung Ji**[1] **Sangwon Jung**[2]
Seungwoo Lyu[1] **Paul Hongsuck Seo**[1]
[1]Korea University [2]Trillion Labs
{happilee12, hscient, dbtmddn41, phseo}@korea.ac.kr
sangwon.jung@trillionlabs.co

Abstract

To mitigate high inference costs for in-context learning, prior works (Hendel et al., 2023; Todd et al., 2024) have proposed activation-space task vectors that encapsulate demonstration contexts into static representations. However, performance occasionally degrades across tasks, leaving it unclear whether this stems from sub-optimal extraction heuristics or fundamental limitations. We resolve this by decomposing Transformer self-attention to establish a theoretical best-case approximation: the Minimal-Gap Approximation. By isolating structural constraints from heuristic flaws, the Minimal-Gap Approximation reveals that task vectors are fundamentally bottlenecked by attention and length gaps. Specifically, because task vectors lock contextual information to fixed attention maps and positional alignments, they structurally fail under dynamic routing, length mismatches, and multi-token generation. We further show that these findings consistently hold across training-free extract-and-inject task vector methods, model architectures, and model scales.

1 Introduction

To mitigate the quadratic inference cost of processing extensive contexts such as in-context demonstrations in LLMs, a growing body of literature has proposed encapsulating these contexts into static, $\mathcal{O}(d)$-dimensional vectors. Such activation-space representations that encapsulate contextual information (Hendel et al., 2023; Todd et al., 2024; Li et al., 2024a) are referred to as *task vectors*, extending the notion originally introduced in weight space (Ilharco et al., 2022). Operating via an extract-and-inject framework, these methods precompute a single-vector representation from a reference request with the full context and inject it into the hidden states of a zero-shot model. The operational goal is highly appealing: bypass the computational overhead of attending to context tokens while preserving the macroscopic behavioral performance of full in-context learning.

Despite their promise, task vectors exhibit severe performance degradation even on seemingly simple classification tasks (Zheng et al., 2024). It remains highly debated *why* these vectors fail (Todd et al., 2024; Yin and Steinhardt, 2025; Tikhonov et al., 2025; Dong et al., 2025). However, because existing methods rely on heuristic extraction strategies—such as searching for optimal injection layers—it is entirely unclear whether their performance bottlenecks are merely artifacts of sub-optimal design choices, or if they represent fundamental, insurmountable constraints of using a static representation to bypass dynamic attention recomputation.

In this work, we resolve this ambiguity by shifting the evaluation of task vectors from a macroscopic behavioral lens to a microscopic representational one. We present a unified mathematical framework that decomposes the exact self-attention computation to establish a *Full Context Baseline*. Concretely, we derive absolute minimum requirements for exact contextual reconstruction without recomputing attention over context tokens. This leads to a theoretical best-case approximation, which we term the **Minimal-Gap Approximation**, isolating the intrinsic limitations of the training-free extract-and-inject paradigm from the quality of heuristic vectors. With this theoretical best-case approximation, we revisit existing challenges with which task vectors struggle. Specifically, we use it to identify which aspects of full-context self-attention are lost in such static approximations, and when those losses translate into performance degradation.

The results show that task vector performance is not primarily constrained by task complexity (Tikhonov et al., 2025; Dong et al., 2025), but rather by structural *Attention and Length Gaps*.

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 34196–34212
October 24-29, 2026 ©2026 Association for Computational Linguistics

Because task vectors mathematically lock context representations to a fixed attention map and positional alignment, they inherently fail on tasks that require dynamic routing or variable sequence lengths. Our empirical analysis validates these bottlenecks, demonstrating that static approximations, including the Minimal-Gap construction, systematically degrade in settings requiring dynamic routing or variable sequence lengths.

In summary, our main contributions are:

- **Revisiting Task Vectors as Approximations of Context:** We show that static vector injection is an inherently lossy approximation of full-context computation, and induces cascading errors.
- **Minimal-Gap Approximation:** We establish a theoretical best-case approximation to isolate the fundamental constraints of task vectors from the flaws of heuristic extraction methods.
- **Re-evaluating Task Vector Behaviors:** We demonstrate that underlying attention gaps explain when task vectors succeed or fail.

2 Related Work

Model Adaptation Methods for adapting or steering large language models have developed along several complementary directions. Traditional approaches modify model parameters through fine-tuning, parameter-efficient tuning, or weight interpolation (Hu et al., 2021; Li and Liang, 2021; Wortsman et al., 2022), while more recent activation steering methods intervene on internal representations at inference time without additional training (Panickssery et al., 2024; Li et al., 2023; Park et al., 2025; Zou et al., 2023). In-context learning (ICL) provides another form of adaptation, as models can adjust to new tasks from demonstrations in the prompt without explicit parameter updates (Brown et al., 2020; Xie et al., 2021; Chan et al., 2022; Akyürek et al., 2023). Complementary to this, a line of work explores compressing or replacing the context through learned or distilled representations, enabling more efficient conditioning without explicitly attending to all input tokens (Li and Liang, 2021; Mu et al., 2023; An et al., 2025). Within this broader context, the notion of a *task vector* first emerged in weight space (Ilharco et al., 2022) and was later extended to activation space, where hidden states formed during ICL are interpreted as reusable task-relevant representations.

Task Vectors in In-Context Learning Recent work has proposed extracting intermediate hidden states or attention-derived representations from ICL and injecting them during zero-shot inference to recover few-shot behavior (Hendel et al., 2023; Todd et al., 2024; Liu et al., 2023). Subsequent work further improves extraction and injection through optimized layers, scaling coefficients, test-time adaptation, or additional training objectives that encourage more robust and query-independent task vectors (Li et al., 2024a; Sağlam et al., 2025; Yang et al., 2025c; Liu and Deng, 2025; Li et al., 2025a; Zheng et al., 2024; Ardoin et al., 2026), and has also been extended to multimodal settings (Huang et al., 2024; Luo et al., 2024; Hojel et al., 2024). A line of work has explored training-based task vectors (Li et al., 2024b; Yang et al., 2025b), which require additional training data, making them conceptually closer to PEFT methods. Our work focuses on the training-free extract-and-inject paradigm and proposes a theoretical tool to revisit its fundamental limitations.

Analysis of Task Vectors A growing body of work studies how task representations emerge and when they are effective (Yin and Steinhardt, 2025; Tikhonov et al., 2025; Zheng et al., 2024; Han et al., 2024; Laiyk et al., 2026). At the same time, some works highlighted clear limitations of fixed task vectors. In particular, for compositional tasks, multi-step generation, or settings where task-relevant representations are temporally or semantically distributed, a single fixed vector often fails to reproduce the full effect of ICL (Li et al., 2025b; Dong et al., 2025; Yang et al., 2025d). Our work is closely related to this line of analysis, but instead of simply cataloging success and failure cases, we frame the problem as one of hidden-state decomposition and approximation in ICL, which makes the limits of fixed task vectors more explicit. Concurrently, Kwon et al. (2026) evaluate task vectors by their next-token distribution discrepancy from ICL. We instead decompose the attention computation that produces those logits.

3 Task Vectors as Static, Lossy Approximations of Context

In this section, we provide a unified mathematical framework for analyzing the behavior of task vectors. We first define their primary objective and decompose the exact attention computation to establish the full context baseline (Section 3.1). We then examine how standard task vectors attempt to approximate this behavior and detail their struc-

tural limitations (Section 3.2). Finally, by deriving the minimal requirements for exact contextual reconstruction via our Minimal-Gap Approximation (Section 3.3), we expose the inherent challenges of the task vector paradigm (Section 3.4). These theoretical findings form the basis of our empirical analysis in Section 4.

3.1 Attention Decomposition in Transformers

The primary objective of task vector methods is to enable a model processing only a user request R to emulate the behavior of a fully contextualized model processing both a fixed context C—such as a set of demonstrations in few-shot prompting—and the request R. This approach aims to retain the guiding influence of the context while entirely bypassing the computational overhead of attending to the context tokens during inference.

Existing literature typically frames this goal broadly as mimicking the macroscopic output behavior of a contextualized model. However, because autoregressive language generation is deterministically driven by the final-layer logits, matching the final output requires matching the intermediate latent states that produce those logits. Therefore, any behavioral gap fundamentally stems from a representation gap. To understand exactly what these methods must approximate at the latent level, we first decompose the exact self-attention computation within a single Transformer layer when the full context is provided.

For an i-th token within the request suffix R, the causal attention range naturally splits into two segments: the fixed context tokens C and the preceding request tokens in R. We define the unnormalized attention contributions ($\mathbf{h}$) and normalization masses (z) for the context segment as:

$$\mathbf{h}_C^{(i)} := \sum_{j \in C} \exp(\mathbf{q}_i^\top \mathbf{k}_j)\mathbf{v}_j \tag{1}$$

$$z_C^{(i)} := \sum_{j \in C} \exp(\mathbf{q}_i^\top \mathbf{k}_j), \tag{2}$$

where $\mathbf{q}_i, \mathbf{k}_j$, and $\mathbf{v}_j$ are the query, key, and value vectors projected from the layer's input representations. For notational brevity, we omit the standard scaling factor $1/\sqrt{d_h}$ as well as the layer and head indices. Similarly, for the request segment, the

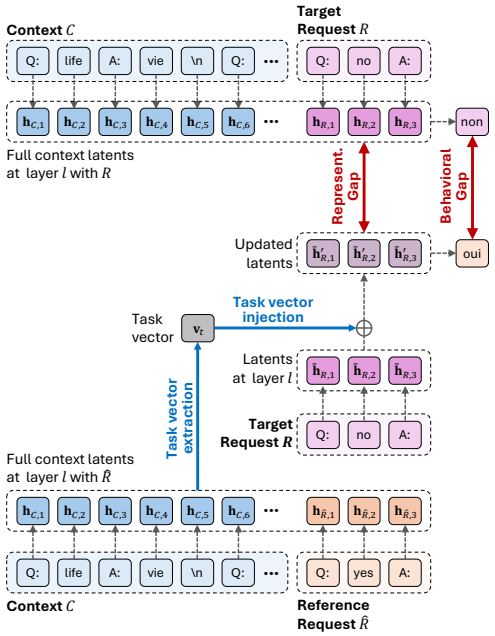


Figure 1: **Task vectors as partial contextual approximations.** Task vectors are extracted from a reference request $\tilde{R}$ with context C (bottom) and injected into a zero-shot target request R (middle) to emulate full-context latent states (top). Incomplete reproduction creates a *representation gap* that drives a downstream *behavioral gap*.

contributions are:

$$\mathbf{h}_R^{(i)} := \sum_{\substack{j \in R \\ j \le i}} \exp(\mathbf{q}_i^\top \mathbf{k}_j)\mathbf{v}_j \tag{3}$$

$$z_R^{(i)} := \sum_{\substack{j \in R \\ j \le i}} \exp(\mathbf{q}_i^\top \mathbf{k}_j). \tag{4}$$

In a Transformer, the final attention output is added back to the input representation $\mathbf{x}^{(i)}$ via a residual connection. The exact layer output $\mathbf{o}^{(i)}$, which aggregates value vectors across both segments using the full context, can thus be written as:

$$\mathbf{o}^{(i)} = \mathbf{x}^{(i)} + \frac{\mathbf{h}_C^{(i)} + \mathbf{h}_R^{(i)}}{z_C^{(i)} + z_R^{(i)}}. \tag{5}$$

3.2 Task Vectors as Partial Approximations

When processing the zero-shot request R without context C during inference, the model intrinsically computes only the zero-shot layer-input $\tilde{\mathbf{x}}^{(i)}$ and its corresponding request-side attention contributions $\tilde{\mathbf{h}}_R^{(i)}$ and $\tilde{z}_R^{(i)}$. Because reproducing the model's final contextual behavior necessitates approximating its internal states, the operational goal of any task vector is ultimately to properly approximate the exact contextual output defined in Eq. (5) from these restricted components.

To achieve this, standard task vectors typically operate via an extract-and-inject framework as shown in Figure 1. During an offline phase, a fixed representation is derived to encapsulate the transformation induced by the demonstration context. This representation is typically precomputed using a specific *reference request*, with the goal of capturing the "shift" in the model's internal state—from a base zero-shot model to one informed by the task context. While the exact derivation varies, the resulting vector $\mathbf{v}_t$ is intended to serve as a heuristic proxy for the contextual information. In the subsequent injection phase, this pre-extracted vector is applied to the hidden states of new, unseen *target requests*. However, this framework relies on a rigid invariance assumption: a single vector derived from a reference request is expected to serve as a universal, static proxy for dynamic task-specific computation.

Examining Eq. (5) reveals the inherent difficulty of this approximation due to three causally linked sources of deviation. The root cause is the complete absence of the exact context-side terms ($\mathbf{h}_C^{(i)}$ and $z_C^{(i)}$) during online inference. At the lowest layers of the Transformer, substituting these dynamic terms with a static task vector immediately alters the layer outputs. As these errors propagate upward, they create a secondary deviation: the input representation to subsequent layers, $\tilde{\mathbf{x}}^{(i)}$, diverges from the exact contextual input $\mathbf{x}^{(i)}$. Consequently, this altered input yields shifted queries, keys, and values ($\tilde{\mathbf{q}}_i, \tilde{\mathbf{k}}_j$, and $\tilde{\mathbf{v}}_j$), which in turn causes the attention contributions computed purely over the request tokens to deviate from their exact counterparts ($\tilde{\mathbf{h}}_R^{(i)} \neq \mathbf{h}_R^{(i)}$ and $\tilde{z}_R^{(i)} \neq z_R^{(i)}$).

Because prior task vector methods rely on partial interventions —targeting isolated terms or applying only to specific layers and final token positions—they have inherent limitations, failing to perfectly reconstruct the full context baseline in Eq. (5) even in the idealized scenario where the target request exactly matches the reference request. Consequently, it remains unclear whether the performance bottlenecks of standard task vectors are merely artifacts of these incomplete approximations, or if they represent deeper, fundamental constraints of bypassing attention recomputation itself.

3.3 Minimal Requirements for Exact Reconstruction

To resolve this ambiguity and systematically expose the hidden challenges inherent to bypassing attention recomputation, we must establish a theoretical best-case approximation. Therefore, we derive what is minimally needed to perfectly reconstruct the output latents $\mathbf{o}^{(i)}$ without requiring any recomputation of attention over the context tokens C, given a specific reference request. By analyzing this absolute best-case scenario, we can look past the heuristic simplifications of standard task vectors and identify the insurmountable limitations inherent to the paradigm itself.

For an exact reconstruction of $\mathbf{o}^{(i)}$ without context attention, an approximation method must neutralize the root source of zero-shot deviation. Because the initial word embeddings of the request tokens are identical to the full context baseline setup, the input to the first layer is naturally exact. To prevent deviations from originating and cascading upward, the method strictly requires extracting and caching two precise token-level statistics from the context: the unnormalized attention contribution $\mathbf{h}_C^{(i)}$ and the normalization mass $z_C^{(i)}$ written in Eq. (1) and (2), respectively.

In a purely zero-shot setting where the model processes only the request R, the attention output is simply $\mathbf{o}_R^{(i)} = \mathbf{h}_R^{(i)} / z_R^{(i)}$. This allows us to re-express the exact output as a rescaled mixture:

$$\mathbf{o}^{(i)} = \mathbf{x}^{(i)} + \underbrace{\frac{1}{z_C^{(i)} + z_R^{(i)}} \mathbf{h}_C^{(i)}}_{\text{Scaled context contribution}} + \underbrace{\left(\frac{z_R^{(i)}}{z_C^{(i)} + z_R^{(i)}} \right) \mathbf{o}_R^{(i)}}_{\text{Scaled zero-shot request contribution}} . \quad (6)$$

This formulation explicitly reveals that perfect reconstruction is achievable without recomputing context attention by simply storing the context-side statistics, $\mathbf{h}_C^{(i)}$ and $z_C^{(i)}$, and injecting them during the zero-shot forward pass. We refer to this theoretically optimal caching strategy as the *Minimal-Gap Approximation*. However, while this approach successfully bypasses context attention and matches the exact output, its reliance on static components introduces profound structural bottlenecks. More specifically, since $\mathbf{h}_C^{(i)}$ and $z_C^{(i)}$ are formed by attention weights whose queries originate from the request tokens, injecting them into a forward pass

driven by a different target request does not constitute a clean transfer.

3.4 Inherent Challenges of the Task Vector

The exact reconstruction in Eq. (6) establishes a clear best-case approximation, but it also delineates the precise, insurmountable limitations of applying task vectors to autoregressive generation.

Primarily, the static nature of the pre-computed statistics mathematically locks the context representation. As shown in Eq. (6), the final output relies on dynamically scaling the context contribution by the combined mass $z_C^{(i)} + z_R^{(i)}$. If a target request requires shifting attention to different semantic parts of the context C, the true context mass $z_C^{(i)}$ and vector $\mathbf{h}_C^{(i)}$ must change accordingly. Because task vectors freeze these variables based on a reference query, they cannot adapt to new attention maps, leading to inevitable performance degradation.

Furthermore, to perfectly halt the cascading deviations, the variables $\mathbf{h}_C^{(i)}$ and $z_C^{(i)}$ must be cached across every layer not only for every token position i in the request, but also for every subsequently generated output token. Because real-world generation involves highly variable output lengths, this requirement fundamentally scales the memory footprint with both the input sequence and the generation length, undermining the core motivation of compressing a prompt into a single $\mathcal{O}(d)$ vector.

Finally, this token-wise injection assumes a strict positional alignment. Since target requests and their corresponding outputs vary wildly in length and semantics, the position-locked injection across arbitrary user queries is highly impractical.

To succeed in this task vector direction and achieve scalable, efficient context compression, future approaches must fundamentally overcome these challenges by moving beyond static, position-locked approximations. In the following section, we present an empirical analysis designed to isolate and quantify these exact degradation factors.

4 Empirical Analysis

We evaluate the representational and behavioral discrepancies of task vectors across diverse tasks. Using the Minimal-Gap Approximation, which isolates the error attributable to structural bottlenecks, we evaluate task vectors on tasks with varying attention gaps and input lengths, revealing the intrinsic limitations of task vectors.

4.1 Experimental Setup

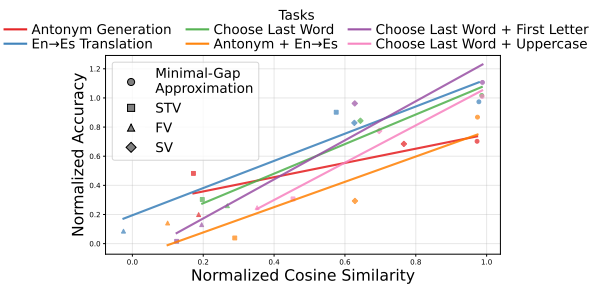
Our evaluation suite comprises a set of tasks from prior work, including atomic tasks and composite tasks (Tikhonov et al., 2025), as well as classification (Zheng et al., 2024) and bidirectional translation (Dong et al., 2025), which have been identified as challenging settings for task vector methods. We additionally evaluate variable length settings to validate the structural hypotheses developed in Section 3.4. The full list of tasks is provided in Appendix B.

Our analysis compares our analytic framework, the **Minimal-Gap Approximation (Min-Gap)**, against the exact **Full Context Baseline**. We further extend these observations to existing training-free task vector methods, including **Standard Task Vectors (STV)** (Hendel et al., 2023), **Function Vectors (FV)** (Todd et al., 2024), and **State Vectors (SV)** (Li et al., 2024a). Note that we report Min-Gap as an analytical upper bound rather than as a competing task vector method: it caches and injects context statistics at every layer and token position, and thus requires more computation than existing task vector methods, which intervene in only a single layer or a few heads. All extractions and injections are performed dynamically during the forward pass without any parameter updates, using a cache extracted once from a single dummy query and reused across all task queries. Training-based task vectors (Yang et al., 2025b; Li et al., 2024b) are based on different underlying mechanisms and thus fall outside the scope of our framework. We provide a separate discussion of these methods in Appendix H.

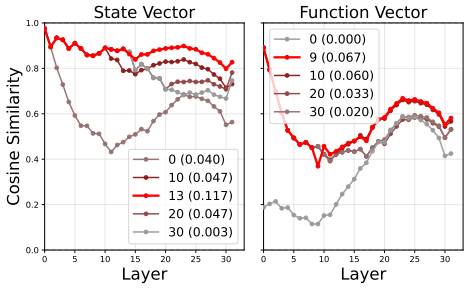
We conduct experiments across diverse architectures and model scales, including LLaMA-3.1-8B (AI@Meta, 2024), Qwen3-8B (Yang et al., 2025a), and Ministral-3-8B and 14B (Liu et al., 2026), and observe consistent trends across these settings. Full results are provided in the Appendix E, while our main results are reported using LLaMA-3.1-8B.

4.2 Task Vectors and Representation Gap

Section 3.2 suggests that existing task vector methods operate as lossy approximations to the latent representations produced by exact full-context inference. Here, we empirically validate this view by examining how design and hyperparameter choices in task vectors affect final latent representations and, ultimately, downstream task performance.

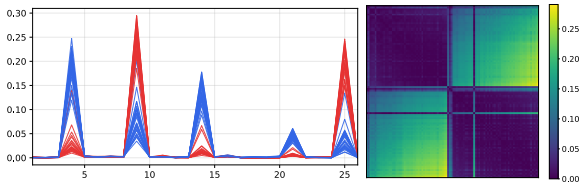


(a) Latent Similarity vs. Performance

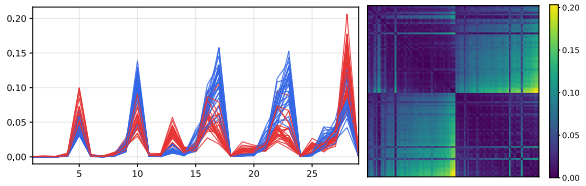


(b) Layer-wise Representational Similarity

Figure 2: **Approximation fidelity to the Full Context Baseline predicts task vector performance.** (a) Performance across various method–task pairs correlates strongly with the cosine similarity between the injected representation and the Full Context Baseline. (b) Layer-wise similarity between the final hidden states of the vector-injected model and the Full Context Baseline. Labels indicate the injection layer along with its final accuracy. Each curve represents a different injection layer; the **red curve** corresponds to the layer that maximizes performance, which also exhibits the highest representational similarity.



(a) English-French Classification



(b) English-French Bidirectional Translation

Figure 3: **Visualization of Attention Gaps.** (Left) Attention distributions of an attention head over context tokens, color-coded by query language: French and English. (Right) Pairwise Jensen-Shannon Divergence ordered by language. The language-based clustering demonstrates the query-dependent attention modes that a single task vector fails to capture.

Method	Reference	EN Input	FR Input	Overall
Baseline	N/A	96.3±0.7	99.7±0.7	98.0±0.6
Min-Gap	EN	97.7±1.7	91.0±6.2	94.3±2.7
Min-Gap	FR	51.0±23.2	100.0±0.0	75.5±11.6

(a) English-French Classification

Method	Reference	EN Input	FR Input	Overall
Baseline	N/A	84.1 ±1.9	81.4±1.7	82.8±1.3
Min-Gap	EN	75.3±23.6	26.2±16.7	50.7±3.5
Min-Gap	FR	5.5±2.6	79.3±1.4	42.4±1.4

(b) English–French Bidirectional Translation

Table 1: **Accuracy (%) across tasks using the Min-Gap.** The **Baseline** row corresponds to the exact Full Context Baseline. The language of the **Reference** query in the approximation is tested with different target query languages (**EN**: English, **FR**: French). We report the mean and standard deviation over 10 runs with different seeds.

Figure 2a examines the relationship between downstream task accuracy and the similarity of the approximated final-layer latents to the Baseline latents. The results reveal strong positive correlations (Pearson $r = 0.936$, Spearman $\rho = 0.867$), where methods that shift the latent state closer to the exact contextual representation achieve correspondingly higher accuracy. This representation-centric perspective not only explains the relative success of different algorithmic designs, but also demystifies their empirical hyperparameter choices. For instance, prior works heavily rely on full-network sweeps to identify an optimal injection layer as illustrated in Figure 2b, which shows the latent similarities of the respective methods to the baseline representation across layers. The results show that the empirically engineered injection layers for individual methods consistently match the layer that induces the highest latent similarity. Consequently, whether through the design of extraction heuristics or the search for an optimal injection point, existing approaches function as empirical optimizers for minimizing the representational gap.

Figure 2a further shows that Min-Gap closely approximates the latent representations of the Full Context Baseline. As a best-case approximation, Min-Gap minimizes errors from heuristic design choices, and isolates the structural limitations of static context injection in the following analyses.

4.3 Validation of Structural Bottlenecks

We now validate the bottlenecks from Section 3.4, demonstrating how static attention and variable

Baseline	Length Match		Length Mismatch	
	Token-wise	Mean	Token-wise	Mean
44.0	42.3 (-1.7)	40.2 (-3.8)	N/A	37.5 (-6.5)

Table 2: **Impact of reference–target length mismatch.** All values indicate accuracy (%), with drops computed relative to the full-context baseline. **Match/Mismatch** denotes whether the reference and target sequence lengths are identical. **Mean** pools context statistics into a single vector, while **Token-wise** injects position-specific statistics. Token-wise is unavailable under mismatched lengths due to strict positional alignment.

token lengths widen this representation gap.

Impact of Attention Gaps Certain tasks require query-dependent attention routing over context. In these scenarios, static task vectors—which mathematically lock the representation to a fixed attention map—cannot reproduce dynamic, input-specific routing. To evaluate this failure, we examine two tasks introduced in prior works (Zheng et al., 2024; Dong et al., 2025): English-French Classification and Bidirectional Translation.

Figure 3 visualizes the attention patterns from a single representative head for both tasks, computed via the Full Context Baseline. French and English inputs (coded in blue and red) reveal drastically different, query-dependent distributions: queries predominantly attend to context examples of the same language. This confirms that the model routes attention based on semantic properties rather than a monolithic pattern. Because task vectors inject a static representation derived from a single reference query, they are entirely blind to these critical, input-dependent shifts.

As shown in Table 1, this mismatch causes severe performance degradation with the Min-Gap compared to the Baseline. Crucially, static approximations become biased toward the reference request's behavior. For instance, in the Bidirectional Translation task, a Min-Gap extracted using an English reference performs well on English test queries (75.3%) but catastrophically fails when the test query switches to French (26.2%). Similarly, in Classification, a French reference defaults to the reference language, failing on English inputs (51.0%) while perfectly classifying French ones (100.0%). This bias is also asymmetric. An English reference in the Classification task preserves 94.3% overall accuracy. Such variance highlights a critical flaw: static approximations are not only incapable of dynamic routing but are acutely sensitive to the arbitrary selection of the reference request.

Impact of Input Length Gaps To strictly isolate the effect of input sequence variations, we evaluate a retrieval task: identifying the second-to-last word in a random list. We compare cases where the target query length matches or mismatches the reference query length used during extraction (Table 2). We apply the Min-Gap via two strategies: **Token-wise** (mapping statistics to exact positional indices) and **Mean** (compressing statistics into a position-agnostic vector). When the lengths match, Token-wise achieves the highest Min-Gap performance (42.3%), closely tracking the full-context Baseline (44.0%), while Mean drops further to 40.2%. Under length mismatch, Token-wise is not applicable because strict positional alignment breaks down, and Mean further decreases to 37.5%. This confirms that locking representations to fixed positions or averaging limits task vector robustness.

Impact of Output Length Gaps We evaluate multi-token generation using a task requiring the model to reverse a list of n words, focusing on output token length rather than input request length. As shown in Figure 4, we compare two strategies: **Token-wise**, which applies the Min-Gap independently at every output position, and **Copy**, which reuses the first-position vector for all subsequent tokens. The results reveal that static injection is acutely sensitive to output length. Token-wise maintains relatively high fidelity but is structurally restricted to the specific token count of the reference output; thus, it is inapplicable when the generated sequence length deviates from the reference. Conversely, while Copy allows for variable-length generation, it suffers a performance collapse as the sequence progresses. This indicates that a single static vector cannot capture the evolving attention requirements of autoregressive generation. The widening gap between these approximations and the **Baseline** confirms that static representations fail to model the dynamic trajectory of multi-token outputs.

4.4 Revisiting Task Vectors with Attention Dynamics

Attention Dynamics Statistics To evaluate the attention dynamics of each task, we quantify the gap between query-specific attention patterns with **Mean Attention Distance (MAD)**. MAD quantifies the mean pairwise Jensen-Shannon Divergence between the attention distributions of different queries within a task; a high MAD indicates large pairwise divergence among query-specific

Task	Min-Gap Analysis				Task Vector Accuracy		
	MAD ($\times 10^3$)	Baseline (A)	Min-Gap (B)	Δ (A−B)	STV	FV	SV
Country→Capital	6.64±0.13	98.4±0.5	98.2±0.5	0.2±0.4	97.9	0.0	96.8
Word-to-Uppercase	9.58±0.60	100.0±0.0	100.0±0.0	0.0±0.0	100.0	97.5	99.6
Antonym Generation	9.62±0.77	70.5±1.8	70.3±1.5	0.2±1.9	65.2	54.7	63.0
En→Es Translation	10.16±0.86	88.0±0.9	87.6±0.8	0.4±1.1	84.1	45.7	82.2
En-Fr Classification	**14.97**±0.92	98.0±0.6	75.5±11.6	**22.5**±11.7	2.5	0.8	12.5
Choose Last Word + First Letter	5.98±0.34	95.5±3.1	92.5±3.3	3.0±3.3	34.7	4.0	17.0
Choose Last Word + Uppercase	7.51±0.54	99.5±0.7	88.7±5.6	10.7±5.8	0.0	75.8	0.0
Antonym + En→Fr	11.70±1.47	45.6±1.4	45.8±1.8	−0.1±2.0	10.5	1.8	8.7
Bidirectional En-Fr	**14.43**±0.72	82.8±1.3	42.4±1.4	**40.4**±0.9	28.5	7.7	46.0

Table 3: **Task-wise comparison of attention statistics and accuracy. MAD** (Mean Attention Distance) denotes the mean pairwise distance between attention patterns. We report the mean and standard deviation over 10 runs with different seeds. The upper block shows atomic tasks and the lower block shows composite tasks. Existing task vector methods (STV, FV, SV) are shown with a gray background.

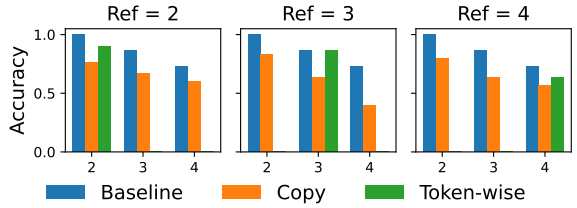


Figure 4: **Accuracy under output length mismatch. Ref** and the x-axis denote the reference and predicted output token counts, respectively. **Baseline** uses the full context C, while two Min-Gap variants, **Token-wise** and **Copy**, approximate C without context attention: **Token-wise** transfers context statistics at every output position, whereas **Copy** reuses the first-position vector. Both variants show a performance gap relative to the baseline, which is substantially larger for **Copy**.

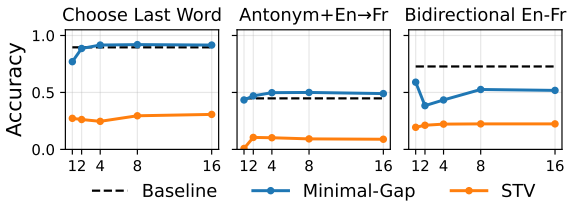


Figure 5: **Denoising and Erasure Effects of Averaging.** Accuracy is shown as a function of the number of averaged task vectors. For tasks with stable attention (left, middle), averaging acts as a denoising mechanism that improves performance. Conversely, for tasks with large attention gaps (right), averaging washes out the distinct representations required for different inputs, leading to systematic performance degradation.

attention distributions. We note that the gap contributes to performance degradation but is not the sole governing factor; accordingly, MAD is not intended as a predictor of task vector performance.

Task Dependency of Attention Gaps Table 3 presents attention statistics and accuracy (%) across tasks. Our results suggest that task vectors tend to be more effective in regimes where this gap remains small. Atomic tasks that have shown high performance in prior task vector studies (Hendel et al., 2023; Todd et al., 2024) (upper block in Table 3) exhibit consistently low MAD. This suggests that the perceived efficacy of task vectors in previous literature is largely a byproduct of evaluating tasks with "lazy" attention dynamics that do not significantly deviate across different queries. In contrast, as the attention gap widens, in English-French Classification and Bidirectional Translation, transferability collapses. This result is consistent across model architectures and sizes (Appendix E).

Revisiting Task Composition Prior literature has suggested that task vectors are fundamentally limited when applied to composite tasks (Tikhonov et al., 2025; Li et al., 2025b), based on experiments with methods that inject at only one or a few layers. Revisiting these tasks with Min-Gap, we find that compositionality is not, by itself, a fundamental barrier. These findings suggest that task vectors fail on composite tasks not because they are complex, but because certain compositions happen to induce large attention gaps we have identified. This result is consistent across model architectures and sizes (Appendix E).

Revisiting Prior Task Vectors The observations derived from our Min-Gap analysis further extend to existing training-free task vector methods. In Table 3, the gray columns report the results of prior methods, including STV (Hendel et al., 2023), FV (Todd et al., 2024), and SV (Li et al., 2024a). Despite occasional unexpected drops caused by heuristic extraction and injection errors, the overall association remains consistent: tasks where Min-Gap degrades under large attention gaps also fail with existing methods, indicating that this limitation is intrinsic to task vectors.

The Denoising vs. Erasure Effect of Averaging In prior work, averaging is typically employed to produce a more "robust" representation (Hendel et al., 2023). Our framework suggests that this works as denoising when attention patterns are stable, suppressing idiosyncratic variations while preserving the core task-relevant component. However, when attention patterns diverge across queries, averaging becomes counterproductive: it washes out query-specific representations and produces a task vector that faithfully represents none of them. As shown in Figure 5, low attention-gap tasks (left, middle) benefit from pooling, whereas high attention-gap tasks (right) suffer performance drop below that of a single task vector.

Implications for Future Directions Our results show that tasks fail when they induce a large attention or length gap, and that the binding constraint is caching a static, position-locked representation after the attention contraction. By making this decomposition explicit, our analysis marks which directions are likely and unlikely to be productive. Future work should either address the dynamic component directly or focus on the low-gap regime where static approximation is structurally sound.

5 Conclusion

We introduce the Minimal-Gap Approximation to isolate the fundamental limitations of task vectors from heuristic artifacts. We isolate the intrinsic constraints of task vectors, revealing that task vector performance is constrained by structural attention and length gaps. Extensive experiments across existing extract-and-inject training-free task vector methods, model architectures, and model scales demonstrate that these limitations are consistent and inherent to static context approximations.

6 Limitations

Although Minimal-Gap Approximation provides an approximation to full-context computation, our analysis and experiments are still conducted in controlled settings centered on task vector benchmarks and synthetic in-context tasks. In addition, while Minimal-Gap Approximation clarifies the best-case behavior of static task vector approximations, it does not by itself solve the practical challenges posed by highly query-dependent attention patterns. Finally, attention and length gaps are not the only factors governing performance. Developing reliable predictive signals for task vector success remains an important direction for future work.

7 Societal Impact

Our analysis shows that static task vector methods are not a uniformly safe replacement for explicit context: on tasks with large attention gaps, performance can collapse and the failures are input-dependent. Moreover, a task vector extracted from a French reference fails more often than one extracted from an English reference (Table 1), meaning the method works less well when the reference context is not in English.

Acknowledgements

This research was supported by MSIT (IITP: IITP-2026-RS-2024-00436857, RS-2024-00398115, IITP-2026-RS-2020-II201819, IITP-2026-RS-2025-02304828, RS-2026-25507282, RS-2026-25585074; NRF: RS-2025-23523979), MCST (KOCCA: RS-2026-25506607, RS-2024-00345025), MND (RS-2026-25551943) and MSS (RS-2026-25549378). This research was also supported by the "Advanced GPU Utilization Support Program" funded by the Government of the Republic of Korea (MSIT).

References

AI@Meta. 2024. Llama 3 model card. *ArXiv*.

Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Yonatan Belinkov. 2023. What learning algorithm is in-context learning? investigations with linear models. In *The Eleventh International Conference on Learning Representations*.

Sumin An, Junyoung Sung, Wonpyo Park, Chanjun Park, and Paul Hongsuck Seo. 2025. Lcirc: A recurrent compression approach for efficient long-form context and query dependent modeling in llms. *arXiv preprint arXiv:2502.06139*.

Thibaud Ardoin, Semira Einsele, Evis Bregu, and Gerhard Wunder. 2026. Prompt compression via activation aggregation. *arXiv preprint arXiv:2607.08399*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901.

Stephanie C. Y. Chan, Adam Santoro, Andrew Kyle Lampinen, Jane X. Wang, Aaditya K Singh, Pierre H. Richemond, J. Mcclelland, and Felix Hill. 2022. Data distributional properties drive emergent in-context learning in transformers. In *Neural Information Processing Systems*.

Yuxin Dong, Jiachen Jiang, Zhihui Zhu, and Xia Ning. 2025. Understanding task vectors in in-context learning: Emergence, functionality, and limitations.

Seungwook Han, Jinyeop Song, Jeff Gore, and Pulkit Agrawal. 2024. Emergence and effectiveness of task vectors in in-context learning: An encoder decoder perspective. *arXiv preprint arXiv:2412.12276*.

Roee Hendel, Mor Geva, and Amir Globerson. 2023. In-context learning creates task vectors. *ArXiv*, abs/2310.15916.

Alberto Hojel, Yutong Bai, Trevor Darrell, Amir Globerson, and Amir Bar. 2024. Finding visual task vectors. In *European Conference on Computer Vision*, pages 257–273. Springer.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Brandon Huang, Chancharik Mitra, Assaf Arbelle, Leonid Karlinsky, Trevor Darrell, and Roei Herzig. 2024. Multimodal task vectors enable many-shot multimodal in-context learning. *ArXiv*, abs/2406.15334.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. 2022. Editing models with task arithmetic. In *International Conference on Learning Representations*.

Jihoon Kwon, Jiwon Choi, and Jy-yong Sohn. 2026. Distributional alignment as a criterion for designing task vectors in in-context learning. *arXiv preprint arXiv:2605.20730*.

Nurkhan Laiyk, Gerard I Gállego, Javier Ferrando, and Fajri Koto. 2026. Exploring language-agnosticity in function vectors: A case study in machine translation. *arXiv preprint arXiv:2604.19678*.

Dongfang Li, Zhenyu Liu, Xinshuo Hu, Zetian Sun, Baotian Hu, and Min Zhang. 2024a. In-context learning state vector with inner and momentum optimization. *Advances in Neural Information Processing Systems*, 37:7797–7820.

Jiaqian Li, Yanshu Li, Ligong Han, Ruixiang Tang, and Wenya Wang. 2025a. Towards generalizable implicit in-context learning with attention routing. *arXiv preprint arXiv:2509.22854*.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. 2023. Inference-time intervention: Eliciting truthful answers from a language model. *https://arxiv.org/abs/2306.03341*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*, pages 4582–4597.

Yuxuan Li, Declan Campbell, Stephanie CY Chan, and Andrew Kyle Lampinen. 2025b. Just-in-time and distributed task representations in language models. *arXiv preprint arXiv:2509.04466*.

Zhuowei Li, Zihao Xu, Ligong Han, Yunhe Gao, Song Wen, Di Liu, Hao Wang, and Dimitris N Metaxas. 2024b. Implicit in-context learning. *arXiv preprint arXiv:2405.14660*.

Alexander H. Liu, Kartik Khandelwal, Sandeep Subramanian, Victor Jouault, Abhinav Rastogi, Adrien Sadé, Alan Jeffares, Albert Jiang, Alexandre Cahill, Alexandre Gavaudan, and 1 others. 2026. Ministral 3. *arXiv preprint arXiv:2601.08584*.

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

Yiting Liu and Zhi-Hong Deng. 2025. Iterative vectors: In-context gradient steering without backpropagation. In *Forty-second International Conference on Machine Learning*.

Grace Luo, Trevor Darrell, and Amir Bar. 2024. Vision-language models create cross-modal task representations.

Jesse Mu, Xiang Li, and Noah Goodman. 2023. Learning to compress prompts with gist tokens. *Advances in Neural Information Processing Systems*, 36:19327–19352.

Arjun Panickssery, Nina Rimsky, Yair Gabrieli, Neel Nanda, Adam Jermyn, and Samuel Marks. 2024. Steering llama 2 via contrastive activation addition. *arXiv preprint arXiv:2312.06681*.

Seongheon Park, Xuefeng Du, Min-Hsuan Yeh andHaobo Wang, and Yixuan Li. 2025. Steer llm latents for hallucination detection. *https://arxiv.org/abs/2503.01917*.

Baturay Sağlam, Zhuoran Yang, Dionysis Kalogerias, and Amin Karbasi. 2025. Learning task representations from in-context learning.

Pavel Tikhonov, Ivan Oseledets, and Elena Tutubalina. 2025. One task vector is not enough: A large-scale study for in-context learning.

Eric Todd, Millicent Li, Arnab Sen Sharma, Aaron Mueller, Byron C. Wallace, and David Bau. 2024. Function vectors in large language models. *ArXiv*, abs/2310.15213.

Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. 2022. Model soups: Averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. *arXiv preprint arXiv:2203.05482*.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. 2021. An explanation of in-context learning as implicit bayesian inference. In *International Conference on Learning Representations*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025a. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Haolin Yang, Hakaze Cho, Kaize Ding, and Naoya Inoue. 2025b. Task vectors, learned not extracted: Performance gains and mechanistic insight. *arXiv preprint arXiv:2509.24169*.

Liu Yang, Ziqian Lin, Kangwook Lee, Dimitris Papailiopoulos, and Robert Nowak. 2025c. Task vectors in in-context learning: Emergence, formation, and benefit.

Zhipeng Yang, Junzhuo Li, Siyu Xia, and Xuming Hu. 2025d. Internal chain-of-thought: Empirical evidence for layer-wise subtask scheduling in llms. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 22547–22575.

Kayo Yin and Jacob Steinhardt. 2025. Which attention heads matter for in-context learning? *ArXiv*, abs/2502.14010.

Bowen Zheng, Ming Ma, Zhongqiao Lin, and Tianming Yang. 2024. Label words as local task vectors in in-context learning. *arXiv preprint arXiv:2406.16007*.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, and 1 others. 2023. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*.

A Jensen–Shannon Divergence Across Tasks

We visualize the pairwise *Jensen–Shannon (JS) divergence* across all layers and heads for different tasks. These patterns reveal how attention distributions vary across inputs. Some tasks exhibit more structured and clustered attention patterns, while others show more diffuse and heterogeneous distributions, reflecting differences in how attention is allocated rather than differences in task quality.

B Task Descriptions

We evaluate our method on a set of synthetic tasks designed to probe different aspects of in-context learning and compositional generalization. We group the tasks into *atomic* tasks, which involve a single transformation, and *composite* tasks, which require sequentially applying multiple operations. Each task contains 100 to 300 samples.

Atomic Tasks. These tasks involve a single transformation applied to the input.

- **Country→Capital**: Return the capital of the given country. e.g., `Hungary` → `Budapest`
- **Word-to-Uppercase**: Capitalize the first letter. e.g., `playful` → `Playful`
- **Extract First Letter**: Return the first character. e.g., `playful` → `p`
- **Antonym Generation**: Return a negated antonym-like form of the input. e.g., `psychic` → `non-psychic`
- **En→Es Translation**: Translate English to Spanish. e.g., `ten` → `diez`
- **Choose Last Word**: Return the last item from a list. e.g., `toward, win, obtain` → `obtain`
- **En-Fr Classification**: Identify whether the word is English or French. e.g., `customs` → `English`, `décès` → `French`

Composite Tasks. These tasks require combination of two operations.

- **Choose Last Word + First Letter**: Select the last item and return its first letter. e.g., `toward, win, obtain` → `o`
- **Choose Last Word + Uppercase**: Select the last item and capitalize it. e.g., `toward, win, obtain` → `Obtain`
- **Antonym + En→Es**: Translate the negated antonym to Spanish. e.g., `psychic` → `no psíquico`
- **Bidirectional En-Fr**: Translate between English and French in both directions. e.g., `deaths` → `décès`, `décès` → `deaths`

Length Generalization Tasks. To further evaluate robustness under varying input and output lengths, we tested following tasks.

- **Second-to-Last Word**: Return the second-to-last item from a list. e.g., `toward, apple, win, obtain` → `win`
- **Reverse List**: Reverse the order of items in the list. e.g., `toward, apple, win, obtain` → `obtain, win, apple, toward`

C Task Vector Methods

Under our unifying view, prior baselines mainly differ along four axes: (1) *what* representation is extracted from the ICL run, (2) *where* that representation is injected, (3) *how* it is aggregated or adjusted across examples, layers, or heads, and (4) whether the interaction between cached task information and the current query representation is fixed, tuned, or explicitly optimized.

- **STV (Hendel et al., 2023).** This is the baseline for the "single cached task vector" view. STV intervenes at a single layer ℓ^*, replacing the query-only hidden state with a state extracted from a contextualized reference request $\tilde{R}$:

$$\mathbf{o}^{(\ell^*)} \leftarrow \mathbf{o}^{(\ell^*)}(C, \tilde{R}). \tag{7}$$

  The gap. It overwrites every term at once, discarding the target query's own computation and carrying the reference query's residue instead.

- **FV (Todd et al., 2024).** FV differs from STV by assuming that task information is not concentrated in one hidden state, but distributed across a small set of causally important attention heads. The extracted vector is therefore not a raw hidden state but an aggregate of mean head outputs. FV also intervenes at a single layer ℓ^*, adding a task vector to the request-only state without rescaling it. The added vector is constructed by

averaging and summing the outputs of causally selected heads:

$$\mathbf{o}^{(\ell^*)} = \mathbf{x}^{(\ell^*)} + \mathbf{v}_t^{\mathrm{FV}} + \mathbf{o}_R^{(\ell^*)}, \qquad \mathbf{o}_R^{(\ell^*)} = \frac{\mathbf{h}_R^{(\ell^*)}}{z_R^{(\ell^*)}}. \tag{8}$$

The gap. The rescaling of the request term is missing. The added vector is also not an isolated context term: the selected heads attend to both the context and the reference request.

- **State Vector (Li et al., 2024a).** State Vector keeps the early-layer compressed-task view, but argues that the extracted vector can itself be improved before reinjection. Its main novelty is a set of refinement strategies: inner optimization, momentum optimization, and divide-and-conquer aggregation. SV replaces the attention activation of the final separator token across all of the first L layers:

$$\mathbf{o}^{(i)} = \mathbf{x}^{(i)} + \mathbf{v}_t^{\mathrm{SV},(i)}, \quad \forall\, i \le L,$$
$$\mathbf{v}_t^{\mathrm{SV},(i)} = \frac{\mathbf{h}_C^{(i)}(\tilde{R}) + \mathbf{h}_{\tilde{R}}^{(i)}}{z_C^{(i)}(\tilde{R}) + z_{\tilde{R}}^{(i)}} \tag{9}$$

The gap. The injected activation fuses the context term with the reference's request term, so the target query's own request-side contribution is discarded at every layer up to L.

D Discussion on Mean Attention Distance (MAD)

While we use MAD to quantify the attention gap of a given task, it is not intended as a predictive metric. Its absolute value depends on the model, on the input and output spaces of the task, and on the context length, while task vector performance additionally depends on properties of the task itself, such as the sensitivity of its output. Once these factors are controlled for, however, MAD provides a measure of the attention gap.

E Cross-Model Generalization

We extend our empirical evaluation across multiple model architectures and scales, including LLaMA, Qwen, and Ministral-Instruct models. The results, summarized in Table 6, confirm that our central observations hold across model families and parameter sizes: tasks that degrade substantially under the Minimal-Gap Approximation, such as **Classification** and **Bidirectional En-Fr**, consistently exhibit high MAD values. En-Fr Classification on Qwen is an exception, showing only a small drop despite a high MAD value. This suggests that a large attention gap does not always translate into failure, as some tasks are less sensitive to perturbations in their output space. This also illustrates the limitation of MAD as a standalone metric, as discussed in Section D.

F Attention Gaps in Non-Linguistic Bidirectional Tasks

The bidirectional translation results in Table 1 involve a single English–French axis, leaving open whether the observed reference-locked collapse is specific to language. We therefore examine two country–capital tasks whose bidirectional axis is non-linguistic: classifying a word as a country or a capital (Table 4), and mapping a country to its capital and a capital to its country (Table 5). Both are expressed entirely in English, and we use Llama-3.1-8B throughout. In both tasks, Min-Gap follows its reference context: accuracy is high on the matching input type and degrades on the other, while ICL performs comparably in both directions. This result shows that the collapse in classification and bidirectional tasks is not tied to language-specific structure.

Method	Reference	Country	Capital	Overall
Baseline	–	81.0	75.0	78.0
Min-Gap	Country	100.0	0.0	50.0
Min-Gap	Capital	0.0	100.0	50.0

Table 4: **Country–Capital Classification.** Given a word, classify whether it is a country or a capital. Columns indicate the input type at evaluation time.

Method	Reference	Country	Capital	Overall
Baseline	–	97.0	98.0	97.5
Min-Gap	Country	96.0	52.0	74.0
Min-Gap	Capital	84.0	96.0	90.0

Table 5: **Country–Capital Bidirectional.** Given a country, return its capital; given a capital, return its country.

G Relationship to KV caching

Min-Gap shares an operational goal with KV caching: both cache quantities computed during a full-context forward pass and reuse them for subsequent requests. The two differ in *where* the cache is taken relative to the query-dependent contraction.

KV caching stores the pre-contraction keys and values $\{(\mathbf{k}_j, \mathbf{v}_j)\}_{j \in C}$, so every new query re-evaluates its attention over the context and the resulting attention map remains free to vary with the input. Min-Gap instead caches $\mathbf{h}_C^{(i)}$ and $z_C^{(i)}$, the *outcome* of that contraction under a single reference query, and existing task vectors further pool these post-contraction aggregates into one $\mathcal{O}(d)$ vector. In this sense, a task vector can be read as an extreme form of KV compression, collapsing the context into a static, position-locked cache. Framed this way, the structural limitations exposed by Min-Gap also delineate which directions KV compression should avoid: any scheme that fixes the attention map over the context will lose performance on tasks with a large attention gap, regardless of how faithfully the cached values themselves are preserved.

H Discussion on Training-Based Task Vectors

Recent work has explored *training-based* task vectors, such as LTV (Yang et al., 2025b) and I2CL (Li et al., 2024b), which are learned through additional supervision. While these approaches show high performance, they are conceptually distinct from the extract-and-inject paradigm studied in this work.

In particular, training-based task vectors function as task-specific parameters optimized with additional data (e.g., 100 samples for (Yang et al., 2025b)), making them more closely related to parameter-efficient fine-tuning methods such as LoRA (Hu et al., 2021). Although they share a similar interface—injecting low-dimensional representations into the model—their underlying mechanisms differ fundamentally.

A primary distinction lies in how the training process alters the token-level latent space. In our preliminary experiments, we observe that the final-token representations of LTV (Yang et al., 2025b) deviate substantially from those of the Full-Context baseline, exhibiting a relatively low cosine similarity of 0.46—comparable to that of a LoRA-tuned model (0.35). In contrast, context-extracted task vectors preserve significantly higher similarities, ranging from 0.82 for TV (Hendel et al., 2023) to 0.65 for FV (Todd et al., 2024).

This suggests that training-based methods effectively construct a new task-adapted model, rather than approximating the original in-context computation. As a result, applying our Minimal-Gap framework—which is designed to characterize the limits of exact self-attention under static compression—to such trained representations falls outside the intended scope of our analysis.

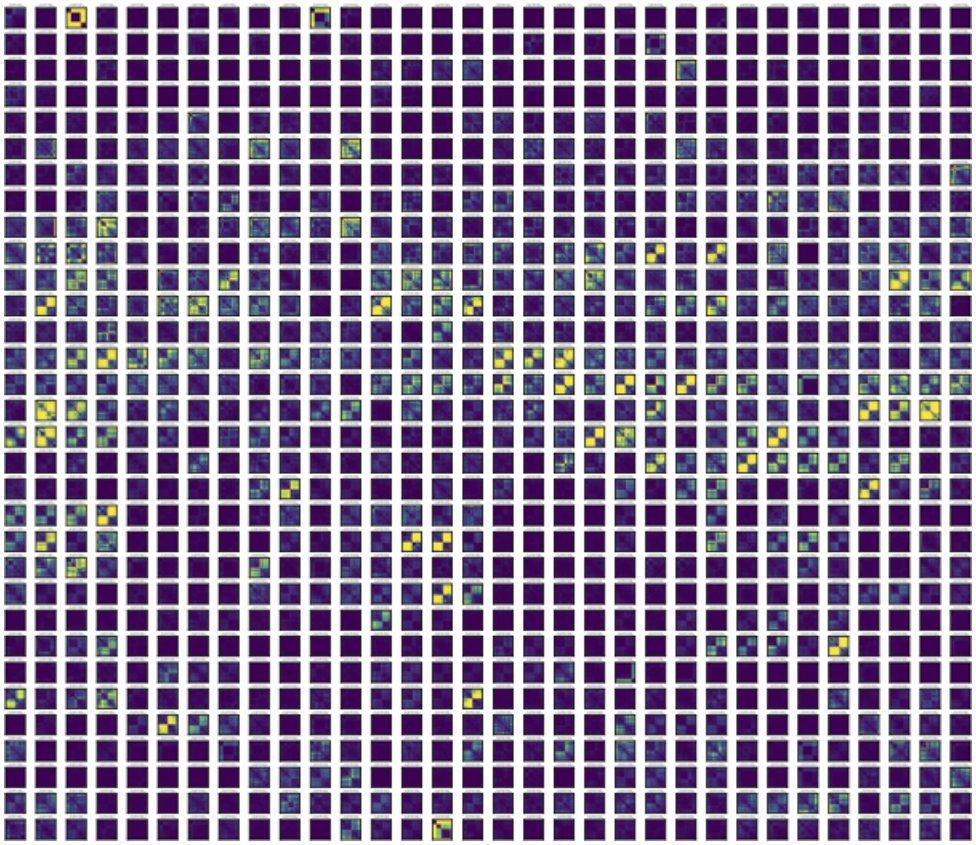
Overall, our work focuses on the intrinsic limitations of the *training-free extract-and-inject* paradigm.

I Use of AI Assistants

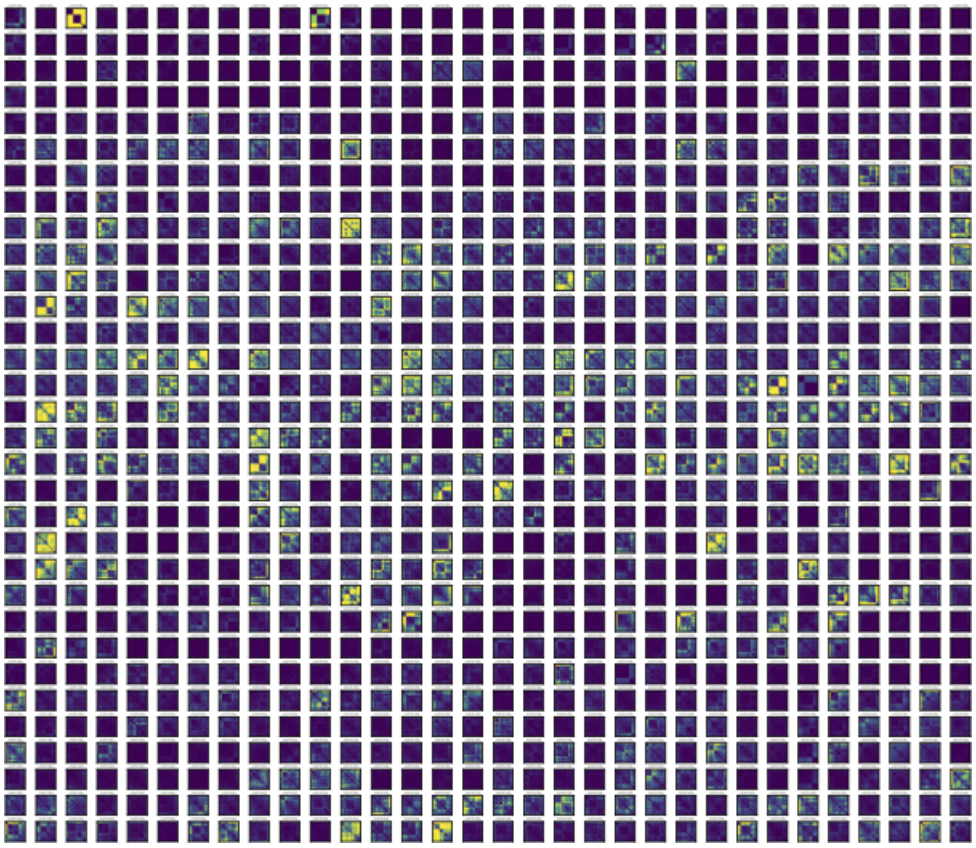
AI assistants were used during manuscript preparation for writing support, such as grammar correction and improving clarity and flow.

		Attention Statistics	Accuracy (%)		
Model	**Task**	**MAD** ($\times 10^3$)	**Baseline (A)**	**Min-Gap (B)**	Δ (A-B)
LLaMA-3.1-8B	Country→Capital	6.46	98.9	97.9	1.0
	Word-to-Uppercase	8.35	100.0	100.0	0.0
	Antonym Generation	9.49	70.7	68.8	1.9
	En→Es Translation	10.81	88.4	87.0	1.4
	En-Fr Classification	13.18	98.3	76.7	21.6
	En-Es Classification	13.67	96.7	58.3	38.4
	Antonym + En→Es	12.01	55.4	48.6	6.8
	Antonym + En→Fr	11.52	45.9	47.7	-1.8
	Choose Last + First Letter	5.01	95.3	90.3	5.0
	Choose Last + Uppercase	7.65	99.3	92.1	7.2
	Bidirectional En-Fr	13.60	82.8	42.5	40.3
Ministral-3-8B	Country→Capital	9.90	97.9	97.9	0.0
	Word-to-Uppercase	16.46	100.0	100.0	0.0
	Antonym Generation	13.74	70.7	69.6	1.1
	En→Es Translation	15.65	86.2	87.0	-0.8
	En-Fr Classification	22.25	95.0	72.5	22.5
	En-Es Classification	21.89	95.0	50.0	45.0
	Antonym + En→Es	14.34	53.6	56.2	-2.6
	Antonym + En→Fr	14.78	50.9	50.9	0.0
	Choose Last + First Letter	10.42	96.8	77.3	19.5
	Choose Last + Uppercase	12.64	98.6	83.0	15.6
	Bidirectional En-Fr	20.01	83.3	44.8	38.5
Ministral-3-14B	Country→Capital	12.77	97.9	97.9	0.0
	Word-to-Uppercase	19.93	100.0	100.0	0.0
	Antonym Generation	16.04	71.4	72.8	-1.4
	En→Es Translation	16.08	88.8	89.1	-0.3
	En-Fr Classification	24.53	93.3	52.5	40.8
	En-Es Classification	25.96	91.7	50.0	41.7
	Antonym + En→Es	16.49	52.2	53.6	-1.4
	Antonym + En→Fr	17.52	53.1	52.3	0.8
	Choose Last + First Letter	9.22	98.6	92.1	6.5
	Choose Last + Uppercase	15.42	99.6	98.6	1.0
	Bidirectional En-Fr	21.90	83.8	45.8	38.0
Qwen3-8B	Country→Capital	15.83	96.8	97.9	-1.1
	Word-to-Uppercase	21.50	100.0	99.6	0.4
	Antonym Generation	17.69	67.4	63.4	4.0
	En→Es Translation	20.99	82.6	83.7	-1.1
	En-Fr Classification	32.42	96.7	93.3	3.4
	En-Es Classification	28.28	90.0	52.5	37.5
	Antonym + En→Es	20.73	48.9	49.6	-0.7
	Antonym + En→Fr	21.17	46.6	48.4	-1.8
	Choose Last + First Letter	11.78	95.7	94.2	1.5
	Choose Last + Uppercase	19.31	97.8	92.1	5.7
	Bidirectional En-Fr	24.50	78.8	40.8	38.0

Table 6: **Cross-model evaluation of attention statistics and performance.**

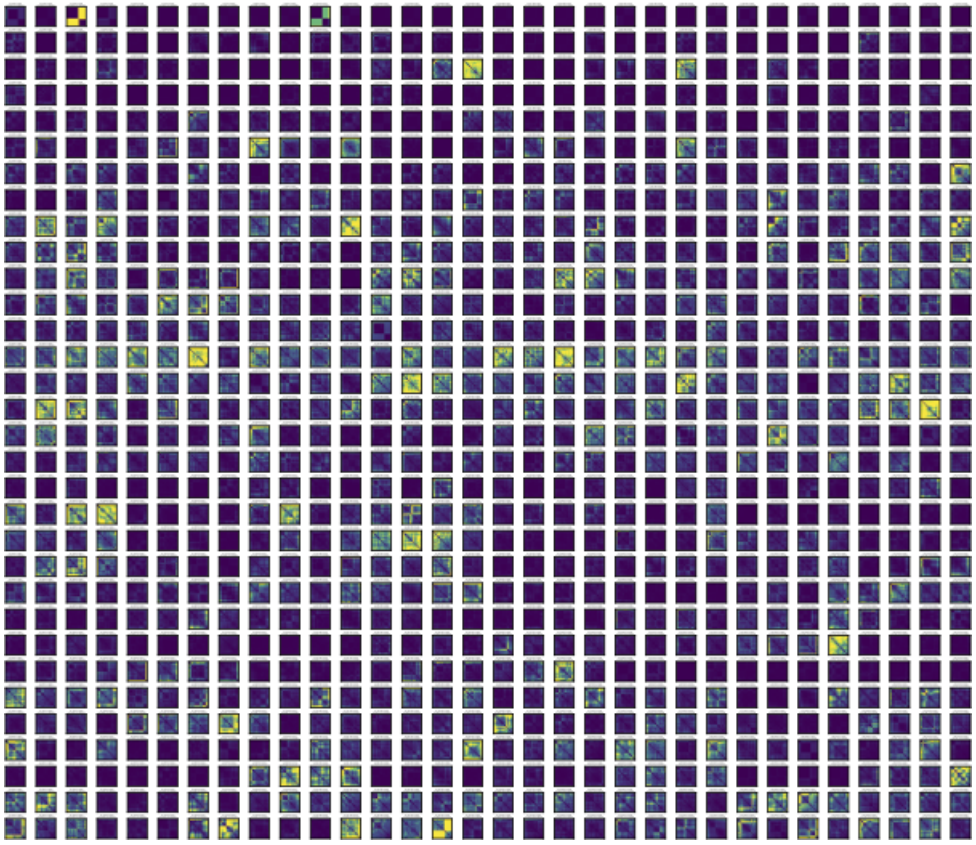


(a) En-Fr Classification

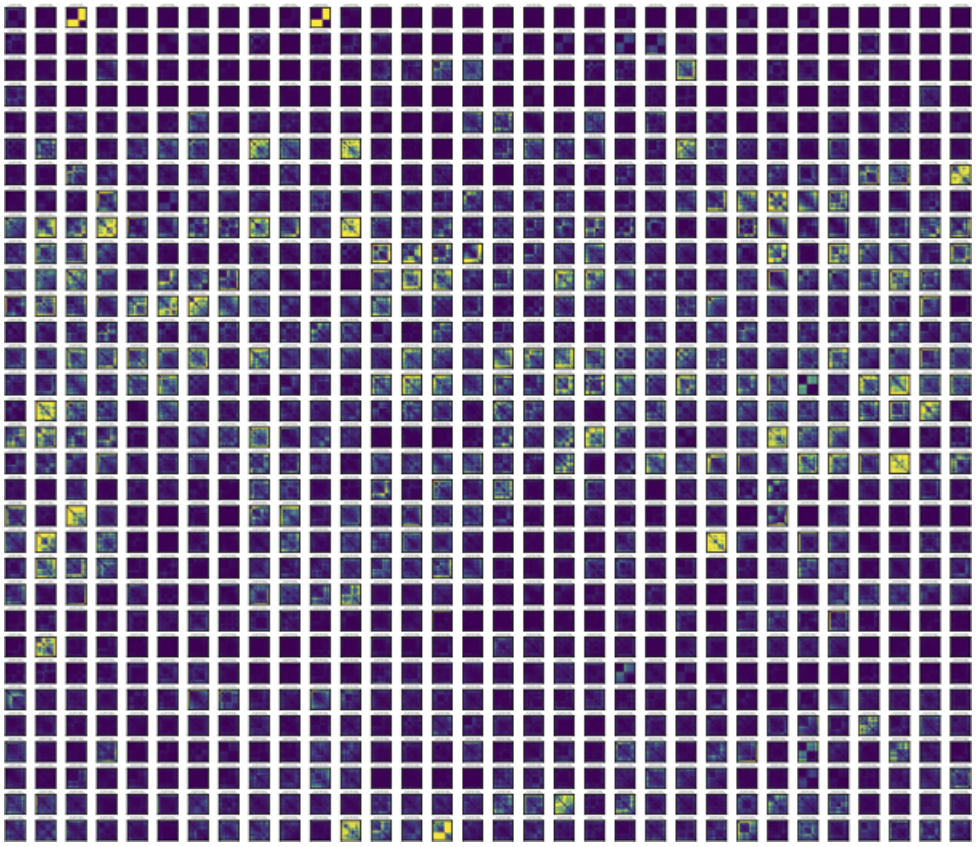


(b) Bidirectional En-Fr

Figure 6: **Pairwise Jensen–Shannon Divergence heatmaps across all layers and heads (1).** Attention patterns form clusters across inputs.



(a) Choose Last Word



(b) En→Es Translation

Figure 7: **Pairwise Jensen–Shannon Divergence heatmaps across all layers and heads (2).** Attention patterns are less clustered and more diffuse.