

The appendix is organized as follows. We first provide detailed descriptions of all tasks in Appendix A. Next, we add more discussion on related works in Appendix B. Appendix C presents additional experiments, including verification of the *Layerwise Compression-Expression* and an ablation study on the choice of the last separation tokens. Appendix D offers a fine-grained token-level analysis, featuring saliency maps and grid TDVN visualizations. Proofs of Theorem 1 are given in Appendix E, followed by an examination of the i.i.d. assumption through repetition experiments in Appendix F. We then provide detailed descriptions and illustrations of task-vector accuracy and early-exit accuracy in Appendix G. Finally, Appendix H presents illustrations of the task-vector contrastive fine-tuning method and PCA visualization of extracted task vectors.

A TASK DESCRIPTION

A.1 SYMBOLIC TASKS

Detailed descriptions of the symbolic tasks used in our empirical studies are provided in Table 1, covering the algorithmic, translation, linguistic, and knowledge domains.

Table 1: Descriptions of symbolic tasks.

Task Domains	Task	Example	Description
Algorithmic(Letter-to-Letter)	Copy Letter	a \rightarrow a	Output the same letter of the given letter.
	Next Letter	a \rightarrow b	Output the next letter of the given letter in the alphabet.
	To Uppercase	a \rightarrow A	Output the corresponding uppercase letter of the given lowercase letter.
	Prev Letter	b \rightarrow a	Output the previous letter of the given letter in the alphabet.
	Next 2 Letter	a \rightarrow c	Output the letter that comes two positions after the given letter in the alphabet.
Algorithmic(List-to-Element)	List First	[a,b,c] \rightarrow a	Output the first item in the given list.
	List Last	[a,b,c] \rightarrow c	Output the last item in the given list.
	List Length	[a,b,c] \rightarrow 3	Output length of the given list.
	List First Upper	[a,b,c] \rightarrow A	Get the first item in the given list, then output the corresponding uppercase letter.
	List Last Upper	[a,b,c] \rightarrow C	Get the last item in the given list, then output the corresponding uppercase letter.
Translation	French \rightarrow English	bonjour \rightarrow hello	Translate the given French word into English.
	Spanish \rightarrow English	gracias \rightarrow thank you	Translate the given Spanish word into English.
	English \rightarrow French	goodbye \rightarrow au revoir	Translate the given English word into French.
	English \rightarrow Italian	music \rightarrow musica	Translate the given English word into Italian.
	English \rightarrow Spanish	thank you \rightarrow gracias	Translate the given English word into Spanish.
Linguistic	Antonyms	hot \rightarrow cold	Output the antonym of the given word.
	Plural \rightarrow Singular	cats \rightarrow cat	Convert the given plural noun to its singular form.
	Present Simple \rightarrow Gerund	run \rightarrow running	Convert the given verb from present simple to its gerund form.
	Present Simple \rightarrow Past Perfect	walk \rightarrow had walked	Convert the given verb from present simple to past perfect tense.
	Present Simple \rightarrow Past Simple	jump \rightarrow jumped	Convert the given verb from present simple to past simple tense.
Knowledge	Singular \rightarrow Plural	dog \rightarrow dogs	Convert the given singular noun to its plural form.
	Country \rightarrow Capital	France \rightarrow Paris	Output the capital city of the given country.
	Football Player \rightarrow Position	Lionel Messi \rightarrow Forward	Output the playing position of the given football player.
	Location \rightarrow Continent	Brazil \rightarrow South America	Output the continent where the given location is found.
	Location \rightarrow Country	Kyoto \rightarrow Japan	Output the country in which the given location is situated.
	Location \rightarrow Language	Egypt \rightarrow Arabic	Output the primary language spoken in the given location.
	Location \rightarrow Religion	India \rightarrow Hinduism	Output the predominant religion of the given location.

A.2 LANGUAGE UNDERSTANDING TASKS

For evaluating TDNV on natural language understanding tasks, we require each query sentence to be assessed across multiple attributes. To enable this, we construct a synthetic natural language dataset in which every sentence can be evaluated on six distinct attributes: length, semantic polarity, tense, sentence type, subject person, and entity type. Each attribute is associated with several categorical labels, all of which are summarized in Table 2. The dataset contains 1,000 samples, and representative examples of these sentences are provided in Table 3.

Attribute	Labels
Length	short, medium, long
Semantic Polarity	positive, negative, neutral
Tense	present, past, future, progressive
Sentence Type	declarative, interrogative, imperative, exclamatory
Subject Person	first_person, second_person, third_person
Entity Type	person, location, organization

Table 2: The attributes and labels in language understanding dataset.

Sentence	Length	Semantic Polarity	Tense	Sentence Type	Subject Person	Entity Type
I enjoy morning walks.	short	positive	present	declarative	first_person	person
Close the window now.	short	neutral	present	imperative	second_person	location
Despite the heavy rain, our research team successfully completed the outdoor experiment and gathered all the required samples before sunset.	long	positive	past	declarative	third_person	organization
Will you be visiting the United Nations headquarters in New York next year to attend the global climate summit?	long	neutral	future	interrogative	second_person	location
While the orchestra rehearsed the challenging new symphony, the conductor meticulously adjusted each section to achieve the perfect balance of sound for the upcoming performance.	long	neutral	progressive	declarative	third_person	organization

Table 3: Example sentences in language understanding dataset, each sentence is annotated with 6 attributes.

A.3 MULTIMODALITY TASKS

For evaluating TDNV on multimodality tasks, we require each query image to be assessed across multiple attributes. To enable this, we construct a synthetic vision-text dataset in which every image can be evaluated on four distinct attributes: color, shape, size and texture. Each attribute is associated with several categorical labels, all of which are summarized in Table 4. The dataset contains 300 samples, and representative examples of these images are provided in Figure 17.

Attribute	Labels
Color	red, green, blue, yellow, black
Shape	circle, square, triangle, pentagon, star
Size	small, medium, large
Texture	solid, stripes, dots, checker

Table 4: The attributes and labels in multimodality dataset.






image	color	shape	size	texture
	yellow	triangle	small	solid
	blue	square	large	dots
	red	star	large	solid
	black	pentagon	large	checker
	green	circle	small	stripes

Figure 17: Example images in multimodality dataset, each image is annotated with 4 attributes.

B RELATED WORKS

Layerwise Representations An intriguing line of research (Ben-Shaul & Dekel, 2022; Fang et al., 2021; Wang et al., 2023b; Rangamani et al., 2023; He & Su, 2024; Zhou et al., 2025) has empirically investigated the role of different layers in feature learning. These studies show that in image classification tasks, features in intermediate layers become increasingly linearly separable as the layers deepen. Specifically, Neural Collapse (\mathcal{NC}) properties emerge in intermediate layers, where the within-class variance decreases compared to the between-class variance as depth increases. This indicates that layerwise compression occurs **monotonically** with layer depth in these settings. However, our hypothesis reveals that in the ICL setting, decoder-only models’ layerwise representations exhibit **dual** encoding-decoding stages: shallow layers compress information while deep layers express it. Furthermore, research by (Skean et al., 2025) shows that intermediate layers consistently outperform both shallow and final layers on downstream tasks. Our research focuses on how intermediate layers achieve maximum information compression in the ICL setting.

In-Context Learning Interpretability Numerous studies have focused on the working mechanisms of ICL (Xie et al., 2021; Chen et al., 2021; Von Oswald et al., 2023; Dai et al., 2022; Ahn et al., 2023; Olsson et al., 2022). Xie et al. (2021) propose that ICL emerges as an implicit form of Bayesian inference. In the realm of meta-learning, Chen et al. (2021) introduce in-context tuning to predict target labels from concatenated sequences of task instructions. A significant line of research connects ICL with gradient descent optimization. Von Oswald et al. (2023) demonstrate that Transformers trained on autoregressive tasks can emulate gradient descent in their forward pass. Dai et al. (2022); Ahn et al. (2023) compare standard gradient descent-based fine-tuning with ICL, revealing that transformer attention in ICL exhibits a dual form of gradient descent-based optimization. Olsson et al. (2022) identify “induction heads”—specific attention heads that facilitate ICL by copying patterns from previous tokens. However, our work focuses on the layer-wise representational analysis of ICL. In addition, Doimo et al. (2024) also reveals different clustering patterns in shallow and deep layers. However, we observed a strikingly different phenomenon and would like to clarify the differences in settings and results. Doimo et al. (2024) uses demonstrations with minimal task cues while the query itself provides ample task information, enabling high zero-shot accuracy and only modest few-shot gains. Our tasks encode the task only in the demonstration pairs, while the query provides no clues, yielding near-random zero-shot performance. As a result, our work shows that the model progressively compresses task information from the demonstration pairs, reaching maximal compression around the intermediate layers. In contrast, Doimo et al. (2024) clusters representations by semantic subject, with ARI peaking in the earliest layers.

Task Representations Researchers have explored various ways to extract compact representations of different ICL tasks, including task vectors (Hendel et al., 2023), function vectors (Todd et al., 2023), and state vectors (Li et al., 2024). Task vectors (Hendel et al., 2023) are extracted from the intermediate hidden state of the final separate token. Function vectors (Todd et al., 2023) are derived from attention activation through causal mediation, while state vectors (Li et al., 2024) concatenate activations from several initial layers of transformers. These representations effectively enable models to perform ICL tasks by injecting to specific transformer layers’ hidden states during inference. Some researchers have explored task manipulation within in-context learning. For instance, Shao et al. (2023) has demonstrated that compositional task representations can be created through composition model training. Additionally, In-context vectors (Liu et al., 2023b) enhance ICL through latent space steering. However, previous works have mainly focused on task representations for individual tasks, and none have provided a layer-wise analysis of task vectors. Our research examines how the model distinguishes between different tasks from a geometric perspective across shallow to deep layers.

C ADDITIONAL EXPERIMENTS

C.1 UNIVERSAL OF LAYERWISE COMPRESSION-EXPRESSION ACROSS DIFFERENT TASKS

In Figure 1, we validate the *Layerwise Compression-Expression* phenomenon across layers on Letter-to-Letter tasks. To further assess its generality, we evaluate the phenomenon on the other algorithmic task groups: (i) List-to-Element tasks (list-first, list-last, list-length, list-first-upper, list-last-upper); (ii) A combination of Letter-to-Letter and List-to-Element tasks.

We use the DeepSeek-Coder-7B model under a 15-shot ICL setting. As shown in Figure 18, the TDNV exhibits a U-shaped curve across layers in both settings, and both task vector accuracy and early-exit accuracy follow patterns similar to those observed in Figure 1. Notably, Figure 18(b) presents results for the combined task groups listed in Table 1, further supporting the conclusion that this phenomenon holds broadly across diverse tasks. These findings further confirm the universality of the *Layerwise Compression-Expression*.

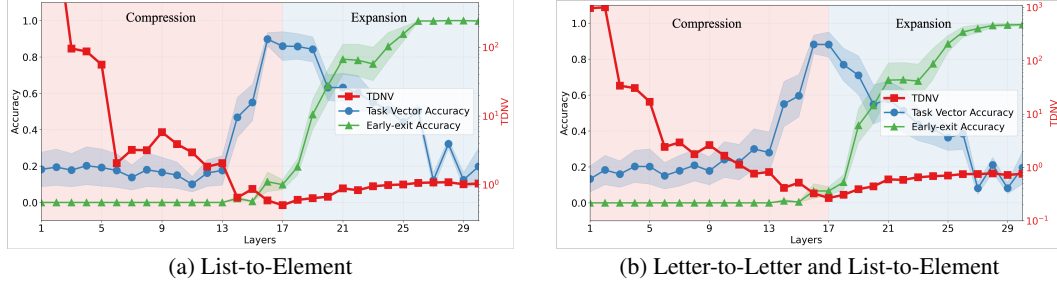


Figure 18: *Layerwise Compression-Expression* phenomenon across different task groups: a) List-to-Element, b) a combination of two task groups, the Letter-to-Letter and List-to-Element. TDNV first decreases then increases from shallow to deep layers, splitting the model into compression and expression stages.

C.2 ALTERNATIVE REPRESENTATIONS FOR TASK VECTOR

We choose the last separator token as the ICL representation following prior work (Hendel et al., 2023), where it serves as a natural anchor point between the demonstrations and the query. For comparison, we evaluate two other aggregation strategies: (i) Mean of All Tokens: the mean of all demonstration token representations (remove the last query and separator), and (ii) Mean of All Separator Tokens: the mean of all separator token representations. To quantitatively evaluate which representation best captures layerwise features, we compare the TDNV changes during both compression and expression stages:

$$\Delta_{\text{Compression}} = \frac{\text{TDNV}_0 - \text{TDNV}_{\ell_{\min}}}{\text{TDNV}_0} \quad \text{and} \quad \Delta_{\text{Expression}} = \frac{\text{TDNV}_L - \text{TDNV}_{\ell_{\min}}}{\text{TDNV}_L}.$$

Where TDNV_0 and TDNV_L are the TDNV of the first and last layer, and $\text{TDNV}_{\ell_{\min}}$ is the minimum TDNV.

Representation	Compression Ratio \uparrow	Expression Ratio \uparrow
Mean All Tokens	0.9671	0.6002
Mean Sep Tokens	0.9879	0.5448
Last Sep Token	0.9926	0.7746

Table 5: Compression and expression ratios for different representations.

As shown Table 5, the Last Sep Token representation demonstrates both higher compression and expression ratios. This evidence supports our conclusion that the last separator token remains the optimal choice for capturing task-relevant information.

The other two alternatives are suboptimal for the following reasons:

- **Mean of All Tokens.** As shown in Figure 20, saliency maps reveal that token contributions to task representation are highly uneven. Early layers focus on label tokens within the demonstrations, while later layers shift attention to the final separator token as the primary carrier of task information. Averaging across all tokens therefore introduces noise from irrelevant content tokens and dilutes the in-context learning (ICL) signal.
- **Mean of All Separator Tokens.** As illustrated in Figure 21, grid-level TDNV analysis across separator tokens shows a monotonic decrease in TDNV from the first to the last

separator. This pattern indicates that the model progressively compresses task-relevant information across successive demonstrations. Because later separators encode richer context and stronger compression, averaging over all separators weakens this effect, whereas using only the final separator captures the fully accumulated task representation.

D TOKEN LEVEL ANALYSIS

D.1 SALIENCY MAPS

In the main pages, we have extensively explored the use of TDNV to quantify layerwise information compression during ICL, revealing important statistical properties of model representations. In this subsection, we complement the understanding from a fine-grained token level. In particular, we will use the method of saliency maps (Wang et al., 2023a), specifically elucidating which parts of the input significantly contribute to the model’s decision-making. By highlighting critical token interactions, saliency maps provide intuitive insights into model behavior. Denoting by I_ℓ saliency map at the ℓ -th layer, we compute it by,

$$I_\ell = \left| \sum_h A_{h,\ell} \odot \frac{\partial \mathcal{L}}{\partial A_{h,\ell}} \right|, \quad (7)$$

where $A_{h,\ell}$ represents the attention map of head h at layer ℓ , and the loss \mathcal{L} is the cross-entropy calculated between the logits of the last token and the ground-truth label. Thus, a saliency map quantifies the importance of internal activations by integrating both attention strength and its gradient-based influence on the model’s outcome. In a nutshell, these maps highlight how token interactions evolve across layers.

We show saliency maps of all layers using three demonstrations in Figure 19. In shallow layers, there is more interaction within demonstrations, indicating that the model extracts task information from each demonstration. In deep layers, there is less interaction within demonstrations and more interaction with the last token, indicating that the model uses the accumulated task information to generate output.

To observe more clear patterns, we group the layers into 3 groups: shallow (average of layers 1-14), intermediate (average of layers 15-16), and deep (layers 17 and above) in Figure 20. We observe that (i) in shallow layers, the model focuses on token-to-label mappings within demonstrations, reflecting local compression, (ii) intermediate layers shift focus toward the final token, indicating integration of task-level information, and (iii) deep layers emphasize interactions between the query and final token, aligning with expression and prediction. Thus, token-level interpretability also aligns with the layerwise compression-expression trajectory.

D.2 GRID-TDNV

In Section 4.3, we find that the position of noisy demonstration affects compression. To investigate why perturbing later demonstrations results in higher TDNV at the final separation token, we conduct a more fine-grained token-level analysis by computing the layerwise TDNV for each separation token across all demonstrations. For each demonstration’s separator, we calculate the layerwise TDNV and organize them into a grid structure, referred to as the grid TDNV. As shown in Figure 21, perturbing a demonstration at a given position most significantly increases the TDNV of the next demonstration. However, this increase gradually diminishes as more correct demonstrations are appended. This pattern suggests that the negative impact of early errors can be partially mitigated by subsequent correct examples.

E PROOF OF THEOREM 1

Proof. (Proof for Variance Decay.) Consider one layer linear attention (Von Oswald et al., 2023; Ahn et al., 2023; Wang et al., 2024; Li et al., 2024) that preserves the normalization property of softmax attention (Katharopoulos et al., 2020; Shen et al., 2021): the hidden state of the query token

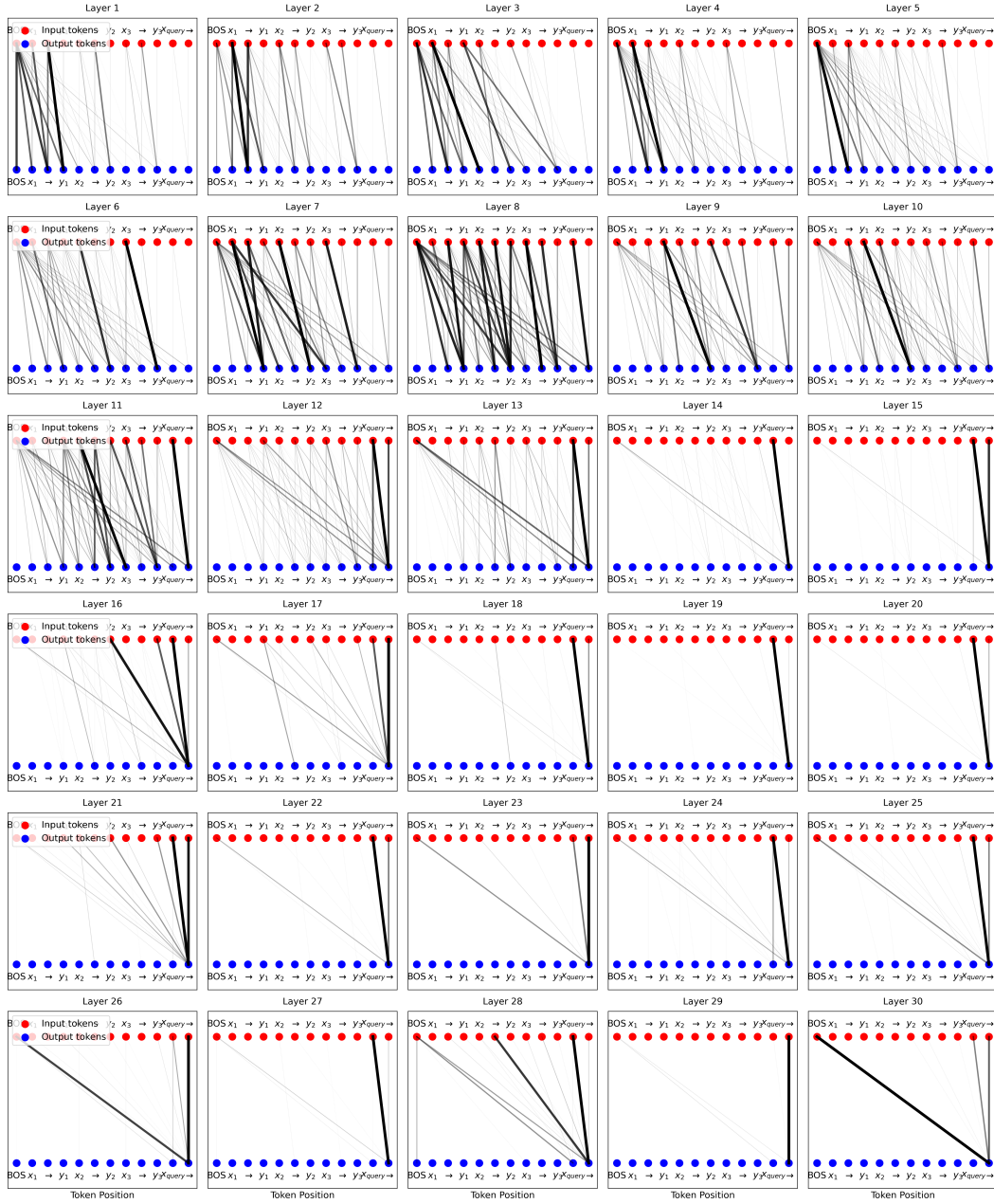


Figure 19: Saliency Maps for all layers.

becomes

$$\begin{aligned}
 h'_q(K) &= [\text{Attn}(h_1, \dots, h_K, h_q)]_{:,K+1} \\
 &= \sum_{i=1}^K \frac{\phi(q_q)^\top \phi(k_i)}{K+1} v_i + \frac{\phi(q_q)^\top \phi(k_q)}{K+1} v_q,
 \end{aligned} \tag{8}$$

where $q = W^Q h$, $k = W^K h$, $v = W^V h$ are the query, key and value vectors, respectively, and $\phi : \mathbb{R}^d \rightarrow \mathbb{R}^r$ denotes the feature map that approximates the softmax. Define $z_j := \phi(q_q)^\top \phi(k_j)$, $z_q := \phi(q_q)^\top \phi(k_q)$, we can rewrite it as

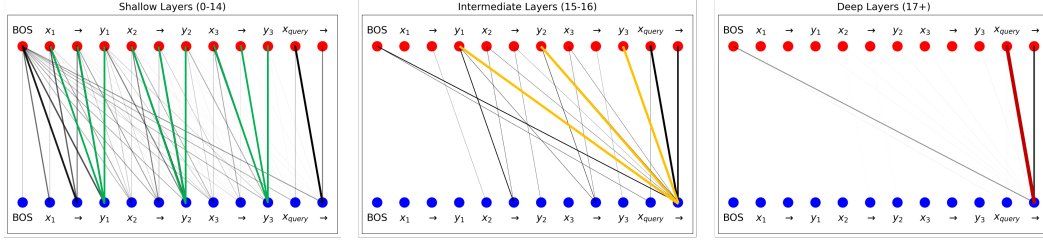


Figure 20: Saliency maps across transformer layers: (a) shallow, (b) intermediate, and (c) deep. The edge widths indicate saliency magnitude from input tokens (red dots) to output tokens (blue dots).

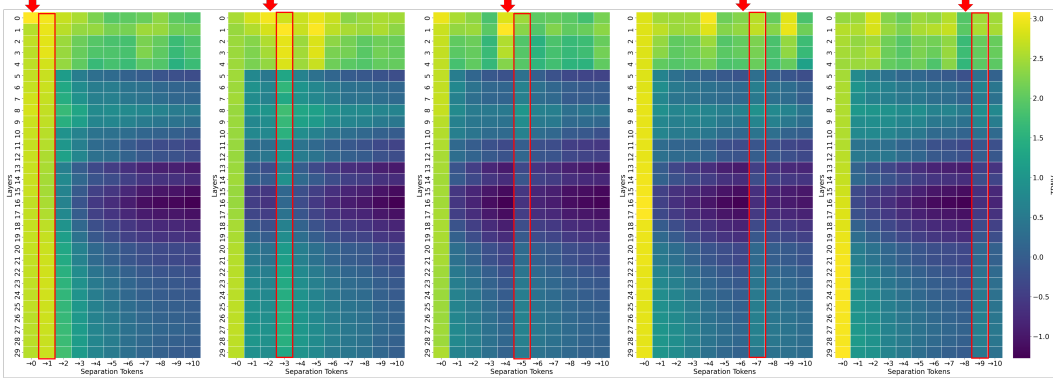


Figure 21: The grid TDNV pattern when perturbing one demonstration at different positions: from left to right, the perturbation is applied to the 0th, 2nd, 4th, 6th, and 8th demonstration.

$$h'_q(K) = \sum_{i=1}^K \frac{z_i}{K+1} v_i + \frac{z_q}{K+1} v_q. \quad (9)$$

To further simplify the notation, we define $\mathbf{a}_i := z_i \mathbf{v}_i$, $i = 1, \dots, K$, and $\mathbf{a}_{K+1} := z_q \mathbf{v}_q$, which gives

$$h'_q(K) = \frac{1}{K+1} \sum_{i=1}^{K+1} \mathbf{a}_i. \quad (10)$$

Since $\mathbf{a}_i := \phi(\mathbf{q}_q)^\top \phi(\mathbf{W}^K \mathbf{h}_i) \mathbf{W}^V \mathbf{h}_i$ only relates to \mathbf{h}_i and \mathbf{h}_i are i.i.d., the \mathbf{a}_i are also i.i.d. with

$$\mathbb{E}[\mathbf{a}_i] = \boldsymbol{\mu}_a, \quad \text{Cov}(\mathbf{a}_i) = \boldsymbol{\Sigma}_a.$$

Finally, we get,

$$\text{Var}(\|h'_q\|_2^2) = \frac{1}{(K+1)^2} \sum_{i=1}^{K+1} \text{Var}(\|\mathbf{a}_i\|_2^2) = \frac{K+1}{(K+1)^2} \text{Tr}(\boldsymbol{\Sigma}_a) = \frac{\text{Tr}(\boldsymbol{\Sigma}_a)}{K+1} \sim \mathcal{O}(1/K). \quad (11)$$

□

Proof. (Proof for Mean Shift.) Define the mean of demonstrations and the zero-shot as

$$\boldsymbol{\mu}_{\text{zero-shot}} := \mathbb{E}[z_q \mathbf{v}_q], \quad \boldsymbol{\mu}_{\text{demo}} := \mathbb{E}[z_i \mathbf{v}_i].$$

Then according to (9), we have

$$\mathbb{E}[h'_q(K)] = \frac{1}{K+1} \underbrace{\mathbb{E}[z_q \mathbf{v}_q]}_{=: \boldsymbol{\mu}_{\text{zero-shot}}} + \frac{K}{K+1} \underbrace{\mathbb{E}[z_i \mathbf{v}_i]}_{=: \boldsymbol{\mu}_{\text{demo}}}. \quad (12)$$

If $K = 0$, we get $\mathbb{E}[\mathbf{h}'_q(0)] = \mathbb{E}[z_q \mathbf{v}_q]$. When $K \rightarrow \infty$, we have $\mathbb{E}[\mathbf{h}'_q(\infty)] = \mathbb{E}[z_i \mathbf{v}_i]$.

In summary, we get,

$$\mathbb{E}[\mathbf{h}'_q(K)] = \lambda_K \mathbb{E}[\mathbf{h}'_q(0)] + (1 - \lambda_K) \mathbb{E}[\mathbf{h}'_q(\infty)] \quad (13)$$

where $\lambda_K = 1/(1 + K) \sim \mathcal{O}(1/K)$.

□

F VALIDATION OF THE I.I.D. ASSUMPTION UNDER REPITITION

In Theorem 1, we assume that each demonstration $\mathbf{h}_i, i = 1, \dots, K$ is i.i.d. randomly generated from a distribution \mathcal{H} on \mathbb{R}^d . To validate the importance of this assumption, we design experiments that increase the ICL length by providing additional demonstrations as input. There are two possible extension methods:

- Repeat Mode: Extending demonstrations by repeating existing examples.
- Distinct Mode: Extending demonstrations by adding new, unique examples.

In the Distinct Mode, new demonstrations are independently and identically distributed (i.i.d.), randomly generated from a distribution. In the Repeat Mode, demonstrations are no longer i.i.d. We examine both performance and compression level across these two modes.

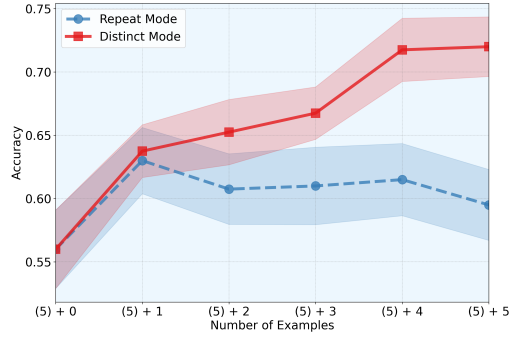


Figure 22: Comparison of accuracy under repeat mode v.s. distinct mode.

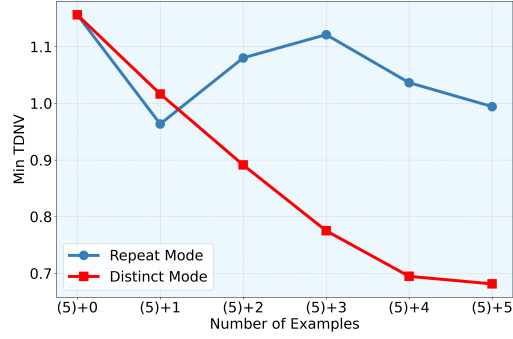


Figure 23: Minimum TDNV under repeat mode v.s. distinct mode.

As shown in Figure 22 and Figure 23, in distinct mode, as new demonstrations introduce novel information, the accuracy increases and TDNV decreases. However, in repeat mode, since repeated demonstrations add no new task information, the accuracy and TDNV remain largely unchanged. This proves that when the i.i.d. condition is violated, increasing the number of demonstrations does not lead to better performance and compression.

G TASK VECTOR ACCURACY & EARLY-EXIT ACCURACY

Task vector accuracy (Hendel et al., 2023) refers to how accurately a LLM can perform a task using only a learned representation of the task (task vector), instead of full in-context demonstrations. As shown in Figure 24, to evaluate the task vector accuracy, we conduct the following steps:

1. **Extract** task vector θ from the demonstration set \mathcal{S}_K using a dummy query x' , avoiding information leakage from the real query as,

$$\theta_{\text{task}} = [f_{\theta(1:\ell)}([\mathcal{S}_K, x'])]_{:,K+1} \quad (14)$$

2. **Inject** θ into a forward pass of the model with only the query x , not the full demonstration set, then predict the output using this modulated forward pass.

$$y = f_{\theta}([x]; \theta_{\text{task}}) \quad (15)$$

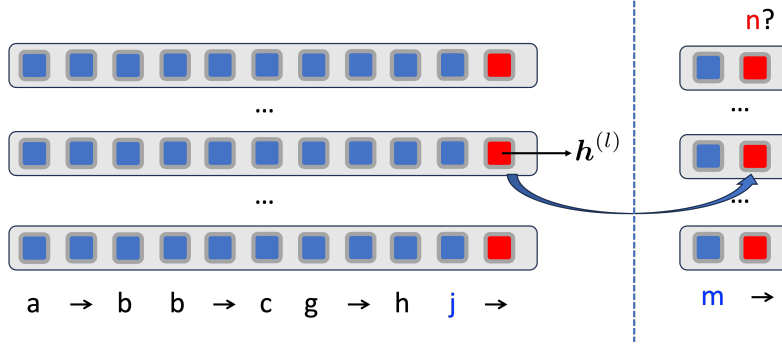


Figure 24: Illustration for task vector accuracy.

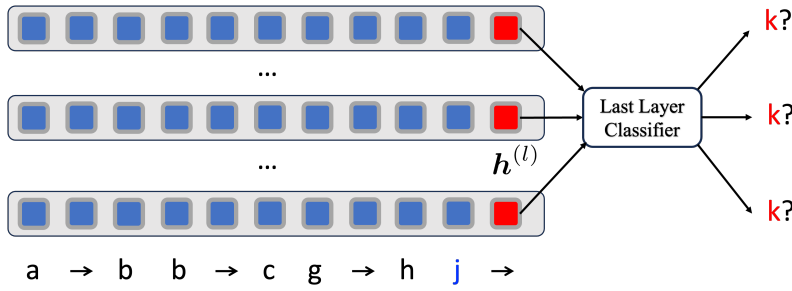


Figure 25: Illustration for early-exit accuracy.

where $[:, K+1]$ means the $(K+1)$ -th column, ℓ is the layer with most compression.

Early-exit accuracy (Xin et al., 2020; Jiang et al., 2024a) measures a model’s prediction accuracy when using intermediate-layer representations instead of the final layer for making predictions. This metric helps assess how effectively different layers encode the information needed to complete the task. Let $\mathbf{h}^{(\ell)} \in \mathbb{R}^d$ be the hidden state of the final token at layer ℓ , and let $\mathcal{C} : \mathbb{R}^d \rightarrow \mathbb{R}^V$ be the last-layer classifier mapping from hidden dimension d to vocabulary size V . Then as shown in Figure 25, the prediction is,

$$\hat{y}^{(\ell)} = \arg \max_{v \in \mathcal{V}} \text{softmax}(\mathcal{C}(\mathbf{h}^{(\ell)}))_v \quad (16)$$

Then, the early-exit accuracy at layer ℓ over N examples is,

$$\text{Acc}^{(\ell)} = \frac{1}{N} \sum_{i=1}^N \mathbf{1}[\hat{y}_i^{(\ell)} = y_i] \quad (17)$$

H TASK-VECTOR CONTRASTIVE FINE-TUNING

As shown in Figure 26, to obtain more compressed and discriminative task vectors, we add an additional contrastive loss on task vectors that explicitly encourages representation compression. Specifically, we fine-tuning a pretrained GPT-2 model on algorithmic ICL tasks domains (next letter, next two letters, previous letter, uppercase, next uppercase letter) using a combined objective as,

$$\mathcal{L} = \mathcal{L}_{\text{CE}} + \beta \mathcal{L}_{\text{Contra}}, \quad (18)$$

where \mathcal{L}_{CE} is the cross-entropy loss computed only on separator tokens to predict the correct label and $\mathcal{L}_{\text{Contra}}$ is a contrastive term that pulls hidden states from the same task closer while pushing

those from different tasks apart:

$$\mathcal{L}_{\text{Contra}} = -\frac{1}{|\Omega|} \sum_t \sum_{\substack{i,j \\ i \neq j}} \log \frac{\exp(\mathbf{h}_{i,t}^\top \mathbf{h}_{j,t} / \tau)}{\sum_{(a,b) \neq (i,t)} \exp(\mathbf{h}_{i,t}^\top \mathbf{h}_{a,b} / \tau)}. \quad (19)$$

where $\mathbf{h}_{i,t}$ denotes the normalized hidden state of the last token in the intermediate layer(task vector) for the i -th sample of task t , τ is the temperature, and Ω is the set of all pairs. We set $\beta = 0.1$, temperature $\tau = 0.07$, and batch size 100 with equal samples from each task. Each training example provides $K = 20$ context demonstrations followed by a separator token. We finetuned the model to achieve 100% ICL accuracy on all tasks.

For evaluation, we extract task vectors with a context length of $K = 20$ and assess task-vector accuracy: performing zero-shot ICL with the extracted task vector injected at the same position. Higher task-vector accuracy reflects more effective task vectors. As shown in Figure 16, task-vector contrastive fine-tuning produces better task vectors than standard fine-tuning.

To illustrate whether task-vector contrastive fine-tuning produces more compressed task vectors, we visualize hidden states from the 7-th layer using PCA. As shown in Figure 27, the left panel depicts task vectors extracted from model finetuned with only the cross-entropy loss, while the right panel shows vectors extracted from model finetuned with both cross-entropy and contrastive losses. The latter exhibits more distinct and tightly clustered task representations, demonstrating the effectiveness of the contrastive objective in compressing task vectors.

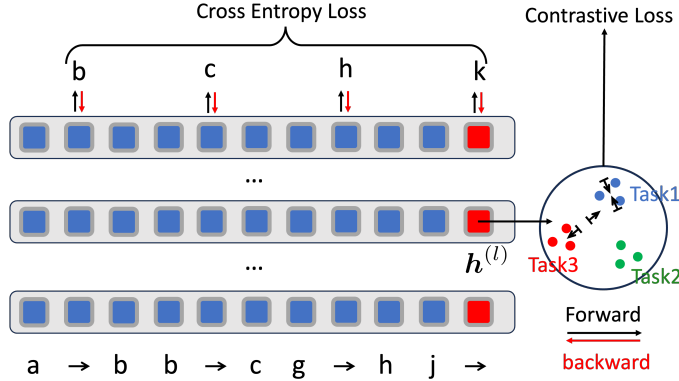


Figure 26: Illustration of task-vector contrastive fine-tuning.

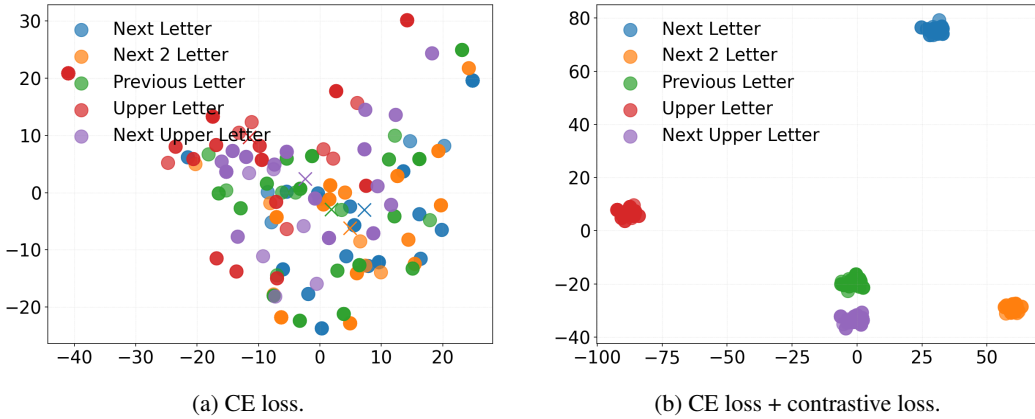


Figure 27: Comparison of PCA visualizations for task vectors. The task vector is extracted from models finetuned with different losses.