Label Words as Local Task Vectors in In-Context Learning

Anonymous ACL submission

Abstract

Large Language Models (LLMs) have demonstrated remarkable abilities, one of the most important being in-context learning (ICL). With ICL, LLMs can derive the underlying rule from a few demonstrations and provide answers that comply with the rule. Previous work hypothesized that the network creates a task vector in specific positions during ICL. The task vector can be computed by averaging across the dataset. It conveys the overall task information and can thus be considered global. Patching the global task vector allows LLMs to achieve zero-shot performance with dummy inputs comparable to few-shot learning. However, we find that such a global task vector does not exist in all tasks, especially in tasks that rely on rules that can only be inferred from multiple demonstrations, such as categorization tasks. Instead, the information provided by each demonstration is first transmitted to its answer position and forms a local task vector associated with the demonstration. In some tasks but not in categorization tasks, all demonstrations' local task vectors converge in later layers, forming the global task vector. We further show that local task vectors encode a high-level abstraction of rules extracted from the demonstrations. Our study provides novel insights into the mechanism underlying ICL in LLMs, demonstrating how ICL may be achieved through an information aggregation mechanism.

1 Introduction

The advent of Large Language Models (LLMs) has enabled machines to understand and generate human-like text with unprecedented accuracy. One of the most remarkable abilities of LLMs is in-context learning (ICL), where the model can abstract the underlying rule defined by a few demonstrations and provide answers that comply with that rule (Brown et al., 2020; Liu et al., 2023; Dong et al., 2023). This capability has garnered significant attention from the research community, as it demonstrates the flexibility of LLMs to adapt to new tasks without extensive training, which is a signature of human cognition (Binz and Schulz, 2023). Unlike prompt fine-tuning (Lester et al., 2021) or chain-of-thought prompting (Wei et al., 2022), ICL simply relies on several demonstrations that share the same structure as the question.

Previous mechanistic work suggests that task information in ICL can be represented as compact vectors localized at specific token positions and layers in the model. (Hendel et al., 2023; Wang et al., 2023a; Liu et al., 2024; Li et al., 2024). Specifically, the task vector is computed by averaging the embedding of the last token at a particular layer across samples from the same task dataset. Researchers have shown that by transferring the task vector to the corresponding positions with zero-shot dummy inputs, LLMs can achieve performance similar to few-shot learning. This task vector is therefore deemed to carry a full abstraction of the task. We term it the global task vector to distinguish it from the local task vector defined below.

While patching the global task vector works as expected in some tasks (e.g., knowledge tasks), we find that it does not work well in other tasks, particularly those in which the rule can only be inferred from multiple demonstrations (e.g. learning an arbitrary categorization boundary). This is consistent with recent large-scale evidence that complex ICL tasks may rely on multiple subtask-specific vectors rather than a single averaged task vector (Tikhonov et al., 2025). Instead, we find that patching the tokens at the *answer* positions of each demonstration to the corresponding positions of a dummy input sequence leads to higher accuracy in these situations. Inspired by the previous study that reveals the importance of label words in the information flow of ICL (Wang et al., 2023a), we propose that these label words serve as local task vectors that

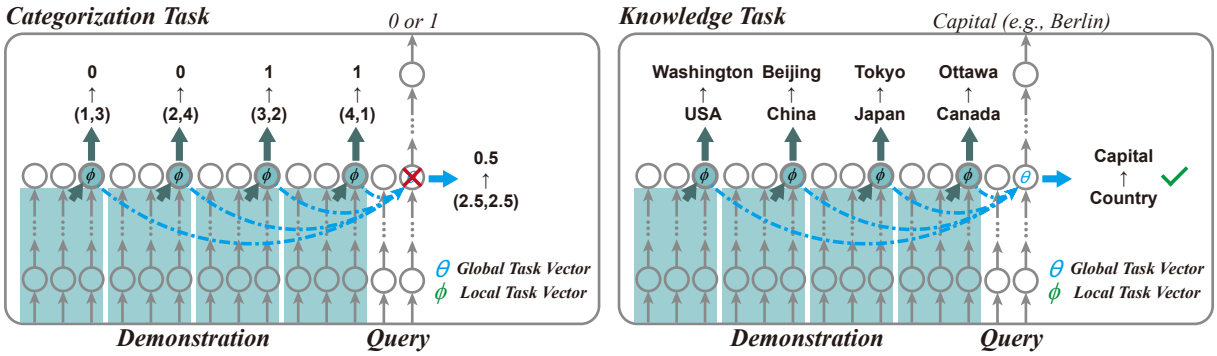

Figure 1. Illustration of local and global task vectors. Left: In a categorization task, local task vectors contain information associated with each demonstration, but they do not converge and form a global task vector. Right: In a knowledge task, local task vectors aggregate into a coherent global task vector that aligns with the LLM's prior knowledge, enabling effective task representation.

carry the task rule information in a distributed manner. In addition, we show that even in tasks where a global task vector exists, the local task vectors appear in earlier layers than the global task vector. The information contained in the local task vectors converges and forms the global task vectors in later layers.

Taken together, these results point to a simple picture of ICL: each demonstration first writes task-relevant information into the hidden state at its *answer* token, forming a *local task vector*, and the model answers the query by aggregating information from these local vectors. This picture also helps explain when global task vectors succeed: in knowledge tasks, local task vectors across demonstrations tend to become increasingly aligned in later layers, making an averaged *global* task vector effective, whereas in categorization tasks they can remain heterogeneous and complementary, so averaging may discard essential information.

2 Preliminaries

2.1 In-context Learning

We use T to denote a decoder-only transformer LLM, S to denote the set of demonstrations used as the inputs to the LLM, and x to denote the query that needs to be answered. We use $T([S, x])$ to denote the output of LLM on the concatenation of S and x. For clarity, we define $S = [Q_0 I_0 A_0 D_0, Q_1 I_1 A_1 D_1, ..., Q_J I_J A_J D_J]$ and $x = [Q_{J+1} I_{J+1}]$, where Q_j, I_j, A_j, and D_j are the tokens for the *query*, the *is*, the *answer* and the *dot* of demonstration j ($j = 1, ..., J$). For example, "cat is 0. monkeys is 1. dog is", each demonstration can be divided into four parts: Q, I, A, and D.

2.2 Global Task Vector

We follow the definition in (Hendel et al., 2023). Assume that ICL operates within a hypothesis space. This mode of operation can be defined as a learning algorithm (denoted $\mathcal{A}$). $\mathcal{A}$ maps S to *the global task vector* θ. Next, the LLM maps the query x to the output through rule application (denoted by f), based on $\theta \equiv \mathcal{A}(S)$, without direct dependence on S. Here, the global task vector θ is at position I_{J+1}. In our experiments, we consider different versions of θ and ϕ_j, including those extracted from a single trial and those averaged across samples. We refer to these as *No Avg* and *Trial Avg*, respectively; see Section A.1 and Table 2 for formal definitions. Consider the mapping from a set of demonstrations and a query to the predicted output:

$$T([S, x]) = f(x; \mathcal{A}(S)) \quad (1)$$

ICL can be viewed as operating on the following hypothesis class: $\mathcal{H} = \{f(\cdot; \theta) \mid \theta\}$.

2.3 Saliency Score

We use the saliency score (Michel et al., 2019; Simonyan et al., 2013; Wang et al., 2023a) to identify the level of attention that the LLM gives to each token in the input. The saliency score is defined as

$$I_l = \left| \sum_h A_{h,l} \odot \frac{\partial \mathcal{L}(x)}{\partial A_{h,l}} \right|, \quad (2)$$

Here, $A_{h,l}$ is the value of the attention matrix of the h-th attention head in the l-th layer, x is the input, and $\mathcal{L}(x)$ is the loss function of the task.

2.4 dPCA

Following (Kobak et al., 2016), we use demixed PCA (dPCA) to isolate the subspace of representations associated with a specific factor. In Sec. 4.2

we use dPCA to identify the subspace predictive of the demonstration's string length and remove it by projecting answer-position states to the null space of that subspace. We find a compressing matrix D and a decompressing matrix F by minimizing:

$$L_{dPCA} = ||X_L - FDX||^2 \quad (3)$$

where X is the $h \times n$ activity matrix, where h is the dimension of the hidden state, and X_L is a label average matrix. X_L has the same size as X, but all rows are replaced by the mean activity vector with the same label.

We then project the local task vectors onto the null space of the informative subspace:

$$B = Null(D) \quad (4)$$

$$\tilde{X} = B(B^T B)^{-1} B^T X \quad (5)$$

where $D \in \mathbb{R}^{d \times h}$ is the compression matrix, $Null(\cdot)$ is the null space computation so that $D \cdot Null(D) = 0$. $\tilde{X}$ is the manipulated activity matrix. The mean is subtracted before the operation and then added back when we put the matrix back into the model.

3 Tasks

We analyze two representative tasks in depth, and evaluate generality on additional tasks listed in Table 1. Throughout the paper, we distinguish **knowledge** tasks, where the input–output mapping is largely consistent with associations likely learned during pretraining (e.g., Country→Capital), from **rule-induction (categorization)** tasks, where the mapping is defined by the demonstrations in the prompt and must be inferred by aggregating multiple examples (e.g., length thresholding, 2-D boundaries).

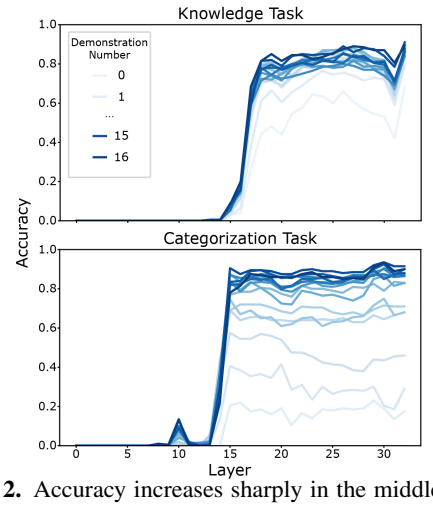

Figure 2. Accuracy increases sharply in the middle layers. The shades of blue indicate demonstration numbers. Top: knowledge task; Bottom: categorization task.

The first task is a country-capital knowledge task (Hendel et al., 2023; Wang et al., 2023a). The model should give the capital of a given country. For example, the prompt "China is Beijing. Japan is Tokyo. Germany is" should lead to the answer "Berlin". In this setup, the final *is* position (after "Germany") corresponds to the **global task vector** location, while each *answer* position (e.g., "Beijing", "Tokyo") serves as a **local task vector**.

Similarly, the second task is a two-alternative categorization task. In this task, the final *is* token before the model prediction is the global task vector position, while each number token (e.g., "0", "1") in the demonstrations serves as a local task vector.

Details of these roles are analyzed in the following sections. In the second task, the model is given a string of random characters, with the length varying between 1 and 10. The model should answer 0 when the string is shorter than 6 characters and 1 otherwise. An example question is "cat is 0. monkeys is 1. dog is". The number of demonstrations ranges from 1 to 16. We show that models are capable of solving these problems (Fig. 2). Consistent with previous work (Hendel et al., 2023), the accuracy increases sharply in the middle layers.

Table 5 includes results for more knowledge tasks and categorization tasks. Unless otherwise noted, we use 4 demonstrations for knowledge tasks and 8 demonstrations for categorization tasks, chosen to yield comparable baseline ICL accuracy and avoid ceiling effects.

Full details on models, tokenization, and evaluation protocols are deferred to Appendix A.1.

4 Local Task Vectors

4.1 The Importance of Answer Positions

Previous research (Hendel et al., 2023; Wang et al., 2023a) shows that the information of each demonstration converges at its respective *answer* position and is important in in-context learning. We further verify this result in the two representative tasks. Using layer 14 as a representative layer, the saliency scores in both tasks are the highest between the `answer`-position tokens of the demonstrations and the `is`-position token of the final query, suggesting that the task-related information from each demonstration is transmitted via the corresponding *answer* position's token to the token at the *is* position of the

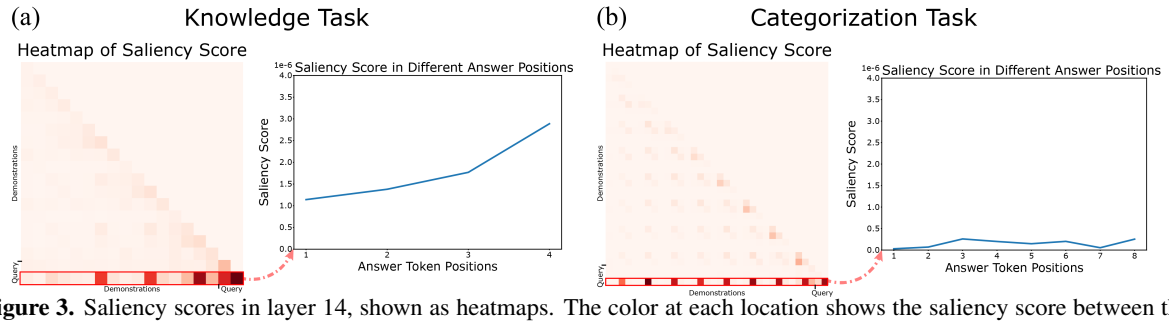

Figure 3. Saliency scores in layer 14, shown as heatmaps. The color at each location shows the saliency score between the positions in the respective row and column. The last row is where the highest scores are observed at the demonstrations' *answer* position, which is plotted on the right as a function of the demonstration index. (a) Knowledge task; (b) Categorization task.

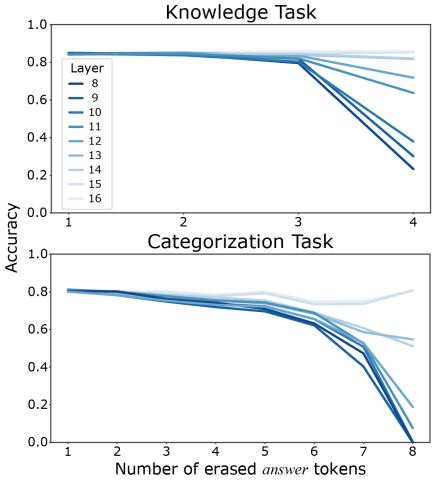

Figure 4. Model accuracy of ablated models. The shades of blue indicate the index of layers. Top: knowledge task; Bottom: categorization task.

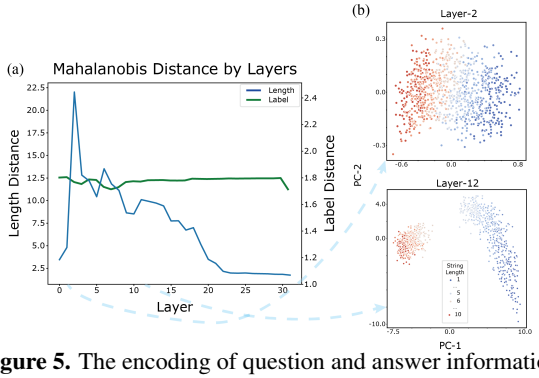

Figure 5. The encoding of question and answer information in *answer* positions. (a) Plotted is the Mahalanobis distance of the clusters defined by the demonstrations' string's length (blue) and by the corresponding answer (green) in the space defined by the two largest PCA components of the *answer* positions. Peaking at layer 2, the Mahalanobis distance for the string length gradually decreases across layers. (b) Two example layers' task spaces defined by the first two largest PCA components. Blue and red indicate answers *0* and *1*, and the color gradient indicates string length. Notice that the segregation between the two answers (blue and red color) is maintained across the layers, while the dots with the same color but different gradients are more mixed in later layers (e.g. layer 12) than in earlier layers (e.g. layer 2).

query to form the output (Fig. 3a). The *is* position token of the query is the global task vector.

In addition, we found that the saliency score is higher for later demonstrations in the knowledge task, suggesting further information aggregation to the *answer* positions of later demonstrations. However, such a trend is not found in the categorization task (Fig. 3b). Lesion experiments on the answer tokens also reveal the difference between the two tasks. While setting the tokens at the **answer** positions to zero or random values does not harm the model's performance unless all answer tokens are erased, the same manipulation leads to a gradual decrease in performance in the categorization task (Fig. 4). This difference suggests that the ICL mechanism may differ in the two tasks.

4.2 Answer Positions As Local Task Vectors

To verify that the information from each demonstration converges at its *answer* position, we use the principal component analysis (PCA) to explore the information encoded at the *answer* positions (Fig. 5). In the categorization task, each demonstration provides a string and its associated category label. For each layer, we collect the hidden states at the demonstrations' *answer* positions from 1000 samples and perform PCA on these vectors (Fig. 5). To quantify which factor is encoded, we group the answer-position vectors (i) by the string length (1–10) and (ii) by the label token (0/1), and compute the *average pairwise* Mahalanobis distance between group centroids in the space spanned by the top two PCs.

The results suggest that the clusters formed by strings with different lengths are well segregated, with the distance peaking at layer 2 and gradually decreasing. In comparison, the category label is encoded at the *answer* positions stably across layers.

To confirm that the string length information contained at the *answer* positions is critical for ICL, we carry out an experiment in which we selectively

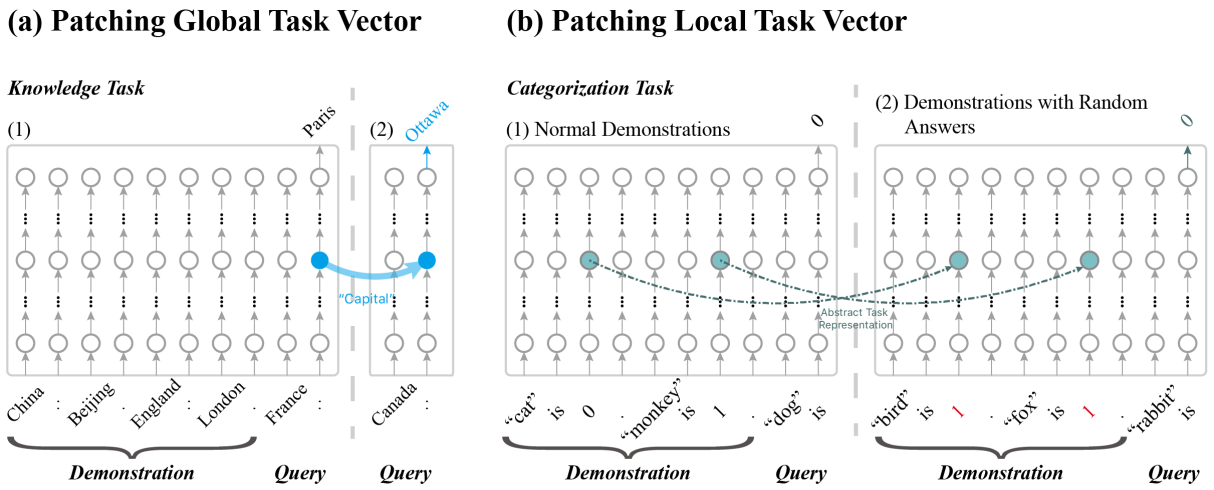

Figure 6. Patching experiments. (a) Patching global task vector. The global task vector, which is the token at the *is* position of the final query, in a network receiving normal demonstrations is copied to a network performing zero-shot inference. (b) Patching distributed local task vectors. The local task vectors, which are tokens at the *answer* position of each demonstration in an LLM receiving normal demonstrations are copied to an LLM receiving the same number of demonstrations but with random *answers*.

remove the string length information from the local task vectors. For each *answer*-position token, we first find its subspace that contains the corresponding demonstration's string length information with dPCA and then project the token onto the null space of the found subspace. We set the dimensionality of the discarded subspace to $d = 10$. This value strikes a balance between effectively isolating the string-length feature in early layers and preserving task-relevant abstract information in later layers. This procedure erases all information from the subspace that contains the string length information while keeping the rest of the information intact.

We apply this selective information removal at each layer (Fig. 7). Interestingly, the procedure leads to poor performance when applied in the early layers, where the string length is best encoded in the PCA space. However, starting from the middle layers, around layer 13, the ablation results in only a minimal decrease in performance. Nevertheless, in the following sections, we demonstrate that these layers encode sufficient task-relevant information for the model to perform the task effectively. These results suggest that in the later layers, the task information encoded in the answer tokens is a high-level abstraction of *query-answer* information that is not affected by selective information removal.

These results suggest that the *answer* position tokens contain task-related information provided by each demonstration and are critical for ICL. As they are local to each demonstration, we term them local task vectors.

5 Patching Local vs. Global Task Vectors

If the local task vectors contain sufficient information for solving the task, copying them to a dummy sequence should yield reasonable performance. Following the previous work (Hendel et al., 2023), we patch the local task vectors to a dummy sequence with those from a normal sequence. Here, the dummy input contains random labels and its demonstrations are different from the original sequence to prevent information leakage. More specifically, given an original sequence S and a dummy sequence D, we patch S_a^j to D_a^j as illustrated in Fig. 6b, where $*_a^j$ denotes the j-th *answer* position (or local task vector) of the sequence.

Table 1. Tasks used in our study. Full input–output examples are provided in Appendix Table 3.

Task Name	Task Rule
Categorization tasks ($y \in \{0, 1\}$)	
Simple String	$y = 1$ iff $\text{len}(s) > 5$.
Complex String	$y = 1$ iff $\text{len}(s) > 5$ (random chars).
Digit	$y = 1$ iff digit ≥ 5.
2-D Data	$y = 1$ iff $y \geq x$ for input (x, y).
Knowledge tasks	
Antonyms	Adjective → Antonym.
Capital	Country → Capital.
Language	Location → Language.
Profession	Person → Profession.
Religion	Person → Religion.

5.1 Accuracy Per Layer

For comparison, we also show the results from patching the global task vector. Given an original sequence S and a dummy sequence D, we patch

$\hat{S}_i^j$ to D_i^0 as illustrated in Fig. 6a, where $*_i^j$ denotes the j-th *is* position of the sequence and $\hat{S}_i^j$ is the **averaged** task vector across samples described in the Preliminaries. This dummy sequence is zero-shot, i.e., it contains no demonstrations. When calculating the accuracy of each layer, we apply the final layer normalization and the language model head to the hidden states in these layers as in (Hendel et al., 2023). Patching with the local task vectors and the global task vector has different effects in the two tasks.

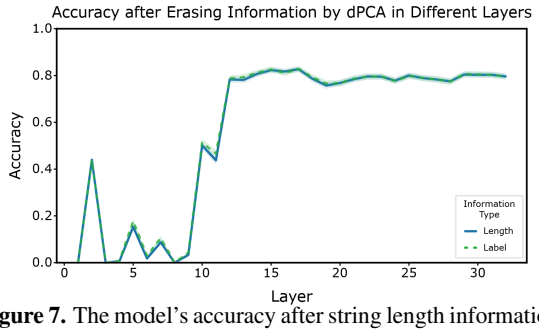

Figure 7. The model's accuracy after string length information in the local task vector is removed in the dPCA space. Note the ablation is only effective in early layers.

Patching with the global task vector restores the ICL performance with dummy inputs in the knowledge task. The layers where patching works the best coincide with the layers where we see the sharp increases in accuracy (Fig. 2). Global task vectors created with more demonstrations lead to higher accuracy (Fig. 9). With global task vectors based on three or more demonstrations, the zero-shot performance of the patched LLM is on par with that receiving the real demonstrations. However, the global task vector does not restore the performance with dummy inputs in the categorization task. The patched LLM performs near the chance level, which is 50% (Fig. 8). In these failed global-vector patching cases, the model often assigns highest probability to non-label tokens (i.e., outputs arbitrary tokens instead of {0, 1}), consistent with an insufficient task representation.

In contrast, patching with the local task vectors in the middle layers rescues the performance of the model in the categorization task (Fig. 8, bottom). The more demonstrations' local task vectors are patched, the better the performance is (Fig. 10). Interestingly, patching with the local task vectors also improves the LLM's performance in the knowledge task.

In addition, the global and local task vectors work differently at different layers (Fig. 8). We

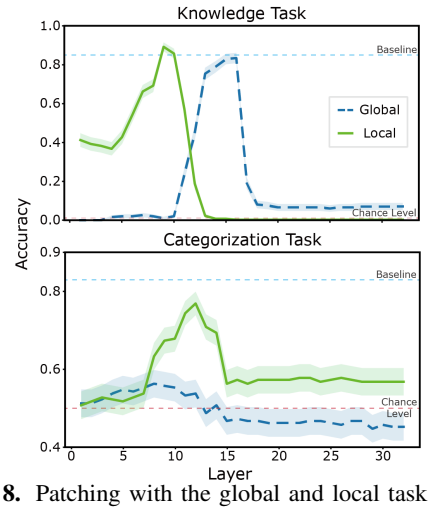

Figure 8. Patching with the global and local task vectors. The accuracy of the network patched with the *global task vector* and the *local task vectors* in different layers in the knowledge task (top) and the categorization task (bottom) is plotted against the layer number. Blue dashed lines indicate the baseline performance, which is the network receiving normal demonstrations. Red dashed lines indicate the chance level.

examine the LLM performance when we apply the patch at different layers and find that patching the local task vectors works best in earlier layers (6-10), and patching the global task vector works better in relatively later layers (12-15). This result supports the previous finding (Wang et al., 2023a) that the information contained within the local task vectors converges toward the global task vector in the later layers during the knowledge task. However, such translation of task information representation is not found in the categorization task (Fig. 8 Bottom).

We carry out further patching experiments on various knowledge-based tasks and categorization tasks and on different LLMs. Main results are summarized in Table 2. We additionally validate

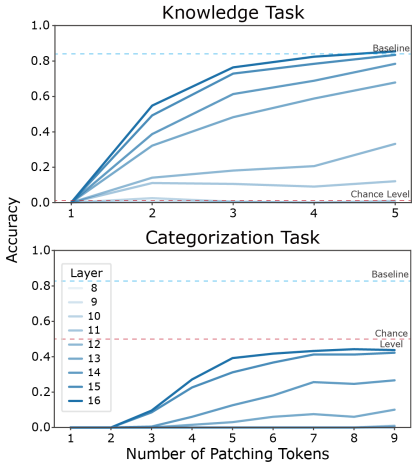

Figure 9. Patching the global task vectors in 8-16 layers. Top: knowledge task; Bottom: categorization task.

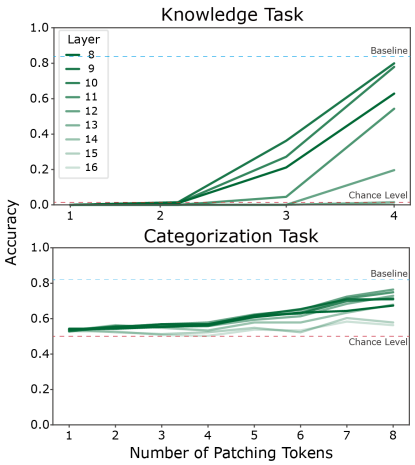

Figure 10. Patching the local task vectors in 8-16 layers. Top: knowledge task; Bottom: categorization task.

the findings on larger and different model families; see Table 4 and 5 in Appendix A.3. For knowledge-based tasks, we selected different types of factual tasks. The categorization tasks require multiple demonstrations to establish the correct association rules. We find that in some tasks, patching the global task vector to the dummy position leads to poor performance, especially in the categorization tasks. In cases where patching with the global task vector does not work well, patching with the local task vectors usually makes up for the performance.

Besides, patching with the global task vector is often more effective when we calculate the global task vector by averaging across the dataset, which makes it not just global for a particular question, but also global in the sense of the whole dataset. This method may extract the most accurate task information. However, in many categorization tasks, local task vectors often outperform the global task vector, even with averaging. Moreover, averaging across datasets is not feasible in normal inference. Therefore, our findings on the local task vector may unveil a more general mechanism underlying In-Context Learning.

Across different architectures, we observe a consistent trend where local task vectors outperform global ones in categorization tasks. Notably, Vicuna-7B shows strong performance with local vector patching in the Simple String and Digit tasks, highlighting the robustness of this mechanism across model families.

6 Limitations

Neither global nor local task vectors work consistently across all tasks and in all types of LLMs tested. We are yet to determine the mechanism causing this inconsistency. Although patching local task vectors works well in most tasks, it fails to restore the few-shot performance in a small number of models and tasks, and the reason for this has yet to be explored.

7 Conclusion

Our results reveal a more general mechanism involving distributed local task vectors that encapsulate the task information contained in the demonstrations during ICL. The local task vectors encode the task information in an abstract manner. Patching the local task vectors to models receiving dummy inputs yields performance levels comparable to few-shot learning. Notably, local task vectors are present in certain tasks where global task vectors are absent, particularly in categorization tasks. This suggests a more nuanced, distributed approach to processing information and extracting task rules in ICL.

8 Related Work

8.1 Neuroscience

Our work is inspired by studies from the field of neuroscience (Barrett et al., 2019; Richards et al., 2019; Yamins and DiCarlo, 2016; Yousefi et al., 2023) that explore the computational mechanisms underlying cognitive functions of the brain. The current study adopts a categorization task similar to what has been used widely in testing animals and humans in neuroscience. These experiments typically provide the demonstrations sequentially. Subjects learn the task through trial-and-error. This process is typically modeled as reinforcement learning (Niv, 2009). However, it has been shown that a system that stores behavior history could model the learning equally well without using explicit reinforcement learning (Zhang et al., 2018). This is conceptually similar to ICL in LLMs. The demixed-PCA (dPCA) (Kobak et al., 2016) used in our study is also initially developed to study how information is encoded in a high-dimensional space represented by population neuronal activities.

8.2 In-Context Learning

Brown et al. (2020) discovered that LLMs possess few-shot learning abilities, enabling them to perform reasoning across different tasks from just a few demonstrations. Meta-learning capabilities may be behind the models' capability to efficiently adapt to new information (Dai et al., 2023). It

Table 2. Best accuracy after patching global vs. local task vectors on LLAMA-7B across tasks. Baseline is standard ICL with real demonstrations; patched models receive dummy demonstrations. Averaging settings follow Sec. A.1. Results for other models are reported in Appendix Table 4 and Table 5. Results are averaged over 5 random trials; variances are small and omitted for brevity.

Model	Task Name	1-shot	Baseline	Global Task Vector		Local Task Vector		
				No Avg	Trial Avg	No Avg	Trial Avg	Pos Avg
LLaMA-7B	Simple String	0.37	0.79	0.49	0.63	**0.66**	0.58	**0.66**
	Complex String	0.57	0.80	0.55	0.56	**0.65**	0.61	0.39
	Digit	0.43	0.69	0.63	0.57	0.65	**0.68**	0.60
	2-D 8 Demo	0.00	0.59	0.53	0.49	**0.59**	0.52	0.38
	2-D 16 Demo	-	0.67	0.53	0.49	**0.55**	0.52	0.43
	Antonyms	0.56	0.86	0.75	0.79	**0.81**	0.80	0.00
	Capital	0.68	0.86	0.83	**0.92**	0.80	0.83	0.00
	Language	0.67	0.82	0.64	0.79	0.77	**0.85**	0.00
	Profession	0.35	0.54	0.32	**0.53**	0.48	0.51	0.00
	Religion	0.60	0.87	0.49	0.71	0.80	**0.92**	0.00

has been further proposed that transformers utilize gradient descent to perform linear regression tasks (Ahn et al., 2024; Akyürek et al., 2023; Von Oswald et al., 2023). Another approach to understanding ICL in LLMs is to view pre-trained models as implicit Bayesian models, and the demonstrations provided in the prompts allow the model to compute the posteriors (Ahuja et al., 2023; Wang et al., 2023b; Xie et al., 2022). Mechanistic analyses further link few-shot ICL to concrete circuits such as induction heads, whose ablation can substantially reduce the benefit from demonstrations (Crosbie and Shutova, 2025). Our work complements these circuit-level analyses by providing a spatial perspective on how information aggregates across different tokens.

8.3 Task Vector

It is proposed that LLMs might be either compressing abstract rule information from demonstrations (Hendel et al., 2023) or aggregating demonstration information layer-by-layer (Wang et al., 2023a). An example is to tell the capital of a country. We term these tasks knowledge tasks. Reference (Hendel et al., 2023) argued that ICL is achieved through a single task vector in the middle layers extracted from the demonstrations. Recent evidence also suggests that complex ICL may rely on multiple task-relevant components rather than a single averaged vector (Saglam et al., 2025; Tikhonov et al., 2025), which our local-vs-global analysis mechanistically contextualizes.

References

Kwangjun Ahn, Xiang Cheng, Hadi Daneshmand, and Suvrit Sra. 2024. Transformers learn to implement preconditioned gradient descent for in-context learning. *Advances in Neural Information Processing Systems*, 36.

Kabir Ahuja, Madhur Panwar, and Navin Goyal. 2023. In-context learning through the bayesian prism. *arXiv preprint arXiv:2306.04891*.

Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. 2023. What learning algorithm is in-context learning? investigations with linear models. *Preprint*, arxiv:2211.15661.

David GT Barrett, Ari S Morcos, and Jakob H Macke. 2019. Analyzing biological and artificial neural networks: challenges with opportunities for synergy? *Current opinion in neurobiology*, 55:55–64.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR.

Marcel Binz and Eric Schulz. 2023. Using cognitive psychology to understand gpt-3. *Proceedings of the National Academy of Sciences*, 120(6):e2218523120.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, and Jared Kaplan. 2020. Language models are few-shot learners. *NeurIPS*.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See https://vicuna. lmsys. org (accessed 14 April 2023)*, 2(3):6.

Joy Crosbie and Ekaterina Shutova. 2025. Induction heads as an essential mechanism for pattern matching in in-context learning. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 5034–5096.

Damai Dai, Yutao Sun, Li Dong, Yaru Hao, and Shuming Ma. 2023. Why can gpt learn in-context? language models implicitly perform gradient descent as meta-optimizers. *Preprint*, arxiv:2212.10559.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, and Zhiyong Wu. 2023. A survey on in-context learning. *Preprint*, arxiv:2301.00234.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407.

Roee Hendel, Mor Geva, and Amir Globerson. 2023. In-context learning creates task vectors. *Preprint*, arxiv:2310.15916.

Dmitry Kobak, Wieland Brendel, Christos Constantinidis, Claudia E Feierstein, Adam Kepecs, Zachary F Mainen, Xue-Lian Qi, Ranulfo Romo, Naoshige Uchida, and Christian K Machens. 2016. Demixed principal component analysis of neural population data. *elife*, 5:e10989.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

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

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35.

Sheng Liu, Haotian Ye, Lei Xing, and James Y. Zou. 2024. In-context vectors: Making in context learning more effective and controllable through latent space steering. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 32287–32307. PMLR.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? *Advances in neural information processing systems*, 32.

Yael Niv. 2009. Reinforcement learning in the brain. *Journal of Mathematical Psychology*, 53(3):139–154.

A Yang Qwen, Baosong Yang, B Zhang, B Hui, B Zheng, B Yu, Chengpeng Li, D Liu, F Huang, H Wei, et al. 2024. Qwen2. 5 technical report. *arXiv preprint*.

Blake A Richards, Timothy P Lillicrap, Philippe Beaudoin, Yoshua Bengio, Rafal Bogacz, Amelia Christensen, Claudia Clopath, Rui Ponte Costa, Archy de Berker, Surya Ganguli, et al. 2019. A deep learning framework for neuroscience. *Nature neuroscience*, 22(11):1761–1770.

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

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*.

Pavel Tikhonov, Ivan Oseledets, and Elena Tutubalina. 2025. One task vector is not enough: A large-scale study for in-context learning. *arXiv preprint arXiv:2505.23911*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Johannes Von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. 2023. Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, pages 35151–35174. PMLR.

Lean Wang, Lei Li, Damai Dai, Deli Chen, and Hao Zhou. 2023a. Label words are anchors: An information flow perspective for understanding in-context learning. *Preprint*, arxiv:2305.14160.

Xinyi Wang, Wanrong Zhu, Michael Saxon, Mark Steyvers, and William Yang Wang. 2023b. Large language models are implicitly topic models: Explaining and finding good demonstrations for in-context learning. In *Workshop on Efficient Systems for Foundation Models@ ICML2023*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. 2022. An explanation of in-context learning as implicit bayesian inference. *Preprint*, arxiv:2111.02080.

Daniel LK Yamins and James J DiCarlo. 2016. Using goal-driven deep learning models to understand sensory cortex. *Nature neuroscience*, 19(3):356–365.

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

Zhewei Zhang, Zhenbo Cheng, Zhongqiao Lin, Chechang Nie, and Tianming Yang. 2018. A neural network model for the orbitofrontal cortex and task space acquisition during reinforcement learning. *PLOS Computational Biology*, 14(1):e1005925.

A Appendix: Supplemental Material

A.1 Analysis Details

Models and Datasets We employ the LLaMA from HuggingFace (Touvron et al., 2023) as our primary model, and the model weights are also sourced from HuggingFace. No further processing on the weights is performed.

We also use the Pythia (Biderman et al., 2023), Vicuna (Chiang et al., 2023), LLaMA 3 (Dubey et al., 2024) and Qwen 2.5 (Qwen et al., 2024) models. Results on LLaMA-7B are reported in Table 2; results on other models are reported in Tables 4 and 5.

The task formats listed in Table 1 follow existing task definitions used in prior works such as Hendel et al. (2023) and Wang et al. (2023a). These references provide detailed descriptions for both factual knowledge and synthetic categorization tasks.

Tokenization Our experiments require us to retrieve tokens at specific positions. However, the tokenizer used with LLaMA generates variable numbers of tokens even for strings of the same length. Therefore, we insert a special character, e.g., "∼", into the original input text to force the tokenizer to generate the same number of tokens for strings of the same length. We remove the token of these special characters after the tokenization step.

We verified that the inclusion of the special character "∼" for length normalization does not introduce inductive biases or alter model performance. Inference results with and without "∼" are identical across all benchmarked tasks, confirming the robustness of this strategy.

Evaluation Details We use 5000 samples for calculating network performance. PCA and dPCA analyses are done with 1000 samples. More samples produce similar results. All experiments are done with a single V100 GPU. Accuracy is evaluated with the token with the largest logit without temperature or any other probabilistic sampling methods.For binary categorization tasks, a prediction is counted as correct only if the top-1 token matches the label token (`0` or `1`); outputs outside {`0`, `1`} are counted as incorrect.

To ensure the reliability of our results, all experiments were conducted across 5 random seeds for demonstration selection and ordering. We observed minimal variance (standard deviation < 0.01 for most tasks), and the performance gaps between local and global patching are statistically significant ($p < 0.05$ under a paired t-test).

We denote the three types of averaging strategies in Table 2 as follows: **No Avg** indicates that the result is from a single trial without averaging across runs. **Trial Avg** refers to averaging results of the same position across multiple trials. **Pos Avg** represents averaging across different positions within a single trial. These settings allow us to assess the consistency and locality of task vector effectiveness.

A.2 Ablation at Answer Position

In Figure 4, we perform an ablation experiment by setting the tokens at the answer positions of the demonstrations to zero. For the knowledge task, the ablation of the answer tokens across all demonstrations is necessary for a significant effect. In contrast, for the categorization task, the performance decreases gradually as the number of ablated answer tokens increases, suggesting a distributed mechanism. This gradual decline indicates that task information is spread across multiple tokens, highlighting the robustness of the model in categorization tasks even when partial information is removed.

Additionally, we performed preliminary tests by shuffling the order of demonstrations within the prompt. We observed that demonstration ordering had minimal impact on the qualitative trends reported in Section 5 (Fig. 8, 9 and 10), further supporting the distributed nature of the identified local task vectors.

A.3 Table of Experiment

We conduct additional experiments across several models and tasks and report the performance both with and without averaging across samples. For the local task vectors, we perform extra experiments by averaging across the demonstration, which leads to a poorer performance. The result suggests that the information contained in different local task vectors cannot be simply averaged. It further supports the distributed nature of these vectors.

A.4 Licenses for existing assets

We follow the license of LLaMA (LLaMA License), Pythia (Pythia License) and Vicuna (Vicuna License) and Apache License 2.0 for HuggingFace.

Table 3. Full task formats with input–output examples (supplement to Table 1).

Task Name	Task Rule	Example
Categorization tasks ($y \in \{0, 1\}$)		
Simple String	1 if len> 5 else 0	abaabb→1, aab→0
Complex String	1 if len> 5 else 0	aFeXGb→1, axvb→0
Digit	1 if digit$\geq$5 else 0	4→0, 7→1
2-D Data	1 if $y \geq x$ else 0	(0,1)→1, (7,4)→0
Knowledge tasks		
Antonyms	Given an English adjective, output an antonym	Adjective→Antonym
Capital	Given a country name, output its capital city	Country→Capital
Language	Given a location name, output its native language	Location→Language
Profession	Given a person name, output their profession	Person→Profession
Religion	Given a person name, output the associated religion	Person→Religion

Table 4. (continued) Results on other 7B-scale models.

Model	Task Name	1-shot	Baseline	Global Task Vector		Local Task Vector		
				No Avg	Trial Avg	No Avg	Trial Avg	Pos Avg
Pythia-6.9B	Simple String	0.37	0.80	0.47	0.59	0.58	**0.61**	0.00
	Complex String	0.52	0.69	0.53	**0.55**	0.49	0.51	0.00
	Digit	0.55	0.66	0.58	0.55	**0.60**	0.52	0.00
	2-D 8 Demo	0.03	0.63	0.55	0.50	**0.59**	0.53	0.00
	2-D 16 Demo	-	0.71	0.55	0.50	**0.59**	**0.59**	0.00
	Antonyms	0.26	0.84	0.75	0.80	**0.81**	0.80	0.01
	Capital	0.68	0.81	0.79	**0.85**	0.73	0.83	0.01
	Language	0.45	0.66	0.62	**0.69**	0.51	0.68	0.00
	Profession	0.2	0.23	0.27	**0.47**	0.33	0.39	0.00
	Religion	0.60	0.84	0.49	**0.87**	0.83	0.85	0.00
Vicuna-7B	Simple String	0.37	0.71	0.48	0.63	0.67	0.58	**0.70**
	Complex String	0.57	0.71	0.59	0.56	**0.62**	0.58	0.09
	Digit	0.55	0.71	0.64	**0.78**	0.64	0.66	0.17
	2-D 8 Demo	0.00	0.57	0.59	**0.80**	0.57	0.52	0.03
	2-D 16 Demo	-	0.58	0.59	**0.80**	0.55	0.55	0.03
	Antonyms	0.47	0.83	0.56	0.77	**0.84**	0.83	0.00
	Capital	0.63	0.87	0.76	**0.92**	0.77	0.85	0.00
	Language	0.5	0.78	0.49	0.70	0.75	**0.83**	0.00
	Profession	0.32	0.44	0.28	**0.53**	0.45	0.52	0.00
	Religion	0.72	0.88	0.50	**0.94**	0.86	0.91	0.00

Table 4. (continued) Results on other 7B-scale models.

Model	Task Name	1-shot	Baseline	Global Task Vector		Local Task Vector		
				No Avg	Trial Avg	No Avg	Trial Avg	Pos Avg
LLaMA3-8B	Simple String	0.42	0.83	0.63	0.65	**0.72**	0.66	0.60
	Complex String	0.60	0.84	0.60	0.62	**0.70**	0.64	0.55
	Digit	0.50	0.75	0.60	**0.68**	**0.68**	0.62	0.58
	2-D 8 Demo	0.04	0.66	0.52	0.54	**0.61**	0.56	0.45
	2-D 16 Demo	-	0.73	0.54	0.56	**0.66**	0.60	0.48
	Antonyms	0.60	0.90	0.86	0.85	**0.88**	0.87	0.01
	Capital	0.72	0.92	0.90	**0.91**	0.88	0.89	0.00
	Language	0.70	0.88	0.83	0.84	**0.86**	0.85	0.00
	Profession	0.40	0.62	0.56	**0.59**	0.58	0.58	0.00
	Religion	0.65	0.91	0.87	0.89	0.88	**0.90**	0.00
Qwen2.5-7B	Simple String	0.46	0.85	0.66	0.67	**0.74**	0.68	0.62
	Complex String	0.62	0.86	0.62	0.63	**0.72**	0.66	0.56
	Digit	0.52	0.78	0.61	0.63	**0.71**	0.64	0.60
	2-D 8 Demo	0.06	0.68	0.54	**0.63**	0.62	0.57	0.46
	2-D 16 Demo	-	0.75	0.56	0.57	**0.67**	0.61	0.50
	Antonyms	0.62	0.91	0.88	**0.90**	0.87	0.89	0.01
	Capital	0.75	0.93	0.91	**0.92**	0.89	0.90	0.00
	Language	0.72	0.90	0.86	0.88	0.87	**0.89**	0.00
	Profession	0.42	0.65	0.60	**0.62**	0.59	0.61	0.00
	Religion	0.68	0.92	0.88	0.90	0.89	**0.91**	0.00

Table 5. Additional results on other model sizes (same setup as Table 2).

Model	Task Name	1-shot	Baseline	Global Task Vector		Local Task Vector		
				No Avg	Trial Avg	No Avg	Trial Avg	Pos Avg
LLaMA-13B	Simple String	0.00	0.72	0.45	0.63	**0.67**	0.64	0.28
	Complex String	0.00	0.77	0.52	0.58	**0.66**	0.62	0.27
	Digit	0.61	0.54	0.51	**0.62**	0.53	0.53	0.53
	2-D 8 Demo	0.00	0.57	**0.60**	0.49	0.53	0.52	0.31
	2-D 16 Demo	-	0.56	**0.61**	0.49	0.53	0.48	0.26
	Antonyms	0.61	0.77	0.65	0.78	**0.85**	**0.85**	0.09
	Capital	0.74	0.84	0.82	**0.88**	0.80	0.83	0.05
	Language	0.56	0.77	0.56	**0.85**	0.75	0.79	0.01
	Profession	0.27	0.44	0.35	**0.49**	0.40	0.48	0.03
	Religion	0.57	0.75	0.53	0.80	0.82	**0.84**	0.00
Pythia-2.8B	Simple String	0.37	0.81	0.48	0.70	**0.72**	0.64	0.00
	Complex String	0.57	0.73	0.55	0.70	0.65	**0.72**	0.00
	Digit	0.55	0.72	0.52	0.55	**0.72**	0.69	0.00
	2-D 8 Demo	0.10	0.67	0.55	0.50	**0.60**	0.56	0.00
	2-D 16 Demo	-	0.73	0.55	0.50	**0.60**	**0.60**	0.00
	Antonyms	0.05	0.70	0.09	0.13	0.63	**0.73**	0.00
	Capital	0.58	0.78	0.78	**0.82**	0.68	0.80	0.01
	Language	0.32	0.70	0.62	**0.76**	0.64	0.72	0.01
	Profession	0.20	0.25	0.29	**0.42**	0.36	**0.42**	0.00
	Religion	0.44	0.84	0.62	0.75	0.85	**0.89**	0.00
Pythia-12B	Simple String	0.37	0.68	0.51	**0.72**	0.59	0.56	0.00
	Complex String	0.56	0.67	0.55	0.57	0.59	**0.62**	0.00
	Digit	0.43	0.67	0.57	0.55	**0.62**	0.61	0.00
	2-D 8 Demo	0.02	0.64	0.54	**0.61**	0.53	0.57	0.00
	2-D 16 Demo	-	0.64	0.53	**0.61**	0.53	0.53	0.00
	Antonyms	0.12	0.55	0.13	0.14	**0.68**	0.67	0.00
	Capital	0.12	0.28	0.29	0.50	0.39	**0.68**	0.00
	Language	0.37	0.69	0.34	0.56	0.67	**0.77**	0.00
	Profession	0.07	0.13	0.18	0.25	0.37	**0.40**	0.00
	Religion	0.26	0.75	0.46	0.78	0.85	**0.92**	0.00
Vicuna-13B	Simple String	0.00	0.53	0.40	0.37	**0.53**	0.51	0.03
	Complex String	0.00	0.34	0.40	**0.56**	0.39	0.41	0.02
	Digit	0.13	0.37	0.48	**0.55**	0.45	0.45	0.05
	2-D 8 Demo	0.00	0.36	0.47	**0.50**	0.46	0.49	0.00
	2-D 16 Demo	-	0.31	0.50	0.50	0.51	**0.52**	0.00
	Antonyms	0.47	0.78	0.59	**0.84**	0.79	0.79	0.04
	Capital	0.59	0.86	0.77	**0.89**	0.83	0.86	0.14
	Language	0.42	0.75	0.33	0.72	0.79	**0.83**	0.04
	Profession	0.71	0.46	0.34	**0.55**	0.52	0.53	0.07
	Religion	0.74	0.88	0.64	0.84	0.86	**0.90**	0.00