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. This task vector can be computed by averaging across the dataset. It conveys the overall task information and can be considered as global. Patching the "global task vector" allows LLMs to achieve zero-shot performance with dummy inputs similar to few-shot learning. However, we find that such a "global task vector" does not underlie all tasks. Patching a "global task vector" in some tasks leads to limited performance. We show that the information provided by each demonstration is first transmitted to its *answer* position and forms a "local task vector", which contains task information associated with each demonstration. The "local task vectors" converge in later layers and form the "global task vector" in some tasks but not the others. The latter include tasks that rely on rules that can only be drawn from multiple demonstrations, such as categorization tasks. We further show that the "local task vector" 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 from 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 involves simply a number of demonstrations that share the same structure as the question.

Previous work on the mechanism of ICL reveals a "task vector" at specific positions within the model(Ilharco et al., 2022; Hendel et al., 2023; Wang et al., 2023a). 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 as 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 its performance degrades in certain tasks, particularly in tasks where the rule can only be drawn from multiple demonstrations (e.g. learning an arbitrary categorization boundary). Instead, we find that patching the tokens at *answer* positions of each demonstration to 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 ICLs (Wang et al., 2023a), we propose that the label words serve as "local task vectors" that carry the information of the task rule in a distributed manner. We show that in tasks where a "global task vector" presents, the "local

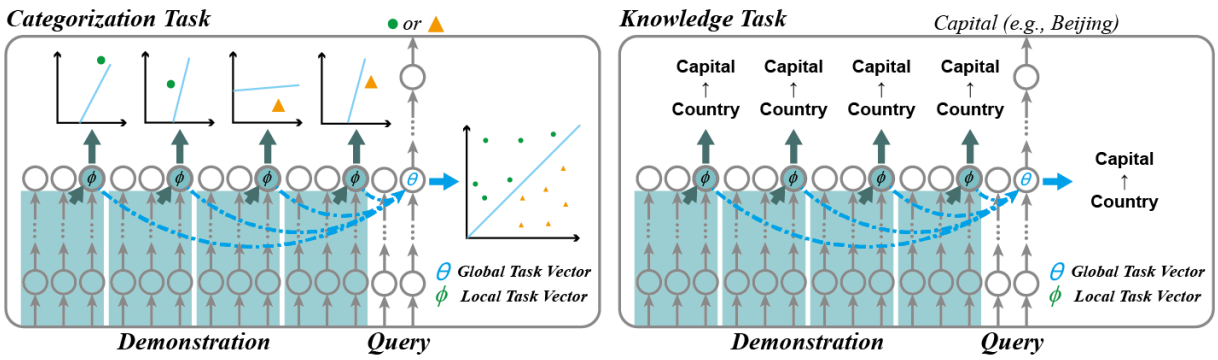

Figure 1: **Illustration of local task vector and global task vector**: A local task vector exists at each *answer* position, encapsulating information about the query, label, and the task rule. Left: In categorization tasks where the rule information extracted from each *answer* position may vary , the global task vector, which averages across samples, may perform poorly. Right: When the rule at each position are similar, as in knowledge tasks, averaging local task vectors can produce a more effective global task vector.

task vectors" start to appear in earlier layers than the "global task vector", and that the information contained in the "local task vectors" converged to the "global task vectors" in later layers.

Together, our findings broaden the previous understanding of "task vectors" and provide a more general mechanism of ICL in LLM.

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 input to ICL, and x to denote the query that ICL is asked to provide an output for. We use $T([S, x])$ to denote the output of ICL on the concatenation of S and x. For clarity, we define $S = [Q_0 I_0 A_0 D_0, Q_1 I_1 A_1 D_1, ..., Q_J I_J]$ 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 ICL operates within a hypothesis space. This mode of operation can be defined as a learning algorithm (denoted by $\mathcal{A}$). $\mathcal{A}$ 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}. Consider the following mapping from a set of demonstrations and a query to the predicted output: $T([S, x]) = f(x; \mathcal{A}(S))$ and 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 (Simonyan et al., 2013; Michel et al., 2019; Wang et al., 2023a) to identify the level of attention the LLM gives to each token. The saliency score is defined as

$$I_l = \left| \sum_h A_{h,l} \odot \frac{\partial \mathcal{L}(x)}{\partial A_{h,l}} \right|, \quad (1)$$

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 Mahalanobis Distance

Given a point $\mathbf{x}$ and a distribution with mean μ and covariance matrix $\mathbf{\Sigma}$, the Mahalanobis distance D_M from $\mathbf{x}$ to the distribution is defined as:

$$D_M(\mathbf{x}) = \sqrt{(\mathbf{x} - \mu)^T \mathbf{\Sigma}^{-1} (\mathbf{x} - \mu)} \quad (2)$$

To calculate Mahalanobis distance, we first group the hidden states into different clusters based on the interested parameter, e.g. string length in our main task. Then we calculate the covariance in each cluster and compute the pairwise distance between each point out side and inside the cluster.

2.5 dPCA

Following Kobak et al. (2016), we find a compressing matrix D and a decompressing matrix F by minimizing:

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

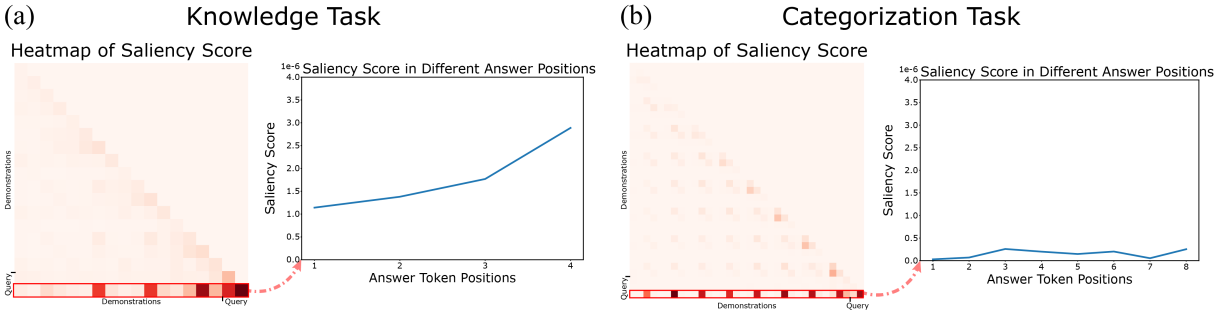

Figure 2: **Saliency score in the middle layers.** Saliency score shown as heatmaps. The color at each location shows the saliency score between the positions in the respective row and column. The last row, where the highest scores are observed. On the right is that the saliency score of the *answer* position of each demonstration. (a) Knowledge task; (b) Categorization task.

where $\mathbf{X}$ is the $n \times h$ activity matrix, where h is dimension of hidden state, and $\mathbf{X_L}$ is a label average matrix. $\mathbf{X_L}$ has the same size as $\mathbf{X}$, but all the 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:

$$\mathbf{B} = Null(\mathbf{D}) \tag{4}$$

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

where $\mathbf{D} \in \mathbb{R}^{d \times h}$ is the compression matrix, $Null(...)$ is the null space computation so that $\mathbf{D}Null(\mathbf{D}) = 0$. $\tilde{\mathbf{X}}$ 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

Two representative tasks are selected as demonstrations. The first task is a country-capital knowledge task (Hendel et al., 2023 and Wang et al., 2023a). The model should give the capital of a given country, e.g., the prompt "China is Beijing. Japan is Tokyo. German is" should lead to the answer "Berlin". The second task is a two-alternative arbitrary categorization task. In this 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 to solve these problems as in Supp Figure 10 and as in previous work (Hendel et al., 2023), the accuracy increase sharply in the middle layers.

Details of categorization task can be found in Appendix Table 1. Results for more knowledge tasks and categorization tasks are also included in Table 2 and Appendix A.

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 (denoted by *a*) token of the demonstrations and the *is* (denoted by *i*) position token of the final query, suggesting 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 query to form the output (Figure 2a). The *is* position token of the query is previously defined as "task vector". As it contains information from all demonstrations, we can further term them as "global task vector".

In addition, we found that the saliency score is larger 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 (Figure 2b). Directly resetting the tokens at the *answer* position in the categorization task decreases the model's performance gradually, confirming the distributed nature of the categorization task (Supp Figure 11). This difference suggest that the mechanism of ICL may be different in the two tasks.

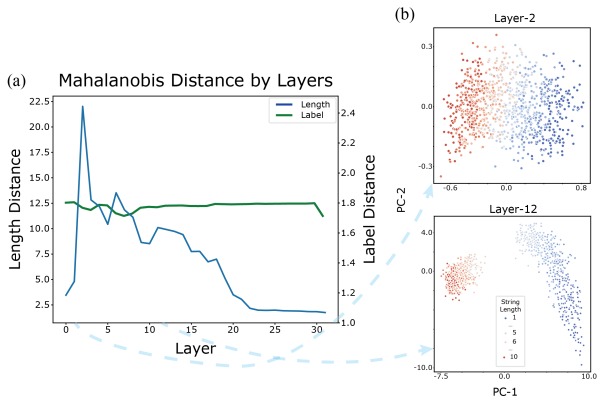

Figure 3: **The encoding of question and answer information in *answer* positions.** (a) Plotted is the Mahalanobis distance of the clusters defined by the demonstrations' string's length (blue) and by the corresponding answer (green) in the space defined by the two largest PCA components of the *answer* positions. Peaking at layer 2, the Mahalanobis distance for the string length gradually decreases across layers. (b) Two examples 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 contrast (string length) are more mixed in later layers (e.g. layer 12) than in earlier layers (e.g. layer 2).

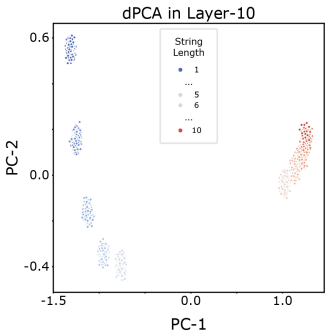

Figure 4: Using layer 10 as the example, both string length and answer information can be well captured in the 2-d dPCA space.

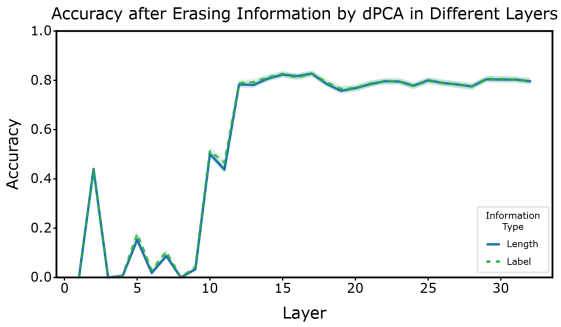

Figure 5: The model's accuracy after string length information in the local task vector are removed in the dPCA space. Note the ablation is only effective in early layers.

4.2 Answer Positions As Local Task Vectors

To verify that each demonstration's information converges at its *answer* position, we use the principle component analysis (PCA) to explore the information encoded at the *answer* positions (Figure 3). In the categorization, each demonstration provides a string and its associated category label. We calculate the Mahalanobis distances between the clusters defined with the string length and with the label for the answer tokens (Figure 3). The results suggest that the clusters formed by strings with different lengths are well segregated, with the distance peaking at layer 2 and decreasing gradually. In comparison, the answer is encoded at the *answer* positions stably across layers.

To confirm the string length information contained at the *answer* positions is critical for ICL, we carry out an experiment in which we selectively remove the string length information from the local task vectors. To do so, for *answer* position token, we first find its subspace that contains the corresponding demonstration's string length information with dPCA (Figure 4), and then project the token onto the null space of the found subspace. This procedure erases all information from the subspace that contains the string length information while keeping the rest of information intact.

We apply this selective information removal at each layer (Figure 5). 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

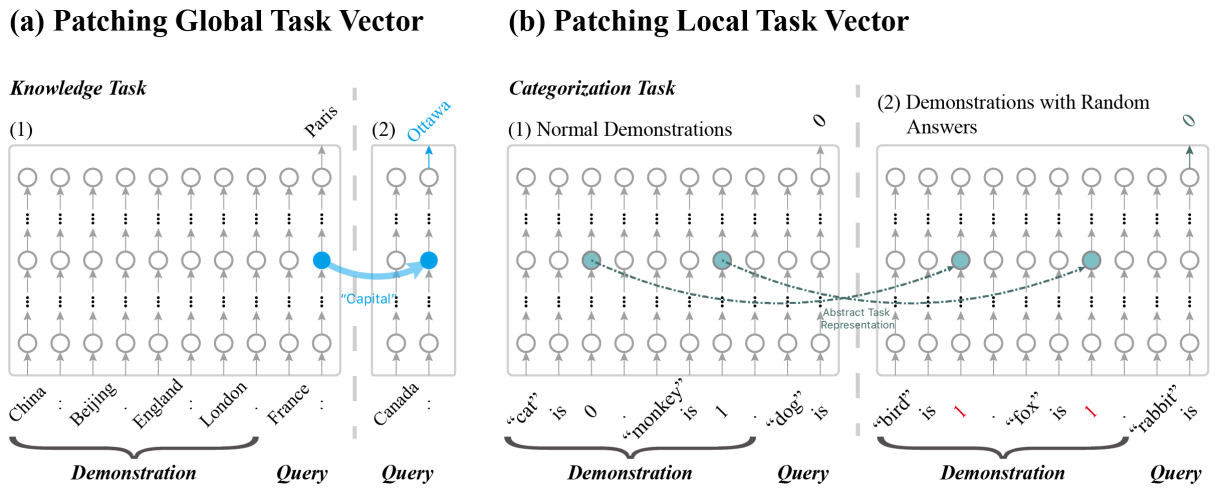

Figure 6: **Patching experiments.** (a) Patching the global task vector. The token at the *is* position of the final query, the task vector, in a network receiving normal demonstrations is copied to a network performing 0-shot inference. (b) Patching the distributed local task vectors. The token at the *answer* position of each demonstration in a network receiving normal demonstrations is copied to a network receiving the same number of demonstrations but with random *answers*.

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 the selective information removal.

These results suggest that the *answer* position tokens contains task-related information provided by each demonstration and are critical for ICL. As they are local to each demonstration, we term them as "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 leak. More specifically, given an original sequence S and a dummy sequence D, we patch S_a^j to D_a^j as illustrated in Figure 6b, where $*_a^j$ donates the j-th *answer* position (or local task vector) of the sequence.

5.1 Accuracy Per Layers

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 Figure 6a, where $*_i^j$ donates the j-th *is* position of the sequence and $\hat{S}_i^j$ is the **averaged** task vector cross samples described in the Preliminaries. This dummy sequence is zero-shot, which contains zero demonstration. 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

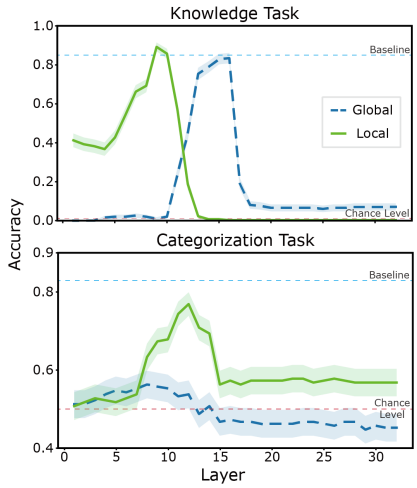

Figure 7: **Patching with the global and local task vectors.** The accuracy of the network patched with the *global task vector* and the *local task vectors* in different layers in the knowledge task (top) and the categorization task (bottom). Blue dash lines indicate the baseline performance, which is the network receiving normal demonstrations. Red dash lines indicate the chance level.

et al. (2023). Patching with the local task vectors and the global task vector have different effects in the two tasks.

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 incerease in accuracy (Figure 10). Global task vectors created with more demonstrations leads to higher accuracy (Figure 8). With global task vectors based on three or more demonstrations, the 0-shot performance of the patched LLM is on par with that receiving the real demonstrations. However, 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% (Figure 7a, bottom).

In contrast, patching with the "local task vectors" in the middle layers rescues the performance of the model in the categorization task (Figure 7b, right). The more demonstrations' local task vectors are patched, the better the performance is (Figure 9). 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 (Figure 7). We 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. But such translation of task information representation is not found in the categorization task (Figure 7 Bottom).

We carry out further patching experiments on various knowledge-based tasks and categorization tasks and on different LLMs. For knowledge-based tasks, we selected different types of factual tasks. The categorization tasks require multiple demonstrations to establish the correct association rules. Results are presented in Appendix A.3. We find that in some tasks, patching the global task vector to dummy position leads to poor performance, especially in the categorization tasks. In cases where patching with the global task vector do not work well, patching with the local task vectors usually make up for the performance.

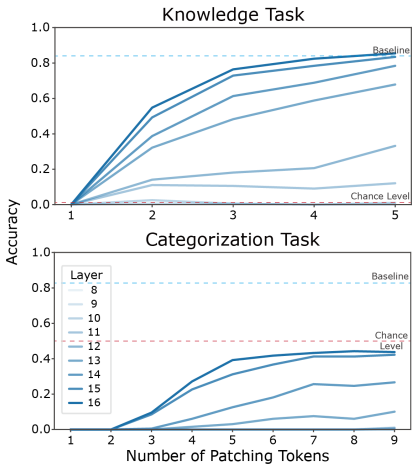

Figure 8: Patching the global task vectors in 8-16 layers. Top: knowledge task; Bottom: categorization task.

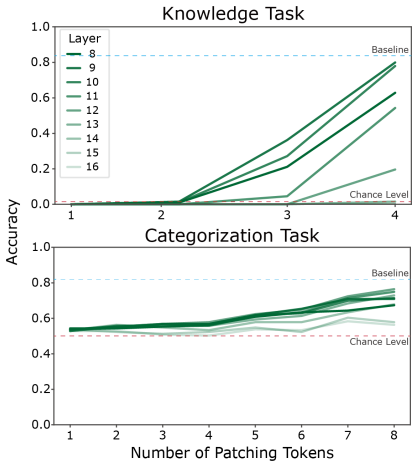

Figure 9: Patching the local task vectors in 8-16 layers. Top: knowledge task; Bottom: categorization task.

Besides, patching with the global task vector is often more effective when we calculate the global task vector by averaging across all dataset (see Appendix A.3), which make it even more global. This method may extract the most accurate task information. However, in many categorization tasks, local task vectors—without averaging—often outperform the global task vector, regardless of whether the averaging is performed across the dataset or not. 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.

6 Limitations

Neither global nor local task vectors work consistently across all tasks and in all types LLM 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 it is yet to explore the reason.

7 Conclusion

Our results reveal a more general mechanism involving distributed local task vectors that encapsulate the task information contained in the demonstrations in 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 Analysis Details

8.1 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) and Vicuna (Chiang et al., 2023) models. The results of all models and the task dataset can be found in supplemental material.

8.2 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. "$\sim$", 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.

8.3 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.

9 Related Work

Neuroscience Our work is inspired by studies from the field of neuroscience(Yamins and DiCarlo, 2016; Richards et al., 2019; Barrett et al., 2019; 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 for studying study how information is encoded in a high-dimensional space represented by population neuronal activities .

ICL 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 has been further proposed that transformers utilize gradient descent to perform linear regression tasks (Von Oswald et al., 2023; Akyürek et al., 2023; Ahn et al., 2024). 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 (Xie et al., 2022; Wang et al., 2023b; Ahuja et al., 2023).

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". Hendel et al. (2023) argued that ICL is achieved through a single task vector in the middle layers extracted from the demonstrations. In contrast, our research focuses on categorization tasks, which rely on multiple demonstrations. We use this distinction to discuss mechanisms in ICL that are different from "task vectors".

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.

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.

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

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. 2022. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*.

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*.

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.

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.

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.

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

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 Accuracy Per Layers

Figure 10 shows the accuracy in two representative tasks. There is a sharp increase in performance in the middle layers. This trend is consistent with prior studies (Hendel et al., 2023) and suggests that the middle layers play a crucial role in processing and encoding key task-specific information, contributing significantly to the model's performance.

A.2 Ablation at Answer Position

In Figure 11, 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.

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.

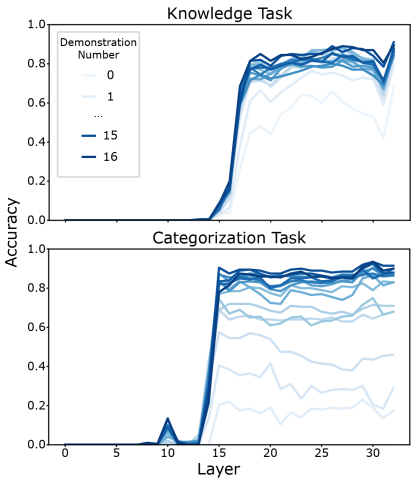

Figure 10: Accuracy increases sharply in the middle layers. The shades of blue indicate demonstration numbers. Top: knowledge task; Bottom: categorization task.

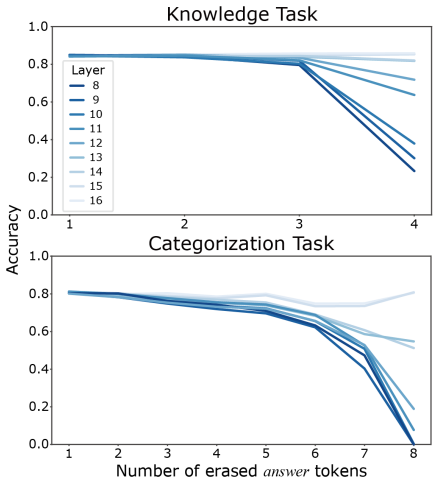

Figure 11: Model accuracy of ablated models. The shades of blue indicate the index of layers. Top: knowledge task; Bottom: categorization task.

Task Name	Task Rule	Example
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>=5,else 0	4->0, 7->1
2-D Data	1 if y>=x,else 0	(0,1)->1, (7,4)->0
Antonyms	Given an English adjective, output an antonym	laugh->cry
Capital	Given a name of a country, output the name of the capital city	French->Paris
Language	Given a name of a location, output their native language	Wyoming: English
Profession	Given a name of a person, output their profession	Marilyn Monroe->actor
Religion	Given a name of a person, output the associated religion	Yazid I: Muslim

Table 1: The tasks used in the study with input to output examples.

Model	Task Name	1-shot	Baseline	Task Vector		Rule 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	0.0	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	0.0
	Capital	0.68	0.86	0.83	**0.92**	0.80	0.83	0.0
	Language	0.67	0.82	0.64	0.79	0.77	**0.85**	0.0
	Profession	0.35	0.54	0.32	**0.53**	0.48	0.51	0.0
Llama-13B	Simple String	0.0	0.72	0.45	0.63	**0.67**	0.64	0.28
	Complex String	0.0	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	0.0	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**	0.0
pythia-2.8B	Simple String	0.37	0.81	0.48	0.70	**0.72**	0.64	0.0
	Complex String	0.57	0.73	0.55	0.70	0.65	**0.72**	0.0
	Digit	0.55	0.72	0.52	0.55	**0.72**	0.69	0.0
	2-D 8 Demo	0.1	0.67	0.55	0.50	**0.60**	0.56	0.0
	2-D 16 Demo		0.73	0.55	0.50	**0.60**	**0.60**	0.0
	Antonyms	0.05	0.70	0.09	0.13	0.63	**0.73**	0.0
	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**	0.0
	Religion	0.44	0.84	0.62	0.75	0.85	**0.89**	0.0

Model	Task Name	1-shot	Baseline	Task Vector		Rule 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**	0.0
	Complex String	0.52	0.69	0.53	**0.55**	0.49	0.51	0.0
	Digit	0.55	0.66	0.58	0.55	**0.60**	0.52	0.0
	2-D 8 Demo	0.03	0.63	0.55	0.50	**0.59**	0.53	0.0
	2-D 16 Demo		0.71	0.55	0.50	**0.59**	**0.59**	0.0
	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	0.0
	Profession	0.2	0.23	0.27	**0.47**	0.33	0.39	0.0
	Religion	0.6	0.84	0.49	**0.87**	0.83	0.85	0.0
Pythia-12B	Simple String	0.37	0.68	0.51	**0.72**	0.59	0.56	0.0
	Complex String	0.56	0.67	0.55	0.57	0.59	**0.62**	0.0
	Digit	0.43	0.67	0.57	0.55	**0.62**	0.61	0.0
	2-D 8 Demo	0.02	0.64	0.54	**0.61**	0.53	0.57	0.0
	2-D 16 Demo		0.64	0.53	**0.61**	0.53	0.53	0.0
	Antonyms	0.12	0.55	0.13	0.14	**0.68**	0.67	0.0
	Capital	0.12	0.28	0.29	0.50	0.39	**0.68**	0.0
	Language	0.37	0.69	0.34	0.56	0.67	**0.77**	0.0
	Profession	0.07	0.13	0.18	0.25	0.37	**0.40**	0.0
	Religion	0.26	0.75	0.46	0.78	0.85	**0.92**	0.0
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	0.0	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	0.0
	Capital	0.63	0.87	0.76	**0.92**	0.77	0.85	0.0
	Language	0.5	0.78	0.49	0.70	0.75	**0.83**	0.0
	Profession	0.32	0.44	0.28	**0.53**	0.45	0.52	0.0
	Religion	0.72	0.88	0.50	**0.94**	0.86	0.91	0.0
Vicuna-13B	Simple String	0.0	0.53	0.40	0.37	**0.53**	0.51	0.03
	Complex String	0.0	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	0.0	0.36	0.47	**0.50**	0.46	0.49	0.0
	2-D 16 Demo		0.31	0.50	0.50	0.51	**0.52**	0.0
	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**	0.0

Table 2: **The highest accuracy after patching the global task vector and the local task vectors in a series of knowledge task and categorization task.** Shown is the performance of the network receiving dummy demonstrations and patched with the average task vector of several samples, the task vector from a network receiving matching normal demonstrations, and the distributed local task vectors from a network receiving matching normal demonstrations. Baseline is the network receiving normal demonstrations.