Finding Visual Task Vectors

Anonymous Authors[1]

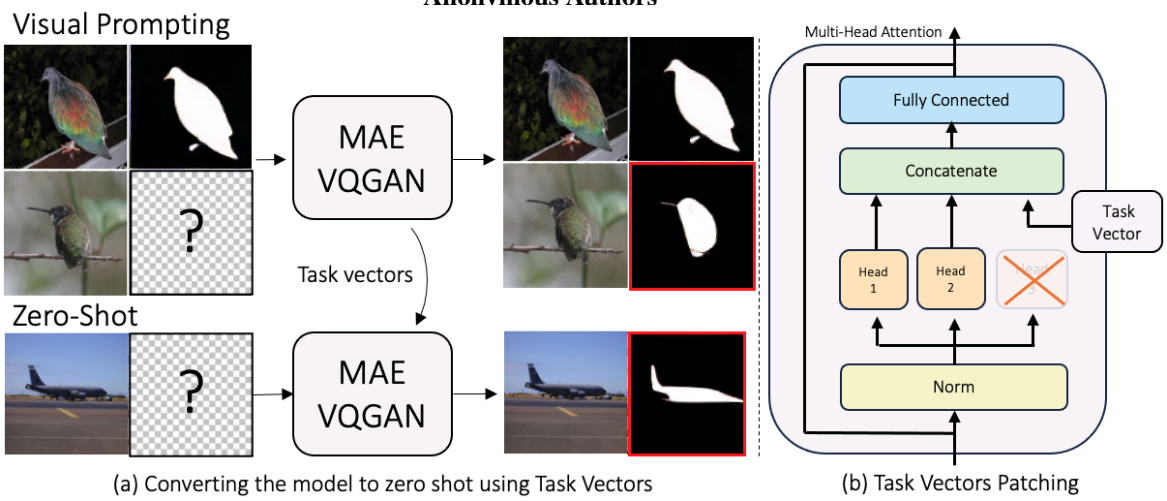

Figure 1. Visual Prompting models like MAE-VQGAN (Bar et al., 2022) require an input-output example to describe the desired task in their forward pass. We analyze the model activations and find *task vectors*, activations that encode task information that can be reused to control the task the model performs (see Figure a). Specifically, we tap into activations of individual attention heads and replace their outputs with task vectors to guide the model to the desired task (see Figure b). Surprisingly, the resulting models perform better than the original model while reducing the need for input-output examples. This confirms that *task vectors* exist in the network activation space and they can guide the model to perform the desired task.

Abstract

Visual Prompting is a technique for teaching models to perform a visual task via in-context examples, and without any additional training. In this work, we analyze the activations of MAE-VQGAN, a recent Visual Prompting model (Bar et al., 2022), and find *task vectors*, activations that encode task specific information. Equipped with this insight, we demonstrate that it is possible to identify the task vectors and use them to guide the network towards performing different tasks without providing any input-output examples. To find task vectors, we compute the average intermediate activations per task and use the REINFORCE (Williams, 1992) algorithm to search for the subset of task vectors. The resulting task vectors can guide the model towards performing a task competitively with the original model while reducing the need for input-output examples.

1. Introduction

In-context learning (ICL) is an emergent capability of large neural networks, first discovered in GPT-3 (Brown et al., 2020). ICL allows models to adapt to novel downstream tasks specified in the user's prompt. In computer vision, Visual ICL (known as Visual Prompting) is still at its infancy but it is increasingly becoming more popular (Bar et al., 2022; Bahng et al., 2022; Bai et al., 2023; Zhang et al., 2024b), mainly due to the appeal of using a single model to perform various downstream tasks without specific finetuning or change in the model weights.

In this work, we ask how does in-context learning work in computer vision. While this question is yet to be explored, there has been a significant body of research in Natural Language Processing (NLP) trying to explain this phenomenon (Akyürek et al., 2022; Dai et al., 2022; Garg et al., 2022; Hahn & Goyal, 2023; Han et al., 2023). Most recently, Hendel et al.(Hendel et al., 2023) suggested that LLMs encode *Task Vectors*, these are vectors that can be patched into the network activation space and replace the

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

ICL examples while resulting in a similar functionality. Concurrently, Todd et al. (Todd et al., 2023) discovered *Function Vectors*, activations of transformer attention heads that carry task representations. Our work is inspired by these observations and aims to study how ICL works in computer vision. We provide a more extended overview of the related works in the Suppl. Section 6.

We hypothesize that Visual ICL models create task vectors too, and aim to identify them. However, finding visual task vectors by relying on previous approaches is challenging. For example, both (Hendel et al., 2023; Todd et al., 2023) restricted their search space to the output activations of the last token in the prompt sequence. However, with images (see Figure 1, "Visual Prompting" input image), it is not obvious what token activations hold task information, and architectures like MAE-VQGAN (Bar et al., 2022) do not process image tokens sequentially. This alone increases the search space significantly because multiple tokens might hold task vectors.

To build intuition regarding the existence of visual task vectors, we visualize intermediate activations ranked by their "taskness", expecting task vectors to remain invariant within a task but vary across different tasks. Then, we propose an approach to find the subset of task vectors using REINFORCE, within the mean task activations of self-attention heads. Patching these identified task vectors leads to competitive performance with the original model across various tasks, confirming them as true task vectors. Our contributions include a method to select visual task vectors and demonstrating that these vectors enhance model performance while reducing the number of FLOPS by 22.5

2. Methods

Our goal is to understand in-context learning for computer vision, and how can existing models adapt to different downstream task in inference time. Specifically, we focus on the MAE-VQGAN (Bar et al., 2022) model, a variant of MAE (He et al., 2021) with a Vision Transformer (Dosovitskiy et al., 2021) encoder-decoder architecture.

The idea behind ICL is that given a an input-output example (x_s, y_s) and a new query x_q, to succeed in this task, the model F has to implicitly apply the transformation from x_s to y_s over x_q to produce y_q:

$$y_q = F(x_s, y_s, x_q) \tag{1}$$

Based on past observations from NLP (Hendel et al., 2023; Todd et al., 2023), we hypothesize that computer vision models encode latent task vectors in their activation space during the forward pass as well. This requires the model to implicitly map the input example into a set of latent task vectors $z = \{z_i\}_{i=1}^{k}$. Then the original function F can be decomposed into extracting the task vectors by a function G then applying F on the query while fixing the computed task activations z:

$$z = G(x_s, y_s) \qquad y_q = F(x_q|z) \tag{2}$$

2.1. Scoring Activations

Intuitively, every task vector is an activation that changes across different tasks but remains invariant to changes within a specific task. Specifically, denote $i = (l, m, k)$ as the l^{th} attention block, m^{th} attention head, and k^{th} token and denote it by F_i the function that outputs the corresponding activation in the attention block residual stream. We sample input-output example and a query triplet from individual tasks as well as across tasks and compute the latent activations corresponding to i:

$$(x_s, y_s, x_q) \sim \mathbf{D_{all\_tasks}} \qquad (x'_s, y'_s, x'_q) \sim \mathbf{D_{task\_j}}$$
$$z^i_{all} = F_i(x_s, y_s, x_q) \qquad z^i_{task_j} = F_i(x'_s, y'_s, x'_q)$$

Then, we define the score and the mean activations for position i and task j, where $\rho_{token}(i)$ is a scalar and $\mu_{i,j}$ has a dimensionality equal to the residual stream:

$$\rho_{token}(i) = \frac{var(z^i_{all})}{\frac{1}{n}\sum_{j=1}^{n} var(z^i_{task_j})} \quad \mu_{i,j} = \mathbb{E}[z^i_{task_j}] \tag{3}$$

Note that since our end goal is to operate in a "zero-shot" setting (e.g, no input-output examples), we only consider the score and task vectors that corresponds to the CLS token, the input query, and the output (present in the decoder only). Also, computing the token score and mean activations is very efficient as it simply requires applying the forward pass of a neural network across a batch without any access to a held out validation set.

Finally, we compute the aggregated per layer score, by summing the individual scores for each attention head and token position corrseponding to that particular layer:

$$\rho_{\text{layer}}(l) = \sum_{i \in I : \text{layer}(i) = l} \rho_{\text{token}}(i) \tag{4}$$

We use this scoring mechanism mainly to build intuition about the existence of task vectors, and find that some activations capture "taskness" (see Figure 2) and cluster the data well with respect to tasks (see Table 2). We further elaborate on this analysis in Section 3 and Section 4. To find visual task vectors, we leverage a more general approach outlined in the following section.

2.2. Finding Visual Task Vectors via REINFORCE

How can task vectors be identified? An exhaustive search over all subsets of activations is intractable. For example,

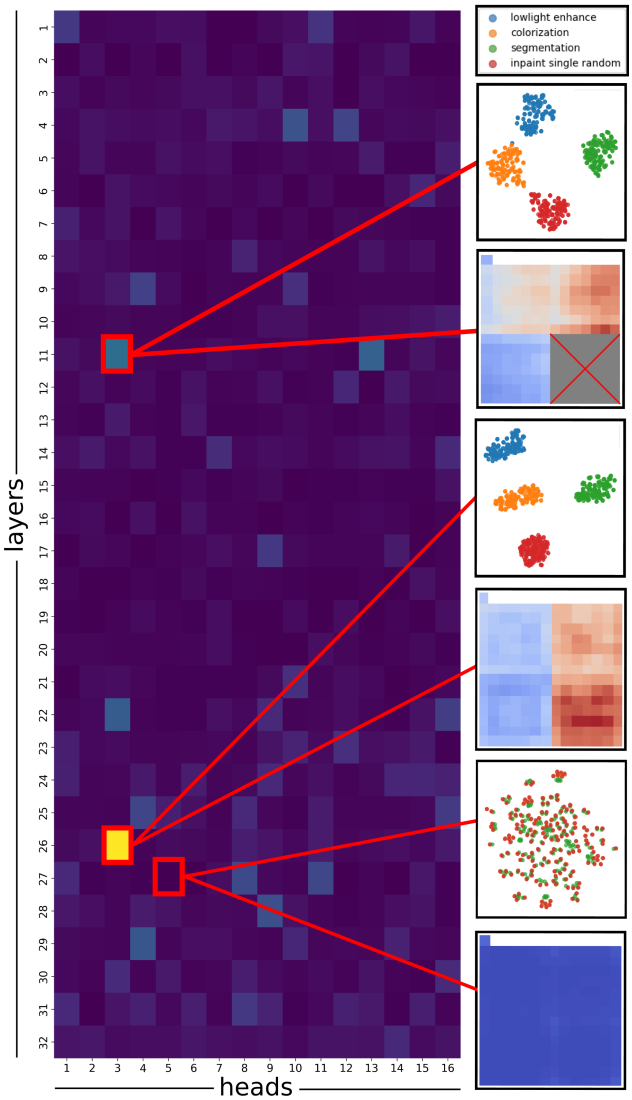

Figure 2. **Activation Scoring Analysis.** Individual scores ($\rho_{token}(i)$) aggregated per Attention Head (left). Individual token scores of specific heads are further visualized in their corresponding spatial arrangements, along with the t-SNE(van der Maaten & Hinton, 2008) clustering of head activations of different tasks (right).

MAE-VQGAN (Bar et al., 2022) which we utilize here has 32 attention blocks, each with 16 heads, therefore, if we consider these to be the potential set of activations then we need to search over 2^{512} options, evaluating each option over a held out validation set.

For every task j we can apply the following procedure to find the task vectors. Recall that $\{\mu_{i,j}\}$ denote the mean activations and $F(\cdot)$ is a pretrained visual prompting model. Denote $\alpha_{i,j} \sim Bernouli(\theta_{ij})$ as random variables that signify whether the mean task activation $\mu_{i,j}$ is a task vector.

Denote z_j as the set of task vectors for task j: $z_j = \{\mu_{i,j} | \alpha_{i,j} = 1\}$. Given pairs of input-output task demonstrations $x_q, y_q \sim \mathbf{D_{task\_j}}$, we want to find the set of task vectors that minimizes the loss function:

$$L(\theta) = E_{z_j \sim q(\theta)} L(y_q, F(x_q | z_j)) \tag{5}$$

Where $q(\theta)$ denotes the sampling distribution of z_j and L denotes the loss function that suits the particular task j. Note that differently than in Equation 1, if we find a good set of task vectors z, then we no longer need to condition F over additional input-output examples.

To find a set of task vectors, we need to estimate the parameters θ. Since the sampling distribution q depends on θ, it is natural to use the REINFORCE algorithm (Williams, 1992) to find the optimal θ. The key idea in REINFORCE is the observation that:

$$\nabla L(\theta) = E_{z_j \sim q(\theta)} L(y_q, F(x_q | z_j)) \nabla log P_\theta(z_j)$$

Thus, we can approximate $\nabla L(\theta)$ by sampling z_j and averaging the above equation. After iteratively optimizing with gradient descent, we select the final task vector positions by sampling once more from the $\alpha_{i,j} \sim Bernouli(\theta_{ij})$ distribution.

This procedure is outlined for finding the task vectors placements for individual tasks, however it is possible to find placements that generalize for multiple tasks by optimizing over examples across datasets and we refer this as "multi-task" placements. Additionally, while above we search for all token activations, it is possible to apply the search in different levels of granularity. For example, by patching groups of tokens from the same quadrant, patching all the tokens in an attention head, or patching the entire layer. We discuss these design choices in the next section.

3. Experiments

The goal of our experiments is to explore whether activation patching of task vectors can guide the model to execute the desired task. In this section, we detail the prompting schemes used, the baselines, tasks of interest, and the experiments conducted. For the full implementation details and experiments, please refer to Suppl. Section 7.

3.1. Downstream Tasks

We evaluate the performance on standard image-to-image tasks like Foreground Segmentation, Low Light Enhancement, In-painting, and Colorization. We use the Pascal-5i (Shaban et al., 2017) dataset and follow a similar evaluation setting as in (Bar et al., 2022).

Table 1. **Quantitative Analysis.** Results comparison across different tasks and splits, indicating the effectiveness of our task specific model. For full results across splits including more baselines, see Suppl. Table 4.

Model	Segmentation ↑ (Mean ± STD)	Lowlight Enhance ↓ (Mean ± STD)	Colorization ↓ (Mean ± STD)	In-painting ↓ (Mean ± STD)
Original MAE-VQGAN	0.338 ± 0.033	0.685 ± 0.032	0.618 ± 0.027	0.550 ± 0.042
Random Quadrants	0.170 ± 0.061	3.000 ± 0.967	3.025 ± 1.190	2.350 ± 0.955
Top Quadrants	0.150 ± 0.023	4.875 ± 0.228	4.250 ± 0.269	3.900 ± 0.141
CMA (Task-specific)	0.230 ± 0.012	0.825 ± 0.043	0.895 ± 0.063	1.750 ± 0.112
CMA (Multitask)	0.150 ± 0.017	1.400 ± 0.122	1.130 ± 0.075	1.225 ± 0.083
Ours (Multitask)	0.325 ± 0.026	0.492 ± 0.025	0.502 ± 0.036	0.558 ± 0.022
Ours (Task-specific)	$\mathbf{0.353} \pm 0.028$	$\mathbf{0.458} \pm 0.032$	$\mathbf{0.453} \pm 0.036$	$\mathbf{0.480} \pm 0.022$

3.2. Zero-shot Task Vector Patching

We seek to explore whether it is possible to implement tasks via zero-shot task vector patching. We investigate this in the following experiments:

Task Specific. For each of the four tasks we execute our task-specific method with 10 images and report the results on corresponding tasks.

Multi-task. We follow the same procedure but in a multi-task scenario where we select two images and perform our method, evaluating the loss across all tasks jointly. We normalize the loss per task in order to account for task-specific differences to ensure no single task dominates the search, and all tasks are weighted equally. Finally, we introduce the identity-copy task to keep the batch size at 10 for equivalent comparisons.

Baselines. We also provide the following baselines to benchmark the performance of our method. First, we compare to **MAE-VQGAN** (Bar et al., 2022), the original model's one-shot performance. We consider other baselines for finding task vectors like **CMA** (Causal Mediation Analysis (Pearl, 2022)), previously considered by Todd et al. (2023). We propose additional simple baselines like using **Random Quadrants**, where we patch into randomly sampled positions across the whole network. Additionally, we test patching into **Top Quadrants**, based on their scoring defined in Section 2. In both Random Quadrants and Top Quadrants we patch the same total amount of patches as task-specific and multitask.

4. Results

4.1. Zero-shot Task Vector Patching

We now share the performance of our method in comparison to the original model and the baselines. Surprisingly, by performing these interventions with task vectors we are able to implement visual ICL tasks with similar or better performance than the original model (see examples in Figure 3).

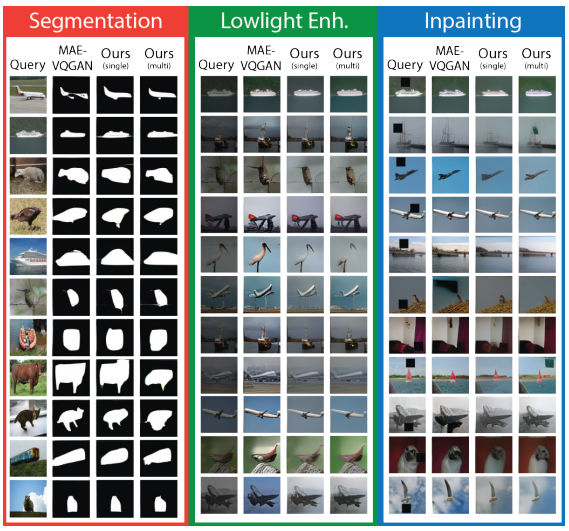

Figure 3. **Qualitative Examples.** We compare our task-specific and multitask methods to MAE-VQGAN

Our task-specific models beat the original MAE-VQGAN in all splits on all tasks. The multitask models are at least as good or better than the original MAE-VQGAN. Furthermore, we provide validation that ranking layers by their $\rho_{layer}(l)$ and selecting the top-k is instrumental to supporting the performance to randomly selecting k layers, which generally does not perform well.

5. Conclusion

In this work we explore the internal mechanisms of visual in-context learning and devise an algorithm to identify Task Vectors, activations present in transformers that can replace the in-context examples to guide the model into performing a specific task. We confirm our approach by adapting MAE-VQGAN to perform specific tasks in a zero-shot fashion by patching the Task Vector identified. We find that different than in NLP, in computer vision Task Vectors are distributed throughout the network's encoder and decoder.

References

Akyürek, E., Schuurmans, D., Andreas, J., Ma, T., and Zhou, D. What learning algorithm is in-context learning? investigations with linear models. *arXiv preprint arXiv:2211.15661*, 2022.

Bahng, H., Jahanian, A., Sankaranarayanan, S., and Isola, P. Exploring visual prompts for adapting large-scale models. *arXiv preprint arXiv:2203.17274*, 2022.

Bai, Y., Geng, X., Mangalam, K., Bar, A., Yuille, A., Darrell, T., Malik, J., and Efros, A. A. Sequential modeling enables scalable learning for large vision models. *arXiv preprint arXiv:2312.00785*, 2023.

Bar, A., Gandelsman, Y., Darrell, T., Globerson, A., and Efros, A. Visual prompting via image inpainting. *Advances in Neural Information Processing Systems*, 35: 25005–25017, 2022.

Bau, D., Zhu, J.-Y., Strobelt, H., Zhou, B., Tenenbaum, J. B., Freeman, W. T., and Torralba, A. Gan dissection: Visualizing and understanding generative adversarial networks. *arXiv preprint arXiv:1811.10597*, 2018.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

Dai, D., Sun, Y., Dong, L., Hao, Y., Sui, Z., and Wei, F. Why can gpt learn in-context? language models secretly perform gradient descent as meta optimizers. *arXiv preprint arXiv:2212.10559*, 2022.

Davies, D. and Bouldin, D. A cluster separation measure. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-1:224 – 227, 05 1979. doi: 10.1109/TPAMI.1979.4766909.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

Esser, P., Rombach, R., and Ommer, B. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12873–12883, June 2021.

Ferry, Q. R., Ching, J., and Kawai, T. Emergence and function of abstract representations in self-supervised transformers. *arXiv preprint arXiv:2312.05361*, 2023.

Gandelsman, Y., Efros, A. A., and Steinhardt, J. Interpreting clip's image representation via text-based decomposition. *arXiv preprint arXiv:2310.05916*, 2023.

Garg, S., Tsipras, D., Liang, P. S., and Valiant, G. What can transformers learn in-context? a case study of simple function classes. *Advances in Neural Information Processing Systems*, 35:30583–30598, 2022.

Hahn, M. and Goyal, N. A theory of emergent in-context learning as implicit structure induction. *arXiv preprint arXiv:2303.07971*, 2023.

Han, C., Wang, Z., Zhao, H., and Ji, H. In-context learning of large language models explained as kernel regression. *arXiv preprint arXiv:2305.12766*, 2023.

He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. B. Masked autoencoders are scalable vision learners. *CoRR*, abs/2111.06377, 2021. URL https://arxiv.org/abs/2111.06377.

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

Jia, M., Tang, L., Chen, B.-C., Cardie, C., Belongie, S., Hariharan, B., and Lim, S.-N. Visual prompt tuning. In *European Conference on Computer Vision*, pp. 709–727. Springer, 2022.

Jin, Z., Cao, P., Yuan, H., Chen, Y., Xu, J., Li, H., Jiang, X., Liu, K., and Zhao, J. Cutting off the head ends the conflict: A mechanism for interpreting and mitigating knowledge conflicts in language models. *arXiv preprint arXiv:2402.18154*, 2024.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization, 2017.

Li, X. L. and Liang, P. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.

Liu, J., Shen, D., Zhang, Y., Dolan, B., Carin, L., and Chen, W. What makes good in-context examples for gpt-3? *arXiv preprint arXiv:2101.06804*, 2021.

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

Lu, S., Schuff, H., and Gurevych, I. How are prompts different in terms of sensitivity? *arXiv preprint arXiv:2311.07230*, 2023.

Lu, Y., Bartolo, M., Moore, A., Riedel, S., and Stenetorp, P. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. *arXiv preprint arXiv:2104.08786*, 2021.

Luo, H. and Specia, L. From understanding to utilization: A survey on explainability for large language models. *arXiv preprint arXiv:2401.12874*, 2024.

Meng, K., Bau, D., Andonian, A., and Belinkov, Y. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35:17359–17372, 2022.

Moraffah, R., Karami, M., Guo, R., Raglin, A., and Liu, H. Causal interpretability for machine learning-problems, methods and evaluation. *ACM SIGKDD Explorations Newsletter*, 22(1):18–33, 2020.

Palit, V., Pandey, R., Arora, A., and Liang, P. P. Towards vision-language mechanistic interpretability: A causal tracing tool for blip. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2856–2861, 2023.

Park, K., Choe, Y. J., and Veitch, V. The linear representation hypothesis and the geometry of large language models. *arXiv preprint arXiv:2311.03658*, 2023.

Pearl, J. Direct and indirect effects. In *Probabilistic and causal inference: the works of Judea Pearl*, pp. 373–392. 2022.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Rousseeuw, P. J. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987. ISSN 0377-0427. doi: https://doi.org/10.1016/0377-0427(87)90125-7. URL https://www.sciencedirect.com/science/article/pii/0377042787901257.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. Imagenet large scale visual recognition challenge, 2015.

Shaban, A., Bansal, S., Liu, Z., Essa, I., and Boots, B. One-shot learning for semantic segmentation. *arXiv preprint arXiv:1709.03410*, 2017.

Singh, C., Inala, J. P., Galley, M., Caruana, R., and Gao, J. Rethinking interpretability in the era of large language models. *arXiv preprint arXiv:2402.01761*, 2024.

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

van der Maaten, L. and Hinton, G. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. URL http://jmlr.org/papers/v9/vandermaaten08a.html.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL http://arxiv.org/abs/1706.03762.

Wang, B. and Komatsuzaki, A. Gpt-j-6b: A 6 billion parameter autoregressive language model, 2021.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35: 24824–24837, 2022.

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

Wu, X. and Varshney, L. R. Transformer-based causal language models perform clustering. *arXiv preprint arXiv:2402.12151*, 2024.

Xie, S. M., Raghunathan, A., Liang, P., and Ma, T. An explanation of in-context learning as implicit bayesian inference. *arXiv preprint arXiv:2111.02080*, 2021.

Xu, J., Gandelsman, Y., Bar, A., Yang, J., Gao, J., Darrell, T., and Wang, X. Improv: Inpainting-based multimodal prompting for computer vision tasks. *arXiv preprint arXiv:2312.01771*, 2023.

Xu, S., Dong, W., Guo, Z., Wu, X., and Xiong, D. Exploring multilingual human value concepts in large language models: Is value alignment consistent, transferable and controllable across languages? *arXiv preprint arXiv:2402.18120*, 2024.

Zhang, F. and Nanda, N. Towards best practices of activation patching in language models: Metrics and methods. *arXiv preprint arXiv:2309.16042*, 2023.

Zhang, K., Lv, A., Chen, Y., Ha, H., Xu, T., and Yan, R. Batch-icl: Effective, efficient, and order-agnostic in-context learning. *arXiv preprint arXiv:2401.06469*, 2024a.

Zhang, Y., Tiňo, P., Leonardis, A., and Tang, K. A survey on neural network interpretability. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 5(5): 726–742, 2021.

Zhang, Y., Zhou, K., and Liu, Z. What makes good examples for visual in-context learning? *Advances in Neural Information Processing Systems*, 36, 2024b.