Iterative Context Vectors: Boost In-Context Learning within Activations

Anonymous ACL submission

Abstract

In-context learning has become a standard learning paradigm for language models. However, current prompt engineering methods, which function within the token space, may restrict their effectiveness. We propose to explore the potential of activation space through Iterative Context Vectors (ICVs), a technique aimed at improving task performance without backpropagation. ICVs are employed by first extracting and iteratively refining activations within a language model, then applying them during inference with minimal computational and memory overhead. We evaluate ICVs across a range of tasks using various models and observe significant improvements. Our findings suggest that activation steering can serve as a promising direction for in-context learning, thereby opening new avenues for future research.

1 Introduction

Few-shot learning has long been a prominent research focus. Recently, language models (LMs) have shown the capability to execute few-shot learning through in-context learning (ICL) (Brown et al., 2020). In this approach, learning a new task involves conditioning on a few support examples and predicting the most suitable tokens to complete a query input, all without the need for any parameter updates. This method is appealing because it relies solely on inference, allowing for quick adaptation to various downstream tasks.

However, it has been noted that despite its potential, the predictions of LMs can be highly volatile when conditioned on prompts. The outcomes depend significantly on the templates, demonstrations, their permutations, and can even ignore or violate the instructions of the prompt (Webson and Pavlick, 2022; Min et al., 2022b).

In this paper, we introduce Iterative Context Vectors (ICVs) to offer a new perspective. As illustrated in Figure 1, rather than remaining in the

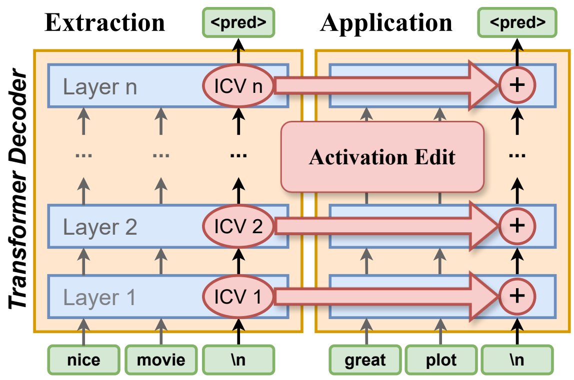

Figure 1: Iterative Context Vectors improve ICL performance by modifying model activations.

discrete prompt space, ICVs delve into the extensive activation space of the model. This exploration reveals a largely uncharted area for developing new methods, with our pioneering efforts to demonstrate how ICL can be enhanced from the representations within the model.

ICV contrasts with existing prompt tuning methods (Li and Liang, 2021; Lester et al., 2021), which operates in a continuous parameter space but still as part of the prompt and requires training via backpropagation. Again, unlike Parameter-Efficient Fine-Tuning (PEFT) methods, e.g. LoRA (Hu et al., 2021), ICV does not seek to tune the parameters of the model but rather modifies the activations during inference.

The essential traits of ICV, which will be elaborated upon in the rest of the paper, are highlighted as follows:

1. ICV has an intuitive theoretical support.
2. ICV is independent of instruction, prompt, label and permutation choices.
3. ICV does not require backpropagation.

To the best of our knowledge, we are the first to investigate the application of activation vectors on diverse real-world in-context learning tasks and to demonstrate their potential with in-context examples during inference.

2 Method

We begin by establishing the evaluation framework.

2.1 Activation Vector Evaluation

We adhere to standard few-shot benchmarking protocols (Vinyals et al., 2016; Finn et al., 2017; Snell et al., 2017) to define the activation vector evaluation setting. For a given split of an n-way k-shot classification task $\mathcal{T} = \{\mathcal{T}_{\text{train}}, \mathcal{T}_{\text{val}}, \mathcal{T}_{\text{test}}\}$, which includes textual query-answer pairs (x, y), an ICL *episode* is sampled as:

$$E = [(x_1, y_1), \ldots, (x_{n\times k}, y_{n\times k}), (x_q, y_q)]\,. \quad (1)$$

Here, (x_q, y_q) represents the query and its label, preceded by the $n \times k$ support examples. To avoid the impact of unbalanced samples during extraction, we uniformly sample k examples from each of the n classes and shuffle them to mitigate any bias arising from sample permutation.

The episode must first be converted into a pure text sequence before the language model $\mathrm{LM}(\cdot)$ can process it. This conversion is handled by a *verbalizer*, which uses a predefined prompt template to instantiate the samples. The template contains two key components: the *input-output separator* that links a question with its answer, and the *example separator* that joins the given support set. To preserve the simplicity of the template, we have chosen to use one newline (\n) for the input-output separator and three newlines for the example separator, as adopted in Min et al. (2022a). In the subsequent text, the verbalizer will be considered an integral component of the Language Model (LM) and will not be explicitly referenced for the sake of conciseness.

When the language model $\mathrm{LM}(\cdot)$ is given an episode E, it executes autoregressive inference on each of its tokens. The input-output tokens are particularly noteworthy because they are responsible for producing the answers. The prediction of the LM can be obtained by applying the softmax function to the logits of the possible labels.

$$\hat{y}_{\text{clean}} = \mathrm{LM}(E). \quad (2)$$

In contrast, an "edited" run utilizes an *activation vector editor* f_{edit}, represented as

$$\hat{y}_{\text{edit}} = \mathrm{LM}(E; f_{\text{edit}}(\boldsymbol{v}, p)), \quad (3)$$

which relies on the extracted vectors $\boldsymbol{v}$ from an *activation vector extractor* f_{ext} with hyperparameters p:

$$\boldsymbol{v} = f_{\text{ext}}(\mathcal{T}_{\text{train}}; p). \quad (4)$$

The extractor retrieves its target vectors $\boldsymbol{v}$ from $\mathcal{T}_{\text{train}}$ and identifies the optimal hyperparameters p^* from $\mathcal{T}_{\text{val}}$ by maximizing the metric M:

$$p^* = \arg\max_p \mathrm{M}_{E\sim\mathcal{T}_{\text{val}}}(\hat{y}_{\text{edit}}, y_q) \quad (5)$$

$$\boldsymbol{v}^* = f_{\text{ext}}(\mathcal{T}_{\text{train}}; p^*). \quad (6)$$

For single-token classification tasks, macro-F1, micro-F1, and weighted-F1 scores can serve as the metrics. The vectors $\boldsymbol{v}^*$ and the optimal hyperparameters p^* are then applied to the test set $\mathcal{T}_{\text{test}}$ to evaluate the final results $\mathrm{M}_{E\sim\mathcal{T}_{\text{test}}}(\hat{y}_{\text{edit}}, y_q)$.

Having outlined the evaluation framework, we will now move on to the theoretical grounds of our method.

2.2 Theoretical Foundation

Given the significance of ICL, many theories have been suggested to explain its mechanism, e.g. Xie et al. (2022); Chan et al. (2022); Ye et al. (2023); Oswald et al. (2023). Drawing inspiration from Irie et al. (2022), we construct our ICV based on the empirical evidence provided by Dai et al. (2023).

Irie et al. (2022) revisited the dual form of the perceptron and applied it in the modern context of deep NNs. They demonstrated that the forward operation of any linear layer in neural networks trained via gradient descent can be viewed as a key-value-query attention mechanism (Vaswani et al., 2017). In this framework, the training data points act as the keys, the corresponding gradients serve as the values, and the test input generates the query. A more detailed introduction to the dual form is provided in Appendix B.

With the help of the dual form, Dai et al. (2023) showed that ICL can be interpreted as a meta-optimization process. This was achieved by reversing the direction of the equivalence and breaking down the attention key and value terms for the ICL query token into its zero-shot and demonstration components. Under the relaxed normalization setting, the pretrained LM acts as a meta-optimizer. Through forward computation, the LM generates meta-gradients from the demonstration examples, which are then applied to the original language model via attention, culminating in the formation of the ICL inference capability. Their experiments

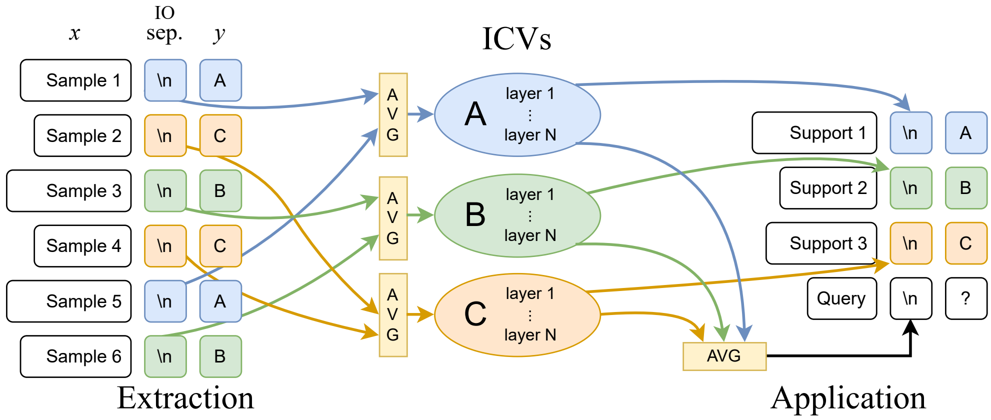

Figure 2: Illustration of the extraction and application phases of ICV. For clarity, contrastive subtraction and iterative updates have been omitted.

verified that ICL behaves similarly to explicit fine-tuning from multiple perspectives.

2.3 Iterative Context Vectors

We have determined that attention layers significantly influence ICL, with demonstrations acting as meta-gradients to help the model adapt to the task during inference. This explanation provides an intuitive understanding of how the LM uses in-context examples, but it also highlights why ICL performance can be unstable. Specifically, meta-gradients derived from limited in-context examples may not fully capture the task and may not fit well with the initial parameters. For this reason, we propose to extract the meta-gradients from the LM's inference process to improve their accuracy and robustness. This would allow us to apply these meta-gradients directly in future inference tasks, eliminating the need to compute them afresh with ICL each time a query is evaluated.

To define ICV, we first specify the extractor f_{ext}. To simulate the gradients, we generate two versions of a given n-way k-shot episode E in a contrastive manner, where $k \geq 1$ is a hyperparameter. The positive sequence is the standard shuffled verbalization, serving as the target for the gradients. The negative sequence can have various design choices; we choose to use a zero-shot query, which provides no information about the task.

The extractor then identifies the activations for all $n \times k$ input-output tokens of the support set (if one exists) and the final input-output token for the query in each attention layer of the LM. When $k > 1$, we initially average the activations for each class. Subsequently, we subtract the negative activations from the positive activations, thereby obtaining the gradients for a single episode. Given that there is no support set in the negative sequence, all activations from the positive support set share a common subtrahend of the negative query. By averaging over the training set, a preliminary version of the vectors can be calculated, as illustrated in Figure 2.

Next, to better utilize the forward pass computation, we propose to apply the vectors during the extraction phase, thus introducing the concept of *Iterative* Context Vectors. Specifically, we implement a batch-like update strategy, simulating standard batched gradient, which has been generally adopted to reduce the instability of single-step gradients. After every b extraction episodes, the vectors extracted from all previous episodes are averaged and used as the ICVs during subsequent extractions, leading us to the definition of the editor f_{edit}.

For the l-th attention layer $\mathrm{Attn}_l(\cdot)$, we have the corresponding extracted ICV $\boldsymbol{v}_l$. During inference, the editing is executed following Eq. 10

$$\mathrm{EditAttn}_l(x) := \mathrm{Attn}_l(x) + \alpha \times \boldsymbol{v}_l, \quad (7)$$

where two additional hyperparameters are introduced: the extraction strength α_1 and the inference strength α_2, adopted during the extraction and inference phrases, respectively. In summary, the hyperparameters for the ICVs are $p = \{k, b, \alpha_1, \alpha_2\}$.

3 Experiments

We apply our ICVs to three popular models across 12 tasks. The results are shown in Table 1. Details of the datasets can be found in Appendix C.

During the few-shot testing process, the model cannot ascertain the true class distribution of the test set, which is often imbalanced. Therefore, we adhere to the one-shot testing design, which sup-

Model	**Task**	agnews	emot.	hate	irony	offe.	sent.	abor.	athe.	clim.	femi.	hill.	trec	**Avg.**
gpt-j-6b	Clean	53.53	24.07	49.38	55.93	51.98	36.94	32.96	25.38	27.11	31.80	35.74	44.83	39.14
	FV	37.95	10.72	36.36	37.80	41.91	21.92	27.07	28.21	28.20	28.03	24.86	11.26	27.86
	TV	62.46	**26.12**	50.17	55.53	52.05	**38.72**	31.56	25.57	29.45	31.67	35.83	**51.94**	40.92
	ICV	**62.63**	20.26	**51.19**	**63.27**	**52.54**	34.55	**37.74**	**33.09**	**36.46**	**38.66**	**38.65**	40.66	**42.48**
llama-2-7b	Clean	61.94	54.45	53.27	58.65	51.86	38.96	27.52	22.13	28.60	29.27	29.42	56.56	42.72
	FV	23.68	16.67	50.76	38.70	21.76	12.60	23.26	10.33	27.05	27.30	20.08	5.92	23.18
	TV	**70.93**	**59.68**	52.44	50.48	54.05	43.67	27.90	21.83	32.04	**29.31**	**32.99**	56.61	44.33
	ICV	67.15	38.19	**57.36**	**66.03**	**58.39**	**45.56**	**31.00**	**22.66**	**32.70**	29.16	30.09	**61.80**	**45.00**
llama-2-13b	Clean	76.23	61.89	53.83	55.17	60.34	38.77	34.96	27.11	20.96	37.13	45.53	61.10	47.75
	FV	51.54	10.15	36.35	54.79	21.76	23.97	9.16	7.48	28.21	8.98	13.65	13.11	23.26
	TV	76.03	63.74	**54.47**	**55.36**	60.55	38.38	**35.12**	**30.08**	28.33	37.15	44.66	65.69	49.13
	ICV	**83.48**	**65.51**	54.43	53.66	**62.01**	**44.03**	34.98	26.11	**36.63**	**47.08**	**54.69**	**72.67**	**52.94**

Table 1: Main experiment results with macro-F1 as the metric. "Clean" denotes a standard one-shot ICL result. The models are GPT-J-6B (Wang and Komatsuzaki, 2021) and Llama 2 (Touvron et al., 2023).

plies the model with minimal yet sufficient information through a uniformly distributed support set. We evaluate over 200 episodes for both extraction and hyperparameter search, with a fixed iterative batch size $b = 10$ for all tasks. For other hyperparameters of ICVs, we search for the extraction shot $k \in \{1, 2, 3, 4\}$, the extraction strength and the inference strength $\alpha_1, \alpha_2 \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$. The final testing results are averaged over 10,000 randomly sampled episodes.

As further proof of concept and baselines for comparison, we also include two recent activation vector proposals: Function Vectors (Todd et al., 2023) and Task Vectors (Hendel et al., 2023). Although these methods were not originally designed to operate under the ICL evaluation setting, we adapted them to utilize the training set by averaging the activations. We search over their respective hyperparameters as well as the extraction shot k to ensure a fair comparison. Please refer to Appendix A for a discussion of their designs.

The results demonstrate that ICVs generally enhance ICL performance, surpassing the baselines in most tasks as well as in the overall average. Despite its simpler design, Task Vectors prove to be surprisingly competitive and can serve as a robust baseline. Conversely, Function Vectors hardly contribute to performance enhancement. Due to FVs' high search time stemming from their design, they may necessitate substantially more effort for optimization in real ICL applications.

To achieve a more comprehensive understanding of ICVs, we compare their performance against the standard ICL using additional shots, as illustrated in Table 2. Firstly, it is observed that ICVs not only surpass the majority of 1-shot performances but also often match or exceed performances with

Task	agnews	hate	irony	offe.	sent.	abor.
0-shot	51.14	45.99	59.25	47.60	32.83	26.51
1-shot	62.89	49.71	44.65	52.85	41.27	27.25
2-shot	76.30	57.74	53.11	57.03	45.01	22.98
3-shot	80.05	60.10	58.31	59.45	45.83	18.90
4-shot	81.44	61.47	54.45	58.10	48.39	18.20
ICV	67.15	57.36	66.03	58.39	45.56	31.00

Table 2: Comparison between ICV and standard ICL on Llama-2-7b with macro-F1 as the metric. See Table 6 for complete results.

higher numbers of shots. Secondly, the data indicates that the performance of standard ICL does not always improve with an increased number of demonstration examples. This suggests potential limitations of solely relying on prompt engineering to enhance performance. Thirdly, it is crucial to note that the temporal cost of standard ICL theoretically scales with $O(n^2)$. Although it is feasible to cache keys and values in practice, these caches cannot be reused when a new set of examples is introduced, which is likely to occur due to the inherent difficulty in identifying and ensuring a single set of examples that are effective for all inputs.

Finally, an ablation study examining the iterative batch size b has been conducted and is presented in Appendix D.

4 Conclusion

In our study, we have derived the Iterative Context Vectors (ICVs) from an intuitive theoretical framework, defined the evaluation protocols and subsequently conducted a series of experiments. Despite ICVs' simplicity, the results obtained are highly encouraging, indicating that activation vectors show significant potential for further exploration.

Limitations

This study examines the application of Iterative Context Vectors in the context of one-shot examples as a compromise between inference time and in-context information. Although applying ICVs to zero-shot inference would be more efficient, a computational sequence of insufficient length might hinder the model's ability to effectively solve the given task. (Feng et al., 2023)

We have opted for classification tasks wherein a single output token is sufficient to distinguish between the classes. The development and application of activation vectors in more complex tasks, as well as in generative tasks, represent areas for future investigation. Nevertheless, it is worth noting that the concept of ICVs and the associated evaluation protocol can potentially be expanded to encompass these more advanced applications.

References

Francesco Barbieri, Jose Camacho-Collados, Luis Espinosa Anke, and Leonardo Neves. 2020. TweetEval: Unified Benchmark and Comparative Evaluation for Tweet Classification. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1644–1650, Online. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Stephanie C. Y. Chan, Adam Santoro, Andrew K. Lampinen, Jane X. Wang, Aaditya Singh, Pierre H. Richemond, Jay McClelland, and Felix Hill. 2022. Data Distributional Properties Drive Emergent In-Context Learning in Transformers. *Preprint*, arxiv:2205.05055.

Damai Dai, Yutao Sun, Li Dong, Yaru Hao, Shuming Ma, Zhifang Sui, and Furu Wei. 2023. Why Can GPT Learn In-Context? Language Models Implicitly Perform Gradient Descent as Meta-Optimizers. *Preprint*, arxiv:2212.10559.

Guhao Feng, Bohang Zhang, Yuntian Gu, Haotian Ye, Di He, and Liwei Wang. 2023. Towards Revealing the Mystery behind Chain of Thought: A Theoretical Perspective. *Advances in Neural Information Processing Systems*, 36:70757–70798.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR.

Roee Hendel, Mor Geva, and Amir Globerson. 2023. In-Context Learning Creates Task Vectors. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9318–9333, Singapore. Association for Computational Linguistics.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*.

Kazuki Irie, Róbert Csordás, and Jürgen Schmidhuber. 2022. The Dual Form of Neural Networks Revisited: Connecting Test Time Predictions to Training Patterns via Spotlights of Attention. In *Proceedings of the 39th International Conference on Machine Learning*, pages 9639–9659. PMLR.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The Power of Scale for Parameter-Efficient Prompt Tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario Šaško, Gunjan Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Clément Delangue, Théo Matussière, Lysandre Debut, Stas Bekman, Pierric Cistac, Thibault Goehringer, Victor Mustar, François Lagunas, Alexander M. Rush, and Thomas Wolf. 2021. Datasets: A Community Library for Natural Language Processing. *Preprint*, arxiv:2109.02846.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. 2023. Inference-Time Intervention: Eliciting Truthful Answers from a Language Model. *Advances in Neural Information Processing Systems*, 36:41451–41530.

Xiang Lisa Li and Percy Liang. 2021. Prefix-Tuning: Optimizing Continuous Prompts for Generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

Sheng Liu, Lei Xing, and James Zou. 2023. In-context Vectors: Making In Context Learning More Effective and Controllable Through Latent Space Steering. *Preprint*, arxiv:2311.06668.

Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2022a. MetaICL: Learning to Learn In Context. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2791–2809, Seattle, United States. Association for Computational Linguistics.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022b. Rethinking the Role of Demonstrations: What Makes In-Context Learning Work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11048–11064, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Johannes Von Oswald, Eyvind Niklasson, Ettore Randazzo, Joao Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. 2023. Transformers Learn In-Context by Gradient Descent. In *Proceedings of the 40th International Conference on Machine Learning*, pages 35151–35174. PMLR.

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical Networks for Few-shot Learning. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Nishant Subramani, Nivedita Suresh, and Matthew Peters. 2022. Extracting Latent Steering Vectors from Pretrained Language Models. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 566–581, Dublin, Ireland. Association for Computational Linguistics.

Eric Todd, Millicent Li, Arnab Sen Sharma, Aaron Mueller, Byron C. Wallace, and David Bau. 2023. Function Vectors in Large Language Models. In *The Twelfth International Conference on Learning Representations*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. *Preprint*, arxiv:2307.09288.

Alexander Matt Turner, Lisa Thiergart, David Udell, Gavin Leech, Ulisse Mini, and Monte MacDiarmid. 2023. Activation Addition: Steering Language Models Without Optimization. *Preprint*, arxiv:2308.10248.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *NIPS 2017*, volume 30. Curran Associates, Inc.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, koray kavukcuoglu, and Daan Wierstra. 2016. Matching Networks for One Shot Learning. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. `https://github.com/kingoflolz/mesh-transformer-jax`.

Mengqiu Wang, Noah A. Smith, and Teruko Mitamura. 2007. What is the Jeopardy Model? A Quasi-Synchronous Grammar for QA. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 22–32, Prague, Czech Republic. Association for Computational Linguistics.

Albert Webson and Ellie Pavlick. 2022. Do Prompt-Based Models Really Understand the Meaning of Their Prompts? In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2300–2344, Seattle, United States. Association for Computational Linguistics.

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.

Xi Ye, Srinivasan Iyer, Asli Celikyilmaz, Ves Stoyanov, Greg Durrett, and Ramakanth Pasunuru. 2023. Complementary Explanations for Effective In-Context Learning. *Preprint*, arxiv:2211.13892.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level Convolutional Networks for Text Classification. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

A Related Work

A.1 Activation Vectors

Some preliminary works have recently explored steering LMs in the representation space. Task Vectors (Hendel et al., 2023) are extracted from one layer of the model during ICL inference and then applied to a zero-shot query to determine whether they can preserve task-relevant information. Function Vectors (Todd et al., 2023), on the other hand, select activations from the top attention heads based on their causal effect in promoting the correct answer, average these activations, and add them to a specific layer.

Although these vectors are designed with intentions similar to ours, they are tested primarily on simple synthetic tasks like antonym, country-capital, and singular-plural pairs. In contrast, we target a practical setting by evaluating on real-world datasets, providing a more comprehensive assessment ground.

A.2 Generative Steering

Another research direction focuses on modifying LMs' activations for generation and transfer purposes. Latent Steering Vectors (Subramani et al., 2022) aim at sentence recovery and sentiment transfer. Inference-Time Intervention (Li et al., 2023) involves probing each attention head and guiding the model with the probe vector to enhance the truthfulness of the generated text. Studies by Turner et al. (2023) and Liu et al. (2023) address style and sentiment transfer by employing positive and negative sentence pairs to extract contrastive guidance.

Despite their similarities, these methods either require training with backpropagation or are specifically tailored for generative or transfer tasks between sentence pairs. Consequently, they cannot be directly integrated into our approach.

B The Dual Form of Attention Layers

Formally, assume a linear layer trained via gradient descent utilizing T training inputs $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T)$ and their corresponding (backpropagated) error signals $(\boldsymbol{e}_1, \ldots, \boldsymbol{e}_T)$, where $\boldsymbol{x}_t \in \mathbb{R}^{d_{in}}$ and $\boldsymbol{e}_t \in \mathbb{R}^{d_{out}}$. If standard gradient descent is applied, a loss function $\mathcal{L}$ produces the error signal $\boldsymbol{e}_t = -\eta_t(\nabla_{\boldsymbol{y}}\mathcal{L})_t$, where $\eta_t \in \mathbb{R}$ is the learning rate, and $\boldsymbol{y}_t = \boldsymbol{W}\boldsymbol{x}_t$ is the output of the linear layer. Its weight matrix is given by

$$\boldsymbol{W} = \boldsymbol{W}_0 + \sum_{t=1}^{T} \boldsymbol{e}_t \otimes \boldsymbol{x}_t, \tag{8}$$

where $\boldsymbol{W}_0 \in \mathbb{R}^{d_{out} \times d_{in}}$ represents the initial value of the weights. This linear layer transforms an input $\boldsymbol{x} \in \mathbb{R}^{d_{in}}$ into an output $S_1(\boldsymbol{x}) \in \mathbb{R}^{d_{out}}$:

$$S_1(\boldsymbol{x}) = \boldsymbol{W}\boldsymbol{x}. \tag{9}$$

Next, consider a composite layer S_2 that stores T key-value pairs, represented by a key matrix $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T) \in \mathbb{R}^{d_{in} \times T}$ and a value matrix $\boldsymbol{E} = (\boldsymbol{e}_1, \ldots, \boldsymbol{e}_T) \in \mathbb{R}^{d_{out} \times T}$, along with a weight matrix $\boldsymbol{W}_0 \in \mathbb{R}^{d_{out} \times d_{in}}$. This layer transforms an input $\boldsymbol{x} \in \mathbb{R}^{d_{in}}$ into an output $S_2(\boldsymbol{x}) \in \mathbb{R}^{d_{out}}$ by

$$S_2(\boldsymbol{x}) = \boldsymbol{W}_0\boldsymbol{x} + \mathrm{Attn}(\boldsymbol{X}, \boldsymbol{E}, \boldsymbol{x}), \tag{10}$$

where the parameters of the unnormalized attention operator $\mathrm{Attn}(\cdot)$ are, in order, the key, value, and query.

It can be shown that S_1 and S_2 are equivalent by expanding the attention operation as

$$\mathrm{Attn}(\boldsymbol{X}, \boldsymbol{E}, \boldsymbol{x}) = \boldsymbol{E}\boldsymbol{X}^\top \boldsymbol{x} = \left(\sum_{t=1}^{T} \boldsymbol{e}_t \otimes \boldsymbol{x}_t\right) \boldsymbol{x}. \tag{11}$$

C Dataset and Tasks

All datasets utilized in this research are obtained from Huggingface (Lhoest et al., 2021). A full list of these datasets, along with their corresponding access labels, is detailed in Table 3.

AG News (Zhang et al., 2015) is a subdataset of AG's corpus of news articles constructed by assembling titles and description fields of articles from the 4 largest classes ("World", "Sports", "Business", "Sci/Tech") of AG's Corpus.

TweetEval (Barbieri et al., 2020) introduces an evaluation framework consisting of a series of Twitter-specific classification tasks. We selected all single-token classification tasks from the dataset.

Text Retrieval Conference Question Answering (TrecQA) (Wang et al., 2007) is a dataset created from the TREC-8 (1999) to TREC-13 (2004) Question Answering tracks.

Our few-shot evaluation methodology employs episodic sampling to regulate the duration of both extraction and inference processes, rather than relying solely on the absolute number of samples.

Name	Abbr.	Huggingface Label
AG News	agnews	ag_news
Emotion	emot.	tweet_eval/emotion
Hate	hate	tweet_eval/hate
Irony	irony	tweet_eval/irony
Offensive	offe.	tweet_eval/offensive
Sentiment	sent.	tweet_eval/sentiment
Abortion	abor.	tweet_eval/stance_abortion
Atheism	athe.	tweet_eval/stance_atheism
Climate	clim.	tweet_eval/stance_climate
Feminist	femi.	tweet_eval/stance_feminist
Hillary	hill.	tweet_eval/stance_hillary
TREC	trec	trec

Table 3: The datasets and tasks employed, along with their corresponding abbreviations used in the result tables, and their respective labels as hosted on Hugging Face.

Consequently, not all available samples are utilized during the experimental procedures. This aspect underscores an additional dimension of efficiency inherent in activation vectors.

We utilize the default labels provided with the datasets to emphasize the independence of prompt formatting. It is noteworthy that, for the TREC dataset, the observed low zero-shot performance can be attributed to the default labels, which are capitalized abbreviations such as "ABBR", "ENTY", and "LOC". This particular dataset functions as a special case within our experiments, aimed at investigating whether ICVs can adapt to labels that are less semantically meaningful.

D Additional Results

More metrics of the main experiment. We present the results on the other two metrics, namely micro-F1 and weighted-F1, derived from our main experiment, in Table 4 and Table 5, respectively. Under these evaluation criteria, ICVs demonstrate performance on par with, and often surpassing, that of FV and TV across the majority of tasks.

All experiments were conducted utilizing a predetermined random seed (42) to mitigate selection bias. To ensure a robust representation of result distributions, the tests from the main experiments were averaged over a substantial number of episodes, specifically 10,000.

Notably, the exception lies in the GPT-J-6B & micro-F1 setting, where FV exhibits superior performance. Given the pronounced underperformance of FVs across other cases, we hypothesize that this anomalous result may indicate a strong bias of FVs towards the majority classes and the specific model. This bias results in an elevated micro-F1 score, while simultaneously failing to perform effectively under other evaluation settings.

Ablation on the extraction batch size. Additionally, we make an experiment that ablate on the extraction batch size b. The results are shown in Figure 3. It is evident that without our iterative batching strategy, whether when $b = 1$ or b is excessively large, the extracted ICVs demonstrate suboptimal performance.

As stated in Section 3, we consistently employ $b = 10$ across all models and tasks in other experiments, concentrating our search efforts on other hyperparameters. This approach is intended to limit the search time and to prioritize the identification and optimization of the more significant parameters.

Furthermore, the results suggest that there exists substantial potential for optimization within the design and hyperparameter space of ICVs. This potential will be reserved for future research endeavors aimed at refining and advancing more sophisticated methodologies.

E Code and Reproducibility

The core code defining the ICVs as well as the evaluation protocol is provided within the supplementary material. We will release the complete code repository necessary for reproducing all of our experiments to promote transparency and facilitate future research endeavors.

Model	**Task**	agnews	emot.	hate	irony	offe.	sent.	abor.	athe.	clim.	femi.	hill.	trec	**Avg.**
gpt-j-6b	Clean	57.97	31.91	49.39	59.86	63.22	38.73	39.17	30.49	30.92	37.70	40.33	54.01	44.48
	FV	41.45	27.30	**57.13**	60.76	**72.16**	**48.98**	**68.37**	**73.35**	**73.31**	**64.94**	**59.47**	18.04	**55.44**
	TV	**66.85**	**33.23**	50.20	59.69	60.83	40.01	37.87	31.27	37.15	37.52	40.80	**61.51**	46.41
	ICV	63.37	31.55	51.29	**64.64**	60.97	38.89	55.97	47.76	53.12	43.62	54.14	53.46	51.57
llama-2-7b	Clean	63.40	57.31	53.64	62.22	53.67	40.02	28.69	24.90	34.88	30.25	30.05	60.77	44.98
	FV	34.46	39.73	57.11	61.01	27.82	19.94	24.71	14.02	33.48	**64.97**	25.82	18.88	35.16
	TV	**71.72**	**62.94**	52.92	56.26	56.48	43.10	29.29	24.63	**65.82**	30.29	33.86	**60.81**	49.01
	ICV	69.72	36.41	**57.38**	**66.58**	**64.78**	**48.22**	**37.44**	**26.12**	54.66	33.60	**37.49**	59.02	**49.28**
llama-2-13b	Clean	77.96	65.42	54.00	55.19	63.56	41.41	52.57	42.78	20.36	55.94	56.83	67.02	54.42
	FV	58.42	25.40	**57.10**	54.85	27.82	**48.83**	15.92	12.64	**73.37**	15.57	25.74	28.43	37.01
	TV	77.89	67.90	54.59	**55.40**	63.70	40.31	**53.25**	44.48	42.68	55.95	56.61	70.52	56.94
	ICV	**83.77**	**69.07**	54.72	54.23	**73.52**	43.51	51.01	**44.95**	49.01	**57.68**	**62.80**	**75.20**	**59.96**

Table 4: Main experiment results with micro-F1 as the metric. "Clean" denotes a standard one-shot ICL result.

Model	**Task**	agnews	emot.	hate	irony	offe.	sent.	abor.	athe.	clim.	femi.	hill.	trec	**Avg.**
gpt-j-6b	Clean	53.69	22.48	49.46	58.64	**62.47**	33.50	42.61	34.82	34.83	40.34	42.14	51.52	43.88
	FV	37.89	11.71	41.54	45.93	60.49	32.21	**55.53**	**62.07**	**62.02**	**51.96**	44.36	11.08	43.07
	TV	62.63	**23.97**	50.34	58.33	61.26	**35.46**	41.46	35.53	42.27	40.15	42.56	**57.36**	45.94
	ICV	**62.74**	23.08	**51.50**	**64.78**	61.40	28.80	55.25	51.70	56.21	46.26	**50.60**	48.26	**50.05**
llama-2-7b	Clean	76.36	65.73	53.46	54.99	65.44	33.47	51.80	45.57	19.77	53.00	55.25	68.46	53.61
	FV	51.57	10.32	41.51	54.43	12.11	32.04	4.37	2.84	**62.10**	4.20	10.54	15.84	25.16
	TV	76.18	68.12	**54.15**	**55.11**	65.54	32.67	**52.24**	**48.36**	45.48	52.99	54.78	70.22	56.32
	ICV	**83.56**	**69.51**	53.91	52.56	**71.28**	**42.13**	51.52	47.12	52.56	**59.37**	**61.99**	**75.31**	**60.07**
llama-2-13b	Clean	62.03	57.45	53.83	61.15	56.07	35.33	30.58	27.50	38.72	31.75	27.79	**64.49**	45.56
	FV	23.85	23.29	52.94	46.74	12.11	8.91	24.87	4.18	34.90	**51.57**	13.49	6.60	25.29
	TV	**71.05**	**63.17**	53.09	53.30	58.79	**41.54**	31.37	27.49	**61.51**	31.87	32.48	64.41	**49.17**
	ICV	67.27	38.23	**57.49**	**66.95**	**65.62**	40.49	**40.43**	**30.16**	55.75	34.57	**36.80**	56.14	49.16

Table 5: Main experiment results with weighted-F1 as the metric. "Clean" denotes a standard one-shot ICL result.

Task	agnews	emot.	hate	irony	offe.	sent.	abor.	athe.	clim.	femi.	hill.	trec	**Avg.**
0-shot	51.14	30.02	45.99	59.25	47.60	32.83	26.51	23.58	13.22	24.75	31.18	0.59	32.22
1-shot	62.89	44.57	49.71	44.65	52.85	41.27	27.25	23.39	28.55	29.65	29.24	56.17	40.85
2-shot	76.30	53.86	57.74	53.11	57.03	45.01	22.98	21.43	30.88	22.57	31.67	68.62	45.10
3-shot	80.05	58.76	60.10	58.31	59.45	45.83	18.90	17.50	32.09	19.92	30.27	69.71	45.91
4-shot	81.44	59.04	61.47	54.45	58.10	48.39	18.20	14.12	27.67	21.03	27.35	71.33	45.22
ICV	67.15	38.19	57.36	66.03	58.39	45.56	31.00	22.66	32.70	29.16	30.09	61.80	45.01

Table 6: Full comparison between ICV and standard ICL on Llama-2-7b with macro-F1 as the metric. The clean results shown here are averaged over 1,000 episodes.

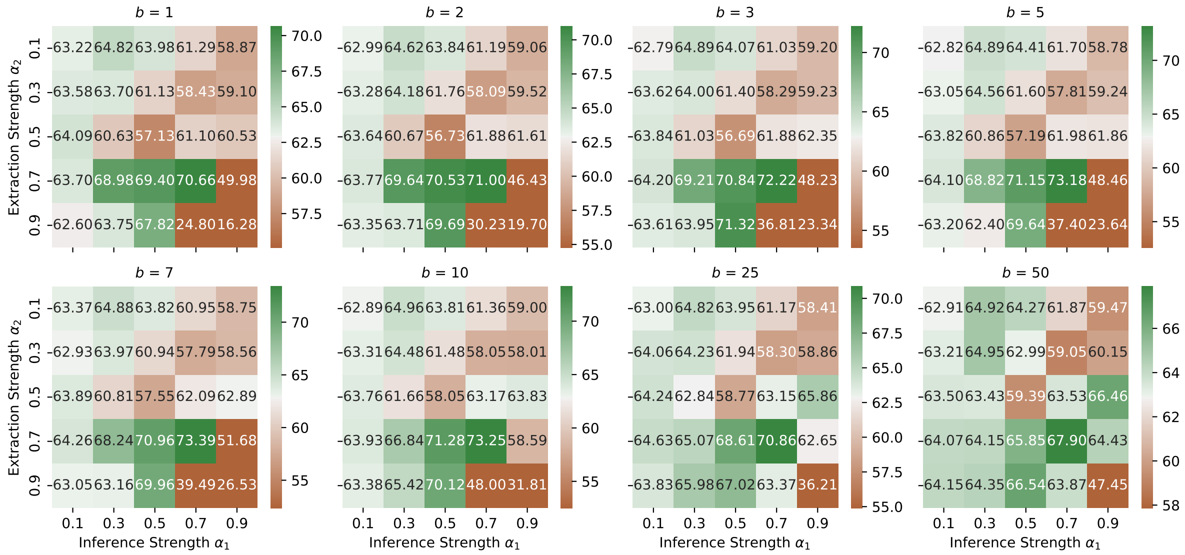

Figure 3: Ablation study on the extraction batch size b. Eight different values of b were evaluated using Llama-2-7b on the AG News dataset. The experiments were conducted under the conditions of 200 one-shot extraction episodes and 1,000 testing episodes. The white color in the heatmaps indicates the corresponding clean one-shot macro-F1 score (62.89, as presented in Table 2).