In-Context Learning State Vector with Inner and Momentum Optimization

Dongfang Li, Zhenyu Liu, Xinshuo Hu, Zetian Sun, Baotian Hu✉, Min Zhang
Harbin Institute of Technology (Shenzhen), Shenzhen, China
{crazyofapple, liuzhenyuhit}@gmail.com
{hubaotian, zhangmin2021}@hit.edu.cn

Abstract

Large Language Models (LLMs) have exhibited an impressive ability to perform In-Context Learning (ICL) from only a few examples. Recent works have indicated that the functions learned by ICL can be represented through compressed vectors derived from the transformer. However, the working mechanisms and optimization of these vectors are yet to be thoroughly explored. In this paper, we address this gap by presenting a comprehensive analysis of these compressed vectors, drawing parallels to the parameters trained with gradient descent, and introducing the concept of state vector. Inspired by the works on model soup and momentum-based gradient descent, we propose inner and momentum optimization methods that are applied to refine the state vector progressively as test-time adaptation. Moreover, we simulate state vector aggregation in the multiple example setting, where demonstrations comprising numerous examples are usually too lengthy for regular ICL, and further propose a divide-and-conquer aggregation method to address this challenge. We conduct extensive experiments using Llama-2 and GPT-J in both zero-shot setting and few-shot setting. The experimental results show that our optimization method effectively enhances the state vector and achieves the state-of-the-art performance on diverse tasks. Code is available at https://github.com/HITsz-TMG/ICL-State-Vector

1 Introduction

In-Context Learning (ICL) has emerged as a powerful capability in tandem with the scaling of large language models (LLMs) [Brown et al., 2020]. By simply conditioning on a few input-label pairs as demonstrations, LLMs yield a significant improvement and deliver remarkable results in various downstream Natural Language Processing (NLP) tasks [Wei et al., 2022, Liu et al., 2023a]. For example, a model prompted with the input *"gaot → goat, sakne → snake, brid →"* can produce the output *"bird"*. Given these successes, it is worthwhile to inquire about the exact internal working mechanisms of ICL. Considering the opaque operation of ICL within the auto-regressive transformer, it is plausible that ICL might function as a general mechanism that leverages both demonstrations and the query to yield the prediction [Dong et al., 2023].

Recently, some studies have found that the ICL mapping function exists in the outputs of the attention layers or attention heads [Liu et al., 2023b, Dai et al., 2023] when applying causal effects analysis on a different set of models and tasks, such as the task vector [Hendel et al., 2023] and the function vector [Todd et al., 2023]. These works show that the functionalities learned through ICL can be encapsulated in compressed vectors derived from transformers, which then can be used to intervene in the transformer to handle queries without demonstrations. This revelation suggests the potential mechanism of ICL that first utilises demonstrations to learn the mapping function from inputs to

✉Corresponding author.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

https://doi.org/10.52202/079017-0250

labels in shallow transformer layers, and then uses the ICL function in deeper transformer layers to make predictions [Hendel et al., 2023]. However, while these compressed vectors encapsulate learned information in a more condensed form and show significant promise in applying ICL, there still exists a considerable gap in understanding the operational mechanisms and optimization strategies of these vectors. This significant gap hinders the further grasping and utilization of ICL.

In this paper, we aim to bridge the existing gap by presenting a comprehensive analysis of compressed vectors. Specifically, we investigate their similarities with parameters trained via gradient descent and introduce the formulation of state vector that encapsulates the processing state of ICL stored in the attention activations. Building on the concept of state vector, and drawing insights from the model soup [Wortsman et al., 2022] and Polyak momentum-based gradient optimization algorithms [Qian, 1999, Sutskever et al., 2013], we propose inner optimization and momentum optimization strategies which are progressively applied to enhance the state vector. Moreover, we further exploit the demonstration compression capabilities of the state vector to address the practical challenges encountered when applying ICL in settings with multiple examples, where demonstrations are typically too lengthy for standard ICL, such as in the 100-shot setting which is prevalent in practice. Specifically, we introduce a divide-and-conquer aggregation method that effectively aggregates the ICL functions of these extensive examples. This approach enables us to scale up for processing extended examples by compressing them into a single state vector. We conduct extensive experiments using Llama-2 [Touvron et al., 2023] and GPT-J [Wang and Komatsuzaki, 2021] in both zero-shot and few-shot settings. The experimental results show that our method effectively enhances the state vector and achieves state-of-the-art performance on diverse tasks. This not only manifests the effectiveness of our approach but also paves the way for a more comprehensive understanding of ICL.

Our contributions are summarized as follows:

- We delve into the working mechanism of compressed vectors in ICL and highlight their similarities with parameters trained via gradient descent. Building on this observation, we propose the formulation of the state vector.
- We propose inner and momentum optimization to progressively refine the state vector as an efficient test-time adaptation. Additionally, we introduce a divide-and-conquer aggregation to effectively scale up to large numbers of examples.
- We show the practicality of our proposed methods across a wide range of tasks through extensive experiments. Our results also offer insights for future research aiming to fully understand the functionalities of ICL.

2 Related Work

Mechanistic Interpretability. Recent works have focused on the working mechanisms of ICL [Chan et al., 2022, Xie et al., 2022, Wang et al., 2023]. Olsson et al. [2022] argue that induction heads may be the mechanistic source of general ICL in transformers. Akyürek et al. [2022] show that transformer-based in-context learners can implicitly implement standard optimization algorithms on linear models. A mainstream assumption posits that ICL has a similarity with the gradient descent. von Oswald et al. [2023] demonstrate how a linear attention-only transformer model can perform a gradient descent-like procedure implicitly. Dai et al. [2023] compare standard gradient descent based fine-tuning and ICL, and figure out that the transformer attention of ICL exhibits a dual form of gradient descent-based optimization. Moreover, some works revisit and modify this theory on the layer causality dependence [Natan et al., 2023] or training batch size [Shen et al., 2023]. In contrast, we focus on the application of the dual form of gradient descent and ICL and present optimization methods with inspiration from the dual form.

Task Representation. Numerous studies have extensively explored the concept of compressing various tasks into task representations as a means of effectively manipulating tasks within ICL ability. Notably, Shao et al. [2023] and Mu et al. [2023] have successfully yielded compositional task representations by training a composition model. In a slightly different vein, some researchers have delved into the art of devising methodologies to compose minor parameter adjustments acquired through task fine-tuning [Ilharco et al., 2022, Panigrahi et al., 2023, Yu et al., 2023, Hu et al., 2024, Merullo et al., 2023]. An alternative line of research finds that the task representation could be extracted in ICL [Liu et al., 2023b, Hendel et al., 2023, Todd et al., 2023, Yang et al., 2023]. Different

from these approaches, our work avoids the need for additional training and focuses more on analysing why these compressed vectors work and how to improve their performance.

3 Formalization

In this section, we first provide a detailed examination of attention activation which is found to contain the compressed ICL function by previous works [Hendel et al., 2023, Todd et al., 2023]. Then, we highlight its inherent similarities with parameters trained through gradient descent. Finally, we introduce the concept of the state vector drawing inspiration from these observations.

A classic template of ICL has the following necessary components: (1) N examples that are used to form the demonstrations and each example contains an input query $\mathcal{X}$ and its corresponding label $\mathcal{Y}$. (2) Separate tokens $\mathcal{S}$ that separate the input query and the label for each example (e.g., $\rightarrow$). (3) A query $\mathcal{X}_q$ for prediction. With the above components, the contextual model input of ICL could be written as follows:

$$\mathcal{X}_1, \mathcal{S}, \mathcal{Y}_1, \mathcal{X}_2, \mathcal{S}, \mathcal{Y}_2, \cdots, \mathcal{X}_N, \mathcal{S}, \mathcal{Y}_N, \mathcal{X}_q, \mathcal{S}.$$

Here we analyse the attention activation of the last separate token. In the l-th transformer layer, the output activation $\mathbf{a}^l$ of the attention heads of the last separate token is:

$$\mathbf{a}^l = W_V[X'; X] \operatorname{softmax}\left(\frac{\left(W_K[X'; X]\right)^T \mathbf{q}}{\sqrt{d}}\right), \tag{1}$$

where X' denotes the hidden state of demonstrations, X denotes the hidden state of the query and the last separate token (called zero-shot input), q denotes the attention query vector of the last separate token, $[X'; X]$ denotes the matrix concatenation, $\sqrt{d}$ is the scaling factor, W_K and W_V are parameter weight matrix.

Consistent with previous works [Dai et al., 2023, Natan et al., 2023], we omit the softmax operation and the scaling factor to approximate standard attention as relaxed linear attention for qualitative analysis. Consequently, the activation can be simplified as follows:

$$\begin{aligned}
\mathbf{a}^l &\approx W_V[X'; X] \left(W_K[X'; X]\right)^T \mathbf{q} \\
&= \left(W_V X \left(W_K X\right)^T + W_V X' \left(W_K X'\right)^T\right) \mathbf{q} \\
&= \left(W_{\text{ZSL}} + \sum_i \left(\left(W_V \mathbf{x}'_i\right) \otimes \left(W_K \mathbf{x}'_i\right)\right)\right) \mathbf{q}.
\end{aligned} \tag{2}$$

We define $W_{\text{ZSL}} = W_V X \left(W_K X\right)^T$ as the initialized parameters since it is the attention result in the Zero-Shot Learning (ZSL) setting.

To draw a meaningful comparison between attention activation and parameters trained through gradient descent, we now shift our focus towards analyzing a simple linear transformation represented by $\mathbf{y}_i = W\mathbf{x}_i$. Given a loss function $\mathcal{L}$ and the learning rate η, the gradient of linear weight is:

$$\nabla_W \mathcal{L}(\mathbf{y}_i) = \frac{\partial \mathcal{L}(\mathbf{y}_i)}{\partial \mathbf{y}_i} \frac{\partial \mathbf{y}_i}{\partial W} = \nabla_{\mathbf{y}_i} \mathcal{L}(\mathbf{y}_i) \mathbf{x}_i^T. \tag{3}$$

Denoting the back-propagated errors as $\mathbf{e}_i = -\eta \nabla_{\mathbf{y}_i} \mathcal{L}$, we can get the full batch gradient with training examples:

$$\Delta W_{GD} = \sum_i \mathbf{e}_i \otimes \mathbf{x}'_i, \tag{4}$$

where $\mathbf{x}'_i$ is the input training examples. Hence, in the previous Eqn. 2, if we substitute $W_K \mathbf{x}'_i$ as training examples, and take $W_V \mathbf{x}'_i \approx \mathbf{e}_i$ corresponding to some meta gradients [Dai et al., 2023, Natan et al., 2023]. The activation can be written as:

$$\mathbf{a}^l = \left(W_{\text{ZSL}} + \sum_i \mathbf{e}_i \otimes W_K \mathbf{x}'_i\right) \mathbf{q} = \left(W_{\text{ZSL}} + \Delta W_{GD}\right) \mathbf{q}. \tag{5}$$

Hence, it can be inferred that the output activation $\mathbf{a}^l$ can be regarded as parameters trained via gradient descent which utilizes the demonstrations as training instances.

https://doi.org/10.52202/079017-0250

With the above dual form between activation and trained parameters, and in light of observations that transformers tend to learn the ICL function primarily in their first L layers [Wang et al., 2023], we have the following hypothesis: During the process of ICL, the first L layers progressively update the flow of information using each example in the demonstration through forward computation. The processing state of ICL is then stored within the activation of the attention head. The subsequent layers access and utilize the processing state to reinstate the ICL function, which is used implicitly for predicting the queries. Therefore we concatenate the activation in the initial L layers and introduce the notation of the state vector:

$$\mathcal{V}_N^L = \Big\Vert_{l=1}^{L} \mathbf{a}^l, \tag{6}$$

where L is the number of layers and N is the number of examples in the demonstration. $\Vert$ denotes the concatenation operation. Note that we have a completely different construction strategy and usage compared to the function vector [Todd et al., 2023]. Although the task vector [Hendel et al., 2023] may be functionally equivalent in the forward process, the proposed state vector differs significantly in its integration into the model, making it easier and more effective to analyse and interpret.

4 Method

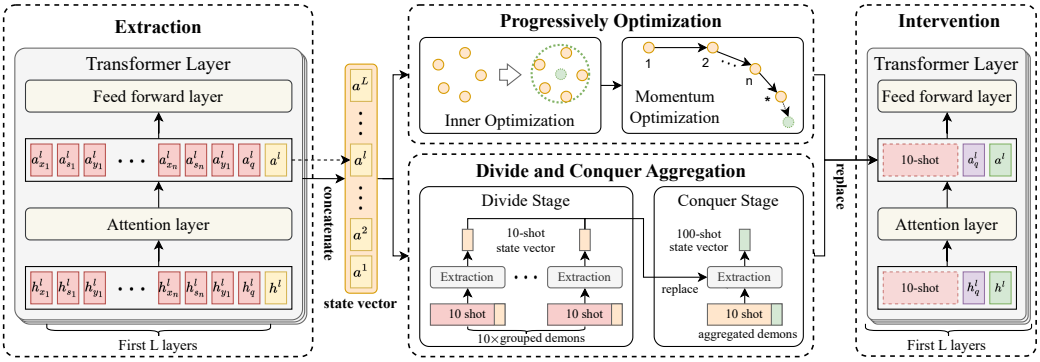


Figure 1: The overall framework of the proposed state vector. The state vectors are extracted from the output activations of attention heads. These state vectors are progressively optimized by *inner optimization* and *momentum optimization*, or be aggregated through a *divide-and-conquer (D&C) aggregation*. Finally, the processed state vector is utilized to intervene the inference forward pass.

4.1 Overview

As illustrated in Figure 1, our approach initially extracts the state vector from the attention head that corresponds to the final separate token in the first L layers using a demonstration and a dummy query. Then, with the view of treating the state vector as trained parameters, coupled with drawing inspiration from the model soup and the momentum-based gradient optimization algorithm, we introduce two methods that progressively optimize the state vector as test-time adaptation [Liang et al., 2023]: (1) inner optimization (§4.2) and (2) momentum optimization (§4.3). Moreover, we propose a divide-and-conquer (D&C) state vector aggregation method for efficiently compressing the ICL function in the multiple example setting (§4.4).

After the state vector optimization or aggregation, we utilize the processed state vector to intervene the model during the forward inference pass. In particular, we first input a test query in the zero-shot setting or with the demonstration in the few-shot setting. During the forward pass in the first L layers, we replace the attention activation of the last separate token with the corresponding activation in the state vector. In other words, the state vector is leveraged to intervene in the output of the first L transformer layers, blocking the attention of the last separate token to the previous context. With state vector intervention, the transformer learns the ICL function from the processing state stored in the state vector, and continues to make the prediction on the test query.

https://doi.org/10.52202/079017-0250

4.2 Inner Optimization

Inspired by the works on the model soup [Wortsman et al., 2022, Chronopoulou et al., 2023] which show that weight-space averaging not only yields performance improvement but also often enhances robustness, we thus ask the following research question (**RQ1**): *Is it possible to optimize our state vector using the model soup approach?* To explore this question, we propose an inner optimization method to improve the effectiveness and robustness of state vector. Specifically, we not only extract the state vector in each separate token of the dummy query but also extract the state vector from each example. Formally, with a forward pass in an N shot ICL setting, we extract the N state vector $\mathcal{V}_i^L$ ($1 \leq i \leq N$) from last N separate token. Subsequently, we apply a uniform averaging process to these state vectors as follows:

$$\overline{\mathcal{V}}_N^L = \frac{1}{N} \sum_{i=1}^{N} \mathcal{V}_i^L, \tag{7}$$

where $\overline{\mathcal{V}}_N^L$ is the inner optimized state vector, which can be directly utilized for inference intervention or serves as the initial state vector for later momentum optimization.

4.3 Momentum Optimization

Since we view the state vector as parameters trained gradually through demonstration examples, the difference between two state vectors with adjacent corresponding separate tokens can also be regarded as the influence of the middle example, akin to the gradient. Motivated by this understanding, coupled with extensive studies of the gradient optimization algorithm [Sutskever et al., 2013, Duchi et al., 2010, Loshchilov and Hutter, 2019], we direct our focus toward a simple momentum-based gradient optimization algorithm, seeking to answer the following research question (**RQ2**): *Can our state vector be optimized using momentum-based optimization algorithm?* To answer this question, we propose a momentum optimization. Formally, we first extract the influence of each example by subtracting two adjacent state vectors:

$$E_i^L = \mathcal{V}_i^L - \mathcal{V}_{i-1}^L, \tag{8}$$

where E_i^L is the influence of i-th ($1 < i \leq N$) example in the early L layer. Then, we apply the momentum gradient optimization algorithm to obtain optimized influence $\widetilde{E}_i^L$, and add it to the last state vector:

$$\widehat{\mathcal{V}}_N^L = \overline{\mathcal{V}}_N^L + \widetilde{E}^L = \overline{\mathcal{V}}_N^L + \texttt{opt}([E_i^L]_{i=1}^N), \tag{9}$$

where $\widehat{\mathcal{V}}_N^L$ is the momentum optimized state vector and $\overline{\mathcal{V}}_N^L$ is the inner optimized state vector. $\texttt{opt}(\cdot)$ denotes the momentum gradient optimization algorithm. We also explore various other gradient optimization algorithms in §6.1.

4.4 Divide-and-Conquer Aggregation

In addition to optimizing the state vector to more effectively represent the ICL function from a small number of examples, we also explore its capacity to encapsulate multiple examples within a single vector. However, regular ICL can not be directly used on multiple examples due to the context length limitation of current LLMs. This leads us to investigate the following question (**RQ3**): *Can we use the state vector to represent multiple examples that are unmanageable for regular ICL?* To address this question, we propose a divide-and-conquer method for state vector aggregation. As depicted in Figure 1, our approach involves distinct aggregation processes (i.e. the divide stage and the conquer stage). In the divide stage, examples are randomly divided into groups, termed grouped demonstrations. Within each group, a random example is selected to serve as a dummy query, which allows us to extract a group-specific state vector. In the conquer stage, these dummy queries are paired with their corresponding labels to form input-label pairs. From these input-label pairs, we form an aggregated demonstration, add an additional dummy query, and subsequently extract the aggregated state vector. It is worth noting that during the forward pass of aggregated state vector extraction, we utilise the group-specific state vector to intervene the attention activation of the separate tokens of their corresponding examples. The divide and conquer approach allows us to aggregate the ICL function of each grouped demonstration into its respective group-specific state vector, and subsequently aggregate the ICL function of each group-specific state vector into a single,

https://doi.org/10.52202/079017-0250

comprehensive aggregated state vector. This aggregated vector is then utilized for interventions during inference, similarly to the optimized state vector discussed in §4.2 and §4.3. Moreover, in the few-shot setting, the aggregated demonstrations are treated as inference demonstrations. The divide-and-conquer approach effectively circumvents the context-length constraints inherent in LLMs, thereby enabling a more effective and efficient aggregation of information across multiple examples.

5 Experiment

5.1 Setup

We conduct the evaluation across 12 datasets that encompass different domains.

- **Linguistics** includes Antonym [Nguyen et al., 2017], Capitalize, Present-Past, and Singular-Plural [Todd et al., 2023], focusing on transformations in the form or meaning of words.
- **Translation** is represented by the English-French [Lample et al., 2018] dataset, which involves translating English words into their French counterparts.
- **Knowledge** comprises Country-Capital [Todd et al., 2023], AG News [Zhang et al., 2015], Person-Sport, Person-Instrument, Person-Occupation, Product-Company, and Landmark-Country [Hernandez et al., 2023], which are centred around question-to-answer mappings for commonsense knowledge queries.

We employ *Llama-2-7B* and *GPT-J-6B* as our LLMs, chosen for their moderate model sizes, open-source and capability for ICL. We also provide the results with larger models (i.e., Llama-2-13B) in the Appendix H. We use Llama-2-7B as the default model unless otherwise specified. Our method is orthogonal to the choice of transformer-based decoder-only autoregressive LLMs.

For simplicity evaluation, we restrict to single-token output and use first output token accuracy as the evaluation metric as in previous work [Hendel et al., 2023, Todd et al., 2023].

5.2 Baseline

In the paper, we compare with the following methods:

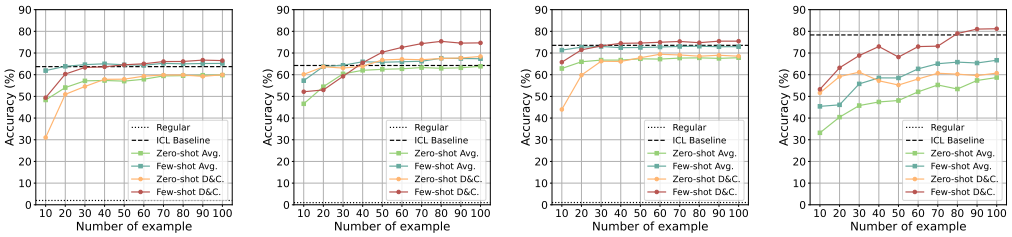
- **Regular** is the baseline for the zero-shot setting that uses only the given query as input, while **ICL baseline** [Wei et al., 2022] makes predictions on the label by taking both the demonstrations and the given query.
- **Function vector** [Todd et al., 2023] is extracted from attention activation using the causal mediation method and is then added to the hidden state of certain transformer layers during inference.
- **Task vector** [Hendel et al., 2023] is extracted from the hidden state of the separate token and is leveraged for blocking the layer when inference.

5.3 Inner Optimization(RQ1)

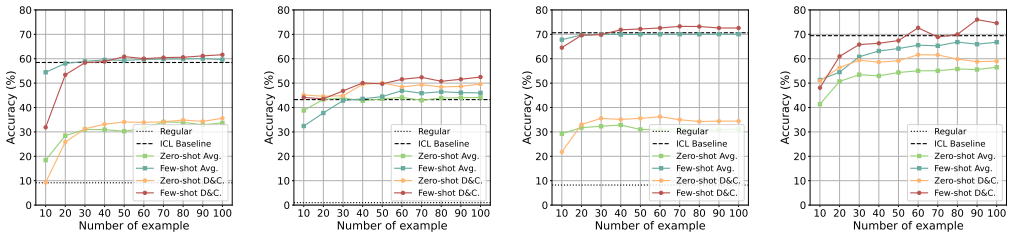
As shown in Table 1, the performance of our inner optimized state vector has a significant improvement comparing the task vector and function vector in both zero-shot and few-shot settings. Our state vector with inner optimization. In the zero-shot setting, the inner optimization shows an average improvement of 10.2% on Llama-2 and 5.9% on GPT-J across six datasets. In the few-shot setting, the inner optimization also achieves a 1.2% improvement on Llama-2 and 1.7% on GPT-J. The improvement demonstrates the effectiveness of inner optimization. However, although state vector (inn.) outperforms task vector, its few-shot performance on some datasets is inferior to the ICL baseline. We attribute this primarily to the introduction of query information from examples. While inner optimization enhances task-relevant information for the state vector, it also introduces noise of other dummy queries, hindering the model's ability to focus on the current predictive query, thereby reducing performance. In addition to the performance improvements, our inner optimization approach also effectively alleviates the phenomenon of high variance in the original task vector in the zero-shot setting. In practical use, the performance of the task vector is influenced by demonstrations and dummy queries, leading to weaker robustness. Our proposed inner optimization approach effectively

Model	Method		Anym	Eng-Fr	Pers-Inst	Pers-Occ	Prod-Comp	Land-Cout	Average
Llama-2-7B	Zero-shot	Regular	1.0± 0.2	0.1±0.1	0.0±0.0	0.0±0.0	0.4±0.2	0.0±0.0	0.3
		Function vector	45.1±2.0	21.6±2.0	11.3±10.7	0.1±0.1	25.6±4.3	32.9±21.6	22.8
		Task vector	56.2±2.8	63.2±3.6	61.8±8.4	27.9±15.2	55.5±20.1	57.8±26.3	53.7
		State vector (inn.)	61.0±1.0	66.5±2.2	67.4±2.6	42.7±4.2	64.5±10.6	81.0±1.7	63.9
		State vector (mom.)	60.4±0.7	67.5±1.8	68.7±1.6	45.6±5.9	71.3±3.6	77.7±1.8	65.2
	Few-shot	ICL baseline	64.8±4.8	74.3±0.8	71.7±3.7	56.1±2.7	80.8±0.8	87.0±0.3	72.5
		Function vector	54.5±0.9	65.2±1.4	60.8±5.6	54.2±2.2	76.0±1.3	84.2±2.9	65.8
		Task vector	65.7±1.8	73.8±0.9	66.6±5.2	56.4±2.3	81.9±1.8	86.7±0.9	71.8
		State vector (inn.)	**66.2**±1.6	**74.6**±0.9	70.1±4.3	57.0±2.2	**82.8**±1.6	87.5±0.9	73.0
		State vector (mom.)	65.8±3.7	74.3±1.1	**74.9**±2.9	**58.2**±0.4	82.0±1.0	**87.6**±0.3	**73.8**
GPT-J-6B	Zero-shot	Regular	8.1±0.6	7.2±0.6	0.0±0.0	0.0±0.0	1.9±0.5	0.9±0.2	3.0
		Function vector	33.1±1.8	29.1±8.5	4.1±5.8	11.1±2.3	46.3±5.7	22.5±10.2	24.4
		Task vector	23.6±3.8	32.2±5.1	44.4±5.0	28.3±18.6	43.8±5.7	41.3±12.3	35.6
		State vector (inn.)	33.4±1.9	31.7±3.8	49.3±2.0	30.0±6.2	42.8±4.3	61.9±1.6	41.5
		State vector (mom.)	31.1±1.0	35.1±2.4	50.3±3.0	42.4±1.5	44.2±1.5	60.3±0.9	43.9
	Few-shot	ICL baseline	59.2±1.4	69.9±2.0	44.7±6.7	29.3±1.0	62.5±1.0	**69.3**±0.5	55.8
		Function vector	56.4±1.9	65.8±1.9	49.1±2.2	30.3±1.9	58.5±3.3	69.2±0.6	54.9
		Task vector	58.5±1.6	70.6±1.2	42.3±6.4	27.8±3.3	66.0±2.6	63.1±5.3	54.7
		State vector (inn.)	58.7±2.2	**70.9**±1.3	46.5±4.9	29.4±1.7	**66.3**±2.1	66.4±2.8	56.4
		State vector (mom.)	**59.6**±1.4	70.1±2.2	**51.9**±2.4	**30.4**±1.1	63.8±0.8	68.6±0.3	**57.4**

Table 1: Performance of state vector optimization. The best results in the zero shot setting are in underline and the best results in the few shot setting are in **bold**. The result of basic state vector is mathematically equivalent to task vector. Note that we only present the results across six tasks here and leave the rest in the Appendix. We also report standard deviation and the results are passed with significance test ($p < .05$).



(a) Llama-2-7B Antonym (b) Llama-2-7B Person-Instrument (c) Llama-2-7B English-French (d) Llama-2-7B AG News



(e) GPT-J-6B Antonym (f) GPT-J-6B Person-Instrument (g) GPT-J-6B English-French (h) GPT-J-6B AG News

Figure 2: Performance of aggregation across number of examples. *Avg.* denotes the average aggregation baseline and *D&C.* denotes the divide-and-conquer aggregation. The **X** axis represents the number of examples, and the **Y** axis represents the accuracy.

mitigates this issue, similarly motivated as the model averaging method, thereby enhancing the robustness of the state vector.

5.4 Momentum Optimization (RQ2)

As depicted in Table 1, building upon the inner optimized state vector, our proposed momentum optimization algorithm further enhances the effectiveness of the state vector, achieving the best performance on average in all settings. In the zero-shot setting, the momentum optimization boosts the performance of the inner-optimized state vector with an average increase of 1.3% on Llama-2 and 2.4% on GPT-J. In the few-shot setting, state vector with momentum optimization achieves a 0.8% average increase on Llama-2 and 1.0% on GPT-J. This reveals the effectiveness of our momentum optimization. With the combination of inner optimization and momentum optimization, our state

https://doi.org/10.52202/079017-0250

Method	Zero-shot	Few-shot
ICL baseline	0.2± 0.4	71.0± 10.8
Task vector	52.9± 9.4	68.5± 10.5
State vector (mom.)	65.2± 10.2	72.2± 10.6
State vector (adag.)	11.7± 12.0	16.1± 10.2
State vector (rms.)	0.8± 0.9	1.5± 1.0
State vector (adam.)	6.7± 6.1	10.6± 8.5

Table 2: Performance comparison of gradient optimization algorithms. The method means the optimization algorithm applied to the $\texttt{opt}(\cdot)$ in Eqn. 9.

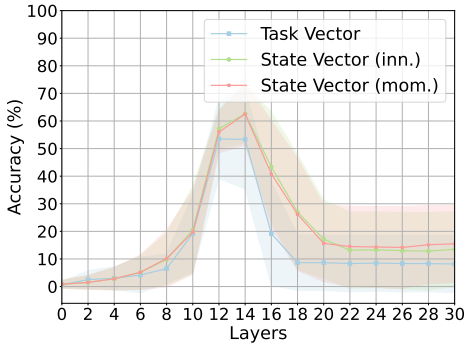


Figure 3: Average zero-shot performance across six datasets for each choice of the intermediate layer L. The solid line means the average value, while the shaded area indicates the standard deviation.

vector (mom.) surpasses the original variant, showcasing a remarkable improvement of 11.5% for Llama-2 and 8.3% for GPT-J in the zero-shot setting. In the few-shot setting, our state vector (mom.) still outperforms the task vector with a 2.0% improvement for Llama-2 and 2.7% for GPT-J. Furthermore, without inputting demonstration during inference, the state vector (mom.) achieves an impressive 90% ICL performance on Llama-2 and 78% ICL performance on GPT-J. When compared to ICL with the same examples as the demonstration, state vector (mom.) outperforms ICL in both Llama-2 and GPT-J. These improvements verify the effectiveness of our progressive optimization strategy. Note that applying momentum optimization directly to task vectors does not yield average improvements across tasks in our preliminary experiment. We speculate that this inconsistency stems from the poor robustness of the task vectors, which hinders the stable optimization by momentum optimization and leads to poor performance in some tasks.

5.5 Divide-and-Conquer Aggregation (RQ3)

In this experiment, we explore the performance of D&C state vector aggregation across varying numbers of examples. Besides the regular and ICL baseline mentioned, we introduce average aggregation as a strong baseline. This approach first extracts state vectors from the example group and subsequently employs their mathematical average for aggregation. We compare our D&C aggregation method with the baseline ranging from 10 to 100 examples across two models. Due to limited computational resources, we were not able to do an exhaustive search over all datasets. Thus, we only present the results for four tasks.

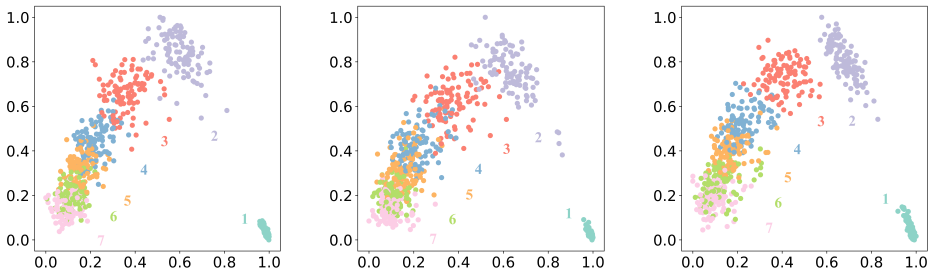
As illustrated in the Figure 2, both the D&C aggregation and average aggregation exhibit similar trends in both few-shot and zero-shot settings. The performance of both aggregation methods initially falls short of the ICL baseline. However, their performance boosts when examples increase. The initial poor performance can be attributed to the limited number of state vectors. Additionally, although the performance of the D&C aggregation initially falls behind that of the average aggregation, it exhibits a more substantial performance improvement when examples increase, ultimately outperforming average aggregation in the multiple example setting, highlighting the efficiency of D&C aggregation.

6 Analysis

6.1 Ablation with Other Optimization Methods

We present an ablation study to investigate various classical gradient optimization algorithms, aiming to delve deeper into the inner state vector optimization. We compare the momentum-based gradient optimization algorithm with following additional first-order gradient optimization algorithms: Adagrad (adag.) [Duchi et al., 2010], RMSprop (rms.) [Graves, 2013] and Adam(adam.) [Kingma and Ba, 2015].

https://doi.org/10.52202/079017-0250



(a) Antonym (b) English-French (c) Product-Company

Figure 4: The 2D PCA visualization of the state vector in the Antonym, English-French, and Product-Company tasks, where each color represents the state vector corresponding to examples occupying specific positions in the demonstration, with the position of each point indicated by the adjacent number.

As shown in Table 2, we observe a significant decrease in state vector performance when using first-order gradient optimization algorithms, in contrast to the more stable results obtained with momentum-based optimization. This discrepancy suggests that current first-order gradient optimization algorithms may not be well-suited for state vector optimization. We believe there are two main reasons for this. First, first-order gradient optimizers typically rely on adaptive learning rates, which depend on a significant amount of historical information. This can lead to instability and reduced effectiveness, especially when the available data is limited. Second, first-order gradient optimizers involve more complex hyper-parameters compared to the Momentum Optimizer, making it more difficult to identify optimal settings. Our experimental results confirm that directly applying hyper-parameter configurations commonly used in gradient descent leads to suboptimal performance in state vector optimization.

6.2 Layer Selection

We investigate the impact of layer selection on the extraction of state vectors in transformer models. We evaluate the average performance across different datasets in the zero-shot setting, as illustrated in Figure 3. Our results reveal a dual-phase trend: initially, increasing the number of layers for state vector extraction improves performance, but this improvement reverses beyond the 14th layer. We correlate this with the dynamics of ICL function processing in transformers in line with previous works [Voita et al., 2019, Wang et al., 2023]. In the initial layers, transformers are primarily engaged in learning and encapsulating the ICL function within state vector, where additional layers enhance the richness of the functional information in the state vector. In contrast, the later layers prioritize applying this learned information for prediction tasks. Here, additional layers tend to introduce noise, especially from predicted labels of dummy queries, which may negatively impact performance.

6.3 Qualitative Study

We provide the visualization by Principal Component Analysis (PCA) of the original state vector in the Antonym, English-French and Product-Company task. As depicted in Figure 4, we have three observations: (1) State vectors corresponding to the examples occupying the same position tend to form distinct clusters. This clustering pattern suggests a high degree of similarity among state vectors within each example position, despite different contexts. (2) A notable separation is evident between the state vectors originating from the first example and other position examples. This demarcation implies that ICL may begin to effectively function with a few examples. (3) An interesting trend is observable in the movement of these clusters as the example position increases. This trend may be indicative of an accumulation of task-specific information, where each additional example contributes to a more nuanced understanding of the model. These findings suggest a progressive enhancement in the ability of model to internalize and reflect the subtleties of the task at hand. Moreover, these observations reflect the efficacy of momentum optimization to leverage the observed clustering trend.

https://doi.org/10.52202/079017-0250

7 Conclusion

In this paper, we reveal that ICL compressed vector can be viewed as parameters trained through gradient descent on the demonstrations. Then, we introduce the concept of state vector coupled with two optimization methods to enhance the capability of ICL and conduct comprehensive experiments across two popular LLMs and multiple tasks to support our claim. Furthermore, our approach demonstrates the ability to compress context while maintaining lower variance. In the future, we aim to extend our methods to more complex ICL scenarios and apply them to larger LLMs and call for more nuanced and realistic studies of ICL.

Acknowledgements

This work is jointly supported by grants: National Natural Science Foundation of China (No. 62376067), National Natural Science Foundation of China (No. 62406088) and Guangdong Basic and Applied Basic Research Foundation (2023A1515110078).

References

Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. What learning algorithm is in-context learning? investigations with linear models. *ArXiv preprint*, abs/2211.15661, 2022.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared 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 M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Stephanie Chan, Adam Santoro, Andrew K. Lampinen, Jane Wang, Aaditya Singh, Pierre H. Richemond, James L. McClelland, and Felix Hill. Data distributional properties drive emergent in-context learning in transformers. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022*, 2022.

Alexandra Chronopoulou, Matthew Peters, Alexander Fraser, and Jesse Dodge. AdapterSoup: Weight averaging to improve generalization of pretrained language models. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 2054–2063, 2023.

Damai Dai, Yutao Sun, Li Dong, Yaru Hao, Shuming Ma, Zhifang Sui, and Furu Wei. Why can GPT learn in-context? language models secretly perform gradient descent as meta-optimizers. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 4005–4019, 2023.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. A survey for in-context learning. *ArXiv preprint*, abs/2301.00234, 2023.

John C. Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. In *COLT 2010 - The 23rd Conference on Learning Theory, Haifa, Israel, June 27-29, 2010*, pages 257–269, 2010.

Alex Graves. Generating sequences with recurrent neural networks. *ArXiv*, abs/1308.0850, 2013.

Roee Hendel, Mor Geva, and Amir Globerson. In-context learning creates task vectors. *ArXiv preprint*, abs/2310.15916, 2023.

Evan Hernandez, Arnab Sharma, Tal Haklay, Kevin Meng, Martin Wattenberg, Jacob Andreas, Yonatan Belinkov, and David Bau. Linearity of relation decoding in transformer language models. *ArXiv preprint*, abs/2308.09124, 2023.

Xinshuo Hu, Dongfang Li, Zihao Zheng, Zhenyu Liu, Baotian Hu, and Min Zhang. Separate the wheat from the chaff: Model deficiency unlearning via parameter-efficient module operation. In *Proc. of AAAI*, 2024.

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

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Machine Learning*, 2015.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In Jason Flinn, Margo I. Seltzer, Peter Druschel, Antoine Kaufmann, and Jonathan Mace, editors, *Proceedings of the 29th Symposium on Operating Systems Principles, SOSP 2023, Koblenz, Germany, October 23-26, 2023*, pages 611–626. ACM, 2023. doi: 10.1145/3600006.3613165.

Guillaume Lample, Alexis Conneau, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. Word translation without parallel data. In *International Conference on Machine Learning*, 2018.

Jian Liang, Ran He, and Tieniu Tan. A comprehensive survey on test-time adaptation under distribution shifts. *ArXiv preprint*, abs/2303.15361, 2023.

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

Sheng Liu, Lei Xing, and James Zou. In-context vectors: Making in context learning more effective and controllable through latent space steering. *ArXiv preprint*, abs/2311.06668, 2023b.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Machine Learning*, 2019.

Jack Merullo, Carsten Eickhoff, and Ellie Pavlick. Language models implement simple word2vec-style vector arithmetic. *ArXiv preprint*, abs/2305.16130, 2023.

Jesse Mu, Xiang Lisa Li, and Noah D. Goodman. Learning to compress prompts with gist tokens. *ArXiv preprint*, abs/2304.08467, 2023.

Tomer Bar Natan, Gilad Deutch, Nadav Magar, and Guy Dar. In-context learning and gradient descent revisited. *ArXiv preprint*, abs/2311.07772, 2023.

Kim Anh Nguyen, Sabine Schulte im Walde, and Ngoc Thang Vu. Distinguishing antonyms and synonyms in a pattern-based neural network. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 76–85, 2017.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. In-context learning and induction heads. *ArXiv preprint*, abs/2209.11895, 2022.

Abhishek Panigrahi, Nikunj Saunshi, Haoyu Zhao, and Sanjeev Arora. Task-specific skill localization in fine-tuned language models. In *International Conference on Machine Learning*, 2023.

Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural Networks*, 12(1): 145–151, 1999.

 https://doi.org/10.52202/079017-0250

Nan Shao, Zefan Cai, Hanwei Xu, Chonghua Liao, Yanan Zheng, and Zhilin Yang. Compositional task representations for large language models. In *International Conference on Learning Representations*, 2023.

Lingfeng Shen, Aayush Mishra, and Daniel Khashabi. Do pretrained transformers really learn in-context by gradient descent? *ArXiv preprint*, abs/2310.08540, 2023.

Ilya Sutskever, James Martens, George E. Dahl, and Geoffrey E. Hinton. On the importance of initialization and momentum in deep learning. In *Proc. of ICML*, volume 28 of *JMLR Workshop and Conference Proceedings*, pages 1139–1147, 2013.

Eric Todd, Millicent Li, Arnab Sen Sharma, Aaron Mueller, Byron C. Wallace, and David Bau. Function vectors in large language models. *ArXiv preprint*, abs/2310.15213, 2023.

Hugo Touvron, Louis Martin, and Kevin R. Stone et al. Llama 2: Open foundation and fine-tuned chat models. *ArXiv preprint*, abs/2307.09288, 2023.

Elena Voita, Rico Sennrich, and Ivan Titov. The bottom-up evolution of representations in the transformer: A study with machine translation and language modeling objectives. In *Proc. of EMNLP*, pages 4396–4406, 2019.

Johannes von Oswald, Eyvind Niklasson, E. Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, 2023.

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

Lean Wang, Lei Li, Damai Dai, Deli Chen, Hao Zhou, Fandong Meng, Jie Zhou, and Xu Sun. Label words are anchors: An information flow perspective for understanding in-context learning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 9840–9855, 2023.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models. *Trans. Mach. Learn. Res.*, 2022, 2022.

Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 23965–23998, 2022.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An explanation of in-context learning as implicit bayesian inference. In *International Conference on Machine Learning*, 2022.

Jiaxi Yang, Binyuan Hui, Min Yang, Binhua Li, Fei Huang, and Yongbin Li. Iterative forward tuning boosts in-context learning in language models. *ArXiv preprint*, abs/2305.13016, 2023.

Le Yu, Yu Bowen, Haiyang Yu, Fei Huang, and Yongbin Li. Language models are super mario: Absorbing abilities from homologous models as a free lunch. *ArXiv preprint*, abs/2311.03099, 2023.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 649–657, 2015.

A Implementation Details

In this paper, we use random sampling to create subsets for each dataset. Each subset consists of 10 instances for demonstrations and one instance for a dummy query since we employ a 10-shot as the default ICL setting. The remaining instances are split into test and development sets with a 7:3 ratio. For experiments with multiple examples, we sample 100 instances instead of 10. We use "$\rightarrow$" as the separate token similar to previous works. We tried other tokens but no significant difference. All the experiments are reported over 5 random seeds. The inference mechanism with state vector we describe in §4.1 has a key hyper-parameter (i.e.the layer L). Previous studies [Hendel et al., 2023] have shown that the choice of L has an influence on performance. We find the best layer for different tasks via the accuracy of the development set. For the inner optimization in §4.2, we choose the last seven state vectors to optimize. This is because the early state vectors yield subpar performance, primarily due to limitations in the available examples. For the momentum optimization, we choose 0.5 as the retention rate for historical momentum from the options of 0.25, 0.5 and 0.75. We run all the experiments on a single NVIDIA A100 80G GPUs. Each of our experiments consumes between 10 minutes to 8 hours of GPU time, depending on the dataset.

B More Details about Baseline

In this section, we present an in-depth and comprehensive analysis of two baselines (i.e. task vector [Hendel et al., 2023] and function vector [Todd et al., 2023]). Furthermore, we offer a more nuanced comparison with our proposed state vector, highlighting the distinct differences and advantages of our approach.

The task vector is designed to extract the ICL function from a specific layer's hidden state within the transformer model. This is achieved by directly replacing the corresponding hidden state during inference for intervention. On the other hand, Todd et al. [2023] first extracts the ICL function from the output activations across all attention heads in all transformer layers. These activations are then prioritized based on their causal effect, quantified by the variance in the model's output space with or without individual activation interventions. The mathematical average of the top 10 causal effect activations is the function vector, which is subsequently added to the hidden state of a specific layer during the inference stage.

In contrast to these methods, our approach for state vector extraction focuses on procuring the ICL procession state from the output activations of the attention heads within the first L layers. During inference, we replace the corresponding activations with optimized ones. While functionally equivalent to the forward process of the task vector when disregarding state vector optimization (i.e., the vanilla state vector), our approach offers enhanced mechanical explainability. This is attributable to its motivation from the dual form of in-context learning and gradient descent, as explicated in previous work [Dai et al., 2023, Natan et al., 2023]. Furthermore, inspired by the dual form, we focus on the further optimization process. On the other hand, unlike the function vector which extracts activations based on the causal effects resulting from individual interventions, our method is rooted in the underlying mechanisms of ICL. This strategy not only improves mechanical explainability but also demonstrates greater performance as evidenced by extensive experiments. Experiments also show notably poor performance of the function vector on certain knowledge-based datasets, such as Person-Occupation.

C More Details about Datasets

Here, we describe in detail the tasks that we use to evaluate the state vectors.

- **Antonym** [Nguyen et al., 2017] contains 2398 word pairs that are antonyms of each other (e.g. "massive" $\rightarrow$ "tiny"). We apply the dataset processed version from the function vector [Todd et al., 2023]. They filter the word pairs where both words can be tokenized as a single token.
- **Capitalize** [Todd et al., 2023] contains 813 word pairs that capitalize the first letter of the given input word (e.g. "plan" $\rightarrow$ "Plan").
- **Present-Past** [Todd et al., 2023] contains 293 word pairs, where simple past tense verbs are output when given simple present tense verbs (e.g. "adapt" $\rightarrow$ "adapted").

https://doi.org/10.52202/079017-0250

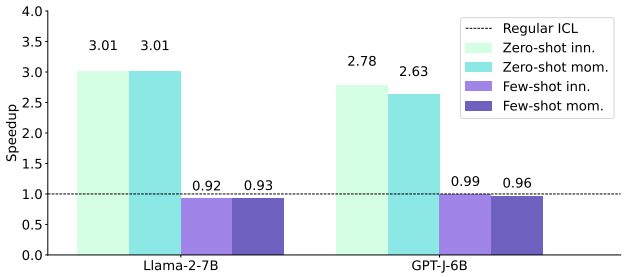


Figure 5: Time efficiency analysis of Llama-2-7B and GPT-J-6B. Inn denotes our state vector with inner optimization. Mom denotes our state vector with momentum optimization

- **Singular-Plural** [Todd et al., 2023] contains 205 word pairs, where the plural form of a given singular word (e.g., "wallet" → "wallets").
- **English-French** [Lample et al., 2018] contains 4698 pairs of words, which consists of a word in English and its translation into French (e.g., "circle" → "cercle"). We apply the processed version from the function vector[Todd et al., 2023].
- **Country-Capital** [Todd et al., 2023] contains 197 instances, which output the name of the capital city of the given country (e.g. "Luanda" → "Angola").
- **AG News** [Zhang et al., 2015] contains 7600 instances. Each instance contains the news headlines and the first few sentences of an article as input, and output corresponding labels include Business, Science, Sports, and World.
- **Person-Sport** [Hernandez et al., 2023] contains 318 instances. Each instance contains the name of a professional athlete and the sport that they play (e.g. "Hank Aaron" → "basketball").
- **Person-Instrument** [Hernandez et al., 2023] contains 510 instances. Each instance contains the name of a professional musician and the instrument they play (e.g. "Tom Fletcher" → "guitar").
- **Person-Occupation** [Hernandez et al., 2023] contains 821 instances. Each instance contains the name of a well-known individual and their occupation (e.g. "Tom Fletcher" → "guitar").
- **Product-Company** [Hernandez et al., 2023] contains 522 instances. Each instance contains the name of a commercial product and the company that sells the product (e.g. "Tom Fletcher" → "guitar").
- **Landmark-Country** [Hernandez et al., 2023] contains 836 instances. Each instance contains the name of a landmark and the country in which it is located.

D Efficiency Analysis

In this section, we present an efficiency analysis of two proposed optimization methods. We evaluate the average inference time using 1000 test data on a single NVIDIA A100 (80G) GPU, covering six main datasets and 10 random seeds per dataset. The results are illustrated in Figure 5. In the zero-shot setting, we compress the ICL function into the state vector which eliminates the need to concatenate demonstrations during inference. As shown in the Figure 5, the proposed inner optimization and momentum optimization, which, while tripling the inference speed, achieve 89% of the regular ICL performance on Llama-2-7B and 78% on GPT-J-6B (see Table 1 in the paper). In the few-shot setting, the proposed inner optimization and momentum optimization achieve better results than standard ICL at the cost of a minimal loss in inference speed (e.g., 99% and 96%). Moreover, our method is orthogonal to attention speedup techniques, such as flash attention [Dao et al., 2022] and page attention [Kwon et al., 2023]. Therefore, our approach can also benefit from the achievements of these works and achieve further efficiency improvement. We leave the exploration of alternative enhancement as future work.

Prompt	Llama-2-7B	+SV
What instrument did X play?	8.7± 0.7	67.3± 2.8
Can you tell me which musical instrument was played by X?	25.1± 0.7	69.0± 4.3
What was the primary instrument of X in their music career?	17.3± 1.4	70.3± 2.9

Table 3: Text portability of momentum optimized state vector. The templates are provided with "X" replaced by a query word. "+SV" denotes adding momentum optimized state vector

English-French	
Prompt	What is the meaning of biography?
Llama-2-7B	A written account of someone's life.
+ state vector	It is biographie.
Antonym	
Prompt	When I think of upright, I think of
Llama-2-7B	I think of a person who is standing up for what they believe in.
+ state vector	I think of down.

Table 4: Natural prompt cases with momentum optimized state vector on Antonym task and English-French task.

E Natural Text Completions

In this study, we evaluate the effectiveness of the momentum optimized state vector on natural text completions. Given a natural text template, we instruct the model to greedily generate 5 tokens with or without intervention in the zero-shot setting. We use exact match accuracy as the metric. Table 3 shows the result of natural text completions on Llama-2. The performance boosts observed with the momentum-optimized state vector on the separate tokens indicate that it can guide the model to generate answers correctly. We include more examples of natural text completions in the Appendix.

F Case Study

In this section, we present a case study shown in Table 4, to demonstrate the efficacy of the momentum-optimized state vector in natural text completions. Consider the query: "What is the meaning of biography?", The vanilla Llama-2 model would directly answer this question. However, when influenced by an English-French state vector, Llama-2 changes its response, translating the question into French instead. Similarly, when presented with the sentence "When I think of upright, I think of". Influenced by an Antonym state vector, Llama-2 completes the sentence with an anonymous pattern. These instances exemplify the model learning the ICL function stored in the momentum optimized state vector, enabling it to generate context relevant to the specified task.

G Full Result

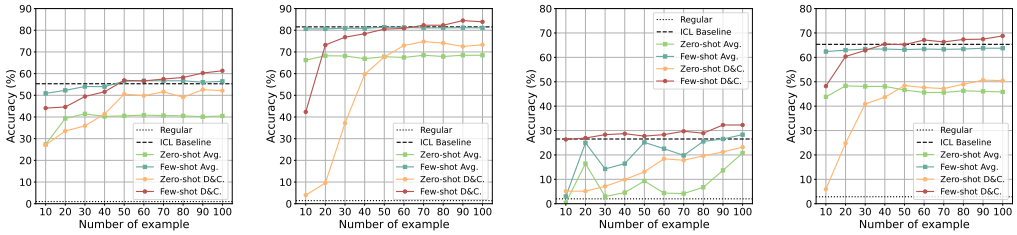
In this section, we provide the additional result with llama-2-7B GPT-J model. We first present the main result of optimization on the other six tasks except the main result, and the average performance across all tasks. As shown in Table 5, our inner optimization and momentum optimization effectively enhance the state vector.

Moreover, we provide the result of state vector aggregation on two additional datasets. As shown in Figure 6, the trends of both D$C and average aggregation follow a similar pattern to the main result shown in Figure 2 as the number of examples increases, illustrating the effectiveness of our aggregation methods.

https://doi.org/10.52202/079017-0250

Model	Method		Capitalize	Country-Capital	Present-Past	Singular-Plural	Person-Sport	AG News	Average (All)
Llama-2-7B	Zero-shot	Regular	0.0± 0.0	0.0± 0.0	0.0± 0.0	0.0± 0.0	0.0± 0.0	0.0± 0.0	0.2
		Function vector	98.6± 0.4	67.4± 20.7	80.2± 4.5	94.2± 0.6	1.4± 0.5	57.7± 0.9	44.7
		Task vector	92.9±6.5	92.8±2.8	95.2±1.7	95.3±1.9	86.9±4.5	47.8±1.3	69.4
		State vector (inn.)	99.6±0.4	94.0±1.3	96.5±1.2	97.1±1.0	89.7±3.2	52.0±5.5	76.0
		State vector (mom.)	99.1±0.3	94.5±0.7	96.5±0.7	96.6±1.0	88.1±2.6	50.0±8.3	76.3
	Few-shot	ICL baseline	**99.9**±0.1	**95.2**±1.0	**98.3**±0.6	98.5±0.1	94.8±0.2	76.0±5.7	83.1
		Function vector	99.7± 0.1	82.2± 3.8	94.6± 1.7	97.3± 0.7	88.4± 1.9	**80.7**±4.6	78.1
		Task vector	98.0±1.0	92.9±3.4	98.2±0.5	98.5±1.3	95.4±0.4	64.3±8.4	81.5
		State vector (inn.)	99.7±0.1	94.4±1.3	**98.3**±0.6	98.5±0.4	95.2±0.2	76.0±8.5	83.3
		State vector (mom.)	99.3±0.1	94.9±0.7	**98.3**±0.6	**98.8**±0.3	**95.7**±0.2	76.3±5.9	**83.8**
GPT-J-6B	Zero-shot	Regular	0.3± 0.1	1.8± 1.7	19.4± 2.1	22.7± 2.9	0.0± 0.0	0.0± 0.0	5.2
		Function vector	66.3± 8.4	57.0± 9.9	63.1± 2.1	69.3± 2.1	0.8± 1.1	46.4± 4.5	37.4
		Task vector	51.0±4.7	31.6±4.8	37.0±5.3	61.6±1.2	46.4±4.0	55.0±3.7	41.4
		State vector (inn.)	58.2±1.3	45.5±8.3	47.3±2.0	61.9±0.7	51.7±1.8	59.7±5.4	47.8
		State vector (mom.)	58.6±0.8	52.9±6.1	45.9±0.2	62.5±0.7	51.4±1.4	61.3±4.8	49.7
	Few-shot	ICL regular	99.3±0.3	88.2±3.4	96.9±0.9	99.3±0.5	82.4±3.5	76.3±1.7	73.1
		Function vector	98.6± 0.6	78.6± 5.1	90.8± 1.3	95.9± 0.9	81.6± 1.4	72.7±3.2	70.6
		Task vector	99.3±0.3	89.8±2.8	97.3±1.0	99.3±0.5	83.3±3.6	63.3±8.7	71.7
		State vector (inn.)	**99.4**±0.3	89.2±3.6	97.3±0.8	99.3±0.5	**83.8**±3.5	75.7±1.2	73.6
		State vector (mom.)	**99.4**±0.2	**90.1**±3.5	**97.6**±0.9	**99.4**±0.3	83.7±3.0	**78.0**±2.2	**74.4**

Table 5: Additional performance of state vector optimization across other six tasks and average performance of all task. The best results in the zero shot setting are in underline and the best results in the few shot setting are in **bold**. The result of basic state vector is mathematically equivalent to task vector. *Average (ALL)* represents the average performance across all 12 datasets.



(a) Llama2-7B Person-Occupation (b) Llama2-7B Product-Company (c) GPT-J-6B Person-Occupation (d) GPT-J-6B Product-Company

Figure 6: Additional performance of aggregation across number of examples. *Avg.* denotes the average aggregation baseline and *D&C.* denotes the divide-and-conquer aggregation. The **X** axis represents the number of examples, and the **Y** axis represents the accuracy.

H Result on Larger Model

In this section, we provide the optimization and aggregation results on the larger model. Here we choose Llama-2-13B and Llama-2-70B (quantized by GPTQ) as their memory requirements suit our hardware conditions. We present the result of the optimization method on three representative datasets shown in Table 6, and the result of the aggregation method on four representative datasets shown in Figure 7 and Figure 8. The result shows that our inner and momentum optimization and D&C aggregation method could also benefit the state vector on the larger model setting.

I Qualitative Study

In Figure 9, we present a Principal Component Analysis (PCA) visualization of the original state vector in GPT-J, applied to both the Antonym task and the English-French translation task. Note that the cluster distributions observed in GPT-J closely mirror those of Llama-2. This similarity indicates a consistent and progressive enhancement in the model capacity, as originally identified in Llama-2 in §6.3, which is also shown on GPT-J. Such findings demonstrate the broad applicability and generalizability of our momentum optimization approach across different models.

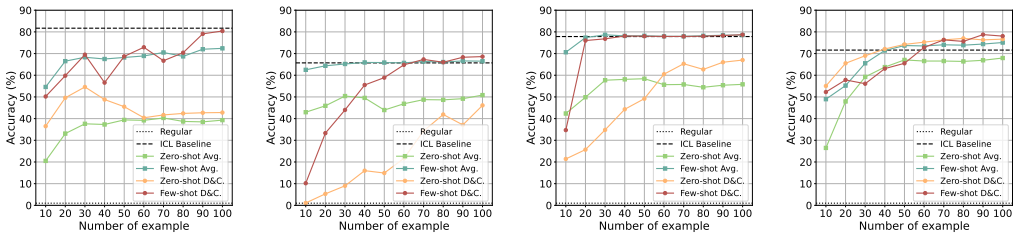
J Robustness Analysis

In this appendix, we examine the robustness of the state vector with inner optimization. Specifically, we evaluate the task vector and the inner optimized state vector on the Llama-2 dataset, focusing on three tasks. We measure and report the performance standard deviation using 100

https://doi.org/10.52202/079017-0250

Model	Method		Antonym	English-French	Person-Instrument	Average
Llama-2-13B	Zero-shot	Regular	1.2± 0.7	0.2± 0.2	0.0± 0.0	0.5
		Function vector	47.1± 1.6	23.2±4.3	0.1± 0.1	23.5
		Task vector	46.0± 2.4	43.1±7.2	58.2±6.3	49.1
		State vector (inn.)	47.0± 1.2	50.5±1.9	66.6±3.1	54.7
		State vector (mom.)	47.9± 1.1	55.9±3.4	68.5±2.0	57.4
	Few-shot	ICL baseline	**67.0± 0.1**	74.5±1.3	75.0±0.2	72.2
		Function vector	65.7± 1.7	75.2±2.6	72.2±0.4	71.3
		Task vector	64.8± 1.2	70.5±3.5	70.6±3.1	68.6
		State vector (inn.)	65.5± 0.8	**75.8±1.6**	77.0±1.3	72.8
		State vector (mom.)	65.9± 0.7	75.6±0.4	**78.6±1.1**	**73.4**
Llama-2-70B	Zero-shot	Regular	2.4±1.5	0.6±0.5	0.0± 0.0	1.0
		Function vector	48.3±9.5	29.9±0.3	5.3±0.3	27.8
		Task vector	63.3±2.1	48.8±9.5	63.5±6.9	58.5
		State vector (inn.)	65.5±1.0	56.0±6.6	67.3±5.6	62.9
		State vector (mom.)	66.7±1.1	57.8±5.3	71.9±4.9	65.5
	Few-shot	ICL baseline	**68.6±2.8**	81.6±1.4	80.6±1.9	76.9
		Function vector	64.8±2.3	81.7±2.1	82.8±4.1	76.4
		Task vector	67.1±2.8	81.5±1.8	82.2±4.7	76.9
		State vector (inn.)	68.0±2.8	**82.9±1.9**	82.6±2.3	77.8
		State vector (mom.)	68.5±2.2	82.5±2.1	**83.2±2.6**	**78.1**

Table 6: Performance of state vector optimization across three tasks on Llama-2-13B and Llama-2-70B. The best results in the zero shot setting are in underline and the best results in the few shot setting are in **bold**. The result of basic state vector is mathematically equivalent to task vector.



(a) AG News (b) Antonym (c) English-French (d) Product-Company

Figure 7: Performance of aggregation on Llama-2-13B across number of examples. *Avg.* denotes the average aggregation baseline and *D&C.* denotes the divide-and-conquer aggregation. The **X** axis represents the number of examples, and the **Y** axis represents the accuracy.

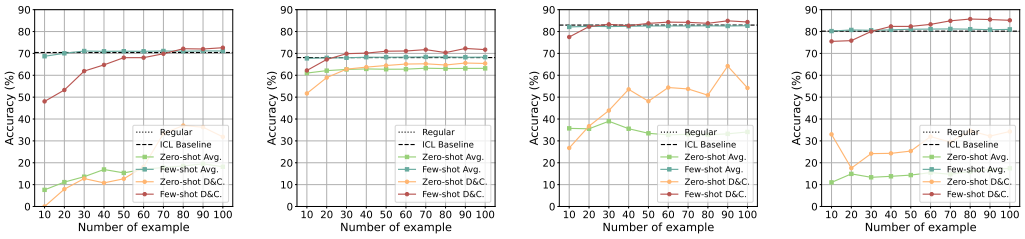
diverse demonstrations or dummy queries. As illustrated in Figure 10, our analysis yields three key observations:

- The task vector and state vector exhibit greater sensitivity to dummy queries than to demonstrations. This finding suggests that dummy queries have a greater impact on performance compared to demonstrations, underscoring the importance of reducing the noise from dummy queries to enhance state vector performance.

- In the few-shot setting, both the task vector and the state vector (inn.) indicate significantly greater robustness compared to their performance in the zero-shot setting. There is a noticeable reduction in the standard deviation across diverse demonstrations or dummy queries when applying demonstrations during ICL inference. This improvement may be attributed to the richer ICL function information provided by demonstrations, which in turn bolsters performance stability.

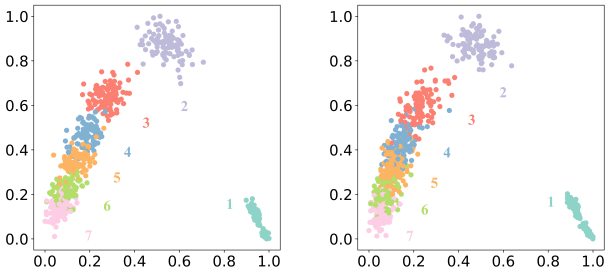
- Compared to the task vector, our inner optimized state vector shows markedly enhanced robustness to the variations in demonstrations and dummy queries, in both zero-shot and few-shot settings. This highlights the effectiveness of our proposed inner optimization in improving the robustness of the state vector.

https://doi.org/10.52202/079017-0250



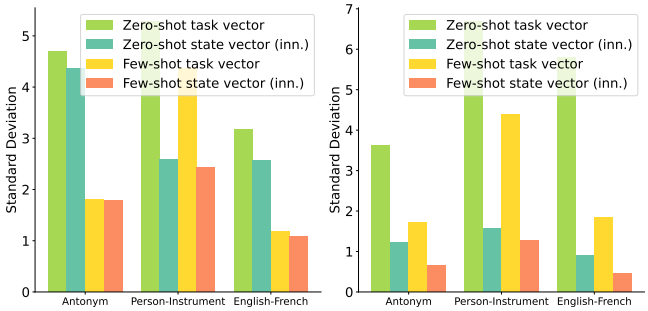
(a) Person-Occupation (b) Antonym (c) English-French (d) Person-Instrument

Figure 8: Performance of aggregation on Llama-2-70B across number of examples. *Avg.* denotes the average aggregation baseline and *D&C.* denotes the divide-and-conquer aggregation. The **X** axis represents the number of examples, and the **Y** axis represents the accuracy.



(a) Antonym (b) English-French

Figure 9: The 2D PCA visualization of the state vector in the Antonym task and English-French task of **GPT-J**, where each color represents the state vector corresponding to examples occupying specific positions in the demonstration and the outlier is of the first order.



(a) Demonstration Robustness (b) Dummy Robustness

Figure 10: Standard deviation of performance on Llama-2 across three datasets.

K Limitation

The definition of state vectors is contingent upon specific assumptions and lacks a rigorous theoretical foundation, which may impact its generalizability and reliability across different NLP tasks. Additionally, the experiments were conducted on a limited scale with moderate-sized models and datasets. These constraints may affect the applicability of the results to larger models or more complex datasets. Further research will explore these aspects to establish a more robust validation of the proposed methods.

https://doi.org/10.52202/079017-0250

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: we provide our discussion of limitations in the Appendix K.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

https://doi.org/10.52202/079017-0250

Justification: the paper does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: the paper fully disclose all the information and details of main experimental results in the Section 5.1 and Appendix A

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: we provide the instructions, data, codes and scripts of the main experimental results in the supplementary material.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: we provide all the training and test detailed description of the experimental setup and details in Section 5.1 and Appendix A.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: we provide the standard error of the main results in Table 1, Table 5, Table 6, Table 3.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

https://doi.org/10.52202/079017-0250

- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: we provide the information on the computer resources in Appendix A.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: the research conducted in the paper fully conforms to the NeurIPS Code of Ethics in every respect.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: there is no societal impact of the work performed.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: the paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: all creators and original owners of assets used in the paper, including code, data, and models, are properly credited, and the licenses and terms of use are explicitly mentioned and fully respected.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

https://doi.org/10.52202/079017-0250

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: the paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: this paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: this paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.