Iterative Forward Tuning Boosts In-Context Learning in Language Models

Jiaxi Yang[1,2,*,‡], **Binyuan Hui**[3,*], **Min Yang**[1†], **Bailin Wang**[4]
Bowen Li[5], **Binhua Li**[3], **Fei Huang**[3], **Yongbin Li**[3†]
[1] Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences
[2] University of Chinese Academy of Sciences
[3] Alibaba Group, [4] MIT CSAIL, [5] Shanghai AI Laboratory
{jx.yang, min.yang}@siat.ac.cn
binyuan.hby@alibaba-inc.com
https://github.com/Yangjiaxi/DeepThinking

Abstract

Despite the advancements in in-context learning (ICL) for large language models (LLMs), current research centers on specific prompt engineering, such as demonstration selection, with the expectation that a single iteration of demonstrations processing can generalize effectively to a given test sample. However, this perspective overlooks the potential benefits derived from multiple iterations involving demonstrations, a practice aligning more closely with the iterative decision-making process exhibited by humans, who often learn through analogy. In this study, we introduce a novel two-stage framework to boost ICL in LLMs. Specifically, our framework delineates the ICL process into two distinct stages: *Deep-Thinking* and test stages. The *Deep-Thinking* stage incorporates a unique attention mechanism, i.e., iterative enhanced attention, which enables multiple rounds of information accumulation. This mechanism operates by manipulating the Key-Value matrices without training, fostering enhanced understanding capabilities in LLMs by "*thinking*" demonstrations multiple times. We evaluated *Deep-Thinking* across a range of benchmarks and LLMs, showing its superior performance over vanilla ICL methods and its effectiveness in challenging tasks where demonstration selection is infeasible.

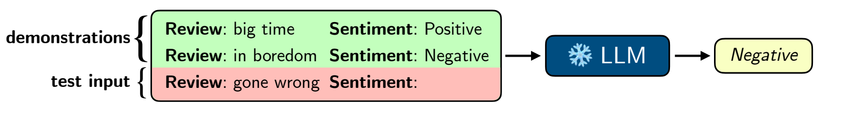
1 Introduction

Large language models (LLMs), e.g. OpenAI GPTs (OpenAI, 2023), LLaMA (Touvron et al., 2023) and Qwen (Bai et al., 2023), demonstrate the mysterious in-context learning (ICL) ability, where LLMs make predictions directly by prepending demonstrations to the original input without updating model parameters. LLMs are expected to learn the patterns hidden in demonstrations and make predictions accordingly. As illustrated in Figure 1 (a), an LLM can correctly perform inference on an unseen task by conditioning on several demonstrations. The ICL paradigm empowers LLMs to achieve impressive results in various downstream tasks with a few demonstrations, making Language-Model-as-a-Service (LMaaS) (Sun et al., 2022) possible.

Since the performance of ICL is sensitive to specific prompt settings, considerable efforts have been developed to improve the performance of ICL by refining the prompt design from different perspectives, such as demonstration selection (Liu et al., 2022; Li and Qiu, 2023), instruction design (Wei et al., 2022a; Ye et al., 2023), and intermediate chain-of-thought (CoT) reasoning (Wei et al., 2022b; Zhang et al., 2023; Lu et al., 2023). These methods can facilitate LLMs to reduce inference variance and avoid poor worst-case accuracy to some extent by performing prompt engineering. The working mechanism of ICL also draws a lot of attention. Dai et al. (2023) shed light on the connections between ICL and explicit fine-tuning. Specifically, ICL computes meta-gradients via forward computation, while explicit fine-tuning obtains gradients by back-propagation. A dual form exists between attention and gradient descent-based optimization (Irie et al., 2022a), directly connecting the test input to demonstrations. Wang et al. (2023a) argue that label words in demonstrations act as anchors, enabling mapping from demonstrations to test input through information aggregation and label propagation.

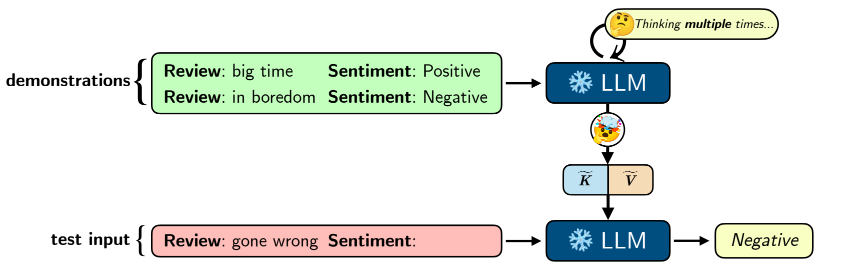
However, these studies assume that the models process demonstrations only once (i.e., perform a single forward computation), which is incoordinate with the human decision-making process by learning from analogy. Humans usually learn from analogy via an iterative thinking process, such as analyzing demonstrations, reflecting on them, and forming abstract concepts. The models

[*] Equal contribution.
[‡]Work done during an intern at Alibaba Group.
[†]Corresponding authors.



demonstrations
Review: big time Sentiment: Positive
Review: in boredom Sentiment: Negative
test input
Review: gone wrong Sentiment:
LLM
Negative

(a) In-Context Learning



Thinking multiple times...
demonstrations
Review: big time Sentiment: Positive
Review: in boredom Sentiment: Negative
LLM
$\widetilde{K}$
$\widetilde{V}$
test input
Review: gone wrong Sentiment:
LLM
Negative

(b) Ours: *Deep-Thinking*

Figure 1: The illustrations of vanilla ICL and our proposed two-stage framework through *Deep-Thinking*. The vanilla ICL method processes demonstrations only once, while our "*Deep-Thinking*" method enables multiple rounds of information accumulation during the reasoning process.

learned from demonstrations in inference time by "*thinking for longer*" or "*thinking multiple times*" (Schwarzschild et al., 2021). These findings inspire us to ask a question: ***Can we boost the performance of ICL by learning from demonstrations through several (iterative) forward inferences?***

In this paper, we propose a two-stage framework to boost the ICL ability in LLMs. Instead of simply concatenating demonstrations and test input together for inference, we decouple the ICL process into a *Deep-Thinking* stage for demonstration training and a test stage, as illustrated in Figure 1 (b). In the *Deep-Thinking* stage, we introduce a new attention module that manipulates the updates of Key-Value matrices (Vaswani et al., 2017) within the Transformer's self-attention (Vaswani et al., 2017) mechanism. This modification leverages Key-Value matrices as a bridge to change the information flow to accumulate and learn information over multiple forward iterations without any training. During the test stage, since the concepts contained in demonstrations are already stored in final Key-Value matrices, we only need to feed the test input into the model and utilize the Key-Value cache for inference. This *Deep-Thinking* strategy is motivated by humans' repeat logical thinking and reasoning process. LLMs are expected to extend their abilities to solve unseen, complex tasks by "*thinking*" demonstrations multiple times.

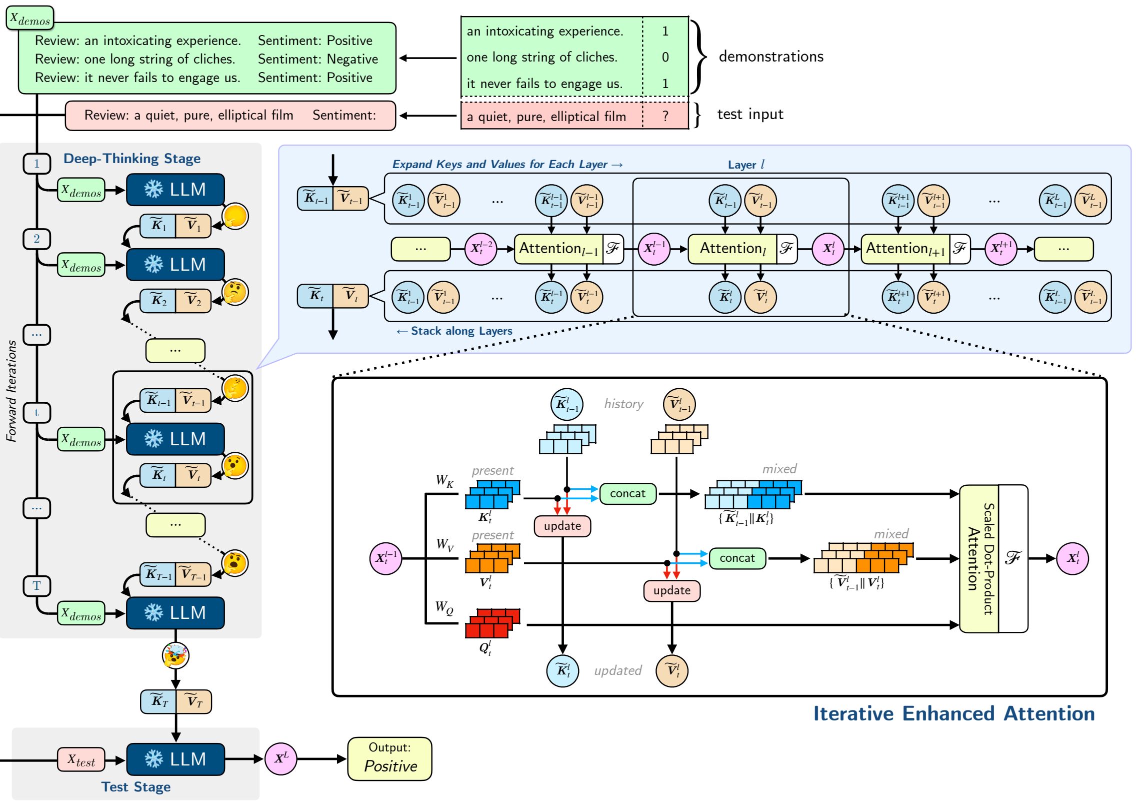
To verify the effectiveness of the proposed *Deep-Thinking*, we initially conduct evaluations on conventional ICL benchmarks across language models of various sizes. The experiments show that *Deep-Thinking* significantly outperforms vanilla ICL in a variety of model sizes and tasks, surpassing previous state-of-the-art (SOTA) methods focused on selecting demonstrations. In addition, we introduce two more challenging benchmarks (i.e., MMLU (Hendrycks et al., 2021) and BBH (Srivastava et al., 2023)) and conduct experiments on advanced LLMs, including LLaMA2 (Touvron et al., 2023) and Pythia (Biderman et al., 2023). We argue that on these challenging benchmarks, demonstration selection becomes impractical due to the lack of a potential candidate pool. *Deep-Thinking* obtains a significant advantage over vanilla ICL.

2 Preliminaries: In-Context Learning

This paper focuses on in-context learning tasks. Formally, given a nature language test input x_{test} with a few (N-shot) input-output demonstrations $\mathcal{C}_{demos} = \{(x_i, y_i)\}_{i=1}^{N}$, the goal of in-context learning is to predict the label $\hat{y}$ of x_{test} from a predefined candidate label set $\mathcal{Y} = \{y_1, y_2, ..., y_m\}$ conditioned on N demonstrations. Given an LLM $\mathcal{M}$ (e.g., a GPT model), the prediction process can be formulated as follows:

$$\hat{y} = \arg\max_{y_j \in \mathcal{Y}} P_{\mathcal{M}}(y_j|\mathcal{C}_{demos}, x_{test}), \tag{1}$$

where P is the output probability of the LLM $\mathcal{M}$. Generally, an LLM adopts the Transformer as the backbone, which consists of a stack of several Transformer blocks (Vaswani et al., 2017).



X_{demos}
Review: an intoxicating experience. Sentiment: Positive
Review: one long string of cliches. Sentiment: Negative
Review: it never fails to engage us. Sentiment: Positive
an intoxicating experience. 1
one long string of cliches. 0
it never fails to engage us. 1
demonstrations
Review: a quiet, pure, elliptical film Sentiment:
a quiet, pure, elliptical film ?
test input
Deep-Thinking Stage
Forward Iterations
LLM
Expand Keys and Values for Each Layer →
Layer l
← Stack along Layers
Attention$_{l-1}$
Attention$_l$
Attention$_{l+1}$
history
present
mixed
concat
update
updated
W_K
W_V
W_Q
Scaled Dot-Product Attention
Iterative Enhanced Attention
X_{test}
Test Stage
Output: Positive

Figure 2: The overview of proposed two-stage ICL framework. It divides the ICL process into *Deep-Thinking* stage and test stage, which take demonstrations and test query as input, respectively. It replaces the vanilla self-attention mechanism with the proposed Iterative Enhanced Attention (IEA). IEA utilizes the Key-Value matrices as bridge of memories, capable of receiving historical (from the previous iteration) memories. It can mix memories with present information to perform attention, and update memories for the next iteration. During testing, predictions are performed using memories that have been refined through multiple iterations. Notably, throughout this process, the LLM parameters remain frozen and no additional parameters are introduced.

3 Methodology

In this paper, we propose a two-stage ICL framework that improves performance through multiple forward iterations. As shown in Figure 2, we assign the demonstrations and test input to the *Deep-Thinking* and test stages, respectively, where Key-Value matrices serve as a bridge between the two stages. Next, we describe these two stages in detail.

3.1 The *Deep-Thinking* Stage

In the *Deep-Thinking* stage, given the demonstrations, we perform multiple forward passes in an iterative way by manipulating the Key-Value matrices in the self-attention (Vaswani et al., 2017) module. We use $\boldsymbol{X}_t^l$ to denote the output representation of the entire demonstration sequence at layer l and the t-th forward pass. Notably, $\boldsymbol{X}_t^l$ receives not only the output $\boldsymbol{X}_t^{l-1}$ from the previous Transformer block, but also the Key-Value matrices $\widetilde{\boldsymbol{K}}_{t-1}^l, \widetilde{\boldsymbol{V}}_{t-1}^l$[*] produced by the same self-attention module at the $(t-1)$-th forward pass. Accordingly, the Key-Value matrices will be updated as $\widetilde{\boldsymbol{K}}_t^l, \widetilde{\boldsymbol{V}}_t^l$.

Iterative Enhanced Attention To handle multiple forward iteration information in *Deep-Thinking*, we proposed a modified attention mechanism, named Iterative Enhanced Attention (IEA). Each block of IEA is illustrated in Figure 2. The information flowing through a block can be observed from both horizontal and vertical processes. The horizontal process represents the calculation of the input parameters in a conventional manner, while the vertical process stands for the manipulation of the Key-Value matrices. Specifically, the input $\boldsymbol{X}_t^{l-1}$ is firstly projected by key, value and query weight matrices, respectively:

$$\boldsymbol{K}_t^l = W_K\boldsymbol{X}_t^{l-1}, \quad \boldsymbol{V}_t^l = W_V\boldsymbol{X}_t^{l-1}, \quad \boldsymbol{Q}_t^l = W_Q\boldsymbol{X}_t^{l-1} \quad (2)$$

[*]Key-Value matrices, represented in blue, act as memory carriers throughout the *Deep-Thinking* iterations and serve as inputs in the final test stage.

where $\boldsymbol{K}_t^l, \boldsymbol{V}_t^l$ represent the **present** Key-Value matrices of vanilla self-attention, projected from input $\boldsymbol{X}$. For the horizontal process, we concatenate the **present** Key-Value matrices with the **history** Key-Value matrices $\widetilde{\boldsymbol{K}}_{t-1}^l, \widetilde{\boldsymbol{V}}_{t-1}^l$ as the **mixed** Key-Value to compute attention map and obtain the output $\boldsymbol{X}_t^l$ of current layer as follows:

$$\boldsymbol{X}_t^l = \mathscr{F}(\textbf{Attention}_l(\{\widetilde{\boldsymbol{K}}_{t-1}^l \| \boldsymbol{K}_t^l\}, \{\widetilde{\boldsymbol{V}}_{t-1}^l \| \boldsymbol{V}_t^l\}, \boldsymbol{Q}_t^l)) \quad (3)$$

where $\mathscr{F}$ refers to the operations after self-attention, namely the Feed-Forward Network (FFN) (Vaswani et al., 2017), layer normalization (Ba et al., 2016) and residual connection (He et al., 2015).

Furthermore, the update process is jointly contributed by the **present** and **history** Key-Value matrices. From a high-level abstract perspective, the update process can be formalized as follows:

$$\begin{aligned} \widetilde{\boldsymbol{K}}_t^l &= \textbf{update}(\widetilde{\boldsymbol{K}}_{t-1}^l, \boldsymbol{K}_t^l) = \eta \boldsymbol{K}_t^l + (1-\eta)\widetilde{\boldsymbol{K}}_{t-1}^l \\ \widetilde{\boldsymbol{V}}_t^l &= \textbf{update}(\widetilde{\boldsymbol{V}}_{t-1}^l, \boldsymbol{V}_t^l) = \eta \boldsymbol{V}_t^l + (1-\eta)\widetilde{\boldsymbol{V}}_{t-1}^l \end{aligned} \quad (4)$$

where $\widetilde{\boldsymbol{K}}_t^l$ and $\widetilde{\boldsymbol{V}}_t^l$ are **updated** Key-Value matrices. For the update function, we adopt a simple gating mechanism that utilizes the hyper-parameter η to control the fusion rate of history and present information.

Modeling demonstrations takes up to T iterations, where the value of T can be predefined by users. After the iterative forward process, we can obtain final **updated** Key-Value matrices $\widetilde{\boldsymbol{K}}_T^l, \widetilde{\boldsymbol{V}}_T^l$. By stacking **updated** Key-Value matrices of all layers in a given LLM, we have

$$\widetilde{\boldsymbol{K}}_T = \{\widetilde{\boldsymbol{K}}_T^l\}_{l=1}^L, \quad \widetilde{\boldsymbol{V}}_T = \{\widetilde{\boldsymbol{V}}_T^l\}_{l=1}^L \quad (5)$$

which can be stored statically. L denotes the number of Transformer blocks in an LLM.

3.2 The Test Stage

Considering that we now have the Key-Value matrices $\widetilde{\boldsymbol{K}}_T, \widetilde{\boldsymbol{V}}_T$ that have been updated for T iterations, the information contained in them can be regarded as a highly condensed modeling of the demonstrations. The inference process can be performed using the same formulation as given by Eq.(2)-Eq.(3). Specifically, the inference process for l-th layer can be formalized as:

$$\begin{aligned} &\boldsymbol{K}_{test}^l = W_K \boldsymbol{X}_{test}^{l-1}, \ \boldsymbol{V}_{test}^l = W_V \boldsymbol{X}_{test}^{l-1}, \ \boldsymbol{Q}_{test}^l = W_Q \boldsymbol{X}_{test}^{l-1} \\ &\boldsymbol{X}_{test}^l = \mathscr{F}(\textbf{Attention}_l(\{\widetilde{\boldsymbol{K}}_T^l \| \boldsymbol{K}_{test}^l\}, \{\widetilde{\boldsymbol{V}}_T^l \| \boldsymbol{V}_{test}^l\}, \boldsymbol{Q}_{test}^l)) \end{aligned} \quad (6)$$

In this way, we can obtain the representation $\boldsymbol{X}_{test}^L$ produced by the final layer, which is used to make predictions.

4 Experiments

4.1 Conventional In-context Learning Tasks

We first evaluate the proposed *Deep-Thinking* against other enhanced ICL methods in a fair comparison of conventional ICL tasks. We select five popular tasks, including **SST2** (Socher et al., 2013), **SST5** (Socher et al., 2013), **MR** (Pang and Lee, 2005), **AGNews** (Zhang et al., 2015) and **TREC** (Li and Roth, 2002; Hovy et al., 2001). For a fair comparison, we choose **GPT2-L** as the base model, which is widely used by previous studies (Li and Qiu, 2023).

Compared Methods We use several demonstration selection methods as baselines, which can be classified into distinct approaches. (1) Geometry-based techniques, such as Herding (Chen et al., 2010) and K-Center Greedy (Sener and Savarese, 2018), concentrate on spatial proximity within the feature space for constructing demonstrations. (2) Uncertainty-based methods posit that demonstrations with higher uncertainty exert more substantial influence on the model, encompassing techniques such as Entropy, Least Confidence, Margin (Coleman et al., 2019), and CAL (Margatina et al., 2021). (3) Gradient matching-based methods, such as CRAIG (Mirzasoleiman et al., 2020) and GradMatch (Killamsetty et al., 2021), aim to replicate the gradient distribution of the full dataset with a subset. (4) Submodularity-based methods assess informativeness and diversity for selection, including such as FacilityLocation and GraphCut (Iyer and Bilmes, 2013). (5) LENS (Li and Qiu, 2023) adopts a "filter-then-search" approach, utilizing the "InfoScore" metric to select the best demonstrations. Notably, our fairest baseline is the Random method (vanilla ICL), where we use the exact same demonstrations without any selection process.

Further Comparison To assess the performance of *Deep-Thinking* across a range of LMs across different sizes, we extend the base model of *Deep-Thinking* on conventional ICL tasks to include **OPT** (125M, 350M, 2.7B) (Zhang et al., 2022), **GPT-2** (Medium, Large, and XL) (Radford et al., 2019), **GPT-Neo** (2.7B) (Black et al., 2021) and **LLaMA2** (7B, 13B) (Touvron et al., 2023). This extension aims to demonstrate the effectiveness of *Deep-Thinking* across a spectrum of LM scales.

Method	**SST2**	**SST5**	**TREC**	**MR**	**AGNews**	**Average**
In-context learning w/o dev set. $^\diamondsuit$						
Random	57.9	27.5	30.3	59.5	33.6	41.8
Herding (Chen et al., 2010)	62.0	24.8	26.4	54.1	38.7	41.2
K-Center Greedy (Sener and Savarese, 2018)	58.6	25.1	31.3	59.0	42.3	43.3
Entropy (Coleman et al., 2019)	62.4	25.5	26.2	54.1	30.6	39.8
LeastConfidence (Coleman et al., 2019)	58.4	26.0	23.5	55.9	31.6	39.1
Margin (Coleman et al., 2019)	62.4	26.1	24.2	54.1	38.1	41.0
CAL (Margatina et al., 2021)	59.3	25.3	31.8	66.2	42.3	45.0
CRAIG (Mirzasoleiman et al., 2020)	63.4	26.4	32.0	59.3	37.4	43.7
GradMatch (Killamsetty et al., 2021)	57.0	26.3	25.8	56.6	32.6	39.7
FacilityLocation (Iyer and Bilmes, 2013)	65.5	23.9	35.7	61.7	42.5	45.9
GraphCut (Iyer and Bilmes, 2013)	65.0	25.3	34.7	66.3	41.9	46.6
Deep-Thinking	**85.7**	**39.2**	**54.2**	**71.6**	**72.9**	**64.7**
In-context learning w/ dev set. $^\heartsuit$						
LENS (Li and Qiu, 2023)	86.3	44.9	59.0	83.1	77.9	70.2
Random$^\clubsuit$	77.9	38.0	56.6	81.8	74.6	65.8
Deep-Thinking$^\clubsuit$	**88.1**	**45.2**	**61.6**	**84.8**	**80.3**	**72.0**

Table 1: Experimental results across conventional ICL tasks with different ICL methods. $^\diamondsuit$ denotes that each method was assessed over ten random seeds, and the reported values are the average performance across these seeds. $^\heartsuit$ signifies the evaluation of multiple sets of random demonstrations on the dev set, with the best-performing set selected (Li and Qiu, 2023). $^\clubsuit$ indicates the methods utilized the same demonstrations, ensuring that any improvement stemmed solely from the *Deep-Thinking* stage.

4.2 Challenging Benchmarks

In contrast to the conventional ICL benchmarks that entail a candidate pool for sample selection, real-world complex tasks frequently present scenarios where only a limited and fixed set of demonstrations is available. This particular challenge renders many existing methods of demonstration selection impractical in such scenarios. To deal with the challenges, we choose **MMLU** (Hendrycks et al., 2021) and **BBH** (Srivastava et al., 2023) to extend the evaluation of *Deep-Thinking* to more rigorous and multifaceted scenarios. Concretely, MMLU encompasses a diverse set of 57 tasks, spanning elementary mathematics, US history, computer science, law, and various other domains. In contrast, BBH is tailored to address a suite of 23 challenging tasks within the BIG-Bench framework. In addressing these challenging benchmarks, we employ more advanced LLMs, including **LLaMA2** (Touvron et al., 2023) (7B and 13B) and **Pythia** (Biderman et al., 2023) (70M, 410M, 1.4B, 6.9B and 12B) as base models, given their balanced ability and versatility in handling a wide range of tasks.

4.3 Implementation Details and Evaluation

All experiments are conducted on a single NVIDIA A100 GPU. For all baselines and *Deep-Thinking*, we run each method over ten random seeds and report the average performance. For conventional ICL tasks, we follow (Li and Qiu, 2023) that the number of demonstrations for SST2, SST5, TREC, MR, and AGNews is 8, 10, 12, 8, and 8, respectively. For MMLU and BBH, the demonstrations come from the dataset's inherent demonstrations. Specifically five demonstrations for MMLU and three demonstrations for BBH per task. In the in-context setting without a dev set, we fix the iteration number T at 5, with the gating parameter η set to 0.01. In the in-context setting with a dev set, we relax the max iteration number T to 15, using the dev set to determine the final hyper-parameters. For the dev set, we randomly select a sample size identical to the test set and keep it fixed. For evaluation, similar to previous methods, we concatenate the test input with each candidate's answer and submit them to the LLM. The final answer is selected by summing the probabilities of the tokens belonging to the answer part and choosing the candidate answer with the highest probability.

4.4 Main Results

Results on Conventional ICL Tasks We first compare *Deep-Thinking* with previous methods on conventional ICL tasks. Table 1 shows that *Deep-Thinking* consistently outperforms baseline methods. In addition, a significant improvement is observed when comparing its performance with

Model	Method	SST2	SST5	TREC	MR	AGNews	Average
***OPT*-125M**	ICL	55.7	26.7	25.0	50.4	41.7	39.9
	Ours	**72.0**	**33.2**	**47.0**	**65.8**	**50.6**	**53.7**
***OPT*-350M**	ICL	54.1	26.6	37.2	71.2	42.9	46.4
	Ours	**79.7**	**31.8**	**45.8**	**73.4**	**64.3**	**59.0**
GPT2-Medium 355M	ICL	59.6	23.7	33.4	65.0	51.7	46.7
	Ours	**86.9**	**38.1**	**43.6**	**80.3**	**80.0**	**65.8**
GPT2-XL 1.5B	ICL	60.3	41.1	34.4	66.7	56.9	51.9
	Ours	**89.3**	**43.8**	**60.6**	**86.1**	**82.6**	**72.5**
***OPT*-2.7B**	ICL	62.4	45.8	37.0	86.2	77.8	61.8
	Ours	**72.4**	**47.7**	**50.0**	**89.0**	**85.8**	**69.0**
GPT-Neo 2.7B	ICL	84.8	39.7	49.6	85.2	71.6	66.2
	Ours	**88.1**	**45.5**	**59.2**	**89.1**	**83.5**	**73.1**
LLaMA2 7B	ICL	89.5	46.3	82.8	91.2	84.6	78.9
	Ours	**90.0**	**48.1**	**84.8**	**92.2**	**88.9**	**80.8**
LLaMA2 13B	ICL	95.2	46.4	84.8	92.5	87.0	81.2
	Ours	**96.0**	**49.9**	**86.2**	**94.7**	**88.8**	**83.1**

Table 2: Experimental results cross different LLMs on conventional ICL tasks. To ensure that any observed improvement stems exclusively from the *Deep-Thinking* stage, all variables are held constant across experiments.

and without the utilization of a development set. Particularly, *Deep-Thinking* surpasses the Random baseline by an average margin of 6.2% under the dev setting. This improvement is solely attributed to the iterative forward operations, providing empirical evidence of the effectiveness of the proposed method.

Transferablity across Different LMs In the aforementioned experiments, we employ the same LM (GPT-L) as the base model. To assess the transferability of *Deep-Thinking* across LMs of varying scales and pre-trained corpora, we expand our experimental scope to include diverse settings. Table 2 validates the transferability and generalizability of *Deep-Thinking* across different base models, maintaining competitiveness even when applied to stronger models such as LLaMA2.

Results on Challenging Benchmarks Table 3 presents the averaged performance of *Deep-Thinking* and vanilla ICL on MMLU and BBH benchmarks. Notably, the performance boost is consistent across all selected models, affirming that *Deep-Thinking* beneficially impacts a wide spectrum of frontier LMs, independent of their specific designs or training data. This effect is particularly evident in smaller-sized models such as Pythia, where the relative performance uplift is significant. This trend aligns with observations from conventional ICL tasks, further highlighting the broad applicability and effectiveness of *Deep-Thinking* in enhancing models' in-context learning ability.

4.5 Fine-grained Task Analysis

To investigate whether *Deep-Thinking* shows greater advantages in tasks requiring complex reasoning, we conduct a detailed analysis as shown in Figure 3. The MMLU benchmark categorizes its 57 subtasks into four major classes: STEM, Humanities, Social Science, and Others. STEM tasks rigorously assess the model's reasoning abilities, whereas the other three categories predominantly serve as tests of knowledge retention. STEM tasks pose greater challenges for vanilla ICL methods; however, *Deep-Thinking* consistently demonstrates improvements across all categories, indicating a relatively more substantial gain in STEM. For instance, LLaMA2-7B exhibits a 3.9% increase over ICL in STEM, while registering improvements of 2.3%, 2.6%, and 2.5% in Humanities, Social Science, and Others, respectively. This highlights the effectiveness of *Deep-Thinking* in enhancing models' capabilities to address complex reasoning and problem-solving tasks.

4.6 Impact of Hyper-parameters

Impact of Demonstration Numbers As depicted in Figure 4, we conducted experiments on the SST2 and AGNews datasets to explore the impact of increasing the number of demonstrations on

Model & Method	*LLaMA2*				*Pythia*									
	7B		13B		70M		410M		1.4B		6.9B		12B	
	ICL	*Ours*	*ICL*	*Ours*	*ICL*	*Ours*	*ICL*	*Ours*	*ICL*	*Ours*	*ICL*	*Ours*	*ICL*	*Ours*
MMLU	41.9	**44.6**	45.1	**47.6**	24.6	**29.6**	27.0	**30.8**	30.6	**33.6**	33.5	**37.1**	36.2	**39.5**
BBH	46.1	**49.8**	49.7	**53.6**	34.9	**39.8**	37.5	**42.3**	38.2	**43.8**	39.2	**43.4**	41.1	**45.3**

Table 3: The results of vanilla ICL and *Deep-Thinking* on challenging benchmarks, including **MMLU** and **BBH**.

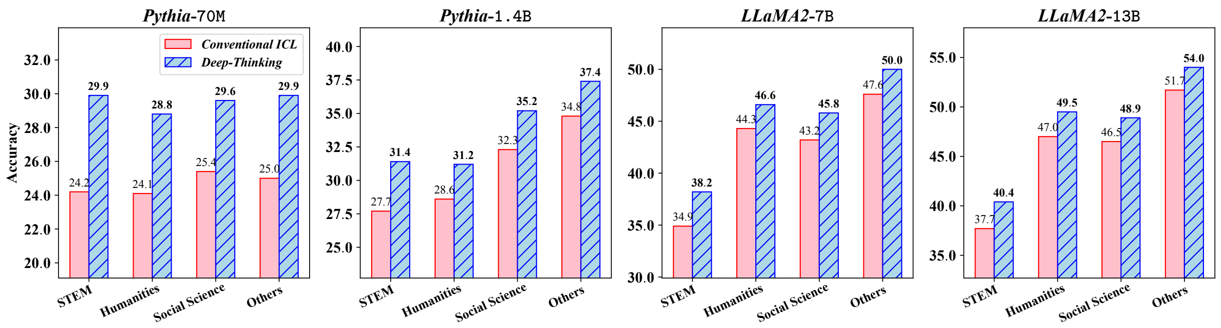


Figure 3: Comparison of model performance across four major classes of the **MMLU** benchmarks. Due to space constraints and to ensure clarity in presentation, we solely report the results of four out of the seven models.

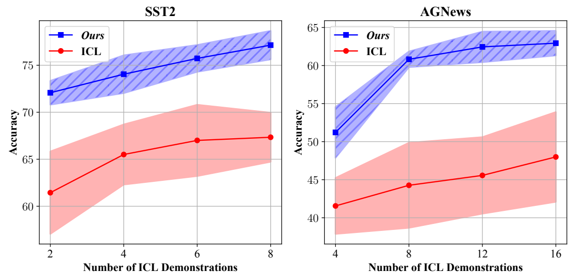


Figure 4: An illustration of the impact of increasing the number of demonstrations on the effectiveness of vanilla ICL and *Deep-Thinking*.

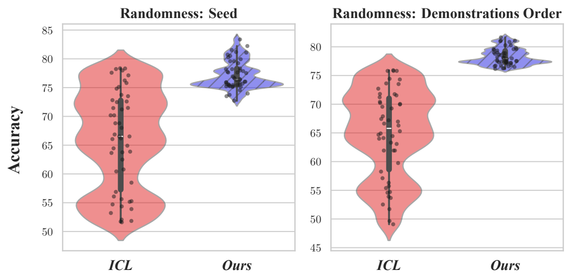


Figure 5: The performance distribution of performance for vanilla ICL and *Deep-Thinking*, comparing effects of random seeds (left) and random orders (right).

the efficacy of ICL. We perform ten runs of the experiments and calculate the variance. The results indicate that both vanilla ICL and *Deep-Thinking* benefit from an increase in the number of demonstrations. However, *Deep-Thinking* consistently outperforms vanilla ICL, achieving significantly better results even with a smaller number of demonstrations. Additionally, *Deep-Thinking* demonstrates a smaller variance, indicating greater robustness. This suggests that it is more cost-effective to "think" more from existing demonstrations than merely increasing the number of demonstrations.

The Sensitivity of Random Seed The randomness in vanilla ICL and *Deep-Thinking* stems solely from the random selection of demonstrations. To further investigate the robustness of the methods, we conduct multiple experiments on the SST dataset by randomly choosing eight different demonstrations, keeping other variables (except the seed) constant. As illustrated in Figure 5 (left), vanilla ICL is significantly affected by randomness, whereas *Deep-Thinking* achieves stronger performance with less variance. *Deep-Thinking*, by iterating multiple times, bridges the gap by maximizing the utility of demonstrations.

The Sensitivity of Demonstrations Orders The order of demonstrations in ICL is crucial and can significantly impact performance (Lu et al., 2022a; Liu et al., 2022). In particular, different orders of demonstrations can lead to performance close to the state-of-the-art or merely random guesses. We examine the effect of demonstration order on ICL and *Deep-Thinking* on the SST-2 dataset. As shown in Figure 5 (right), the results show that vanilla ICL

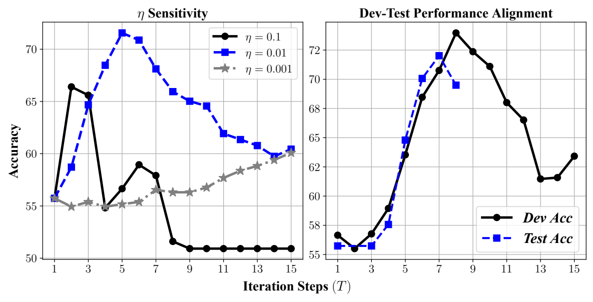


Figure 6: Sensitivity of accuracy to hyperparameter η and alignment of development (Dev) and test set performance over iteration steps (T).

is highly sensitive to the order, with a significant variance in outcomes indicating large instability. *Deep-Thinking* benefits from iterative learning of demonstrations, overcoming order biases, and thus shows more consistent performance.

Impact of Gate η and Iteration Steps T The gate η is crucial in dictating how much of the memory is retained during the *Deep-Thinking* stage and the degree of openness to new information for the next iteration. A larger η signifies greater changes, thus requiring fewer iterations T, and vice versa. To investigate the optimal η, we enumerate values in [0.001, 0.01, 0.1]. As shown in Figure 6 (left), setting $\eta = 0.01$ achieves a balance between the number of iterations and performance. We can analogize η to the learning rate and T to the number of training steps. Inspired by this comparison, as described in Table 1, we use a dev set to determine the optimal number of iterations T. Figure 6 (right) shows that there is a basic alignment between the dev and test sets.

5 Related Work

In-context learning (ICL) with LLMs has made a breakthrough and become mainstream in tackling various tasks (Li et al., 2023; Dong et al., 2022; Qiao et al., 2023). Recently, great efforts have been made to improve the performance of ICL from different perspectives, such as demonstrations selection (Liu et al., 2022; Li and Qiu, 2023), prompt template design (Wei et al., 2022a), and intermediate chain-of-thought (CoT) reasoning (Wei et al., 2022b; Zhang et al., 2023).

For demonstration selection, Liu et al. (2022) performed demonstration selection through a kNN-based retriever, choosing the closest example to test input. Wu et al. (2022) proposed self-adaptive ICL with a general select-and-rank framework for demonstration selection. In addition to example selection, Lu et al. (2022b) investigated the sensitivity of ICL to the permutation of demonstrations and proposed entropy metrics to determine their order. The above ICL methods are usually restricted by the number of demonstrations. To mitigate such a challenge, Hao et al. (2022) attempted to scale ICL by grouping demonstrations, which could increase the number of demonstrations to 1,000.

The formatting function also plays a crucial role in ICL, especially for tasks requiring complex reasoning steps, such as commonsense reasoning. Wei et al. (2022b) introduced chain-of-thoughts (CoT) prompting to provide guidance. Zhang et al. (2023) stimulated the model's ability for gradual reasoning by adding the "*Let's think step-by-step*" prefix. Instead of generating reasoning steps, Press et al. (2023) investigated the compositional reasoning abilities by allowing LLMs to generate follow-up questions. Subsequently, Madaan et al. (2023) introduced a new framework to enhance the initial outputs generated by LLMs via iterative feedback and refinement. Meanwhile, some studies (Xie et al., 2022; Dai et al., 2023; Wang et al., 2023b) attempt to uncover the underlying working mechanism of ICL. In particular, Xie et al. (2022) showed that ICL happened via Bayesian inference, where certain concepts were implicitly predicted before the final prediction. Subsequently, Dai et al. (2023) revealed that there are connections between ICL and explicit fine-tuning and explained LLMs as meta-optimizers (Irie et al., 2022b).

Unlike existing methods, to the best of our knowledge, we are the first to decouple ICL into two stages and focus on how to deeply learn from fixed demonstrations rather than on demonstration selection or prompt engineering. This is advantageous for the world situation where provided samples are scarce, i.e., there is no large candidate set of demonstrations.

6 Conclusion

In this paper, we introduce a novel two-stage framework aimed at enhancing the ICL capabilities of LLMs by leveraging iterative forward inferences to learn from demonstrations. By decoupling the ICL process into a dedicated *Deep-Thinking* stage for demonstration training and a subsequent test stage, we effectively mimic the decision-making process of humans by learning from analogy. This approach aligns with how humans engage in repeated logical thinking. The empirical evaluations across conventional ICL benchmarks and more challenging datasets demonstrate that our *Deep-Thinking* strategy significantly outperforms previous ICL approaches, particularly in scenarios where demonstration selection is impractical.

Limitations

While our method has demonstrated promising results and significant advancements across various aspects, it is imperative to conduct a thorough analysis of its limitations. In this section, we explore the potential constraints of our method. Firstly, owing to limited computational resources and time constraints, we were unable to evaluate our method on larger language models, such as LLaMA2-70B. This limitation may impact the generalizability of our findings to larger-scale language models. Secondly, our evaluation primarily focused on conventional ICL tasks and challenging benchmarks. To enhance the comprehensiveness of our findings, we intend to broaden the scope of evaluation to encompass a diverse range of dataset types, including math reasoning, code generation, and open-ended text generation. This extension aims to provide further validation of our method's generalizability.

Acknowledgments

Min Yang was supported by National Key Research and Development Program of China (2022YFF0902100), National Natural Science Foundation of China (62376262), the Natural Science Foundation of Guangdong Province of China (2024A1515030166), Shenzhen Science and Technology Innovation Program (KQTD20190929172835662), Shenzhen Basic Research Foundation (JCYJ20210324115614039). This work was supported by Alibaba Group through Alibaba Research Intern Program.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

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

Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. 2021. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow.

Yutian Chen, Max Welling, and Alex Smola. 2010. Super-samples from kernel herding. In *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*, pages 109–116.

Cody Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter Bailis, Percy Liang, Jure Leskovec, and Matei Zaharia. 2019. Selection via proxy: Efficient data selection for deep learning. In *International Conference on Learning Representations*.

Damai Dai, Yutao Sun, Li Dong, Yaru Hao, Shuming Ma, Zhifang Sui, and Furu Wei. 2023. 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. Association for Computational Linguistics.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. 2022. A survey for in-context learning. *arXiv preprint arXiv:2301.00234*.

Yaru Hao, Yutao Sun, Li Dong, Zhixiong Han, Yuxian Gu, and Furu Wei. 2022. Structured prompting: Scaling in-context learning to 1,000 examples. *arXiv preprint arXiv:2212.06713*.

Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Eduard Hovy, Laurie Gerber, Ulf Hermjakob, Chin-Yew Lin, and Deepak Ravichandran. 2001. Toward semantics-based answer pinpointing. In *Proceedings of the First International Conference on Human Language Technology Research*.

Kazuki Irie, Róbert Csordás, and Jürgen Schmidhuber. 2022a. The dual form of neural networks revisited: Connecting test time predictions to training patterns via spotlights of attention. In *International Conference on Machine Learning*, pages 9639–9659. PMLR.

Kazuki Irie, Róbert Csordás, and Jürgen Schmidhuber. 2022b. The dual form of neural networks revisited: Connecting test time predictions to training patterns via spotlights of attention. In *International Conference on Machine Learning*, pages 9639–9659. PMLR.

Rishabh K Iyer and Jeff A Bilmes. 2013. Submodular optimization with submodular cover and submodular knapsack constraints. *Advances in neural information processing systems*, 26.

Krishnateja Killamsetty, Ganesh Ramakrishnan, Abir De, and Rishabh Iyer. 2021. Grad-match: Gradient matching based data subset selection for efficient deep model training. In *International Conference on Machine Learning*, pages 5464–5474. PMLR.

Jinyang Li, Binyuan Hui, GE QU, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin Chang, Fei Huang, Reynold Cheng, and Yongbin Li. 2023. Can LLM already serve as a database interface? a BIg bench for large-scale database grounded text-to-SQLs. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Xiaonan Li and Xipeng Qiu. 2023. Finding support examples for in-context learning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 6219–6235, Singapore. Association for Computational Linguistics.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2022. What makes good in-context examples for GPT-3? In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 100–114, Dublin, Ireland and Online. Association for Computational Linguistics.

Pan Lu, Baolin Peng, Hao Cheng, Michel Galley, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, and Jianfeng Gao. 2023. Chameleon: Plug-and-play compositional reasoning with large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022a. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8086–8098, Dublin, Ireland. Association for Computational Linguistics.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022b. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8086–8098.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Katerina Margatina, Giorgos Vernikos, Loïc Barrault, and Nikolaos Aletras. 2021. Active learning by acquiring contrastive examples. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 650–663, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Sewon Min, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Noisy channel language model prompting for few-shot text classification. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5316–5330, Dublin, Ireland. Association for Computational Linguistics.

Baharan Mirzasoleiman, Jeff Bilmes, and Jure Leskovec. 2020. Coresets for data-efficient training of machine learning models. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 6950–6960. PMLR.

OpenAI. 2023. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 115–124.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. 2023. Measuring and narrowing the compositionality gap in language models. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

Shuofei Qiao, Yixin Ou, Ningyu Zhang, Xiang Chen, Yunzhi Yao, Shumin Deng, Chuanqi Tan, Fei Huang, and Huajun Chen. 2023. Reasoning with language model prompting: A survey. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5368–5393, Toronto, Canada. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Avi Schwarzschild, Eitan Borgnia, Arjun Gupta, Furong Huang, Uzi Vishkin, Micah Goldblum, and Tom Goldstein. 2021. Can you learn an algorithm? generalizing from easy to hard problems with recurrent networks. *Advances in Neural Information Processing Systems*, 34:6695–6706.

Ozan Sener and Silvio Savarese. 2018. Active learning for convolutional neural networks: A core-set approach. In *International Conference on Learning Representations*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*.

Tianxiang Sun, Yunfan Shao, Hong Qian, Xuanjing Huang, and Xipeng Qiu. 2022. Black-box tuning for language-model-as-a-service. In *Proceedings of The International Conference on Machine Learning*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Lean Wang, Lei Li, Damai Dai, Deli Chen, Hao Zhou, Fandong Meng, Jie Zhou, and Xu Sun. 2023a. Label words are anchors: An information flow perspective for understanding in-context learning. In *Conference on Empirical Methods in Natural Language Processing*.

Xinyi Wang, Wanrong Zhu, Michael Saxon, Mark Steyvers, and William Yang Wang. 2023b. Large language models are latent variable models: Explaining and finding good demonstrations for in-context learning. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022a. Finetuned language models are zero-shot learners. In *The Tenth International Conference on Learning Representations*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022b. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

Zhiyong Wu, Yaoxiang Wang, Jiacheng Ye, and Lingpeng Kong. 2022. Self-adaptive in-context learning: An information compression perspective for in-context example selection and ordering. *arXiv preprint arXiv:2212.10375*.

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

Yunhu Ye, Binyuan Hui, Min Yang, Binhua Li, Fei Huang, and Yongbin Li. 2023. Large language models are versatile decomposers: Decomposing evidence and questions for table-based reasoning. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 174–184.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, pages 649–657.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2023. Automatic chain of thought prompting in large language models. In *The Eleventh International Conference on Learning Representations*.

A Data Statistics and Templates For In-context Learning Tasks

We choose five datasets for evaluating in-context learning methods following (Min et al., 2022; Li and Qiu, 2023). We show the prompting templates and dataset statistics in Table 4.

Task	*Original Dev Size*	*Test Size*	*Template*	*Labels*
SST2	67349	873	`Review: {query}` `Sentiment: {label}`	`negative / positive`
SST5	8544	2210	`Review: {query}` `Sentiment: {label}`	`terrible / negative / neutral / positive / great`
TREC	5452	500	`Question: {query}` `Type: {label}`	`Abbreviation / Entity / Description / Person / Location / Number`
MR	8530	1066	`Review: {query}` `Sentiment: {label}`	`negative / positive`
AGNews	120000	7601	`Article: {query}` `Category: {label}`	`World / Sports / Business / Technology`

Table 4: The statistics of standard in-context learning tasks, including detailed task sizes, prompting templates, and labels.

B Data Statistics For MMLU

We obtained the MMLU dataset from the Hugging Face Hub, specifically from the repository `cais/mmlu`[†]. According to the dataset's card, MMLU encompasses 57 tasks spanning diverse knowledge domains. Each task includes a minimum of 100 test examples. For in-context demonstrations, five examples per task provided by original dataset are used. We present detailed statistics for each sub-task, including the classification scheme, in Table 5.

C Data Statistics For BBH

For the BigBench-Hard (BBH) dataset, we sourced the data from the `maveriq/bigbenchhard`[‡]. We have excluded certain tasks due to their incompatibility with multiple-choice or classification formats. Specifically, the tasks omitted include: Dyck Languages, Multistep Arithmetic, Object Counting, Word Sorting, and Reasoning about Colored Objects. Table 6 provides a detailed statistics for each selected task.

[†]`https://huggingface.co/datasets/cais/mmlu`
[‡]`https://huggingface.co/datasets/maveriq/bigbenchhard`

STEM	**astronomy**: 152, **college_physics**: 102, **conceptual_physics**: 235, **high_school_physics**: 151, **college_chemistry**: 100, **high_school_chemistry**: 203, **college_biology**: 144, **high_school_biology**: 310, **college_computer_science**: 100, **computer_security**: 100, **high_school_computer_science**: 100, **machine_learning**: 112, **abstract_algebra**: 100, **college_mathematics**: 100, **elementary_mathematics**: 378, **high_school_mathematics**: 270, **high_school_statistics**: 216, **electrical_engineering**: 145
Humanities	**high_school_european_history**: 165, **high_school_us_history**: 204, **high_school_world_history**: 237, **prehistory**: 324, **formal_logic**: 126, **logical_fallacies**: 163, **moral_disputes**: 346, **moral_scenarios**: 895, **philosophy**: 311, **world_religions**: 171, **international_law**: 121, **jurisprudence**: 108, **professional_law**: 1534
Social Sciences	**high_school_government_and_politics**: 193, **public_relations**: 110, **security_studies**: 245, **us_foreign_policy**: 100, **human_sexuality**: 131, **sociology**: 201, **econometrics**: 114, **high_school_macroeconomics**: 390, **high_school_microeconomics**: 238, **high_school_geography**: 198, **high_school_psychology**: 545, **professional_psychology**: 612
Others (Business, Health, misc.)	**global_facts**: 100, **miscellaneous**: 783, **professional_accounting**: 282, **business_ethics**: 100, **management**: 103, **marketing**: 234, **anatomy**: 135, **clinical_knowledge**: 265, **college_medicine**: 173, **human_aging**: 223, **medical_genetics**: 100, **nutrition**: 306, **professional_medicine**: 272, **virology**: 166

Table 5: The number of samples for each subtask in the **MMLU** benchmark.

BBH	**boolean_expressions**: 250, **causal_judgement**: 187, **date_understanding**: 212, **disambiguation_qa**: 247, **formal_fallacies**: 250, **geometric_shapes**: 200, **hyperbaton**: 250, **logical_deduction_three_objects**: 250, **logical_deduction_five_objects**: 250, **logical_deduction_seven_objects**: 250, **movie_recommendation**: 231, **navigate**: 250, **penguins_in_a_table**: 145, **ruin_names**: 247, **salient_translation_error_detection**: 250, **snarks**: 177, **sports_understanding**: 250, **temporal_sequences**: 250, **web_of_lies**: 250 **tracking_shuffled_objects_three_objects**: 250, **tracking_shuffled_objects_five_objects**: 250, **tracking_shuffled_objects_seven_objects**: 250

Table 6: The number of samples for each selected subtask in the **BBH** benchmark.

D Minimal Implementation

The provided minimal implementation code showcases our proposed method, primarily integrating with the Hugging Face Transformers library.

```python
class GatedOptim:
    def __init__(self, step_size=0.01):
        self.step_size = step_size

    def upd_x(self, old_x, g): return old_x + self.step_size * g
    def __call__(self, old_xs, new_xs):
        pesudo_gs = [new_x - old_x for old_x, new_x in zip(old_xs, new_xs)]
        updated_kv = [self.upd_x(old_x, g) for old_x, g in zip(old_xs, pesudo_gs)]
        return updated_kv

class AttnOptimWrapper:
    def __init__(self, llm, **optimizer_args):
        self.model = llm
        self.kv = None
        self.update_k = GatedOptim(**optimizer_args)
        self.update_v = GatedOptim(**optimizer_args)

    def step(self, ctx_ids):
        L = len(ctx_ids)
        ctx_ids = ctx_ids.unsqueeze(0) # [1, L]
        mask = torch.ones_like(ctx_ids).repeat(1, 2 if self.kv else 1)

        out = self.model(ctx_ids, mask, past_key_values=self.kv, use_cache=True)
        out_kv = out.past_key_values # kv @ (old_ctx + new_ctx)
        cur_kv = [[k[:,:,-L:,:], v[:,:,-L:,:]] for k, v in out_kv] # kv @ (new_ctx)

        if not self.kv:
            self.kv = cur_kv
        else:
            (old_ks, old_vs), (cur_ks, cur_vs) = zip(*self.kv), zip(*cur_kv)
            upd_ks = self.update_k(old_ks, cur_ks)
            upd_vs = self.update_v(old_vs, cur_vs)
            self.kv = list(zip(upd_ks, upd_vs)) # kv @ (merged_ctx)
        return self.kv

def main():
    # ... initialization (dataset, model, tokenizer, logger, etc)
    ex_str = load_exemplar()
    ex_ids, ex_mask = tokenize(ex_str)
    meta_optim = AttnOptimWrapper(model)
    for idx in range(args.meta_steps):
        ex_kv = meta_optim.step(ex_ids)
        for B_query_ids, B_query_mask in inference_loader:
            bs = len(B_query_ids)
            # [B(expanded), L+L'(concat)]
            B_merged_mask = torch.cat((ex_mask.expand(bs, -1), B_query_mask), dim=1)
            B_kv_shape = (bs, -1, -1, -1)
            B_kv = [[k.expand(B_kv_shape), v.expand(B_kv_shape)] for k, v in ex_kv]

            B_out = model(B_query_ids, B_merged_mask, past_key_values=B_kv).logits
            B_out = F.log_softmax(B_out, dim=-1)
            # ...
```