ELICIT: LLM AUGMENTATION VIA EXTERNAL IN-CONTEXT CAPABILITY

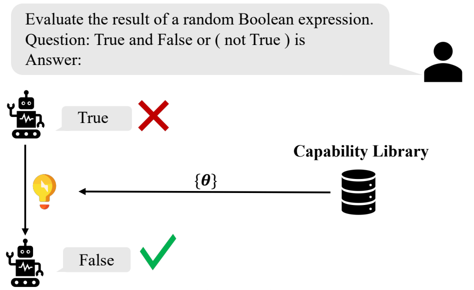
Anonymous authors
Paper under double-blind review

ABSTRACT

Enhancing the adaptive capabilities of large language models is a critical pursuit in both research and application. Traditional fine-tuning methods require substantial data and computational resources, especially for enhancing specific capabilities, while in-context learning is limited by the need for appropriate demonstrations and efficient token usage. Inspired by the expression of in-context learned capabilities through task vectors and the concept of modularization, we propose ELICIT, a framework consisting of two modules designed to effectively store and reuse task vectors to elicit the diverse capabilities of models without additional training or inference tokens. Our comprehensive experiments and analysis demonstrate that our pipeline is highly transferable across different input formats, tasks, and model architectures. ELICIT serves as a plug-and-play performance booster to enable adaptive elicitation of model capabilities. By externally storing and reusing vectors that represent in-context learned capabilities, ELICIT not only demonstrates the potential to operate modular capabilities but also significantly enhances the performance, versatility, adaptability, and scalability of large language models.

1 INTRODUCTION

Large Language Models (LLMs) have revolutionized the field of Natural Language Processing (NLP), demonstrating remarkable versatility in tackling a wide array of tasks and real-world challenges (Devlin, 2018; Brown, 2020; Han et al., 2021; Achiam et al., 2023; Touvron et al., 2023). The power of these models lies in their ability to seamlessly integrate various *capabilities*, from logical reasoning (Bommasani et al., 2021) to common sense understanding (Talmor et al., 2018). In our rapidly evolving world, a crucial aspect of LLM is the ability to efficiently adapt to new tasks or scenarios.



Evaluate the result of a random Boolean expression.
Question: True and False or (not True) is
Answer:

Figure 1: **Illustration of ELICIT**, which dynamically retrieves and integrates task vectors from a capability library to augment a language model's performance on arbitrary queries, without increasing token usage during inference.

Traditional fine-tuning methods, while effective in enhancing specific model capabilities (Devlin, 2018; Thirunavukarasu et al., 2023; Gururangan et al., 2020), often fall short in providing the necessary adaptability. These approaches are computationally intensive, leaving LLMs ill-equipped to handle the dynamic nature of real-world applications. In-Context Learning (ICL) (Brown, 2020) has emerged as a promising alternative, allowing LLMs to adapt to new tasks without additional training by leveraging their inherent capabilities (Team et al., 2023; Vacareanu et al., 2024; Agarwal et al., 2024). ICL, while effective, relies on carefully crafted demonstrations, requires extra overhead for each inference, and interrupts the context, potentially limiting its efficiency and flexibility (Lu et al., 2021; Zhao et al., 2021; Dong et al., 2022; Liu et al., 2023b).

We envision a next step in LLM adaptation: *Can we elicit and harness the potential of LLMs' inherent capabilities when adapting to new tasks, as demonstrated by ICL, while simultaneously maintaining efficiency and flexibility?*

Our research explores this question by introducing a novel approach ELICIT inspired by the concept of modularization (Pfeiffer et al., 2023; Fedus et al., 2022; Ding et al., 2023; Zhang et al., 2023b; Xiao et al., 2024). ELICIT involves the establishment of a comprehensive library of task vectors, each eliciting one in-context capability within the LLM. For any arbitrary query text, ELICIT dynamically leverage this capability library to selectively elicit the capability, effectively and flexibly tapping into the model's inherent capabilities on an as-needed basis. We empirically verify the advantages of ELICIT under 20 tasks and 4 models:

- **Efficient Capability Elicitation**: ELICIT aims to improve the model's task-specific capabilities with minimal additional computational cost during inference. Across 20 tasks and 4 models, ELICIT achieves an average improvement of 11.4% over zero-shot performance while maintaining the same token usage (Section 4.2).
- **Flexible Task Handling**: ELICIT can adapt to various tasks without requiring task-specific demonstrations or rigid templates, enhancing performance on both in-domain and unseen tasks (Sections 4.2 and 4.4).
- **Selective Capability Activation**: ELICIT allows for targeted activation of specific model capabilities based on the input query In our experiments with a math-only capability library, ELICIT boosted Math performance dramatically (e.g., from 2.6% to 21.3% for Mistral) while maintaining or slightly improving performance in other domains (Section 4.3).
- **Complementary Integration**: ELICIT shows potential for complementary use with existing methods, offering further performance gains. When combined with BM25 retrieval, ELICIT enhanced Pythia's average performance from 21.9% to 30.1% (Section 4.5).

Our key contributions are summarized as follows:

- We introduce a novel, modular framework for enhancing the adaptive capabilities of LLMs on demand with minimal computational overhead.
- We conduct extensive experiments to evaluate our method, showcasing its effectiveness across different query formats, language models, and tasks.
- We provide a thorough analysis of our method, offering insights into the design choices and their contributions to overall performance.

2 RELATED WORK

In-Context Learning. While Brown (2020) introduced In-Context Learning (ICL) as a simple yet effective way to enhance LLM performance by incorporating demonstrations into prompts, its applications have rapidly expanded across diverse domains. ICL enables model to adapt to a wide array of tasks ranging from traditional NLP benchmarks to more specialized tasks such as egression (Vacareanu et al., 2024), kNN classification (Agarwal et al., 2024; Dinh et al., 2022), and even jailbreaking (Anil et al., 2024). Researchers have actively explored various avenues to further enhance ICL's adaptability and effectiveness. These efforts include increasing demonstration quantity (Bertsch et al., 2024; Agarwal et al., 2024; Zhang et al., 2023a; Team et al., 2023), fine-tuning models for ICL (Min et al., 2021), leveraging prompt engineering (Nie et al., 2022), and implenmenting demonstration retrieval (Liu et al., 2021; Rubin et al., 2021; Li et al., 2023; Shi et al., 2022).

Concurrently, deeper insights into ICL's underlying mechanisms have been sought through diverse perspectives. Some researchers view ICL as a process of compressing training data into task-specific vectors (Hendel et al., 2023), while others relate it to gradient descent (Von Oswald et al., 2023) or analyze it through the lens of repetition (Yan et al., 2023) and memorization (Golchin et al., 2024). Building upon ICL advancements, we explored eliciting and harnessing LLMs' inherent capabilities for new task adaptation, akin to ICL, while maintaining efficiency and flexibility.

Task representation for ICL. Inspired by findings that intermediate representations in LLMs encode semantic meaning (Zou et al., 2023), researchers have explored injecting in-context learning demonstrations, encoded as function vectors, into intermediate representations to trigger desired predictions (Liu et al., 2023b; Hendel et al., 2023; Todd et al., 2023; Li et al., 2024). The scope of this research line has broadened to include different modalities,with recent work demonstrating its effectiveness in both visual (Hojel et al., 2025) and multimodal domains (Huang et al., 2024). However, this line of work focuses on manipulating internal representations. We are the first to comprehensively explore the modular approach of externally storing and retrieving such task representations to augment large language model capabilities.

Modular LLM. Examining and understanding the modular nature of large language models (LLMs) has become a crucial area of study for researchers (Pfeiffer et al., 2023; Fedus et al., 2022; Ding et al., 2023; Zhang et al., 2023b; Xiao et al., 2024). Initial investigations suggest that LLMs possess the capability to be broken down into distinct specialized components or modules. Some approaches introduce additional modules or parameters for optimization, including parameter-efficient tuning techniques like adapter layers (Houlsby et al., 2019; Pfeiffer et al., 2020), prompt tuning (Liu et al., 2023a; Ding et al., 2021), and parameter subset optimization methods such as BitFit (Zaken et al., 2021) and binary masks (Guo et al., 2020; Zhao et al., 2022). Other approaches involve training dedicated models for task composition (Shao et al., 2023; Mu et al., 2024) or merging fine-tuned parameter adjustments (Ilharco et al., 2022; Panigrahi et al., 2023; Merullo et al., 2023; Yu et al., 2024). Inspired by such modular perspectives, we explore the question of using task vectors in a modular way to dynamically elicit capabilities within the model.

3 METHOD

To elicit the hidden capability inside LLMs, we build our ELICIT by introducing a capability library which condenses each in-context learned capability into a task vector, and utilizing a retrieval module to strengthen the model when a task vector is helpful.

This section describes our implementation of ELICIT. We first formally define in-context learning Task Vectors (Section 3.1), and motivate our work. Then, we discuss the design choices of building capabilities libraries (Section 3.2), including the layer selection and intervention strategies. Finally, we introduce our retrieval module (Section 3.3) to dynamic elicit and leverage model's capability.

3.1 FROM ICL TO TASK VECTORS: FORMAL DEFINITIONS

In-Context Learning (ICL). Firstly, we define the framework for ICL. Let $\mathcal{T}$ represent a collection of tasks. For each task $t \in \mathcal{T}$, there exists a dataset $\mathbf{P}_t$ of in-context prompts. Each prompt $\mathbf{p}_i^t \in \mathbf{P}_t$ is a sequence of tokens that represents the i-th prompt for task t. Specifically, each prompt $\mathbf{p}_i^t$ consists of two components: (1) a set of N input-output demonstrations $\mathbf{D} = (\mathbf{x}_{ij}, \mathbf{y}_{ij})_{j=1}^{N}$ from task t, where j indexes the sequence of pairs ranging from 1 to N, and (2) a query input $\mathbf{x}_{iq}$, which is distinct from the inputs in $\mathbf{D}$. We formally represent an ICL prompt $\mathbf{p}_i^t$ as:

$$\mathbf{p}_i^t = [(\mathbf{x}_{i1}, \mathbf{y}_{i1}), \ldots, (\mathbf{x}_{iN}, \mathbf{y}_{iN}), \mathbf{x}_{iq}] \,. \tag{1}$$

The Language Model (LM) aims to predict the corresponding target response $\mathbf{y}_{iq}$ for the query input $\mathbf{x}_{iq}$. Through learning from the demonstrated input-output mappings in $\mathbf{D}$, ICL can enhances the model's capability to perform this task. We firstly introduce the hidden state in Transformers below.

Task Vector. Previous research (Hendel et al., 2023) introduced the concept of a task vector in the context of ICL. We build upon this foundation in our work. We first introduce the definition of hidden state representations in transformer models and task vector is derived from it.

Definition 3.1 (**Hidden State Representation in Transformers**). *Let T be an auto-regressive transformer language model with L layers. For each layer $l \in 1, \ldots, L$, we define $\mathbf{h}_l \in \mathbb{R}^d$ as the vector representation of the last token at layer l. The computation of $\mathbf{h}_l$ follows the recurrence relation (Vaswani, 2017): $\mathbf{h}_l = \mathbf{h}_{l-1} + \mathbf{m}_l + \mathbf{a}_l$, where $\mathbf{m}_l$ is the output of a multilayer perceptron at layer l, and $\mathbf{a}_l$ is the projection of the attention output into the hidden state at layer l.*

Having established the notion of hidden states in transformer models, we can now formally define the task vector within the ICL framework.

Definition 3.2 (**Task Vector $\boldsymbol{\theta}$**). *ICL functions by learning a task-specific mapping from demonstrations. This mapping is represented as a task vector $\boldsymbol{\theta}$. The task vector is derived from the activation state $\mathbf{h}_l$ (as defined in Definition 3.1) at a specific layer l, corresponding to the last token of the prompt. This vector subsequently steers the transformer to yield pertinent outputs for given queries.*

The task vector, as defined, encapsulates the essence of the task. This leads to the following lemma, highlighting its role in simulating ICL behavior.

Lemma 3.3 (Task Vector for ICL Simulation). *Given a task vector $\boldsymbol{\theta}$ that effectively captures the information from demonstrations in an ICL setting, we can simulate the behavior of regular ICL with only query as follows:*

$$T[\mathbf{p}_i^t] \approx f(\boldsymbol{\theta}; \mathbf{x}_{iq}),$$

where:

- $T[\mathbf{p}_i^t]$ *represents the output of the transformer model given a ICL prompt* $\mathbf{p}_i^t$ *defined as* (1).
- $f(\boldsymbol{\theta}; \mathbf{x}_q)$ *denotes a function that processes the query input* $\mathbf{x}_q$ *in a zero-shot manner, guided by the information encoded in the task vector* $\boldsymbol{\theta}$.

Remark 3.4 (Intervention of Task Vector $\boldsymbol{\theta}$). *The function* $f(\boldsymbol{\theta}; \mathbf{x}_q)$ *mentioned in Lemma 3.3 is an abstract concept expressing that the task vector can be used to influence the model's inference process. In practice,* $f(\boldsymbol{\theta}; \mathbf{x}_q)$ *is implemented through operations on the hidden states* $\mathbf{h}_l$ *and the task vector* $\boldsymbol{\theta}$. *Specifically, these operations can take the following forms:*

1. ***Replacement*** *(Hendel et al., 2023): The task vector* $\boldsymbol{\theta}$ *directly replaces the hidden state* $\mathbf{h}_l$, *i.e.,* $\tilde{\mathbf{h}}_l = \boldsymbol{\theta}$.
2. ***Linear combination*** *(Todd et al., 2023): The task vector* $\boldsymbol{\theta}$ *is combined linearly with the hidden state* $\mathbf{h}_l$, *i.e.,* $\tilde{\mathbf{h}}_l = \mathbf{h}_l + \alpha\boldsymbol{\theta}$, *where* α *is an adjustable scalar parameter.*

While previous research has demonstrated the existence and extractability of task vectors, it also has shown the potential for serving a technique to elicit the inherent capabilities when adapting to difference tasks as ICL, while simultaneously maintaing computational efficiency and flexibility.

We investigate the research problem through task vectors by proposing ELICIT, a framework designed to leverage these vectors for enhancing model capabilities. As shown in Figure 2, ELICIT consists of two main components:

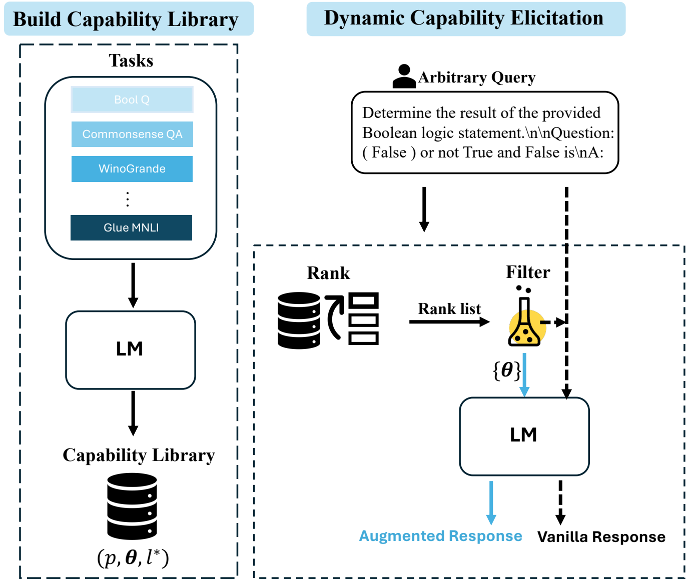
- **Build Capability Library**: A capability library that stores task vectors representing various in-context learned capabilities.
- **Dynamic Capability Elicitation**: A dynamic retrieval module that selectively activates relevant task vectors based on the input query.

3.2 BUILDING CAPABILITIES LIBRARY

To investigate the idea of ELICIT, we first create a library of in-context learned capabilities $\boldsymbol{\Theta} = \{\{\boldsymbol{\theta}_i^t\}_{i=1}^k\}_{t\in\mathcal{T}}$. Each element in this library is represented by a task vector (as defined in Definition 3.2). Here, k denotes the number of ICL prompts for each task t, and we use $k = 10$ for illustration. Definition 3.2 describes $\boldsymbol{\theta} \in \mathbb{R}^d$ as a single layer's hidden state. In our implementation, we collect $\boldsymbol{\theta} \in \mathbb{R}^{L\times d}$, which includes representations from all L layers, to enable the exploration of various designs for the sequential components of ELICIT.

The implementation of creating capability library involves two critical considerations: ① *Dynamic Layer Selection for* l^*, and ② *Intervention Strategies*. ① determines the appropriate layer l to intervene into during further reuse, utilizing the corresponding task vector, while ② decides how to appropriately intervene the task vector to influence the model's behavior (possible methods are described in Remark 3.4). Our framework addresses these considerations as follows.

① Dynamic Layer Selection for l^*. The selection of the optimal layer for task vector intervention is crucial for maximizing the effectiveness of our approach (Todd et al., 2023; Hendel et al., 2023). Appendix B further illustrates the variation in the optimal layer across different tasks. We implement a dynamic layer selection strategy to determine the optimal layer l^* for task vector intervention. While using a validation set to identify the optimal layer is not a novel concept, *our contribution lies in addressing the challenge of determining the intervention layer when applying the library in our proposed pipeline: we equip each task vector with its corresponding optimal layer, pre-identified during the library construction phase, thereby enabling efficient and effective reuse of task vectors during inference*. Our process is as follows:



Determine the result of the provided Boolean logic statement.\n\nQuestion: (False) or not True and False is\nA:

Figure 2: **Overview of the proposed ELICIT framework for Large Language Model Augmentation.** ELICIT consists of two modular components: (1) Build Capability Library - constructing a library of task-specific task vectors by learning from diverse task; (2) Dynamic Capability Elicitation - dynamically retrieving and integrating relevant task vectors from the library to augment the model's capability for an arbitrary input query.

- We obtain the task vector $\boldsymbol{\theta} \in \mathbb{R}^{L \times d}$ by processing the ICL prompt $\mathbf{p}_i^t$ (defined in (1), using randomly selected $N = 16$ demonstrations). This is done by extracting the hidden states $\{\mathbf{h}_l\}_{l=1}^{L}$ from all layers of the language model. We also store these ICL prompts for future use.
- Using the validation set, we perform a layer-by-layer analysis:
  - For each layer l, we intervene $\boldsymbol{\theta}_l$ during zero-shot query processing.
  - We measure zero-shot accuracy on the validation set for each intervention.
  - We identify l^* as the layer yielding the best accuracy.
- We repeat this process for each task-model combination, creating a library where each task vector $\boldsymbol{\theta}$ has its corresponding best layer l^*.

When reusing task vectors in library, for any $\boldsymbol{\theta} \in \boldsymbol{\Theta}$, we intervene at the pre-identified optimal layer l^* for each task-model combination. This dynamic selection method ensures the performance of the task vector and provides a generalizable framework adaptable to different tasks.

② **Intervention Strategies.** The concept of intervention, formally introduced in Remark 3.4, also outlines two methods for incorporating the task vector $\boldsymbol{\theta}_l$ into the model's inference process. We evaluate these two intervention strategies: 1) linear combination of the original hidden state and task vector with varying intervene strength α, and 2) direct replacement of the original hidden state with the task vector. We examine the impact of these intervention strategies on both task performance and language modeling capability, with the latter measured using cross-entropy loss on the pre-training dataset (i.e, WikiText).

Figure 3 provides a detailed visualization of how varying α affect both accuracy and cross-entropy loss in the Llama3-8B model across a diverse set of 20 tasks. *Results reveal a clear trade-off between task performance and language modeling capability as intervention strength increases.* Among the strategies tested, the additive approach $\tilde{\mathbf{h}}_l = \mathbf{h}_l + 2 \times \boldsymbol{\theta}_l$ consistently demonstrates superior performance across a wide range of tasks while minimizing degradation in language modeling ability. Results for other models are presented in the Appendix A, showing similar trends.

In conclusion. Our library contains $k \times |\mathcal{T}|$ items for each model, each consisting of three key components: (1) the ICL prompt $\mathbf{p}_i^t$, (2) the corresponding task vector $\boldsymbol{\theta} \in \mathbb{R}^{L \times d}$, and (3) the pre-identified optimal layer l^*.

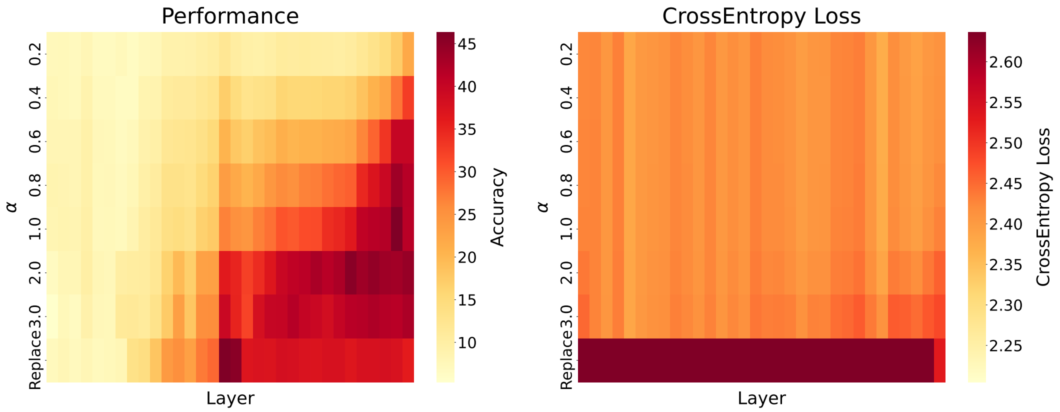
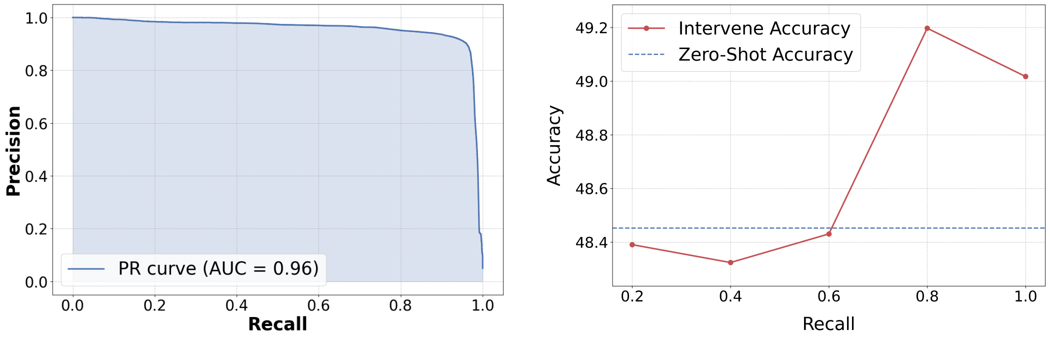


Figure 3: **Varying intervention strengths affect accuracy and cross-entropy loss in Llama3-8B on valid set of** 20 **tasks across different layer**. Higher in intervention strengths improve average task performance across layers but negatively impact language modeling capabilities. This reveals a trade-off between task-specific enhancement and general language modeling proficiency using task vectors.



(a) Precision-Recall Curves

(b) Valid Performance varying different recall.

Figure 4: **Precision-Recall Curves and recall sweeping on Llama3-8B in valid set across 20 tasks.** (a) Precision-Recall curves for the retriever across 20 tasks (AUC=0.96), guiding threshold selection for high recall and precision. (b) Validation set accuracy after intervention using different recall thresholds.

3.3 DYNAMIC CAPABILITY ELICITATION

After the creation of the capability library, as described in Section 3.2, we consider two considerations: (i) *Relevant Task Vector Selection*, and (ii) *Threshold-Based Filtering*. (i) Relevant Task Vector Selection focuses on identifying the most relevant task vectors from the library for a given test query q. We aim to find the most relevant task vectors $\boldsymbol{\theta}^q \in \boldsymbol{\Theta}$ stored in the library. Unlike traditional in-context learning (ICL), we lack meta-information about the query. (ii) Threshold-Based Filtering determines whether to utilize a retrieved task vector or not, to avoid compromising performance when no suitable task vectors are available in the library. Our framework addresses these challenges as follows:

(i) **Relevant Task Vector Selection.** We address the challenge of selecting the most relevant task vectors by employing a binary classifier to calculate similarity scores. This classifier is built upon the SimCSE RoBERTa model[1], augmented with a 2-layer Multi-Layer Perceptron (MLP) head. The architecture incorporates ReLU activation functions and a dropout rate of 0.2 for regularization.

We fine-tuned this model over 15 epochs using a learning rate of $2e{-}5$ on our curated dataset of 10,000 examples. The trained classifier is then used to compute similarity scores between a given query and each ICL prompt in our library. These scores are used to rank all library items, producing a similarity list of size $k \times |\mathcal{T}|$. The top-ranked task vector from this list is selected as our target for further processing.

(ii) **Threshold-Based Filtering.** To determine whether to utilize stored task vectors from our library, we implement a threshold-based approach using similarity scores. This threshold is established through a comprehensive analysis of the recall-precision trade-off across our validation set, as

---

[1] princeton-nlp/sup-simcse-roberta-base

Table 1: **Models used in this work.** We consider decoder-only auto-regressive language models and recurrent neural networks that are capable of ICL. For each model, we present the number of parameters, context window during training, and the number of layers $|L|$.

Model	HuggingFace ID	Citation	Parameters	Train Length	$\|L\|$
Llama 3	meta-llama/Meta-Llama-3-8B	Dubey et al. (2024)	8B	8k	32
Mistral	TIGER-Lab/Mistral-7B-Base-V0.2	Jiang et al. (2023)	7B	32k	32
Pythia	EleutherAI/pythia-2.8b	Biderman et al. (2023)	2.8B	2k	32
Mamba	state-spaces/mamba-2.8b-hf	Gu & Dao (2023)	2.8B	2k	64

illustrated in Figures 4a, utilizing the aggregated similarity lists for all samples. The AUC scores of our precision-recall curves (i.e., 0.96) demonstrate the high effectiveness of our threshold-based approach in accurately determining whether stored task vectors require intervention. Our evaluation of various recall levels, as shown in Figures 4b, reveals that a recall of 0.8 provides the optimal balance for our pipeline, other models' results shown in Appendix D.

Our decision process of how to utilize the similarity list and threshold to choose whether to use the stored task vectors and what task vectors to apply is as follows:

- We implement Dynamic Top-K Thresholding (DTT). If the highest similarity score exceeds the threshold, we select the top 10 task vectors from the ranked list for further processing.
- We then employ a majority voting mechanism among the optimal layers suggested by these top vectors to determine the final layer for intervention.
- In cases where the highest similarity score falls below the threshold, we refrain from using any stored task vector, relying instead on the model's base capabilities.

4 EXPERIMENTS

To comprehensively evaluate the effectiveness of ELICIT, we conduct a series of experiments designed to explore the following key questions:

- *Capability Elicitation Efficiency:* Can ELICIT effectively elicit the model's capabilities without incurring significant additional computational costs?
- *Selective Activation:* Is ELICIT capable of selectively activating relevant capabilities as needed for specific tasks?
- *Complementarity:* How well does ELICIT integrate with and complement existing methods in the field?
- *Generalization:* Can ELICIT handle novel queries, particularly those that diverge significantly from the task vectors currently stored in the library?

4.1 EXPERIMENT SETUP

Model. We utilize decoder-only auto-regressive language models (Pythia-2.8B (Biderman et al., 2023), LLaMA3-8B (Dubey et al., 2024), and Mistral-7B (Jiang et al., 2023)) and recurrent neural network (Mamba-2.8B (Gu & Dao, 2023)). Table 1 provides a comprehensive overview of these models, detailing their key characteristics including the number of parameters, layer numbers, and training context window size. For all models, we use the corresponding `huggingface` implementations (Wolf et al., 2020).

Tasks. To assess the efficacy of our proposed pipeline across a diverse array of scenarios, we have meticulously constructed a benchmark comprising 20 distinct tasks. This benchmark is designed to evaluate the model's performance on both classification and multiple-choice problems, spanning a wide spectrum of applications and complexities. The tasks are categorized into five domains:

- **Knowledge**: CommonsenseQA (Talmor et al., 2018), OpenBookQA (Mihaylov et al., 2018), HellaSwag (Zellers et al., 2019), and BoolQ (Clark et al., 2019);
- **Reasoning**: Four subsets from Big-Bench Hard (BBH) (Suzgun et al., 2022) and ARC-Challenge (Clark et al., 2018);
- **Mathematics**: MathQA (Amini et al., 2019) and MMLU Pro-MATH (Wang et al., 2024);
- **Safety**: Crows-Pairs (Nangia et al., 2020), BBQ-Age (Parrish et al., 2021), Ethics-Commonsense, and Ethics-Justice (Merity et al., 2016);

Table 2: Performance of ELICIT across model and tasks. ELICIT significantly enhances performance while maintaining the same token usage as Zero-shot, often achieving results comparable to or better than 16-shot and 16-shot BM25 ICL retriever methods. This improvement is consistent across various models and tasks, demonstrating ELICIT's efficiency and effectiveness in boosting model capabilities without increasing computational demands. We sample 100 examples per task across three random seeds.

Model		# Tokens	NLU	Reasoning	Knowledge	Math	Safety	Avg.
Llama3	16-shot	1574.7 ± 0.2	42.4 ± 0.9	36.1 ± 0.4	59.9 ± 0.2	21.1 ± 1.9	50.9 ± 0.6	42.1 ± 0.2
	bm25	2145.5 ± 29.1	44.3 ± 1.4	47.1 ± 0.3	61.6 ± 0.8	25.7 ± 1.8	45.6 ± 2.1	44.9 ± 0.4
	Zero-shot	108.2 ± 1.4	32.2 ± 1.2	32.9 ± 0.2	42.5 ± 1.2	14.0 ± 1.0	35.5 ± 1.2	31.4 ± 0.7
	ELICIT	108.2 ± 1.4	**38.1 ± 0.9**	**46.1 ± 0.3**	**60.7 ± 1.2**	**19.4 ± 1.1**	**49.4 ± 2.1**	**42.7 ± 0.8**
Mistral	16-shot	1797.3 ± 2.8	50.0 ± 0.9	41.6 ± 0.5	60.8 ± 0.9	17.6 ± 1.6	55.5 ± 1.3	45.1 ± 0.5
	bm25	2403.5 ± 32.1	44.5 ± 1.3	43.1 ± 0.4	60.1 ± 1.0	18.6 ± 0.8	52.3 ± 2.6	43.7 ± 0.7
	Zero-shot	123.4 ± 1.7	29.5 ± 1.2	26.2 ± 0.4	45.5 ± 1.3	2.8 ± 0.1	36.1 ± 0.3	28.0 ± 0.5
	ELICIT	123.4 ± 1.7	**36.9 ± 1.0**	**47.1 ± 0.6**	**60.4 ± 1.1**	**20.4 ± 1.1**	**47.9 ± 1.6**	**42.5 ± 0.5**
Pythia	16-shot	1597.6 ± 2.4	48.1 ± 0.4	22.2 ± 0.8	12.5 ± 0.7	10.3 ± 0.9	28.2 ± 0.9	24.3 ± 0.4
	bm25	2190.5 ± 30.2	38.8 ± 0.6	20.3 ± 0.1	12.5 ± 0.6	11.7 ± 0.8	26.2 ± 1.3	**21.9 ± 0.2**
	Zero-shot	109.8 ± 1.5	42.9 ± 0.3	18.9 ± 0.2	21.8 ± 1.6	7.3 ± 0.2	32.5 ± 1.2	24.7 ± 0.4
	ELICIT	109.8 ± 1.5	**60.1 ± 0.1**	**25.7 ± 0.9**	**20.9 ± 1.2**	**14.4 ± 1.3**	**40.9 ± 2.5**	**32.4 ± 0.4**
Mamba	16-shot	1601.5 ± 1.7	31.0 ± 0.9	24.0 ± 0.7	27.2 ± 1.1	12.6 ± 1.0	36.4 ± 0.4	26.2 ± 0.3
	bm25	2190.5 ± 30.2	34.6 ± 1.8	23.3 ± 0.6	23.8 ± 1.1	12.9 ± 2.1	29.3 ± 1.6	24.8 ± 0.3
	Zero-shot	109.8 ± 1.5	36.1 ± 1.5	19.5 ± 0.5	17.3 ± 1.2	5.8 ± 1.2	30.2 ± 0.2	21.8 ± 0.2
	ELICIT	109.8 ± 1.5	**45.6 ± 1.4**	**29.3 ± 0.7**	**30.9 ± 1.4**	**15.4 ± 1.4**	**47.3 ± 1.6**	**33.7 ± 0.7**

- **Natural Language Understanding (NLU)**: GLUE (SST-2, QNLI, MNLI) (Wang, 2018) and SuperGLUE (WIC, RTE) (Wang et al., 2019).

Evaluation. To evaluate ELICIT under a real usage scenario, where the demonstrations can hardly be at the same format with the test query, we augment the test query with two additional formats different from the demonstration in library. Furthermore, in our preliminary experiments, we find that zero-shot LLMs cannot answer properly with contextual guidance. Thus, to ensure a fair comparison with the zero-shot scenario, we add task templates before the test query. More details and examples can be found in Appendix E.

Baselines. Our primary baseline is the zero-shot performance of LLMs, as our method maintains the same token usage. For reference, we also include in-context learning (ICL) and BM25 (Robertson et al., 2009) retrieval of 16 examples from the same pool of examples used in constructing the capability library. However, these are not directly comparable to our method, due to the raised nearly 20 times more tokens consuming. The ICL baseline is task-specific, requiring knowledge of each query's task type to use corresponding demonstrations. In contrast, our method is task-agnostic, applicable across various tasks without needing task-specific information or prompts.

4.2 EFFICIENT CAPABILITY ELICITATION

ELICIT *achieves efficiently eliciting models' capabilities.* From Tables 2, comparing the zero-shot baseline and ELICIT, we observe that ELICIT significantly elicits model capabilities across most tasks without increasing token usage. Across the 20 tasks, ELICIT achieves an average improvement of 11.4% across different models. For Llama3, ELICIT improves over zero-shot by 15.1% while using the same 108.2 tokens. ELICIT demonstrates substantial gains in Reasoning (e.g., +14.4% for LLama3) and Safety tasks (e.g., +14.0% for LLama3). In many cases, ELICIT's performance is comparable to or surpasses that of 16-shot and BM25 methods, despite their higher token requirements. Furthermore, it exhibits robustness across various template formats, highlighting its versatility.

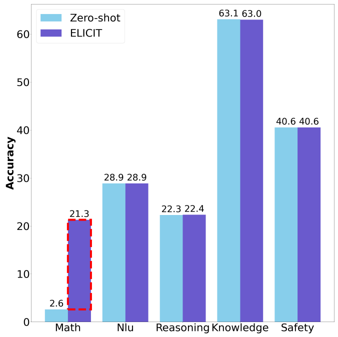


Figure 5: Performance on ELICIT across different domains when the library only contains math-related task vectors on Mistral.

4.3 SELECTIVE ADAPTIVE ACTIVATION OF CAPABILITIES

ELICIT *elicits capability when necessary.* We demonstrate selective activation by constructing a library containing only math-related task vectors, as shown in Figure 5. The results clearly illustrate that ELICIT significantly boosts performance in the Math domain, with a dramatic increase from

Table 3: **ELICIT can generalize to unseen tasks.** ELICIT achieves significant performance gains without additional token usage across different models and unseen tasks. We sample 100 examples per task across three random seeds. We use BM25 retrieval of 16 examples as baseline.

		# Tokens	**GLUE COLA**	**BBQ Religion**	**Deepmind**	**MMLU-Psychology**	**BBH-five-objects**	**Avg**
Llama	**BM25**	2066.9 ± 22.5	**83.6 ± 0.8**	54.5 ± 0.7	**29.1 ± 1.9**	81.6 ± 0.4	29.6 ± 0.0	55.7 ± 0.2
	Zero-shot	102.9 ± 0.7	72.0 ± 0.7	38.6 ± 1.1	17.5 ± 2.6	**54.2 ± 0.3**	**14.3 ± 0.0**	39.3 ± 0.8
	ELICIT	102.9 ± 0.7	**73.4 ± 0.6**	**47.8 ± 0.8**	**23.1 ± 3.4**	**75.9 ± 0.2**	**26.2 ± 0.0**	**49.3 ± 0.5**
Mistral	**BM25**	**2300.3 ± 23.5**	84.6 ± 1.2	**57.6 ± 0.6**	19.5 ± 2.2	**74.7 ± 0.9**	**15.1 ± 0.0**	50.3 ± 0.5
	Zero-shot	114.6 ± 0.8	43.3 ± 1.1	35.4 ± 3.3	9.0 ± 0.4	57.9 ± 0.7	6.9 ± 0.0	30.5 ± 1.0
	ELICIT	114.6 ± 0.8	**46.2 ± 2.0**	**45.0 ± 2.7**	**25.4 ± 1.7**	**71.2 ± 0.4**	**24.0 ± 0.0**	**42.4 ± 1.1**
Pythia	**BM25**	2140.4 ± 23.8	67.4 ± 0.7	30.4 ± 2.9	19.6 ± 1.3	15.9 ± 0.4	14.6 ± 0.0	29.6 ± 0.8
	Zero-shot	**105.9 ± 0.7**	70.3 ± 1.0	21.3 ± 2.9	10.9 ± 0.7	13.9 ± 0.5	6.2 ± 0.0	24.5 ± 0.4
	ELICIT	105.9 ± 0.7	**72.4 ± 0.4**	**33.2 ± 2.7**	**23.3 ± 1.7**	**26.4 ± 0.4**	**13.9 ± 0.0**	**33.8 ± 0.4**
Mamba	**BM25**	2140.4 ± 23.8	54.3 ± 1.3	29.7 ± 0.5	13.1 ± 2.0	15.1 ± 0.3	5.8 ± 0.0	23.6 ± 0.4
	Zero-shot	105.9 ± 0.7	48.5 ± 0.6	22.0 ± 2.0	9.8 ± 1.3	19.9 ± 0.9	10.0 ± 0.0	22.0 ± 0.2
	ELICIT	105.9 ± 0.7	**51.5 ± 1.0**	**31.0 ± 4.5**	**13.5 ± 3.0**	**24.1 ± 1.6**	**18.3 ± 0.0**	**27.7 ± 1.1**

2.6% to 21.3%, while maintaining performance in other domains for Mistral. Notably, the Reasoning domain also shows a slight improvement, increasing from 22.3% to 22.4%. This behavior stems from ELICIT's selective application of task vectors from library, which are not applied when no relevant tasks vectors are detected. More discussion is presented in Appendix J. Results for other models, presented in Appendix F, demonstrate a similar trend. The striking improvement in Math performance, coupled with the subtle gain in Reasoning and the stability in other domains, demonstrates ELICIT's capacity for targeted capability activation, making it a flexible and efficient performance enhancer.

4.4 ELICIT GENERALIZE TO UNSEEN TASKS WITHOUT ADDTIONAL INFORMATION

ELICIT *generalizes across unseen tasks.* In Table 3, we observe that ELICIT significantly improves model performance on unseen tasks (GLUE-COLA, BBQ Religion, Deepmind (Saxton et al., 2019), MMLU-Psychology, and BBH-Logical-Deduction-Five-objects) not present in its capability library. Across all models, ELICIT consistently outperforms the Zero-shot baseline. In several cases, it even approaches or surpasses the BM25 retrieval baseline, despite using substantially fewer tokens. For instance, on the MMLU-Psychology task, ELICIT achieves a 21.7% absolute improvement over Zero-shot, surpassing BM25 retrieval by 5.7% for Llama3 Model. These results are achieved without additional token usage and task information, demonstrating ELICIT's efficiency, flexibility, and generalization ability.

4.5 COMPLEMENTARY INTERGRATION

ELICIT *serves as a versatile plug-and-play performance booster.* Tables 4 and 9 showcase ELICIT's remarkable ability to function as an orthogonal, plug-and-play enhancement to existing solutions like BM25 retrieval. In Table 4, we observe consistent performance gains across all models and tasks when ELICIT is combined with BM25 retrieval. Notably, Pythia demonstrates substantial improvements, with its average performance increasing from 21.9% to 30.1% for in-domain tasks, and from 23.6% to 28.4% on unseen tasks. For instance, the Llama3 model on the MMLU-Psychology task achieves 81.2% with ELICIT, surpassing BM25 retreival alone (81.6%), persisting across various models and tasks. The combination with BM24 Retrieval results on unseen tasks is shown in H. Its plug-and-play nature allows for seamless integration with existing methods, enhancing model performance in both in-domain and unseen task scenarios without compromising the benefits of current approaches.

5 ABLATION STUDY

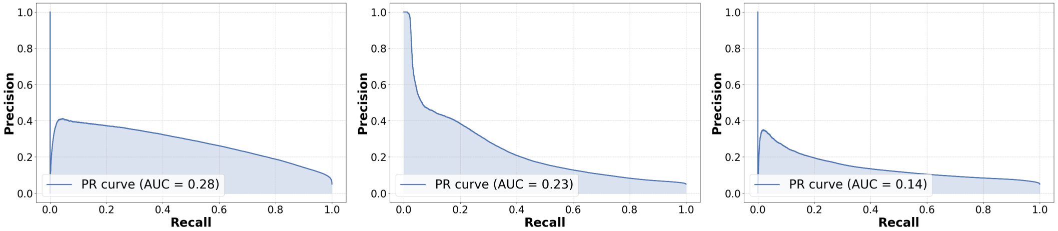
5.1 SIMILARITY-BASED RETRIEVE

We also explored similarity-based retrieval methods, such as cosine similarity, t-SNE distance, and Euclidean distance between the query embedding and the task vectors $\boldsymbol{\theta}$ in capability library. However, as illustrated in Figure 6, the precision-recall curves for these methods on Llama3 exhibit very low AUC scores, with the highest being a mere 0.28. These poor AUC values indicate that the discrimination ability of these similarity-based approaches is inadequate for effectively identifying relevant task vectors from the library. The precision-recall curves for similarity-based methods on

Table 4: **ELICIT as an plug-and-play performance booster**: Performance Gains When Combined with BM25 on In-Domain Tasks.

Model		NLU	Reasoning	Knowledge	Math	Safety	Avg.
Llama	**BM25**	**44.3 ± 1.3**	47.1 ± 0.3	**61.6 ± 0.8**	**25.7 ± 1.8**	45.6 ± 2.1	44.9 ± 0.4
	BM25+ELICIT	43.9 ± 1.7	**49.9 ± 0.5**	60.5 ± 1.0	25.2 ± 1.2	**48.7 ± 3.2**	**45.6 ± 0.7**
Mistral	**BM25**	44.5 ± 1.3	43.1 ± 0.4	60.0 ± 1.0	18.6 ± 0.8	**52.3 ± 2.6**	43.7 ± 0.8
	BM25+ELICIT	**47.7 ± 1.3**	**47.1 ± 0.6**	**61.8 ± 1.5**	**23.5 ± 2.2**	49.2 ± 3.0	**45.9 ± 0.2**
Pythia	**BM25**	38.8 ± 0.7	20.4 ± 0.2	12.6 ± 0.6	11.7 ± 0.7	26.2 ± 1.3	21.9 ± 0.2
	BM25+ELICIT	**60.9 ± 1.0**	**23.4 ± 1.0**	**15.4 ± 0.8**	**14.3 ± 1.5**	**36.3 ± 0.9**	**30.1 ± 0.3**
Mamba	**BM25**	34.6 ± 1.8	23.4 ± 0.6	23.7 ± 1.1	12.9 ± 2.0	29.3 ± 1.6	24.8 ± 0.3
	BM25+ELICIT	**44.4 ± 1.9**	**31.1 ± 0.3**	**30.4 ± 1.7**	**15.9 ± 2.7**	**42.3 ± 2.1**	**32.8 ± 0.3**

other models are presented in Appendix G, further highlighting their suboptimal performance. In stark contrast, the trained retriever in our proposed design can achieve a remarkably high AUC of 0.96 (Figure 4a). This substantial improvement in retrieval performance underscores the benefits of our design, which effectively leverages learning-based techniques to intelligently retrieve and integrate relevant capabilities from the library.



(a) Cosine Similarity (b) Euclidean Distance (c) t-SNE Distance

Figure 6: **Precision-Recall Curves for Similarity-based methods for Llama3**.

5.2 SELECTION AFTER THRESHOLD FILTERING

In Table 5, we investigate the impact of the number of selected states n when the top1 similarity score reaches the threshold, as well as the effect of using Dynamic Top-K Thresholding (DTT) or not as Section 3.3 mentioned. The results show that selecting $n = 10$ and using Dynamic Top-K Thresholding achieves the best performance on the validation set. Choosing fewer vector quantities (e.g., n=5) would limit the method's potential, while selecting too many (e.g., n=15) could introduce irrelevant noise, thereby degrading performance. Dynamic Top-K adaptively sets the similarity threshold to ensure that only sufficiently relevant vectors are utilized. This ablation study highlights the rationale and effectiveness of our design choices.

Table 5: Ablation Study on DDT on Valid for Llama3-8B.

	TP
zs-shot	33.62
n=5	44.71
n=15	45.02
n=10,w/o DTT	45.17
n=10	**45.29**

6 CONCLUSION

In this paper, we explore the vision of eliciting and harnessing the potential of large language models' inherent capabilities when adapting to new tasks, akin to in-context learning (ICL), while maintaining efficiency and flexibility. We propose ELICIT, a novel framework consisting of two key modules: Build Capability Library and Dynamic Capability Elicitation. ELICIT achieves consistent improvements across diverse tasks, input formats, and model architectures. Our results show that ELICIT not only has the potential to harness models' latent abilities without introducing substantial additional computational cost, but also advances language models' performance, versatility, adaptability, and scalability.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Rishabh Agarwal, Avi Singh, Lei M Zhang, Bernd Bohnet, Stephanie Chan, Ankesh Anand, Zaheer Abbas, Azade Nova, John D Co-Reyes, Eric Chu, et al. Many-shot in-context learning. *arXiv preprint arXiv:2404.11018*, 2024.

Aida Amini, Saadia Gabriel, Peter Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. *arXiv preprint arXiv:1905.13319*, 2019.

Cem Anil, Esin Durmus, Mrinank Sharma, Joe Benton, Sandipan Kundu, Joshua Batson, Nina Rimsky, Meg Tong, Jesse Mu, Daniel Ford, et al. Many-shot jailbreaking. *Anthropic, April*, 2024.

Amanda Bertsch, Maor Ivgi, Uri Alon, Jonathan Berant, Matthew R Gormley, and Graham Neubig. In-context learning with long-context models: An in-depth exploration. *arXiv preprint arXiv:2405.00200*, 2024.

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. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pp. 2397–2430. PMLR, 2023.

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

Tom B Brown. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Stephanie Chan, Adam Santoro, Andrew Lampinen, Jane Wang, Aaditya Singh, Pierre Richemond, James McClelland, and Felix Hill. Data distributional properties drive emergent in-context learning in transformers. *Advances in Neural Information Processing Systems*, 35:18878–18891, 2022.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*, 2019.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Jacob Devlin. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Ning Ding, Shengding Hu, Weilin Zhao, Yulin Chen, Zhiyuan Liu, Hai-Tao Zheng, and Maosong Sun. Openprompt: An open-source framework for prompt-learning. *arXiv preprint arXiv:2111.01998*, 2021.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 5(3):220–235, 2023.

Tuan Dinh, Yuchen Zeng, Ruisu Zhang, Ziqian Lin, Michael Gira, Shashank Rajput, Jy-yong Sohn, Dimitris Papailiopoulos, and Kangwook Lee. Lift: Language-interfaced fine-tuning for non-language machine learning tasks. *Advances in Neural Information Processing Systems*, 35: 11763–11784, 2022.

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

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

William Fedus, Jeff Dean, and Barret Zoph. A review of sparse expert models in deep learning. *arXiv preprint arXiv:2209.01667*, 2022.

Changjiang Gao, Shujian Huang, Jixing Li, and Jiajun Chen. Roles of scaling and instruction tuning in language perception: Model vs. human attention. *arXiv preprint arXiv:2310.19084*, 2023a.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 12 2023b. URL https://zenodo.org/records/10256836.

Shahriar Golchin, Mihai Surdeanu, Steven Bethard, Eduardo Blanco, and Ellen Riloff. Memorization in in-context learning. *arXiv preprint arXiv:2408.11546*, 2024.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

Demi Guo, Alexander M Rush, and Yoon Kim. Parameter-efficient transfer learning with diff pruning. *arXiv preprint arXiv:2012.07463*, 2020.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. Don't stop pretraining: Adapt language models to domains and tasks. *arXiv preprint arXiv:2004.10964*, 2020.

Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Yuan Yao, Ao Zhang, Liang Zhang, et al. Pre-trained models: Past, present and future. *AI Open*, 2:225–250, 2021.

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

Alberto Hojel, Yutong Bai, Trevor Darrell, Amir Globerson, and Amir Bar. Finding visual task vectors. In *European Conference on Computer Vision*, pp. 257–273. Springer, 2025.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pp. 2790–2799. PMLR, 2019.

Brandon Huang, Chancharik Mitra, Assaf Arbelle, Leonid Karlinsky, Trevor Darrell, and Roei Herzig. Multimodal task vectors enable many-shot multimodal in-context learning. *arXiv preprint arXiv:2406.15334*, 2024.

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

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

Dongfang Li, Zhenyu Liu, Xinshuo Hu, Zetian Sun, Baotian Hu, and Min Zhang. In-context learning state vector with inner and momentum optimization. *arXiv preprint arXiv:2404.11225*, 2024.

Xiaonan Li, Kai Lv, Hang Yan, Tianyang Lin, Wei Zhu, Yuan Ni, Guotong Xie, Xiaoling Wang, and Xipeng Qiu. Unified demonstration retriever for in-context learning. *arXiv preprint arXiv:2305.04320*, 2023.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. What makes good in-context examples for gpt-3? *arXiv preprint arXiv:2101.06804*, 2021.

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 arXiv:2311.06668*, 2023b.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. *arXiv preprint arXiv:2104.08786*, 2021.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

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

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*, 2018.

Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. Metaicl: Learning to learn in context. *arXiv preprint arXiv:2110.15943*, 2021.

Jesse Mu, Xiang Li, and Noah Goodman. Learning to compress prompts with gist tokens. *Advances in Neural Information Processing Systems*, 36, 2024.

Nikita Nangia, Clara Vania, Rasika Bhalerao, and Samuel R Bowman. Crows-pairs: A challenge dataset for measuring social biases in masked language models. *arXiv preprint arXiv:2010.00133*, 2020.

Feng Nie, Meixi Chen, Zhirui Zhang, and Xu Cheng. Improving few-shot performance of language models via nearest neighbor calibration. *arXiv preprint arXiv:2212.02216*, 2022.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, et al. In-context learning and induction heads. *arXiv preprint arXiv: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*, pp. 27011–27033. PMLR, 2023.

Alicia Parrish, Angelica Chen, Nikita Nangia, Vishakh Padmakumar, Jason Phang, Jana Thompson, Phu Mon Htut, and Samuel R Bowman. Bbq: A hand-built bias benchmark for question answering. *arXiv preprint arXiv:2110.08193*, 2021.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. Mad-x: An adapter-based framework for multi-task cross-lingual transfer. *arXiv preprint arXiv:2005.00052*, 2020.

Jonas Pfeiffer, Sebastian Ruder, Ivan Vulić, and Edoardo Maria Ponti. Modular deep learning. *arXiv preprint arXiv:2302.11529*, 2023.

Stephen Robertson, Hugo Zaragoza, et al. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389, 2009.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. Learning to retrieve prompts for in-context learning. *arXiv preprint arXiv:2112.08633*, 2021.

David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. Analysing mathematical reasoning abilities of neural models. *arXiv preprint arXiv:1904.01557*, 2019.

Nan Shao, Zefan Cai, Chonghua Liao, Yanan Zheng, Zhilin Yang, et al. Compositional task representations for large language models. In *The Eleventh International Conference on Learning Representations*, 2023.

Peng Shi, Rui Zhang, He Bai, and Jimmy Lin. Xricl: Cross-lingual retrieval-augmented in-context learning for cross-lingual text-to-sql semantic parsing. *arXiv preprint arXiv:2210.13693*, 2022.

Aaditya K Singh, Ted Moskovitz, Felix Hill, Stephanie CY Chan, and Andrew M Saxe. What needs to go right for an induction head? a mechanistic study of in-context learning circuits and their formation. *arXiv preprint arXiv:2404.07129*, 2024.

Hongjin Su, Jungo Kasai, Chen Henry Wu, Weijia Shi, Tianlu Wang, Jiayi Xin, Rui Zhang, Mari Ostendorf, Luke Zettlemoyer, Noah A Smith, et al. Selective annotation makes language models better few-shot learners. *arXiv preprint arXiv:2209.01975*, 2022.

Jiuding Sun, Chantal Shaib, and Byron C Wallace. Evaluating the zero-shot robustness of instruction-tuned language models. *arXiv preprint arXiv:2306.11270*, 2023.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*, 2022.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *arXiv preprint arXiv:1811.00937*, 2018.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Arun James Thirunavukarasu, Darren Shu Jeng Ting, Kabilan Elangovan, Laura Gutierrez, Ting Fang Tan, and Daniel Shu Wei Ting. Large language models in medicine. *Nature medicine*, 29(8): 1930–1940, 2023.

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

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Robert Vacareanu, Vlad-Andrei Negru, Vasile Suciu, and Mihai Surdeanu. From words to numbers: Your large language model is secretly a capable regressor when given in-context examples. *arXiv preprint arXiv:2404.07544*, 2024.

A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

Johannes Von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, pp. 35151–35174. PMLR, 2023.

Alex Wang. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, 32, 2019.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *arXiv preprint arXiv:2406.01574*, 2024.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pp. 38–45, 2020.

Chaojun Xiao, Zhengyan Zhang, Chenyang Song, Dazhi Jiang, Feng Yao, Xu Han, Xiaozhi Wang, Shuo Wang, Yufei Huang, Guanyu Lin, et al. Configurable foundation models: Building llms from a modular perspective. *arXiv preprint arXiv:2409.02877*, 2024.

Jianhao Yan, Jin Xu, Chiyu Song, Chenming Wu, Yafu Li, and Yue Zhang. Understanding in-context learning from repetitions. *arXiv preprint arXiv:2310.00297*, 2023.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *Forty-first International Conference on Machine Learning*, 2024.

Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv preprint arXiv:2106.10199*, 2021.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

Wenxuan Zhang, Yue Deng, Bing Liu, Sinno Jialin Pan, and Lidong Bing. Sentiment analysis in the era of large language models: A reality check. *arXiv preprint arXiv:2305.15005*, 2023a.

Zhengyan Zhang, Zhiyuan Zeng, Yankai Lin, Chaojun Xiao, Xiaozhi Wang, Xu Han, Zhiyuan Liu, Ruobing Xie, Maosong Sun, and Jie Zhou. Emergent modularity in pre-trained transformers. *arXiv preprint arXiv:2305.18390*, 2023b.

Kang Zhao, Hua Xu, Jiangong Yang, and Kai Gao. Consistent representation learning for continual relation extraction. *arXiv preprint arXiv:2203.02721*, 2022.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. Calibrate before use: Improving few-shot performance of language models. In *International conference on machine learning*, pp. 12697–12706. PMLR, 2021.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, et al. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*, 2023.

A INTERVENTION STRATEGIES

As mentioned in Section 3.2, we choose $\tilde{\mathbf{h}}_l = \mathbf{h}_l + \alpha \cdot \boldsymbol{\theta}_l$ as the intervention strategy, where α is a scaling factor that controls the intervention strength. We observe the performance and cross-entropy loss across a diverse set of 20 tasks by varying α. The results for Mistral, Mamba, and Pythia are shown in Figures 7, 8, and 9, respectively. The results reveal a similar trade-off between task performance and language modeling capability as the intervention strength increases. Among the strategies tested, the additive approach consistently demonstrates superior performance across a wide range of tasks while minimizing degradation in language modeling ability. Across different models, α can be set to 2.0 to achieve a good balance between task performance and language modeling capability.

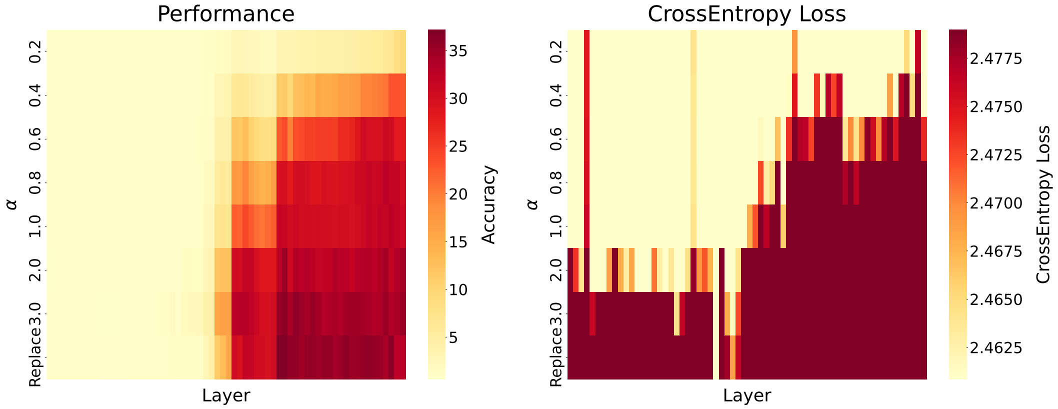


Figure 7: Varying intervention strengths affect accuracy and cross-entropy loss in Mamba on valid set of 20 tasks across different layer.

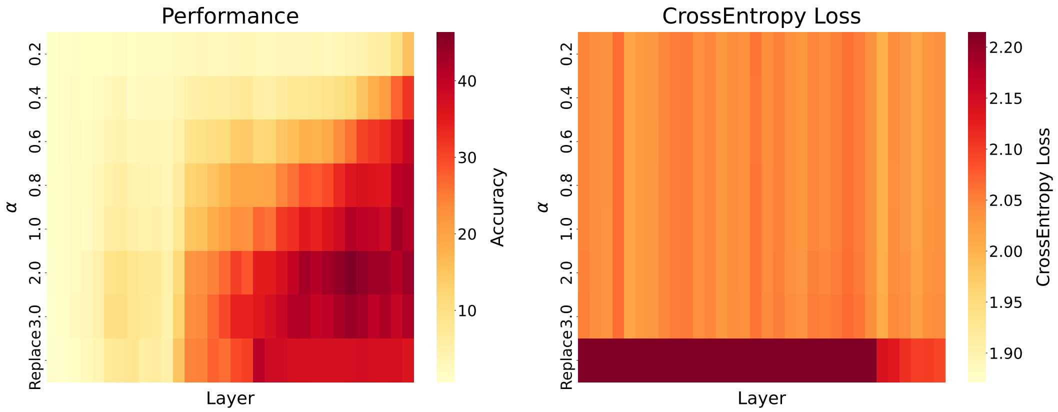


Figure 8: Varying intervention strengths affect accuracy and cross-entropy loss in Mistral on valid set of 20 tasks across different layer.

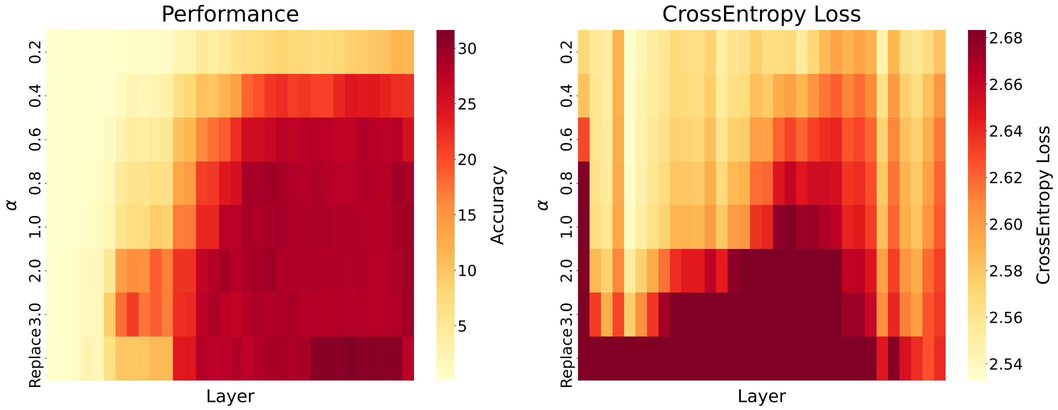


Figure 9: Varying intervention strengths affect accuracy and cross-entropy loss in Pythia on valid set of 20 tasks across different layer.

A.1 POSSIBLE INTERPRETATION OF TRADE-OFF

The trade-off between intervention strength and language modeling performance can be explained through neural circuit interactions. Prior work suggests ICL and general language modeling operate with different circuits (Chan et al., 2022; Olsson et al., 2022; Singh et al., 2024). We hypothesize that stronger interventions redirect activation patterns from pretrained language modeling circuitry towards ICL-specific patterns, creating tension between these distinct computational paths. As intervention strength increases, activations appear to deviate further from their pretrained configurations, potentially explaining the degraded performance on general language modeling tasks like WikiText. This observation suggests that optimal intervention strength requires careful balancing rather than maximization. Further investigation of these circuit-level understanding remains an important direction for future research.

B BEST LAYER FOR DIFFERENT TASKS.

As mentioned in Section 3.2, we identify the optimal intervention layer for different tasks. Figures 10, 11, and 12 illustrate that the optimal intervention layer varies significantly across tasks. For instance, the ARC challenge task achieves the best performance when intervening in the middle layers, while the GLUE SST2 task performs best when intervening in the later layers for the Llama-3 model. Furthermore, intervening at different layers leads to substantial performance variations. Therefore, instead of fixing the intervention layer as in Todd et al. (2023), we propose a dynamic layer selection approach to identify the optimal intervention layer for each task.

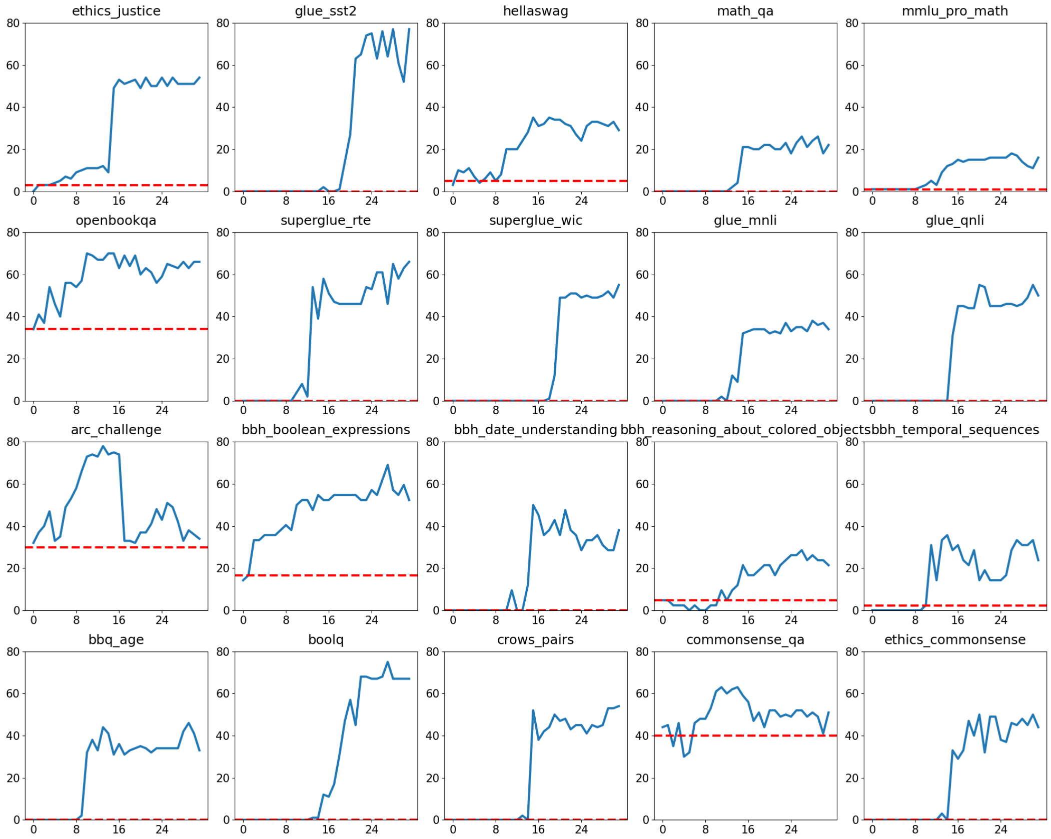


Figure 10: Performance distribution varying intervention layer on Llama3 8B in 20 tasks on Valid Set.

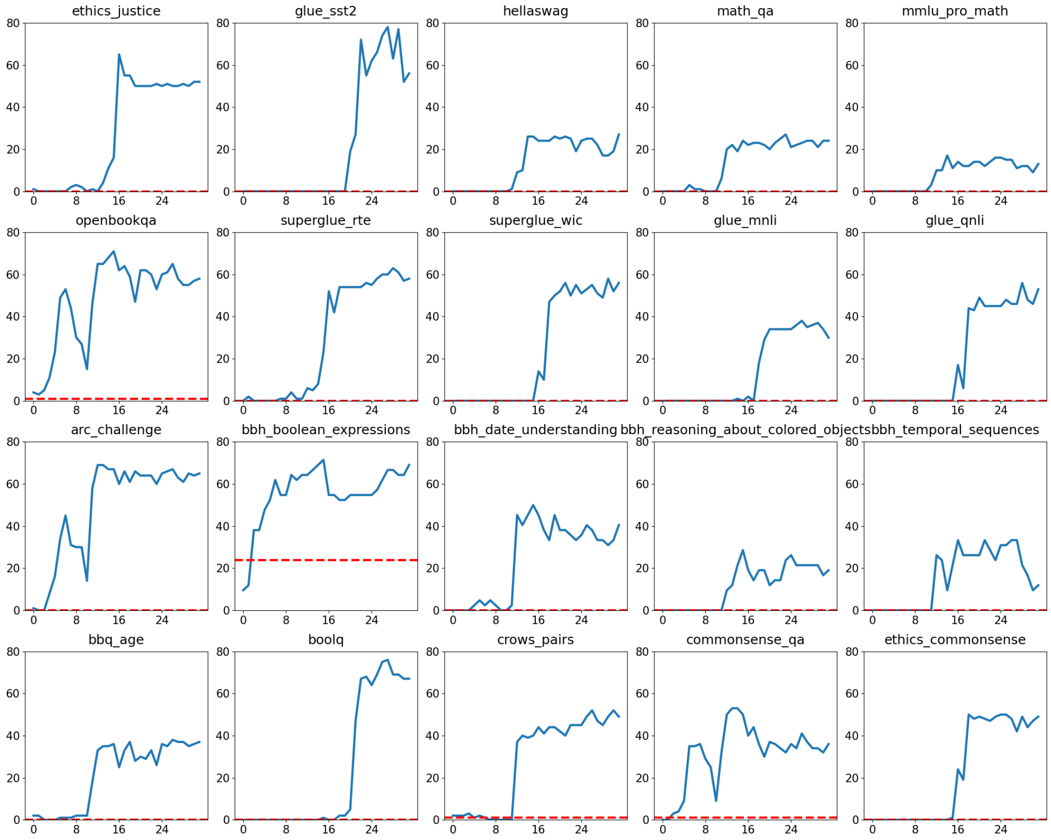


Figure 11: Performance distribution varying intervention layer on Mistral in 20 tasks on Valid Set.

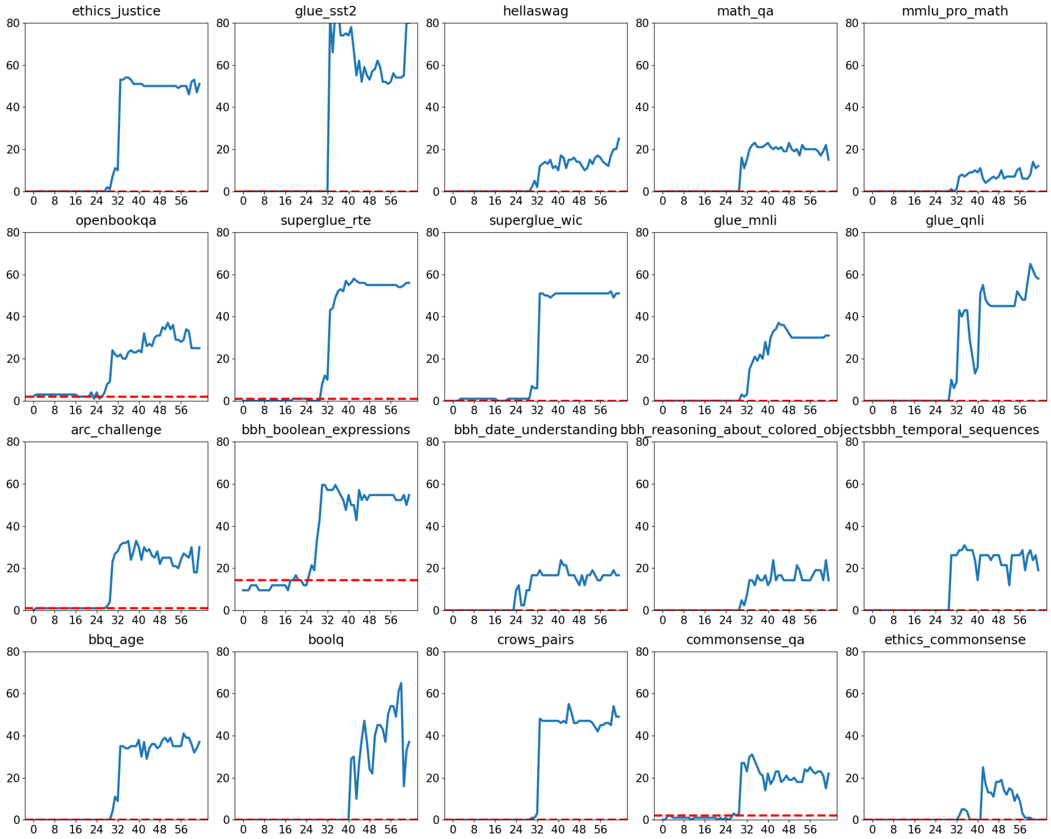


Figure 12: Performance distribution varying intervention layer on Mamba in 20 tasks on Valid Set.

C DATA CONSTRUCTION FOR RETRIEVER

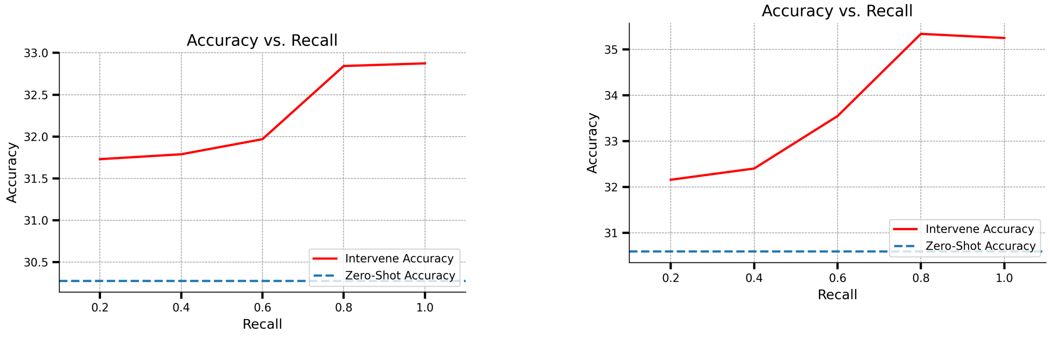
As mentioned in Section 3.3, we train a retriever on the constructed data. To train the retriever as a classifier, we construct data pairs to determine whether a task-specific prompt and an ICL prompt belong to the same task. For each task and each template, we sample two examples from the validation set. We create positive and negative pairs, where each pair consists of a task-specific prompt and an ICL prompt from the library. Negative examples are formed by randomly pairing task-specific prompts with ICL prompts from different tasks. We balance the data distribution across different tasks, ensuring that each task has an equal number of positive and negative pairs.

D RECALL SWEEP

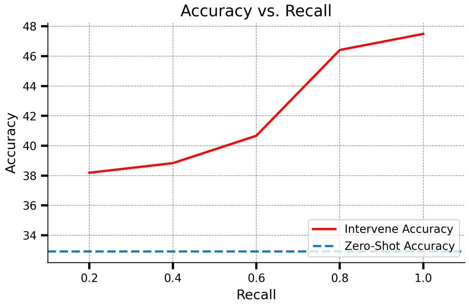
As mentioned in Section 3.3, we sweep the threshold determined by recall, and as shown in Figure13, the results reveal that a recall of 0.8 provides an optimal balance between accuracy and recall for our pipeline across Pythia, Mamba, and Mistral models.

We observe a similar increasing trend in performance as recall increases for all three models. At lower recall values, the intervene accuracy is close to the zero-shot accuracy, indicating that the retrieved prompts may not be relevant to the task. As recall increases, the intervene accuracy improves significantly, demonstrating the effectiveness of the proposed approach in selecting appropriate prompts for intervention.

Based on these observations, we choose a recall value of 0.8 to determine the threshold for filtering prompts across different models, as it strikes a balance between maximizing accuracy and maintaining a reasonable recall level.



(a) Pythia (b) Mamba



(c) Mistral

Figure 13: Accuracy vs. Recall curves for Pythia, Mamba, and Mistral models, illustrating the performance trade-off at different recall levels.

E EVALUATION SETTING

E.1 ICL SETTING

As 4.1 states, we first primarily implement ELICIT on the traditional In-Context Learning (ICL) setting. We find that zero-shot Large Language Models (LLMs) cannot answer properly with contextual guidance. Although ELICIT works on such a traditional ICL setting, as shown in Table 6 and 7, the zero-shot accuracy is almost 0, which is not plausible to evaluate model's performance. Therefore, we think it's not fair to augment model performance on such traditional zero-shot queries.

Table 6: Performance of ELICIT across model and tasks in ICL setting.

model		**#Tokens**	**NLU**	**Reasoning**	**Knowledge**	**Math**	**Safety**	**Avg.**
Llama	16-shot	1553.4 ± 2.8	60.3 ± 1.2	55.6 ± 0.2	69.2 ± 2.0	27.0 ± 0.0	60.9 ± 0.5	54.6 ± 0.6
	BM25	1799.2 ± 26.8	59.4 ± 0.3	54.7 ± 0.2	66.7 ± 0.2	30.3 ± 0.9	55.4 ± 1.2	53.3 ± 0.5
	Zero-shot	87.6 ± 0.8	0.0 ± 0.0	21.6 ± 0.2	26.2 ± 0.2	0.0 ± 0.0	3.1 ± 0.6	10.2 ± 0.1
	Ours	87.6 ± 0.8	**45.5 ± 1.5**	**45.2 ± 0.2**	**57.6 ± 0.1**	**12.7 ± 0.9**	**43.6 ± 2.0**	**40.9 ± 0.1**
Mistral	**16-shot**	1779.0 ± 3.4	**59.5 ± 1.5**	51.7 ± 0.7	**69.4 ± 1.6**	24.0 ± 3.7	**62.2 ± 1.8**	**53.4 ± 1.3**
	BM25	2045.0 ± 27.9	57.5 ± 1.3	51.8 ± 0.5	66.2 ± 2.5	23.8 ± 1.6	59.2 ± 1.3	51.7 ± 0.6
	Zero-shot	100.9 ± 1.7	0.1 ± 0.1	22.6 ± 0.8	16.9 ± 0.9	0.7 ± 0.5	2.7 ± 0.4	8.6 ± 0.1
	Ours	100.9 ± 1.7	**31.8 ± 0.5**	**44.4 ± 0.6**	**48.0 ± 0.7**	**17.0 ± 3.1**	**41.8 ± 1.0**	**36.6 ± 0.1**
Pythia	**16-shot**	1581.1 ± 0.3	52.7 ± 1.7	21.7 ± 1.0	13.7 ± 1.2	12.2 ± 1.5	34.1 ± 0.2	26.8 ± 0.2
	BM25	1848.6 ± 26.4	47.9 ± 1.7	20.8 ± 0.8	20.2 ± 0.8	12.3 ± 2.8	36.4 ± 1.0	27.5 ± 0.3
	Zero-shot	88.3 ± 1.5	0.2 ± 0.0	7.7 ± 0.3	3.2 ± 0.4	0.3 ± 0.2	2.0 ± 0.0	2.7 ± 0.1
	Ours	88.3 ± 1.5	**46.3 ± 0.8**	**23.2 ± 1.1**	**11.1 ± 0.8**	**15.2 ± 2.0**	**36.9 ± 3.0**	**26.5 ± 0.8**
Mamba	**16-shot**	1581.1 ± 1.3	40.4 ± 1.1	31.9 ± 1.1	**34.2 ± 1.0**	15.0 ± 2.9	40.9 ± 2.1	32.5 ± 0.3
	BM25	1848.6 ± 26.4	43.7 ± 1.9	31.4 ± 0.0	25.2 ± 1.9	14.7 ± 2.1	38.3 ± 0.8	30.7 ± 0.5
	Zero-shot	88.3 ± 1.5	0.2 ± 0.2	12.3 ± 0.2	0.6 ± 0.2	0.0 ± 0.0	5.1 ± 1.5	3.6 ± 0.3
	Ours	88.3 ± 1.5	**33.7 ± 0.6**	**26.5 ± 0.6**	**25.6 ± 1.1**	**16.3 ± 2.2**	**39.2 ± 0.5**	**28.3 ± 0.2**

Table 7: Unseen task for ICL Setting

		# Tokens	GLUE COLA	BBQ Religion	Deepmind	MMLU-Psychology	BBH-five-objects	Avg
Llama	**BM25**	1684.3 ± 3.3	33.3 ± 1.9	69.3 ± 2.4	32.0 ± 1.4	83.7 ± 0.5	35.0 ± 0.0	50.7 ± 0.1
	Zero-shot	76.5 ± 0.2	0.7 ± 0.5	5.3 ± 2.4	0.0 ± 0.0	62.7 ± 0.5	0.0 ± 0.0	13.7 ± 0.7
	Ours	76.5 ± 0.2	**1.3 ± 0.5**	**24.3 ± 3.8**	**19.3 ± 2.4**	**66.0 ± 0.0**	**5.0 ± 0.0**	**23.2 ± 1.1**
Mistral	**BM25**	1913.3 ± 20.0	27.0 ± 1.6	69.0 ± 1.4	28.3 ± 3.9	79.7 ± 1.2	26.2 ± 0.0	46.1 ± 0.6
	Zero-shot	85.5 ± 0.8	**1.7 ± 1.2**	1.0 ± 0.8	1.3 ± 1.2	35.3 ± 1.2	0.0 ± 0.0	7.9 ± 0.3
	Ours	85.5 ± 0.8	1.3 ± 1.2	**18.3 ± 2.1**	**20.0 ± 0.8**	**54.7 ± 1.2**	**10.0 ± 0.0**	**20.9 ± 1.1**
pythia	**BM25**	1747.6 ± 20.2	13.7 ± 1.9	33.7 ± 1.7	20.3 ± 2.6	18.7 ± 1.7	6.2 ± 0.0	18.5 ± 1.2
	Zero-shot	78.1 ± 0.7	**43.0 ± 2.9**	0.0 ± 0.0	0.3 ± 0.5	3.3 ± 0.5	0.0 ± 0.0	9.3 ± 0.7
	Ours	78.1 ± 0.7	37.0 ± 2.4	**14.7 ± 4.6**	**15.0 ± 0.8**	**13.0 ± 1.4**	**6.2 ± 0.0**	**17.2 ± 0.8**
Mamba	**BM25**	1747.6 ± 20.2	36.7 ± 1.2	33.3 ± 2.6	25.3 ± 4.5	25.3 ± 0.5	21.2 ± 0.0	28.4 ± 1.1
	Zero-shot	78.1 ± 0.7	**19.7 ± 2.5**	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	3.9 ± 0.5
	Ours	78.1 ± 0.7	18.0 ± 1.4	**19.3 ± 3.4**	**13.7 ± 3.8**	**13.0 ± 0.8**	**7.5 ± 0.0**	**14.3 ± 1.1**

E.2 TASK SPECIFIC PROMPT

To ensure fair comparisons in zero-shot scenarios, we prepend task-specific prompts before each test query. These task-specific prompts are manually crafted following guidelines from `lm-harness` (Gao et al., 2023b) and the `chain-of-thought-hub` [2]. The complete set of prompts used is provided in Figure 8.

Table 8: Task Specific Prompts for Various Tasks

Task	**Prompts**
bbh_date_understanding	• Infer the date from context. Finish your answer with 'X' where X is the correct letter choice. Question: {input} • Determine the date based on contextual clues. End your response with 'X', where X represents the correct option. Question: {input} • Use the given context to deduce the date. Conclude your answer with 'X', X being the right letter choice. Question: {input}
bbh_boolean_expressions	• Evaluate the result of a random Boolean expression. Question: {input} • Calculate the outcome of a given Boolean expression. Question: {input} • Determine the result of the provided Boolean logic statement. Question: {input}

Continued on next page

---

[2] https://github.com/FranxYao/chain-of-thought-hub.git

Prompts for Various Tasks (Continued)

Task	**Prompts**
bbh_date_understanding	• Infer the date from context. Finish your answer with ’X’ where X is the correct letter choice. Question: {input} • Determine the date based on contextual clues. End your response with ’X’, where X represents the correct option. Question: {input} • Use the given context to deduce the date. Conclude your answer with ’X’, X being the right letter choice. Question: {input}
bbh_boolean_expressions	• Evaluate the result of a random Boolean expression. Question: {input} • Calculate the outcome of a given Boolean expression. Question: {input} • Determine the result of the provided Boolean logic statement. Question: {input}
mmlu_pro_math	• The following are multiple choice questions (with answers) about math. Finish your answer with ’X’ where X is the correct letter choice. Question: {input} • Below are multiple-choice math questions. Conclude your response with ’X’, X being the correct option. Question: {input} • Answer these math multiple-choice questions. Answer with ’X’, where X is the right letter choice. Question: {input}

Continued on next page

Prompts for Various Tasks (Continued)

Task	**Prompts**
superglue_rte	• {input} Question: Is the hypothesis correct based on the premise? Answer True or False. • {input} Question: Based on the premise, is the hypothesis accurate? Respond with True or False. • {input} Question: Does the premise support the hypothesis? Answer True or False.
bbh_temporal_sequences	• Answer questions about which times certain events could have occurred. Finish your answer with 'X' where X is the correct letter choice. Q: {input} • Determine possible occurrence times for specific events. Answer with 'X', X being the correct option. Q: {input} • Identify when certain events could have happened. Conclude with 'X', where X is the right letter choice. Q: {input}
bbh_reasoning_about_colored_objects	• Answer extremely simple questions about the colors of objects on a surface. Finish your answer with 'X' where X is the correct letter choice. Q: {input} • Respond to basic questions about object colors on a surface. Answer with 'X', X being the correct option. Q: {input} • Address simple queries regarding the colors of items on a surface. Conclude with 'X', where X is the right letter choice. Q: {input}

Continued on next page

Prompts for Various Tasks (Continued)

Task	**Prompts**
ethics_commonsense	• {input} Question: Is this wrong? Answer Yes or No. • {input} Question: Is this ethically incorrect? Respond with Yes or No. • {input} Question: Does this violate ethical norms? Answer Yes or No.
ethics_justice	• {input} Question: Would most people believe this reasonable or unreasonable to say? Answer Yes or No. • {input} Question: Would the majority consider this statement reasonable or unreasonable? Answer Yes or No. • {input} Question: Is this statement generally perceived as reasonable or unreasonable? Respond with Yes or No.
glue_mnli	• {input} True, False or Neither? • {input} Answer selecting from: True, False, or Neither? • {input} Choose one as answer: True, False, or Neither?
glue_sst2	• {input} Question: Is this sentence positive or negative? Answer positive or negative. • {input} Question: Does this sentence convey a positive or negative sentiment? Answer positive or negative. • {input} Question: Is the tone of this sentence positive or negative? Respond with positive or negative.
glue_qnli	• {input} Question: Does this response answer the question? Answer Yes or No. • {input} Question: Is the response answer to the question? Answer Yes or No. • {input} Question: Does the given response address the question? Respond with Yes or No.

Continued on next page

Prompts for Various Tasks (Continued)

Task	Prompts
hellaswag	• The following are multiple choice questions (with answers) about commonsense NLI. Finish your answer with ’X’ where X is the correct letter choice. Question: {input} • The following are multiple-choice questions about commonsense natural language inference. Answer with ’X’, X being the correct option. Question: {input} • Answer these multiple-choice questions on commonsense language understanding. Conclude with ’X’, where X is the right letter choice. Question: {input}
crows_pairs	• {input} Question: Which sentence is more stereotypical? Answer A or B. • {input} Question: Which sentence contains more stereotypical content? Answer A or B. • {input} Question: Identify the sentence with stronger stereotypical elements. Respond with A or B.
boolq	• {input} Answer True or False. • {input} Respond with True or False. • {input} Is this statement correct? Answer True or False.

Continued on next page

Prompts for Various Tasks (Continued)

Task	Prompts
math_qa	- The following are multiple choice questions (with answers) about math word problem. Finish your answer with ’X’ where X is the correct letter choice. Question: {input} - Below are multiple-choice questions about math word problems. Answer with ’X’, X being the correct option. Question: {input} - Solve these multiple-choice math word problems. Conclude with ’X’, where X is the right letter choice. Question: {input}
superglue_wic	- {input} Question: Is the word used in the same way in the two sentences above? Answer Yes or No. - {input} Question: Is the word used similarly in both sentences above? Respond with Yes or No. - {input} Question: Does the word have the same meaning in the two given sentences? Answer Yes or No.
openbookqa	- The following are multiple choice questions (with answers) about multi-step reasoning. Finish your answer with ’X’ where X is the correct letter choice. Question: {input} - The following are multiple-choice questions testing multi-step reasoning. Answer with ’X’, X being the correct option. Question: {input} - Answer these multiple-choice questions involving multi-step logical thinking. Conclude with ’X’, where X is the right letter choice. Question: {input}

Continued on next page

Prompts for Various Tasks (Continued)

Task	**Prompts**
commonsense_qa	- The following are multiple choice questions (with answers) about commonsense knowledge reasoning. Finish your answer with 'X' where X is the correct letter choice. Question: {input} - Below are multiple-choice questions about commonsense reasoning. Answer with 'X', X being the correct option. Question: {input} - Respond to these multiple-choice questions on commonsense knowledge. Conclude with 'X', where X is the right letter choice. Question: {input}
bbq_age	- The following are multiple choice questions (with answers) about social bias on age. Finish your answer with 'X' where X is the correct letter choice. Question: {input} - The following are multiple-choice questions about age-related social bias. Answer with 'X', X being the correct option. Question: {input} - Answer these multiple-choice questions on social bias regarding age. Conclude with 'X', where X is the right letter choice. Question: {input}
arc_challenge	- The following are multiple choice questions (with answers) about science. Finish your answer with 'X' where X is the correct letter choice. Question: {input} - Below are multiple-choice science questions. Answer with 'X', X being the correct option. Question: {input} - Respond to these multiple-choice questions on scientific topics. Conclude with 'X', where X is the right letter choice. Question: {input}

Continued on next page

Prompts for Various Tasks (Continued)

Task	**Prompts**
glue_cola	• {input} Question: Does this sentence make sense? Answer Yes or No. • {input} Question: Is this sentence logically coherent? Respond with Yes or No. • {input} Question: Evaluate if this sentence is meaningful. Reply with Yes or No.
bbh_logical_deduction_five_objects	• A logical deduction task which requires deducing the order of a sequence of objects. Finish your answer with 'X' where X is the correct letter choice. Question: {input} • This challenge involves logically determining the sequence of a set of objects. Conclude your response with 'X', where X is the appropriate letter option. Question: {input} • In this logical reasoning exercise, deduce the correct order of a series of objects. End your answer with 'X', X being the right letter choice. Question: {input}
mmlu_high_school_psychology	• The following are multiple choice questions (with answers) about high school psychology. Finish your answer with 'X' where X is the correct letter choice. Question: {input} • Below are multiple-choice questions testing high school level psychology knowledge. Conclude your response with 'X', X representing the correct option. Question: {input} • These questions assess understanding of high school psychology concepts. End your answer with 'X', where X is the letter of the correct choice. Question: {input}

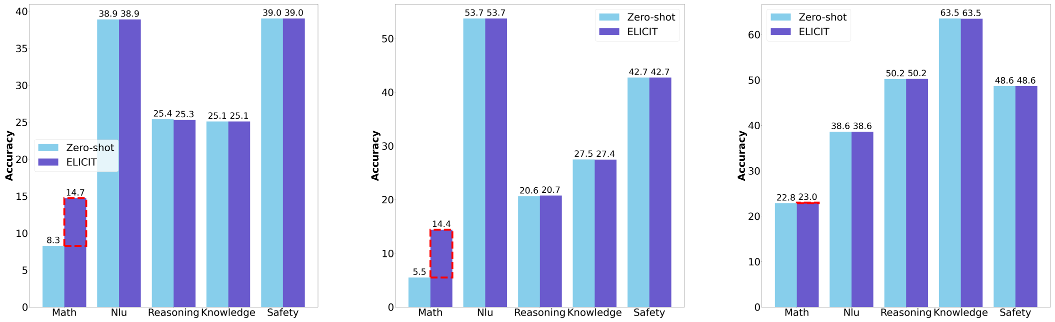
Continued on next page

Prompts for Various Tasks (Continued)

Task	Prompts
bbq_religion	• The following are multiple choice questions (with answers) about social bias on religion. Finish your answer with ’X’ where X is the correct letter choice. Question: {input} • Here are multiple-choice questions addressing social biases related to religion. Conclude your answer with ’X’, X being the correct letter option. Question: {input} • These questions explore social biases in the context of religion. End your response with ’X’, where X represents the right letter choice. Question: {input}
deepmind	• The following are multiple choice questions (with answers) about algebraic word problems. Finish your answer with ’X’ where X is the correct letter choice. Question: {input} • Below are multiple-choice questions testing algebraic word problem solving skills. Conclude your answer with ’X’, X being the correct option letter. Question: {input} • These questions assess your ability to solve algebraic word problems. End your response with ’X’, where X is the letter of the right choice. Question: {input}

F ADAPTIVE ELICITATION

As mentioned in Section 4.3, we show that when provided the library with only math-related task vectors, performance shows a significant improvement on the math domain while retaining or slightly improving in other domains for Mistral. Figure 14 illustrates similar results on other models such as Mamba, Pythia, and Llama3.

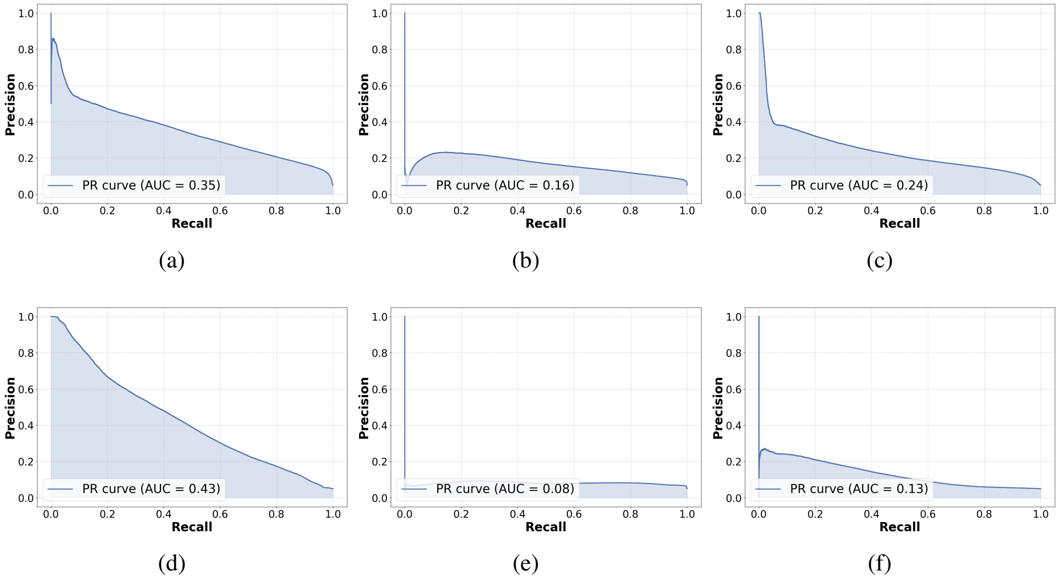


(a) Mamba (b) Pythia (c) Llama3

Figure 14: Performance on ELICITacross different domains when the li-brary only contains math-related taskvectors on Mamba, Pythia, and Llama3.

G SIMILARITY-BASED RETRIEVE METHOD

Section 5.1 demonstrates the poor precision-recall performance of similarity-based retrieval methods on the Llama3 model. Figure 15 presents the Precision-Recall curves for Mistral and Mamba under different similarity-based approaches, which also exhibit poor results. In contrast, our proposed retrieval module achieves significantly higher precision and recall across all models. This highlights the effectiveness of our method in accurately retrieving relevant task vectors to support different tasks.



(a) (b) (c)

(d) (e) (f)

Figure 15: Precision-Recall Curves for Similarity-based Prompt Retrieval Methods on Mistral and Mamba Models. Top Row: Mistral, Bottom Row: Mamba. From Left to Right: Cosine Similarity, t-SNE, and Distance-based Methods.

H ORTHOGONAL AUGMENTATION IN UNSEEN TASKS.

As shown in Section 4.5, ELICIT demonstrates its plug-and-play capability by seamlessly integrating with existing methods such as BM25 Retrieval for in-domain tasks. Table 9 showcases ELICIT combined with BM25 in unseen tasks, combined with BM25 for unseen tasks, where we can observe performance improvements across various models and tasks. This highlights the versatility and effectiveness of ELICIT in augmenting existing methods and tasks.

Table 9: Orthogonal Augmentation in Unseen Tasks.

Model		**GLUE COLA**	**BBQ Religion**	**Deepmind**	**MMLU-Psychology**	**BBH-five-objects**	**Avg**
Llama	**BM25**	83.6 ± 0.8	**54.5 ± 0.7**	29.1 ± 1.9	**81.6 ± 0.4**	29.6 ± 0.0	55.7 ± 0.2
	BM25+ELICIT	**84.7 ± 0.9**	54.4 ± 0.1	**30.0 ± 4.3**	81.2 ± 0.3	**33.5 ± 0.0**	**56.7 ± 0.6**
Mistral	**BM25**	84.6 ± 1.2	**57.6 ± 0.6**	19.5 ± 2.2	**74.7 ± 0.9**	15.1 ± 0.0	50.3 ± 0.5
	BM25+ELICIT	**85.6 ± 1.4**	54.3 ± 2.9	**26.1 ± 2.4**	74.2 ± 1.0	**23.8 ± 0.0**	**52.8 ± 0.9**
Pythia	**BM25**	54.5 ± 1.0	29.7 ± 0.7	13.3 ± 2.2	14.9 ± 0.2	5.7 ± 0.1	23.6 ± 0.3
	BM25+ELICIT	**56.1 ± 0.9**	**33.9 ± 4.0**	**13.9 ± 2.8**	**22.3 ± 1.3**	**16.0 ± 0.0**	**28.4 ± 1.2**
Mamba	**BM25**	67.4 ± 0.7	30.3 ± 2.7	19.4 ± 1.2	16.0 ± 0.4	14.5 ± 0.1	29.5 ± 0.7
	BM25+ELICIT	**69.3 ± 0.9**	**30.8 ± 1.6**	**23.0 ± 1.4**	**22.8 ± 0.2**	**16.5 ± 0.0**	**32.5 ± 0.1**

I DATASET SPLITS

We provide detailed information about our dataset curation and splitting strategies to ensure reproducibility. Our primary objective was to maintain robust evaluation capabilities while ensuring sufficient training data for ICL prompt construction. For datasets with pre-existing splits (ARC-Challenge, Ethics, GLUE, MathQA, OpenbookQA), we preserved the original partitioning. When handling datasets with only train-valid splits, we employed two approaches: for those with validation sets exceeding 350 samples (e.g., BoolQ, Hellaswag), we split the validation set into new validation and test sets at a 7:3 ratio; for those with smaller validation sets (e.g., CommonsenseQA), we divided the training set into new train and test sets (7:3). For test-only datasets, we implemented different strategies based on size: smaller datasets like BBH (250 samples) were split to ensure 128 samples for training and 80-100 samples for testing, with remaining samples allocated to validation. Larger test-only datasets (>1000 samples) such as MMLU-Pro-Math, BBQ, and Crows Pairs were split into train-valid-test sets at a 7:2:1 ratio. The same 7:2:1 split was applied to train-only datasets like SuperGLUE and DeepMind. This systematic approach ensures a minimum of 80 test samples for reliable evaluation metrics and at least 128 training samples for ICL prompt construction across all tasks.

J ANALYSIS OF ELICIT'S SELECTIVE ACTIVATION

We investigate why ELICIT can selectively activate capability in Figure 5 and the importance of this mechanism. Using a library containing only math-related task vectors on Mistral, we analyzed the number of chosen states per domain, shown in Table 10. Math-related tasks showed consistent high utilization (9.8 ± 0.1 chosen states), while other domains maintained minimal selection (approximately 0.0). This pattern confirms that ELICIT's performance improvements stem from its dynamic retrieval and selective activation of relevant capabilities.

Table 10: The average number of chosen numbers per domain per sample. The statistics come from Mistral when the capability library only contains math-related task vectors.

	in-domain				
	NLU	**Reasoning**	**Knowledge**	**Math**	**Safety**
chosen nums	0.0 ± 0.0	0.1 ± 0.0	0.0 ± 0.0	**9.8 ± 0.1**	0.0 ± 0.0
	Out-of-domain				
	GLUE COLA	**BBQ Religion**	**Deepmind**	**MMLU-Psychology**	**BBH-five-objects**
chosen nums	0.0 ± 0.0	0.0 ± 0.0	**9.9 ± 0.1**	0.0 ± 0.0	0.0 ± 0.0

We observed minor improvements in reasoning tasks, exemplified by this ARC Challenge case in Table 11. It demonstrates our pipeline's ability to selectively activate relevant capabilities based solely on query and handle unseen tasks flexibly, without requiring explicit task information.

Experiments forcing the application of top task vectors to all queries (Table 12), showed significant performance degradation in NLU and knowledge tasks, highlighting the importance of selective activation.

Table 11: A successful case from Arc-Challenge when capability library only contains math-related task vectors on Mistral.

input	Below are multiple-choice science questions.Answer with 'X', X being the correct option.\n\nQuestion: An unbalanced equation for the reaction of methane gas (CH_{4}) with oxygen is shown below. CH_{4} + \\Box O_{2} ->2CO_{2} + 4H_{2}O How many molecules of oxygen gas (O_{2}) are needed to properly balance this equation?\nOptions:\nA. 1\nB. 2\nC. 3 \nD. 4\nAnswer:
chosen task vectors	10 task vectors from MathQA
Original Output	B
ELICIT Output	**D (correct)**

Table 12: The results of forcibly applying the top task vectors for each query. The experiments were conducted on Mistral. Domains with degraded performance are marked in bold.

	nlu	reasoning	knowledge	math	safety
Zero-shot	28.8	27.4	58.8	4.0	42.2
ELICIT	**15.7**	31.4	**47.8**	18.3	53.1

These experimental results demonstrate that ELICIT's performance improvement stems from its selective activation mechanism and the importance of selectively using only task-relevant vectors to dynamically activate capabilities.

We analyzed the usage frequency of task vectors in the capability library, which contains 20 distinct task vector types. The analysis was performed on Pythia-6.9B while evaluating 25 tasks in total: 20 in-domain tasks and 5 out-of-domain tasks. Our findings confirmed that all 20 task vector types in the library were utilized during the evaluation.

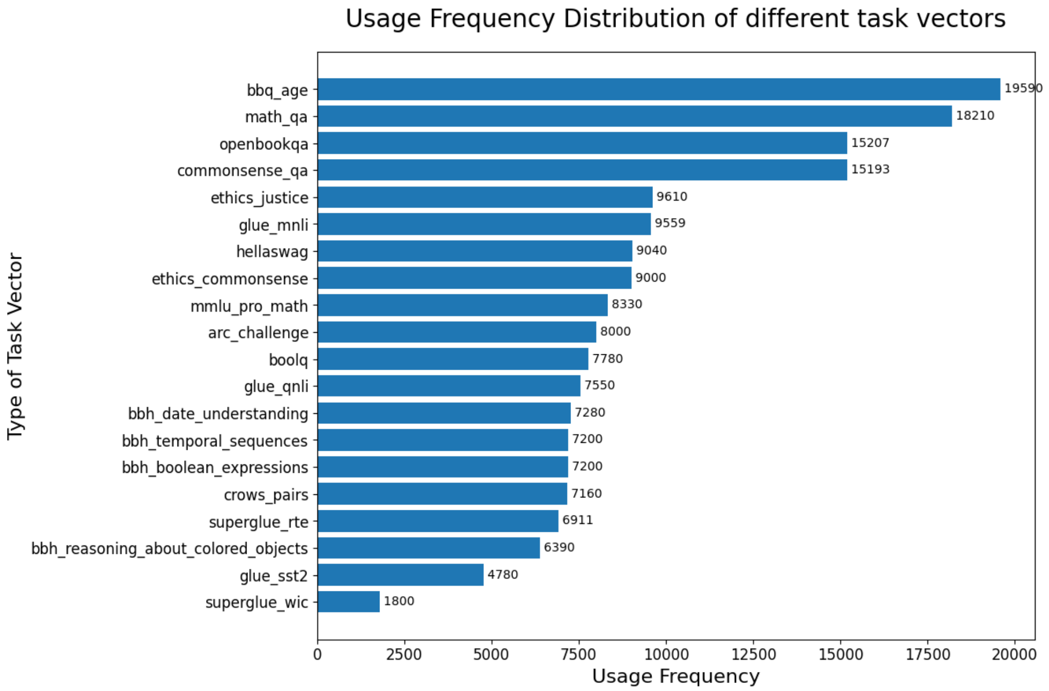


Figure 16: Usage Frequency Distribution of Different Types of Task Vectors Across all In-Domain and OOD (Out-of-Domain) Samples. The results is based on Pythia-6.9B.

K More models and tasks on ELICIT

Beyond our primary experiments, we evaluate the scalability and generalizability of ELICIT across larger language models and more challenging tasks. As shown in Table 13, ELICIT maintains its performance advantages when applied to more base models. Furthermore, Table 14 demonstrates that ELICIT achieves consistent improvements across a diverse set of complex tasks, validating its effectiveness and versatility.

We further explore the applicability of ELICIT to instruction-tuned models, with preliminary results shown in Table 15. While this initial experiment suggest the potential compatibility of ELICIT with instruction-tuned models, several challenges remain. Instruction-tuned models exhibit heightened sensitivity to prompts and instructions (Sun et al., 2023; Gao et al., 2023a), necessitating more investigation and analysis. Key challenges include identifying effective task vectors for in-context learning (ICL) and developing robust methods for zero-shot performance evaluation. We leave the comprehensive adaptation of ELICIT for instruction-tuned models as promising future work.

Table 13: Performance of ELICIT on more different models. ELICIT are effective for larger models.

		Length	**nlu**	**reasoning**	**knowledge**	**math**	**safety**	**avg**
Pythia-6.9B	**16shots**	1598.4 ± 2.0	27.3 ± 0.3	27.3 ± 0.3	27.3 ± 0.3	27.3 ± 0.3	27.3 ± 0.3	27.3 ± 0.3
	bm25	2184.2 ± 29.7	47.1 ± 0.2	24.6 ± 0.5	19.8 ± 0.7	12.1 ± 1.5	28.5 ± 0.9	26.5 ± 0.4
	zs	109.8 ± 1.5	37.6 ± 0.4	16.1 ± 0.5	17.4 ± 0.6	5.9 ± 0.7	31.7 ± 0.5	21.8 ± 0.1
	ELICIT	109.8 ± 1.5	**38.7 ± 1.4**	**28.1 ± 0.5**	**27.9 ± 1.0**	**18.2 ± 2.6**	**47.8 ± 2.0**	**32.2 ± 0.7**
Pythia-12B	**16shots**	1598.2 ± 1.4	47.8 ± 1.2	15.8 ± 0.4	23.7 ± 0.3	14.3 ± 1.5	35.0 ± 1.9	27.3 ± 0.3
	bm25	2184.2 ± 29.7	42.6 ± 1.4	20.3 ± 0.2	21.0 ± 0.4	14.0 ± 1.5	30.1 ± 0.8	25.6 ± 0.4
	zs	109.8 ± 1.5	34.7 ± 0.6	20.7 ± 0.2	18.1 ± 0.6	7.9 ± 1.7	34.6 ± 0.6	23.2 ± 0.2
	ELICIT	109.8 ± 1.5	**38.5 ± 0.5**	**29.7 ± 0.7**	**29.8 ± 0.6**	**17.5 ± 2.1**	**46.8 ± 0.2**	**32.5 ± 0.5**
Llama3-70B	**bm25**	1823.1	42.8	78.1	80.9	49.2	71.5	64.5
	16shot	1411.5	40.1	76.3	83.3	51.4	70.9	64.4
	zs	101.1	50.9	66.8	59.7	37.6	44.2	51.8
	ELICIT	101.1	**55.9**	**80.5**	**84.6**	**52.4**	**67.4**	**68.2**

Table 14: The results of ELICIT on GSM8K and MMLU-Professional-Law on Llama3-8B. GSM8K is as in-domain task and MMLU-Profeesional-Law is out-of-domain.

	GSM8K	**MMLU-Professional-Law**
zs	30.44	31.67
ELICIT	**32.44**	**41.11**
16shot	42.87	30.78
16shot+ELCIT	**43.22**	**41.89**

Table 15: The preliminary experiment of ELICIT on Llama3-8B-Instruct.

	nlu	**reasoning**	**knowledge**	**safety**	**avg**
zs	45.0	4.9	31.9	42.5	31.1
ELICIT	**52.7**	**36.2**	**70.9**	**49.0**	**52.2**

L Diversity-Optimizaed Capability Library

We conduct an experiment on maximizing the diversity of prompts in the given capability library. Instead of random demonstration selection, we construct a new capability library of diversity-optimized prompts as described in Su et al. (2022).

Spefically, we used Sentence-BERT to generate embeddings by averaging the resulting vectors over the words in each text input. For each task, after computing embeddings for all training data, we implemented an iterative approach to find diverse examples to construct ICL prompts. Starting with a random example, we selected examples that maximized the distance from previously chosen examples in each iteration. We then conducted a new capability library using these more diverse ICL prompts.

As shown in Table 16, the diversity-optimized prompts yielded mixed results. Compared to the original ELICIT, while performance improved in reasoning (+1.1%), math (+0.5%) and NLU tasks (+4.5%), there was a decline in Knowledge (-5.9%) and Safety (-2.3%) ability.

This result suggests the potential for future work to improve our pipeline by enhancing the quality of task vectors through better demonstration selection methods.

Table 16: The comparison of ELICIT using different capability library based on different ICL prompts. The experiments are conducted on Llama3-8B.

	NLU	**Reasoning**	**Knowledge**	**Math**	**Safety**	**Avg.**
Zero-shot	32.2 ± 1.2	32.9 ± 0.2	42.5 ± 1.2	14.0 ± 1.0	35.5 ± 1.2	31.4 ± 0.7
ELICIT	38.1 ± 0.9	46.1 ± 0.3	**60.7 ± 1.2**	19.4 ± 1.1	**49.4 ± 2.1**	**42.7 ± 0.8**
ELICIT (diversity)	**42.6 ± 0.3**	**47.2 ± 0.1**	54.8 ± 1.5	**19.9 ± 0.8**	47.1 ± 2.6	42.3 ± 0.9

M MULTI-LAYER INTERVENTION

While our primary analysis focuses on single-layer intervention, we also conduct preliminary experiments on multi-layer intervention, with the intervention strength $\alpha = 2$ distributed evenly across layers. We evaluated four settings: (1) the zero-shot baseline, (2) intervention on three consecutive layers (centered on the previously identified optimal layer), (3) intervention across all layers, and (4) our original single-layer implementation. Results from Llama3-8B (Table 17) reveal an intriguing pattern: distributing intervention across multiple layers tends to yield better performance. This observation opens promising directions for future research into the mechanisms and benefits of multi-layer interventions.

Table 17: Comparison of multiple intervention layers on ELICIT. The experiments are conducted on Llama3-8B.

	nlu	**reasoning**	**knowledge**	**math**	**safety**	**avg**
zs	32.4	31.8	42.8	15.4	36.6	31.8
ELICIT (1 layer)	38.3	46.9	60.7	20.6	51.1	43.5
ELICIT (3 layers)	38.2	**47.1**	61	21.6	51.6	43.9
ELICIT (all layers)	**40.9**	46.3	**61.4**	**21.7**	**52.4**	**44.5**

N ANALYSIS OF COMPUTATIONAL EFFICIENCY WITH RETRIEVAL MODULE

To demonstrate the effciency of ELICIT, We conducted a detailed analysis of ELICIT's computational efficiency using the Pythia-6.9B model, measuring the average processing time per sample across different pipeline stages. The results are shown in Table 18. Our quantitative results demonstrate that the integration of the retrieval module maintains the method's efficiency. Specifically, the retrieval module adds only 0.105 seconds of computational overhead per sample. The total inference time, including retrieval operations, remains efficient at 0.172 seconds per sample. ELICIT demonstrates superior efficiency compared to baseline approaches, processing samples 2-3 times faster than both 16-shot inference and BM25-based inference methods. These results validate that ELICIT's performance improvements do not come at the cost of computational efficiency, even with the addition of retrieval module.

Table 18: The running time of different stages per sample across different domains.

	zs inference time	**ELCIT inference time**	**retrieve time**	**bm25 inference time**	**16shot inference time**
nlu	0.063	0.064	0.097	0.302	0.181
reasoning	0.065	0.066	0.104	0.349	0.315
knowledge	0.066	0.069	0.108	0.517	0.371
math	0.065	0.067	0.111	0.351	0.352
safety	0.067	0.069	0.104	0.611	0.366
avg	0.065	0.067	0.105	0.426	0.317