ALPS: Attention Localization and Pruning Strategy for Efficient Alignment of Large Language Models

Hao Chen[♠], **Haoze Li**[♠], **Zhiqing Xiao**[♠], **Lirong Gao**[♠], **Qi Zhang**[♠]
Xiaomeng Hu[♠], **Ningtao Wang**[♣], **Xing Fu**[♣], **Junbo Zhao**[♠]
[♠]Zhejiang University [♣]Ant Group
{h.c.chen, j.zhao}@zju.edu.cn

Abstract

Aligning general-purpose large language models (LLMs) to downstream tasks often incurs significant training adjustment costs. Prior research has explored various avenues to enhance alignment efficiency, primarily through minimal-data training or data-driven activations to identify key attention heads. However, these approaches inherently introduce data dependency, which hinders generalization and reusability. To address this issue and enhance model alignment efficiency, we propose the ***A**ttention **L**ocalization and **P**runing **S**trategy (**ALPS**)*, an efficient algorithm that localizes the most task-sensitive attention heads and prunes by restricting attention training updates to these heads, thereby reducing alignment costs. Experimental results demonstrate that our method activates only **10%** of attention parameters during fine-tuning while achieving a **2%** performance improvement over baselines on three tasks. Moreover, the identified task-specific heads are transferable across datasets and mitigate knowledge forgetting. Our work and findings provide a novel perspective on efficient LLM alignment. The code is available at https://github.com/VoiceBeer/ALPS.

1 Introduction

Pre-trained Large Language Models (LLMs) have demonstrated impressive performance across various tasks (Achiam et al., 2023; OpenAI, 2025; Guo et al., 2025). A way to harness the potential of pre-trained foundation models on downstream tasks is task alignment, which mainly incorporates tailored task knowledge. Recently, task alignment has garnered considerable attention in both industry and academia with a diverse range of specialized models having been developed, spanning domains such as mathematics (Shao et al., 2024; Yang et al., 2024), code generation (Guo et al., 2024; Roziere et al., 2023), medical diagnostics (Singhal et al., 2023, 2025) and protein structure prediction (Schmirler et al., 2024). However, despite the effectiveness of task alignment, it still poses a significant resource-intensive challenge. The alignment process demands extensive efforts to construct task-specific instruction datasets and training adjustments, and further, its computational demands exacerbate the costs (Zhao et al., 2023) These challenges highlight the pressing need for more efficient alignment strategies that not only minimize resource consumption costs but also maintain enhance model performance (Wan et al., 2023).

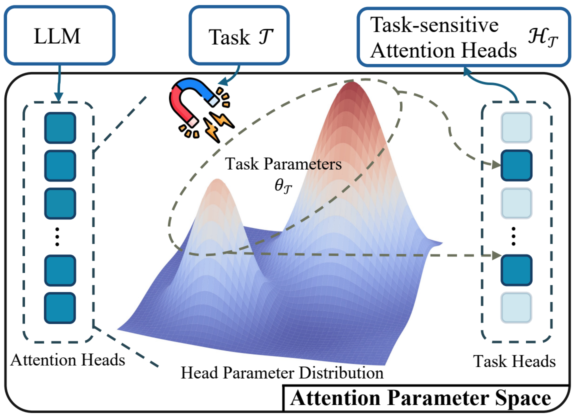

LLM
Task $\mathcal{T}$
Task-sensitive Attention Heads $\mathcal{H}_\mathcal{T}$
Task Parameters $\theta_\mathcal{T}$
Attention Heads
Head Parameter Distribution
Task Heads
Attention Parameter Space

Figure 1: Process of localizing task-sensitive attention heads. The left side represents the attention heads of LLM, and the center illustrates the distribution of head parameters, where task parameters $\theta_\mathcal{T}$ capture task information and lead to task-sensitive heads $\mathcal{H}_\mathcal{T}$.

To improve this efficiency problem of model alignment, recent LLM studies are interested in attention mechanisms and attempt to localize task-specific attention heads (Clark, 2019; Michel et al., 2019; Zhou et al., 2024b; Wu et al., 2024; Tang et al., 2024). However, these methods mainly rely on task-specific data to activate model parameters for attention head localization, inadvertently introducing data dependency to identified heads, which not only hinders generalization but also complicates reusability. Moreover, these approaches over-

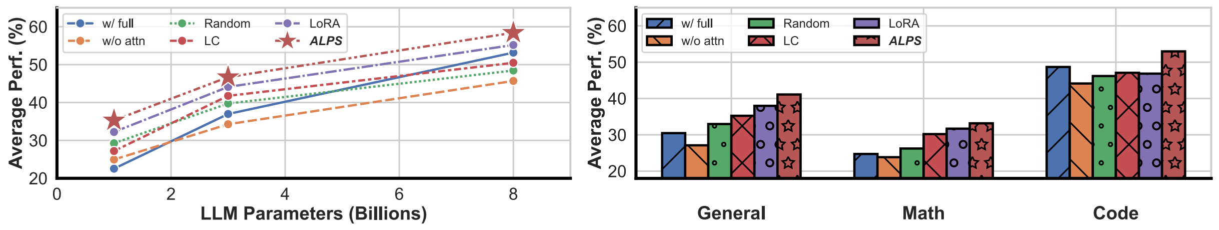


Figure 2: Preliminary results comparing **ALPS** against baselines across different LLM scales (**Left**), and downstream tasks (**Right**). **ALPS** consistently outperforms other methods, demonstrating its efficiency in diverse settings.

look the intrinsic functionality of the model weight parameters, which inherently encode task-relevant information (Zhong et al., 2022; Tam et al., 2024), and can be leveraged to identify task-sensitive attention heads without relying on activation data.

In this work, to enhance alignment efficiency, utilize weight parameters, and introduce reusability of identified attention heads, we propose the ***A**ttention **L**ocalization and **P**runing **S**trategy (**ALPS**)*, a heuristic-guided algorithm that localizes task-sensitive attention heads and prunes by restricting attention training updates to these heads only. This algorithm draws inspiration from prior work leveraging model weight parameters to localize task-sensitive attention heads (Voita et al., 2019; Zheng et al., 2024; Shi et al., 2024), as shown in Fig 1. Unlike previous work utilizing whole model parameters, ALPS focuses on the distribution of each attention head. Specifically, given a base model and its task-related model, we extract and normalize the distribution of each head, then quantify the distributional shift between corresponding heads using our proposed parameter alignment distribution score. The heads with the highest scores are then selected as task-sensitive heads, while the remaining heads are frozen during fine-tuning. This yields a pruned, task-specialized model with enhanced efficiency. Empirical results demonstrate that our method updates only **10%** of attention parameters during fine-tuning while achieving a **2%** performance gain over baselines across general, math, and code tasks, as shown in Figure 2. Further analysis reveals that the selected task-sensitive heads exhibit transferability and reusability, enhancing performance across datasets within the same task, reducing computational overhead, and avoiding data dependency. Moreover, the induced sparsity in attention heads helps mitigate knowledge forgetting in the model alignment.

In summary, our contributions are as follows:

- We offer a novel perspective on efficient LLM alignment, pioneering an approach using model weight parameters to identify task-sensitive attention heads.
- We propose **ALPS**, a novel algorithm that identifies task-sensitive attention heads by measuring shifts in their weight distributions and then restricts training to these heads.
- Empirical results demonstrate that our method improves alignment efficiency and task performance while introducing sparsity and head transferability.

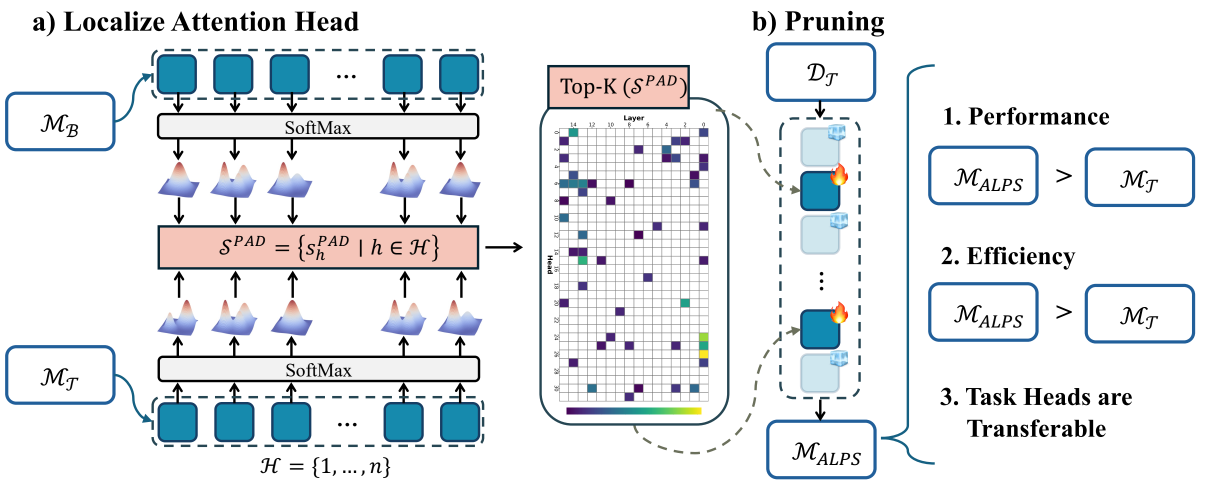
2 Related Work

2.1 Efficient LLMs

To mitigate the high costs of aligning foundation LLMs to downstream tasks, researchers have explored various strategies for efficient alignment (Wan et al., 2023). For example, DeepSeek-R1 (Guo et al., 2025) relies solely on reinforcement training to reduce overhead. While data-focused approaches either aim to minimize the need for task-specific data (Zhou et al., 2024a; Chen et al., 2023a,b) or augment downstream datasets (Li et al., 2023b; Wang et al., 2022; Xu et al., 2023), parameter-focused approaches improve efficiency by avoiding full-model updates (Hu et al., 2021), compressing the model (Jacob et al., 2018; Hinton, 2015), and streamlining key-value storage (Ainslie et al., 2023; Liu et al., 2024). Our work leverages the attention mechanism to localize and prune task-sensitive attention heads.

2.2 Attention Localization and Pruning

Recent studies have investigated attention mechanisms at various granularities to improve the efficiency and interpretability of LLM, including neurons, layers, and heads (Geiger et al., 2021; Gurnee et al., 2023; Zou et al., 2023; Zheng et al., 2024; Zhao et al., 2024). Notably, many works have concentrated on attention heads for downstream



a) Localize Attention Head
b) Pruning
$\mathcal{M}_{\mathcal{B}}$
$\mathcal{M}_{\mathcal{T}}$
SoftMax
SoftMax
$\mathcal{S}^{PAD} = \{s_h^{PAD} \mid h \in \mathcal{H}\}$
$\mathcal{H} = \{1, \dots, n\}$
Top-K ($\mathcal{S}^{PAD}$)
Layer
Head
$\mathcal{D}_{\mathcal{T}}$
$\mathcal{M}_{ALPS}$
1. Performance
$\mathcal{M}_{ALPS}$ > $\mathcal{M}_{\mathcal{T}}$
2. Efficiency
$\mathcal{M}_{ALPS}$ > $\mathcal{M}_{\mathcal{T}}$
3. Task Heads are Transferable

Figure 3: Overview of **ALPS** framework. **a)** Given a base model $\mathcal{M}_{\mathcal{B}}$ and a task fine-tuned model $\mathcal{M}_{\mathcal{T}}$, we extract their attention heads, compute weight distributions using softmax, and calculate their **PAD** score s_h^{PAD}. The Top-K attention heads are then selected based on $\mathcal{S}^{PAD}$, while **b)** pruning the remaining heads by freezing their gradient updates during fine-tuning. The resulting model $\mathcal{M}_{ALPS}$ outperforms $\mathcal{M}_{\mathcal{T}}$ in both performance and efficiency, with transferable task-relevant heads that enhance alignment across datasets within the same task.

tasks, including general linguistic and factuality capability (Clark, 2019; Wu et al., 2024; Zheng et al., 2024), model safety (Chen et al., 2024; Zhou et al., 2024b), KV-cache compression (Ainslie et al., 2023; Fu et al., 2024), etc. Among them, Michel et al. (2019) demonstrated that most attention heads can be dropped without compromising performance. While most methods rely on the activation of model parameters by feeding task-specific data, Voita et al. (2019) analyzed the weight matrices of attention in models, revealing that specialized heads contribute to tasks, while the rest can be pruned. Similarly, Shi et al. (2024) and He et al. (2024) also observed that pruning certain attention layers enhances alignment efficiency, indicating redundancy within the attention mechanism. In contrast, our approach directly localizes task-relevant attention heads from weight matrices (Voita et al., 2019; Shi et al., 2024), thereby reducing data dependency and improving generalizability and efficiency in downstream alignment.

3 Methodology

In this section, we first provide a preliminary overview of the attention mechanism with task parameters and then present **ALPS** in detail. Fig 3 illustrates the overall framework.

3.1 Preliminary

In this work, we adopt the Llama-3 series of models, which employ grouped-query attention (GQA) (Ainslie et al., 2023) to reduce KV cache overhead. Formally, given an input sequence matrix $\boldsymbol{X} \in \mathbb{R}^{t \times d}$, where t represents the sequence length, GQA divides the n attention heads into g groups, making all heads within the same group share a single pair of KV projections. Each head $h \in \{1, \dots, n\}$ learns projections $\{\boldsymbol{W}_q^h, \boldsymbol{W}_k^{\lceil hg/n \rceil}, \boldsymbol{W}_v^{\lceil hg/n \rceil}\}$, where $\lceil \cdot \rceil$ is the ceiling function, and mapping $\boldsymbol{X}$ into queries $\boldsymbol{Q}^h$, keys $\boldsymbol{K}^{\lceil hg/n \rceil}$, and values $\boldsymbol{V}^{\lceil hg/n \rceil}$:

$$\begin{aligned} \boldsymbol{Q}^h &= \boldsymbol{X}\boldsymbol{W}_q^h \in \mathbb{R}^{t \times d_k}, \\ \boldsymbol{K}^{\lceil hg/n \rceil} &= \boldsymbol{X}\boldsymbol{W}_k^{\lceil hg/n \rceil} \in \mathbb{R}^{t \times d_k}, \\ \boldsymbol{V}^{\lceil hg/n \rceil} &= \boldsymbol{X}\boldsymbol{W}_v^{\lceil hg/n \rceil} \in \mathbb{R}^{t \times d_v}, \end{aligned} \tag{1}$$

where a common choice sets $d_k = d_v = d_{model}/h$, and if $g = h$, this GQA setting reduces to standard multi-head attention. In the Llama-3 collection, a typical choice sets $g = 8$ for KV sharing. The output of head s is then computed as:

$$\begin{aligned} \boldsymbol{O}^h = \text{Softmax}\left(\frac{\boldsymbol{Q}^h (\boldsymbol{K}^{\lceil hg/n \rceil})^\top}{\sqrt{d_k}}\right) \boldsymbol{V}^{\lceil hg/n \rceil}, \\ \boldsymbol{O}^h \in \mathbb{R}^{t \times d_v}. \end{aligned} \tag{2}$$

3.2 Task Head Parameters

Downstream task-related head parameters are critical components in LLMs that directly influence performance on specific tasks. Inspired by mechanistic interpretability studies (Voita et al., 2019; Zhao et al., 2024; Lindner et al., 2023), we define these task head parameters as those whose ablation results in a significant degradation of task-specific performance. Formally, given a downstream task $\mathcal{T}$ with evaluation metric p, we quantify the impact of ablating a head parameter θ^h by the relative performance drop:

$$\Delta p(\theta^h) = p\left(\theta^{\mathcal{M}}; \mathcal{T}\right) - p\left(\theta^{\mathcal{M}} \setminus \theta^h; \mathcal{T}\right), \quad (3)$$

where $\theta^{\mathcal{M}}$ denotes the original model parameters, and $\theta^{\mathcal{M}} \setminus \theta^h$ represents the model after ablating head parameters θ^h. The downstream task-related head parameters are then identified as the Top-K heads that maximize this performance drop:

$$\Theta_{\mathcal{T},K} = \text{Top-K}\left\{\theta^{\mathcal{T}} : \underset{\theta^h \in \theta^{\mathcal{M}}}{\text{argmax}}\, \Delta p(\theta^h)\right\}. \quad (4)$$

The ablation of $\Theta_{\mathcal{T},K}$ leads to a measurable decline in task performance, underscoring their importance.

3.3 Attention Localization and Pruning Strategy (ALPS)

To localize task-sensitive attention heads that contribute to downstream task alignment, we introduce a heuristic search algorithm named *Attention Localization and **P**runing Strategy (**ALPS**)*. The algorithm is divided into two stages: **Localizing** task-sensitive attention heads and **Pruning** non-critical heads in task alignment.

Localizing. To identify task-sensitive attention heads, we introduce the *parameter alignment distribution score* s^{PAD}, which is based on the Wasserstein-1 distance (Vaserstein, 1969) between the weight matrices of a fine-tuned task model and its base counterpart, quantifying the variation in attention head parameters across different tasks.

Given a pre-trained model $\mathcal{M}_{\mathcal{B}}$ and its task fine-tuned version $\mathcal{M}_{\mathcal{T}}$, we analyze the static projection matrices of each attention head h. For head h with parameters $\{\boldsymbol{W}_q^h, \boldsymbol{W}_k^{\lceil hg/n \rceil}, \boldsymbol{W}_v^{\lceil hg/n \rceil}\}$, the projection matrix is computed as:

$$\boldsymbol{W}_o^h = \boldsymbol{W}_q^h \boldsymbol{W}_k^{\lceil hg/n \rceil \top} \boldsymbol{W}_v^{\lceil hg/n \rceil} \in \mathbb{R}^{d \times d}, \quad (5)$$

which captures the transformation behavior of one head(Kobayashi et al., 2020).

To measure task-induced parameter shifts, we first convert $\boldsymbol{W}_o^{(s)}$ into a probability distribution via tempered softmax:

$$\boldsymbol{P}^h = \text{Softmax}\left(\frac{\boldsymbol{W}_o^h}{\tau}\right), \quad \tau > 0, \quad (6)$$

Algorithm 1 Attention Localization and Pruning Strategy (ALPS)

```
Input: $\mathcal{M}_{\mathcal{B}}, \mathcal{M}_{\mathcal{T}}, \mathcal{D}_{\mathcal{T}}$, retention ratio $r \in (0, 1]$
Output: $\mathcal{H}_r, \mathcal{M}_{ALPS}$
 1: Let $\mathcal{H} \leftarrow \{1, 2, \ldots, n\}$ ▷ Set of all attention heads
 2: for all $h \in \mathcal{H}$ do
 3:     $\boldsymbol{P}_{\mathcal{B}}^h \leftarrow \text{Softmax}(\boldsymbol{W}_{o,B}^h)$
 4:     $\boldsymbol{P}_{\mathcal{T}}^h \leftarrow \text{Softmax}(\boldsymbol{W}_{o,T}^h)$
 5:     Compute $s_h^{PAD} \leftarrow \mathcal{W}_1\left(\boldsymbol{P}_{\mathcal{B}}^h, \boldsymbol{P}_{\mathcal{T}}^h\right)$
 6: end for
 7: $K \leftarrow \lceil r \cdot n \rceil$
 8: $\mathcal{S}^{PAD} \leftarrow \{s_h^{PAD} \mid h \in \mathcal{H}\}$
 9: $\mathcal{H}_{\mathcal{T}} \leftarrow \text{Top-K}(\mathcal{S}^{PAD})$ ▷ the set of heads with Top-$K$ $s_h^{PAD}$
10: for all $\hat{h} \in \mathcal{H} \setminus \mathcal{H}_{\mathcal{T}}$ do
11:     $\nabla_{\theta^{\hat{h}}} \mathcal{L} = 0$ ▷ Ablate non-critical heads during fine-tuning
12: end for
13: Fine-tune model on $\mathcal{D}_{\mathcal{T}}$ to obtain $\mathcal{M}_{\text{ALPS}}$
14: return $\mathcal{H}_r, \mathcal{M}_{\text{ALPS}}$
```

where τ controls distribution granularity, and we set τ to 1 for simplicity. The Wasserstein-1 distance between base and task distributions then quantifies head sensitivity:

$$\begin{aligned} s_h^{PAD} &= W_1\left(\boldsymbol{P}_{\mathcal{B}}^h, \boldsymbol{P}_{\mathcal{T}}^h\right). \\ &= \inf_{\gamma \in \Gamma(\boldsymbol{P}_{\mathcal{B}}^h, \boldsymbol{P}_{\mathcal{T}}^h)} \mathbb{E}_{(x,y)\sim\gamma}\left[\|x - y\|\right], \end{aligned} \quad (7)$$

where Γ denotes all joint distributions with marginals $\boldsymbol{P}_{\mathcal{B}}^h$ and $\boldsymbol{P}_{\mathcal{T}}^h$, and we compute s_h^{PAD} for each attention head. Results in Section 4.3 show that heads with higher s_h^{PAD} consistently achieve stronger task alignment and thus better performance, confirming that the magnitude of parameter variation reflects their importance for downstream tasks.

Pruning. After obtaining a set of $\mathcal{S}^{PAD} = \{s_h^{PAD} \mid h \in \mathcal{H}\}$, where $\mathcal{H} = \{1, \ldots, n\}$ denote all attention heads in the model, we select a group of task-sensitive heads $\mathcal{H}_{\mathcal{T}}$ with $\text{Top-K}(\mathcal{S}^{PAD})$ to retain. During task fine-tuning, the remaining heads will be pruned by freezing gradients of these heads, which masks their parameter updates to 0. Specifically, for all attention heads, only parameters associated with $\mathcal{H}_{\mathcal{T}}$ are updated:

$$\nabla_{\theta^h} \mathcal{L} = \begin{cases} \nabla_{\theta^h} \mathcal{L} & \text{if } h \in \mathcal{H}_S \\ 0 & \text{otherwise} \end{cases}, \quad (8)$$

Model	Method	Attn%	General		Math		Code				Avg.
			IFEval	GPQA	GSM8K	MATH	HEval	HEval+	MBPP	MBPP+	
Llama-3.2-1B	w/ full	100%	34.53	18.75	17.82	4.60	45.12	39.08	28.63	23.78	26.54
	w/o attn	0%	35.85	19.64	23.35	5.20	41.51	35.28	**29.91**	**24.88**	26.95 ↑0.41
	Random	10%	33.29	25.89	22.82	5.02	41.51	37.22	28.12	24.13	27.25 ↑0.71
	LC	10%	32.15	27.88	21.73	4.81	42.05	38.07	27.03	23.88	27.20 ↑0.66
	LoRA	~10%	**36.75**	26.14	23.67	5.92	42.34	37.99	28.92	24.10	28.22 ↑1.68
	ALPS	10%	36.69	**26.16**	**23.74**	**7.28**	**46.89**	**40.22**	29.13	24.21	**29.29 ↑2.75**
Llama-3.2-3B	w/ full	100%	42.81	22.32	31.77	16.68	56.18	50.53	50.00	41.50	38.97
	w/o attn	0%	40.29	24.33	33.21	13.76	55.52	50.62	48.71	39.72	38.27 ↓0.70
	Random	10%	42.21	20.54	35.03	14.56	56.18	49.43	54.57	45.28	39.73 ↑0.76
	LC	10%	41.75	21.86	**36.42**	13.94	55.82	50.67	55.13	42.50	39.76 ↑0.79
	LoRA	~10%	42.90	21.22	34.20	15.36	56.85	50.09	**55.27**	**45.03**	40.12 ↑1.15
	ALPS	10%	**44.96**	**24.33**	34.27	**17.12**	**58.28**	**51.67**	52.21	42.78	**40.70 ↑1.73**
Llama-3.1-8B	w/ full	100%	51.20	25.22	63.00	20.52	69.50	64.28	63.21	52.38	51.16
	w/o attn	0%	40.41	26.16	64.06	15.48	70.72	**65.88**	63.27	52.55	49.82 ↓1.34
	Random	10%	49.52	26.34	62.40	23.52	70.12	63.39	59.00	49.21	50.44 ↓0.72
	LC	10%	48.67	27.10	63.83	22.47	70.68	63.05	58.28	49.79	50.48 ↓0.68
	LoRA	~10%	50.02	26.82	62.92	**24.07**	72.50	63.89	59.55	49.69	51.18 ↑0.02
	ALPS	10%	**51.33**	**27.29**	**64.13**	22.48	**72.68**	65.03	**63.67**	**52.88**	**52.41 ↑1.25**

Table 1: Performance of our method compared to other baselines with Llama-3.2-1B, Llama-3.2-3B, and Llama-3.1-8B in general, math, and code tasks. The best results are highlighted in **bold**. ↓↑ indicates change relative to the w/ full baseline. Note that each task uses a separately fine-tuned model trained on its respective dataset. **ALPS** achieves better performance while updating only **10%** of attention parameters, highlighting its efficiency.

where $\theta^h = \{\boldsymbol{W}_q^{(h)}, \boldsymbol{W}_k^{(h)}, \boldsymbol{W}_v^{(h)}\}$. This gradient masking ensures that non-sensitive heads $\mathcal{H} \setminus \mathcal{H}_\mathcal{T}$ retain pre-trained knowledge without interference, and the loss objective turns to:

$$\mathcal{L} = \mathbb{E}_{(x,y)\sim\mathcal{D}_\mathcal{T}} \left[-\log p \left(y \mid x; \{\theta^h\}_{h\in\mathcal{H}_\mathcal{T}} \right) \right], \tag{9}$$

where $\{\theta_s\}$ denotes the parameters of heads in $\mathcal{H}_S$, and $\mathcal{D}_\mathcal{T}$ represents the downstream task data. This approach reduces optimization redundancy by focusing updates on heads critical to $\mathcal{T}$.

The overall procedure is detailed in Algorithm 1, and the final output of the algorithm is the set of sensitive attention heads $\mathcal{H}_\mathcal{T}$ with modified fine-tuned model $\mathcal{M}_{ALPS}$.

4 Experiments

4.1 Setup

Datasets. We evaluate our approach on three representative datasets covering general, math, and code tasks: UltraChat (Ding et al., 2023), MathInstruct (Yue et al., 2023), and MagiCoder (Wei et al., 2023). To assess the transferability of our method, we further conduct experiments on Alpaca (Taori et al., 2023), Camel-math (Li et al., 2023a), and CodeAlpaca (Chaudhary, 2023). Details are listed in Appendix A.

Models. We conduct our method on the Llama-3 series models (Llama-3.2-1B, Llama-3.2-3B, and Llama-3.18B), all pre-trained with GQA for improved efficiency. Each model shares key and value projections across 8 heads (g=8), while queries remain head-specific. Thus, when computing s^{PAD}, each key-value pair corresponds to 8 query heads.

Evaluation. We use lm-eval (Gao et al., 2024; meta llama, 2023) to evaluate general capabilities in instruction following, reasoning, and dialogue, as well as mathematical abilities. For code generation, we employ EvalPlus (Liu et al., 2023). All evaluations follow the official implementations, with details on benchmarks and metrics provided in Appendix A.

Baseline. To evaluate the effectiveness of **ALPS** , we compare it against the following baselines. **w/ full**: standard full supervised fine-tuning where all attention parameters are updated. **w/o attn**: freeze all attention heads by preventing updates to the QKV weight matrices. **Random**: selects a fixed proportion of random attention heads for training. **LC**: layer consistency selection, extending random by ensuring a uniform distribution of selected heads across layers, maintaining fairness in head allocation. **LoRA** (Hu et al., 2021): a

Model	Method	Attn%	General		Math		Code				Avg.
			IFEval	GPQA	GSM8K	MATH	HEval	HEval+	MBPP	MBPP+	
Llama-3.2-1B	C.S.	10%	28.12	17.97	18.18	4.15	28.22	23.97	18.89	13.67	19.15
	Eu.	10%	34.05	24.68	21.91	6.89	44.55	39.12	27.14	22.54	27.61
	KL	10%	35.22	**26.45**	**23.95**	7.05	45.83	**40.45**	28.72	**24.55**	29.03
	s^{PAD}	10%	**36.69**	26.16	23.74	**7.28**	**46.89**	40.22	**29.13**	24.21	**29.29**
Llama-3.2-3B	C.S.	10%	32.57	19.88	22.05	10.67	46.12	38.83	38.97	30.25	29.92
	Eu.	10%	42.36	22.45	32.11	16.23	55.67	49.25	49.82	40.15	38.51
	KL	10%	43.78	**24.50**	33.14	16.78	56.89	50.33	50.47	41.12	39.63
	s^{PAD}	10%	**44.96**	24.33	**34.27**	**17.12**	**58.28**	**51.67**	**52.21**	**42.78**	**40.70**
Llama-3.1-8B	C.S.	10%	41.85	22.47	48.22	19.17	61.45	52.93	55.34	49.05	43.81
	Eu.	10%	48.94	26.03	62.78	20.84	69.12	63.47	61.02	50.37	50.32
	KL	10%	50.12	26.75	63.91	21.26	**73.05**	64.12	62.15	51.43	51.60
	s^{PAD}	10%	**51.33**	**27.13**	**64.13**	**22.48**	72.68	**65.03**	**63.67**	**52.88**	**52.41**

Table 2: Ablation study on metrics for **ALPS** with 10% of attention heads. **Bold** indicates the best performance.

Model	Ratio	General		Math		Code				Avg.
		IFEval	GPQA	GSM8K	MATH	HEval	HEval+	MBPP	MBPP+	
Llama-3.2-1B	100%	34.53	18.75	17.82	4.60	45.12	39.08	28.63	23.78	26.54
	70%	31.80	16.91	20.63	6.35	44.20	39.50	28.40	24.90	26.59 ↑0.05
	50%	33.46	18.07	21.08	7.15	43.95	39.69	28.92	24.64	27.12 ↑0.58
	30%	**37.12**	25.98	22.56	7.12	46.34	39.84	28.95	24.05	29.00 ↑2.46
	10%	36.69	**26.16**	**23.74**	**7.28**	**46.89**	**40.22**	29.13	24.21	**29.29 ↑2.75**
	0%	35.85	19.64	23.35	5.20	41.51	35.28	**29.91**	**24.88**	26.95 ↑0.41
Llama-3.2-3B	100%	42.81	22.32	31.77	16.68	56.18	50.53	50.00	41.50	38.97
	70%	39.83	21.91	31.63	15.35	56.21	51.55	49.40	39.97	38.23 ↓0.74
	50%	40.93	22.18	32.17	16.10	55.91	49.72	50.05	40.33	38.42 ↓0.55
	30%	43.53	23.46	33.91	16.54	57.28	51.69	51.61	41.93	39.99 ↑1.02
	10%	**44.96**	**24.33**	**34.27**	**17.12**	**58.28**	**51.67**	**52.21**	**42.78**	**40.70 ↑1.73**
	0%	40.29	24.33	33.21	13.76	55.52	50.62	48.71	39.72	38.27 ↓0.70
Llama-3.1-8B	100%	51.20	25.22	63.00	20.52	69.50	64.28	63.21	52.38	51.16
	70%	50.80	24.91	62.63	20.35	69.20	61.50	63.40	51.90	50.59 ↓0.57
	50%	**52.55**	**27.28**	61.11	20.95	69.69	63.35	61.71	51.02	50.96 ↓0.20
	30%	51.85	26.47	62.72	22.15	71.15	64.55	62.88	52.23	51.62 ↑0.46
	10%	51.33	27.13	**64.13**	**22.48**	**72.68**	**65.03**	**63.67**	**52.88**	**52.41 ↑1.25**
	0%	40.41	26.16	64.06	15.48	70.72	64.88	63.27	52.55	49.69 ↓1.47

Table 3: Ablation study on attention head selection ratio for **ALPS** . The best results are highlighted in **bold**. ↓↑ indicates change relative to the w/ full baseline.

widely used parameter-efficient fine-tuning method that freezes all original weights and updates only a low-rank decomposition of weight matrices. We set the LoRA rank to 128 and only to QKV matrices. For consistency in head indexing, all head selections follow the query index. For a Q index of 20, the K and V indices are determined by $\lceil 20 * g/h \rceil$, where g is the number of KV groups, and h is the total number of heads.

Implementation Details. We optimize our models using the AdamW (Loshchilov, 2017) optimizer with hyperparameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$, and a weight decay of 0.1. A cosine decay schedule is employed, gradually reducing the learning rate to 10% of its initial value throughout training, following a linear warmup ratio of 0.1. We set the effective batch size to 128 and initialize the learning rate at 2×10^{-5}. All models are trained for 3 epochs using mixed precision (fp16) on 8 A100 GPUs.

4.2 Main Results

Table 1 presents the main results comparing **ALPS** against other baselines across three model scales (1B, 3B, and 8B) and three downstream tasks (general, math, and code). Note that each task uses a separately fine-tuned model trained on its respective dataset, that is, the results of general tasks are from the general fine-tuned model, while the results of math and code are from math and code fine-tuned models, respectively. Notably, our method consistently outperforms all baselines, showing a 2.75% improvement over the *w/ full* baseline on the 1B model and maintains a 1.25% gain at the 8B scale. This indicates the redundancy among attention heads in downstream alignment. In contrast, the *w/o attn* settings fall significantly behind the *w/ full* baseline, highlighting the critical role of attention heads in downstream alignment. Furthermore, results from *Random*, *LC*, *LoRA*, and our method indicate that selecting a subset of heads can indeed enhance alignment efficiency. However, how these heads are chosen remains pivotal, and selecting more task-sensitive attention heads can outperform training on all heads, underscoring the importance of precise head identification for maximizing both efficiency and performance.

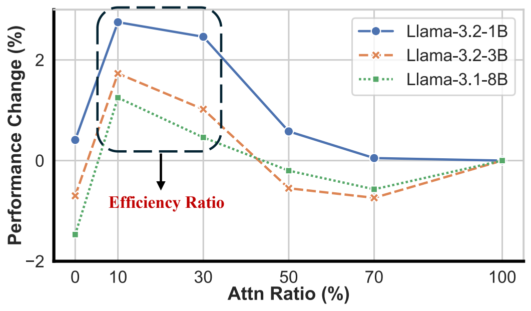


Figure 4: Ablation study on attention head selection ratio. The efficiency ratio represents that activating 10% and 30% of attention heads yields better performance than full fine-tuning across three models.

4.3 Ablations Study

Head Metric Ablation. Table 2 compares four metrics: *C.S.* (cosine similarity), *Eu.* (Euclidean distance), *KL* (Kullback-Leibler Divergence (Kullback and Leibler, 1951)), and our proposed *parameter alignment distribution score* s^{PAD}, for identifying task-sensitive attention heads. Notably, *C.S.* yields the weakest performance, likely because it focuses on overall matrix similarity, which can overlook significant local shifts tied to specific task requirements. In contrast, across all model scales (1B, 3B, 8B) and tasks (general, math, code), our proposed metric s^{PAD} consistently achieves the highest average scores, indicating its effectiveness in capturing the most critical distribution shifts between the base and task fine-tuned models.

Head Ratio Ablation. The ablation study on the proportion of attention heads selected for our method ranging from *0%* to *100%*, is presented in Table 3 and Figure 4. We observe that both *10%* and *30%* ratios consistently yield strong performance across all model scales and tasks, as highlighted by the efficiency ratio region in Figure 4. Notably, updating 10% of the heads already achieves comparable to or better results than higher ratios, and strikes a better trade-off between performance and computational cost by reducing the number of trainable parameters. Consequently, we adopt *10%* as our default setting for improved efficiency.

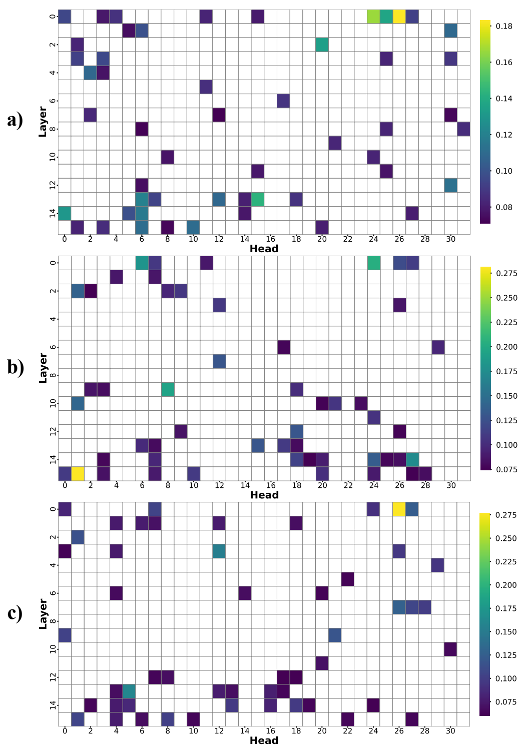


Figure 5: Heatmaps of task-sensitive attention heads (10%) selected by our method on Llama-3.2-1B for **a)** general, **b)** math, and **c)** code tasks. The color intensity indicates the value of s^{PAD} of each head.

4.4 Head Impact Analysis

Figure 5 shows heatmaps of the top 10% attention heads identified by our method across three tasks, full heatmaps are listed in Appendix F. Each grid cell corresponds to a layer-head combination, and the color intensity reflects the importance of the head. We observe that general and code tasks exhibit notable overlap in their most sensitive heads, potentially due to the presence of extensive natural language instructions in the code datasets. Meanwhile, math reveals a more distinct pattern, suggest-

Model	Method	General	Math	Code	Avg.
Llama-3.2-1B	w/ full	14.28	5.87	22.28	14.14
	w/o attn	19.73	8.39	19.98	16.03 ↑1.89
	Random	21.80	7.91	19.63	16.45 ↑2.31
	ALPS	**23.43**	**11.51**	**24.11**	**19.68 ↑5.54**
Llama-3.2-3B	w/ full	21.57	15.23	38.55	25.12
	w/o attn	22.31	13.49	35.72	23.84 ↓1.28
	Random	22.38	17.80	**41.37**	27.18 ↑2.06
	ALPS	**24.65**	**18.70**	40.24	**27.86 ↑2.74**
Llama-3.1-8B	w/ full	27.21	28.76	42.34	32.77
	w/o attn	18.29	25.77	**46.11**	30.06 ↓3.38
	Random	27.93	**32.96**	40.43	33.77 ↑1.00
	ALPS	**28.21**	31.31	44.57	**34.67 ↑1.90**

Table 4: Performance of identified heads with a new set of general, math, and code datasets. The best results are highlighted in **bold**. ↓↑ indicates change relative to the baseline. Our method consistently outperforms other methods, demonstrating the transferability of selected task heads. Full details are listed in Appendix G.

ing specialized attention requirements for mathematical reasoning. Some heads appear consistently important across all three tasks, indicating partial head sharing that aligns with similarities in data distributions as illustrated in Appendix A. These findings highlight that our method effectively identifies both task-specific and task-agnostic heads, shedding light on how different downstream objectives influence attention-head specialization.

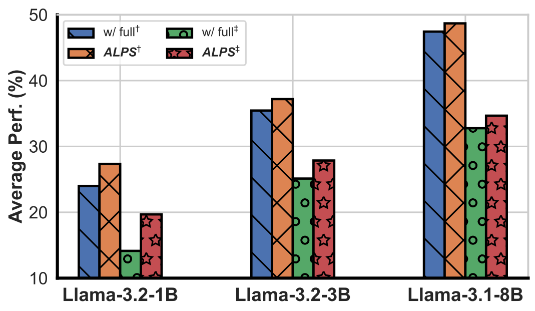


Figure 6: Evaluation on transferability of **ALPS** with a new set of general, math, and code datasets. Results with † correspond to results from Table 1, while ‡ represents performance from Table 4.

4.5 Does Heads Transfer?

To assess whether ALPS-selected heads generalize across different datasets within the same task domain, we evaluate them on a new set of general, math, and code datasets, as illustrated in Section 4.1. Table 4 and Figure 6 show that our method consistently outperforms all baselines, indicating the transferability of the task-sensitive attention heads identified by **ALPS** that can be reused effectively on new datasets within the same tasks. This highlights the potential of our method for reducing alignment costs without sacrificing performance.

Model	Method	MMLU	ARC-C	Avg.
Llama-3.2-1B	vanilla	32.2	32.8	32.5
	w/ full	28.14	26.95	27.55
	w/o attn	27.80	23.18	25.49
	Random	27.86	26.27	27.07
	ALPS	**29.88**	**28.73**	**29.31**
Llama-3.2-3B	vanilla	58	69.1	63.55
	w/ full	44.22	50.82	47.52
	w/o attn	46.16	46.95	46.56
	Random	45.55	47.30	46.43
	ALPS	**48.83**	**52.37**	**50.60**
Llama-3.1-8B	vanilla	66.7	79.7	73.2
	w/ full	55.40	60.34	57.87
	w/o attn	22.95	49.01	35.98
	Random	54.59	62.92	58.76
	ALPS	**57.87**	**64.29**	**61.08**

Table 5: Performance comparison on two common general benchmarks between our method and baselines with the vanilla models. Full details are listed in Appendix G.

4.6 Fewer Heads Avoid Forgetting

Table 5 presents results on MMLU and ARC-C, two commonly used benchmarks to assess general knowledge, making them vulnerable to knowledge forgetting when the alignment dataset is limited in quality or quantity (Chang et al., 2024). The *w/ full* baseline shows a noticeable drop in performance, reflecting the loss of pre-trained knowledge. In contrast, **ALPS** , which restricts updates to a small fraction of attention heads, better preserves this general knowledge and consistently outperforms both *w/ full*, *w/o attn*, and *Random* baselines. By limiting the number of trained heads that introduce sparsity, our method reduces overfitting to the smaller task-specific dataset, thereby mitigating knowledge forgetting from the pre-training phase.

5 Conclusion

This work presents **ALPS** , an approach that leverages model weight parameters to localize and prune task-sensitive attention heads, leading to more efficient LLM alignment while avoiding data dependency. Extensive experiments on general, math, and code tasks demonstrate that our method consistently outperforms baselines, exhibits head transferability, and mitigates knowledge forgetting. Overall, **ALPS** offers a promising perspective on efficient LLM alignment and paves the way for further research into parameter-efficient alignment strategies that capitalize on the role of attention heads.

6 Limitation

While **ALPS** demonstrates promising improvements in efficiency and task performance, our investigation into its transferability and its ability to mitigate knowledge forgetting remains preliminary. Due to training overhead constraints, our evaluations have been limited to a select set of downstream tasks and datasets. Moreover, further investigation is needed to elucidate the underlying mechanism by which **ALPS** preserves pre-trained knowledge. Despite these limitations, our approach still effectively avoids data dependency and highlights a promising direction for future research in efficient LLM alignment.

7 Acknowledgment

This work is mainly supported by the National Key Research and Development Program of China (No. 2022YFB3304100), and partially by the Fundamental Research Funds for the Central Universities (226-2024-00049).

References

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

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. 2023. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245*.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein gan. *Preprint*, arXiv:1701.07875.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. 2024. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, 15(3):1–45.

Sahil Chaudhary. 2023. Code alpaca: An instruction-following llama model for code generation. `https://github.com/sahil280114/codealpaca`.

Hao Chen, Yiming Zhang, Qi Zhang, Hantao Yang, Xiaomeng Hu, Xuetao Ma, Yifan Yanggong, and Junbo Zhao. 2023a. Maybe only 0.5% data is needed: A preliminary exploration of low training data instruction tuning. *arXiv preprint arXiv:2305.09246*.

Jianhui Chen, Xiaozhi Wang, Zijun Yao, Yushi Bai, Lei Hou, and Juanzi Li. 2024. Finding safety neurons in large language models. *arXiv preprint arXiv:2406.14144*.

Lichang Chen, Shiyang Li, Jun Yan, Hai Wang, Kalpa Gunaratna, Vikas Yadav, Zheng Tang, Vijay Srinivasan, Tianyi Zhou, Heng Huang, et al. 2023b. Alpagasus: Training a better alpaca with fewer data. *arXiv preprint arXiv:2307.08701*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Kevin Clark. 2019. What does bert look at? an analysis of bert's attention. *arXiv preprint arXiv:1906.04341*.

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

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. 2023. Enhancing chat language models by scaling high-quality instructional conversations. *Preprint*, arXiv:2305.14233.

Yu Fu, Zefan Cai, Abedelkadir Asi, Wayne Xiong, Yue Dong, and Wen Xiao. 2024. Not all heads matter: A head-level kv cache compression method with integrated retrieval and reasoning. *arXiv preprint arXiv:2410.19258*.

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, Muennighoff, et al. 2024. A framework for few-shot language model evaluation.

Atticus Geiger, Hanson Lu, Thomas Icard, and Christopher Potts. 2021. Causal abstractions of neural networks. *Advances in Neural Information Processing Systems*, 34:9574–9586.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Yu Wu, YK Li, et al. 2024. Deepseek-coder: When the large language model meets programming–the rise of code intelligence. *arXiv preprint arXiv:2401.14196*.

Wes Gurnee, Neel Nanda, Matthew Pauly, Katherine Harvey, Dmitrii Troitskii, and Dimitris Bertsimas. 2023. Finding neurons in a haystack: Case studies with sparse probing. *arXiv preprint arXiv:2305.01610*.

Shwai He, Guoheng Sun, Zheyu Shen, and Ang Li. 2024. What matters in transformers? not all attention is needed. *arXiv preprint arXiv:2406.15786*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Geoffrey Hinton. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, et al. 2024. Qwen2.5-coder technical report. *arXiv preprint arXiv:2409.12186*.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, et al. 2018. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2704–2713.

Goro Kobayashi, Tatsuki Kuribayashi, Sho Yokoi, and Kentaro Inui. 2020. Attention is not only a weight: Analyzing transformers with vector norms. *arXiv preprint arXiv:2004.10102*.

Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023a. Camel: Communicative agents for "mind" exploration of large scale language model society. *Preprint*, arXiv:2303.17760.

Xian Li, Ping Yu, Chunting Zhou, Timo Schick, Omer Levy, Luke Zettlemoyer, Jason Weston, and Mike Lewis. 2023b. Self-alignment with instruction backtranslation. *arXiv preprint arXiv:2308.06259*.

David Lindner, János Kramár, Sebastian Farquhar, Matthew Rahtz, Thomas McGrath, and Vladimir Mikulik. 2023. Tracr: Compiled transformers as a laboratory for interpretability. *Preprint*, arXiv:2301.05062.

Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, et al. 2024. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. 2023. Is your code generated by chatGPT really correct? rigorous evaluation of large language models for code generation. In *Thirty-seventh Conference on Neural Information Processing Systems*.

I Loshchilov. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

meta llama. 2023. Llama recipes: Examples to get started using the llama models from meta. `https://github.com/meta-llama/llama-cookbook/tree/archive-main`.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? *Advances in neural information processing systems*, 32.

OpenAI. 2025. Openai o3-mini.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2023. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*.

Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, et al. 2023. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*.

Robert Schmirler, Michael Heinzinger, and Burkhard Rost. 2024. Fine-tuning protein language models boosts predictions across diverse tasks. *Nature Communications*, 15(1):7407.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Guangyuan Shi, Zexin Lu, Xiaoyu Dong, Wenlong Zhang, Xuanyu Zhang, Yujie Feng, and Xiao-Ming Wu. 2024. Understanding layer significance in llm alignment. *arXiv preprint arXiv:2410.17875*.

Karan Singhal, Shekoofeh Azizi, Tao Tu, S Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, et al. 2023. Large language models encode clinical knowledge. *Nature*, 620(7972):172–180.

Karan Singhal, Tao Tu, Juraj Gottweis, Rory Sayres, Ellery Wulczyn, Mohamed Amin, Le Hou, Kevin Clark, Stephen R Pfohl, Heather Cole-Lewis, et al. 2025. Toward expert-level medical question answering with large language models. *Nature Medicine*, pages 1–8.

Derek Tam, Mohit Bansal, and Colin Raffel. 2024. Merging by matching models in task parameter subspaces. *Transactions on Machine Learning Research*.

Hanlin Tang, Yang Lin, Jing Lin, Qingsen Han, Shikuan Hong, Yiwu Yao, and Gongyi Wang. 2024. Razorattention: Efficient kv cache compression through retrieval heads. *arXiv preprint arXiv:2407.15891*.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`.

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

Leonid Nisonovich Vaserstein. 1969. Markov processes over denumerable products of spaces, describing large systems of automata. *Problemy Peredachi Informatsii*, 5(3):64–72.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. *arXiv preprint arXiv:1905.09418*.

Zhongwei Wan, Xin Wang, Che Liu, Samiul Alam, Yu Zheng, Jiachen Liu, Zhongnan Qu, Shen Yan, Yi Zhu, Quanlu Zhang, et al. 2023. Efficient large language models: A survey. *arXiv preprint arXiv:2312.03863*.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. Self-instruct: Aligning language models with self-generated instructions. *arXiv preprint arXiv:2212.10560*.

Yuxiang Wei, Zhe Wang, Jiawei Liu, Yifeng Ding, and Lingming Zhang. 2023. Magicoder: Source code is all you need. *arXiv preprint arXiv:2312.02120*.

Wenhao Wu, Yizhong Wang, Guangxuan Xiao, Hao Peng, and Yao Fu. 2024. Retrieval head mechanistically explains long-context factuality. *arXiv preprint arXiv:2404.15574*.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023. Wizardlm: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, et al. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, et al. 2024. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*.

Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. 2023. Mammoth: Building math generalist models through hybrid instruction tuning. *arXiv preprint arXiv:2309.05653*.

Haiyan Zhao, Hanjie Chen, Fan Yang, Ninghao Liu, Huiqi Deng, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, and Mengnan Du. 2024. Explainability for large language models: A survey. *ACM Transactions on Intelligent Systems and Technology*, 15(2):1–38.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

Zifan Zheng, Yezhaohui Wang, Yuxin Huang, Shichao Song, Mingchuan Yang, Bo Tang, Feiyu Xiong, and Zhiyu Li. 2024. Attention heads of large language models: A survey. *arXiv preprint arXiv:2409.03752*.

Weishun Zhong, Ben Sorscher, Daniel Lee, and Haim Sompolinsky. 2022. A theory of weight distribution-constrained learning. *Advances in Neural Information Processing Systems*, 35:14113–14127.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. 2024a. Lima: Less is more for alignment. *Advances in Neural Information Processing Systems*, 36.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*.

Zhenhong Zhou, Haiyang Yu, Xinghua Zhang, Rongwu Xu, Fei Huang, Kun Wang, Yang Liu, Junfeng Fang, and Yongbin Li. 2024b. On the role of attention heads in large language model safety. *arXiv preprint arXiv:2410.13708*.

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

A Datasets and Benchmarks

Datasets. Table 6 provides an overview of the datasets used in our experiments. Each dataset consists of ⟨*instruction, answer*⟩ pairs, which serve as the basis for supervised fine-tuning. These datasets span diverse domains, ensuring a comprehensive evaluation of our approach across general, math, and code tasks

Name	Task	# Samples
UltraChat (Ding et al., 2023)	general	200k
MathInstruct (Yue et al., 2023)	math	262k
Magicoder (Wei et al., 2023)	code	110k
Alpaca (Taori et al., 2023)	general	52k
Camel-math (Li et al., 2023a)	math	50k
CodeAlpaca (Chaudhary, 2023)	code	20k

Table 6: Overview of the datasets employed in our experiments, including dataset name, task domain, and number of samples.

Data Distribution. Figure 7 shows a 3D visualization of the data distribution for UltraChat, MathInstruct, and Magicoder, as embedded by Llama-3.1-8B model. Notably, the code and math datasets exhibit some overlaps, whereas the general dataset displays a distinct distribution, reflecting domain-specific characteristics.

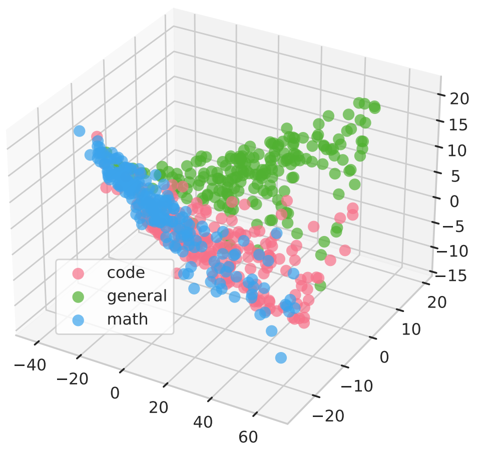


Figure 7: 3D visualization of data distribution for UltraChat, MathInstruct, and Magicoder. All data is embedded using Llama-3.1-8B model.

Evaluations. We evaluate model performance using lm-eval (Gao et al., 2024; meta llama, 2023) to assess general capabilities such as instruction following, reasoning, dialogue, and mathematical abilities, and EvalPlus (Liu et al., 2023) to evaluate the capabilities of code generation. All settings are with the official report, and the details regarding the evaluation benchmarks, including the number of shots and metrics, are summarized in Table 7, with further metric-specific details provided in Table 8

B Training Details

For all experiments, we use the Alpaca (Taori et al., 2023) template for fine-tuning across all datasets and models. Table 9 shows the training template employed, which standardizes the input format and instruction style for all tasks. This consistency facilitates effective fine-tuning across diverse domains.

C Does ALPS Require a Full Fine-tuning Cycle to Obtain a Task-related Model?

For better clarification, the task-related model in our method serves merely as a recipe to extract the parameter shifts needed for head identification, and **the primary focus of ALPS is on the identification of task-sensitive attention heads for subsequent parameter-efficient fine-tuning (PEFT)**, rather than on the process of obtaining a task-related model. For domains such as code and math, well-established task-specific models, e.g., CodeLlama (Roziere et al., 2023) and Qwen-2.5-Coder (Hui et al., 2024), are readily available and can be used directly in conjunction with the same size and architecture model, e.g., Llama-2 (Touvron et al., 2023), Qwen-2.5 (Yang et al., 2025) to perform **ALPS** head identification, and thereby obviating the need for additional full fine-tuning. Since there is no widely recognized task fine-tuned model with the Llama-3.2 series, we then opted to perform full SFT to obtain a task-related baseline. Importantly, this choice was made to ensure a fair comparison and does not affect the core contribution of **ALPS**, which lies in its efficient head identification strategy. Furthermore, transferable experiments in Table 4 demonstrate that once the task-sensitive heads are identified, they remain transferable across subsequent fine-tuning cycles. This one localization enables significant reductions in training cost for all downstream tasks, which serves as a key strength of **ALPS**.

To further support our claim that ALPS does not necessarily involve a full fine-tuning cycle to obtain a task-related model, we conducted experiments showing that even with 0-cost or low-cost training (10% parameter, 10% data size) to obtain the task-related model, **ALPS** still achieves com-

Benchmark	Capability	# Shots	Metric
IFEval (Zhou et al., 2023)	Instruction Following	-	Avg(Prompt/Instruction acc Loose/Strict)
GPQA (Rein et al., 2023)	Reasoning	0	acc
GSM8K (Cobbe et al., 2021)	Math	8	em_maj1@1
MATH (Hendrycks et al., 2021)	Math	0	final_em
Humaneval (Chen et al., 2021)	Code	0	pass@1
Humaneval+ (Liu et al., 2023)	Code	0	pass@1
MBPP (Austin et al., 2021)	Code	0	pass@1
MBPP+ (Liu et al., 2023)	Code	0	pass@1
MMLU (Hendrycks et al., 2020)	General	5	macro_avg/acc
ARC-C (Clark et al., 2018)	Reasoning	0	acc

Table 7: Overview of benchmarks utilized in our experiments, detailing the assessed capability, number of shots, and evaluation metrics.

Metric	Detail
macro_avg/acc (Macro Average Accuracy)	The mean accuracy across all classes or tasks
acc_char (Character-Level Accuracy)	Evaluates character-level correctness
em (Exact Match)	Measures how often the model output, exactly matches the reference answer
f1 (F1 Score)	Balances precision and recall, particularly useful for imbalanced data
pass@1	Assesses the correctness of generated code, on the first attempt
em_maj1@1	Measures exact-match on the first major attempt, especially in complex reasoning or math problems
final_em	The final exact-match score, commonly used in challenging benchmarks

Table 8: Overview of evaluation metric details.

Field	Content
System prompt	Below is an instruction that describes a task. Write a response that appropriately completes the request.
User prompt	### Instruction: {{content}} ### Response:

Table 9: Training template used for fine-tuning.

petitive performance, as shown in Table 10.

These results illustrate that training a model via head identification using an existing task-related model (0-cost) yields comparable performance, even though these readily available models may not have been fully trained. Moreover, a task model obtained with less data or fewer parameters (lower cost) also performs well after **ALPS** head identification, demonstrating that ALPS does not necessarily require a complete full fine-tuning cycle to obtain

Method	$\mathcal{M}_{\mathcal{T}}$	Math		Code			
		GSM8K	MATH	HEval	HEval+	MBPP	MBPP+
Llama3.2-1B							
w/full	SFT	17.82	4.60	45.12	39.08	28.63	23.78
ALPS_Readily	Readily $\mathcal{M}$	21.81	6.87	45.78	39.29	27.93	24.01
ALPS_Fewer_$\mathcal{D}$	10% data	22.96	7.06	46.91	39.87	28.73	23.62
ALPS_Fewer_θ	attn only (10%)	**23.85**	7.19	**47.23**	**41.01**	28.87	23.98
ALPS	SFT	23.74	**7.28**	46.89	40.22	**29.13**	**24.21**
Llama3.2-3B							
w/full	SFT	31.77	16.68	56.18	50.53	50.00	41.50
ALPS_Readily	Readily $\mathcal{M}$	33.28	16.23	58.27	51.42	**54.87**	**43.33**
ALPS_Fewer_$\mathcal{D}$	10% data	33.87	16.98	56.97	49.68	52.77	41.98
ALPS_Fewer_θ	attn only (10%)	34.17	**17.33**	56.89	50.05	51.79	40.32
ALPS	SFT	**34.27**	17.12	**58.28**	**51.67**	52.21	42.78

Table 10: Experiments on variants of task-specific model acquisition.

the task-related model.

In summary, **ALPS is a method focused on efficient task head identification**. The task-related model serves merely as a component in the pipeline and **the acquisition process does not necessarily require a full fine-tuning cycle,** especially when leveraging readily available task models.

D Efficiency of ALPS

To better understand the efficiency of **ALPS**, we list the training time cost in Table 11. Our method shows significant reductions in training time, thus improving the efficiency.

Model	Method	Avg. Time Cost (hrs)
Llama-3.2-1B	w/ full	2.02
	ALPS	**0.40**
Llama-3.2-3B	w/ full	4.40
	ALPS	**1.05**
Llama-3.1-8B	w/ full	7.67
	ALPS	**2.18**

Table 11: Training time cost.

E Why is Parameter Alignment Distribution Score Better?

Let us assume that the change from $\boldsymbol{W}_{o,B}^{h}$ to $\boldsymbol{W}_{o,T}^{h}$ can be modeled as a small translation in the parameter space:

$$\boldsymbol{W}_{o,T}^{h} = \boldsymbol{W}_{o,B}^{h} + \Delta, \tag{10}$$

where Δ is a small perturbation that reflects task-specific adjustments. The W1 distance between $\boldsymbol{P}_{\mathcal{T}}^{h}$ and $\boldsymbol{P}_{\mathcal{B}}^{h}$ is defined as:

$$W_1\left(\boldsymbol{P}_{\mathcal{B}}^{h}, \boldsymbol{P}_{\mathcal{T}}^{h}\right) = \inf_{\gamma \in \Gamma(\boldsymbol{P}_{\mathcal{B}}^{h}, \boldsymbol{P}_{\mathcal{T}}^{h})} \mathbb{E}_{(x,y)\sim\gamma}\left[\|x-y\|\right]. \tag{11}$$

For the special case where $\boldsymbol{P}_{\mathcal{T}}^{h}$ is simply a translated version of $\boldsymbol{P}_{\mathcal{B}}^{h}$, i.e., $\boldsymbol{P}_{\mathcal{T}}^{h} \approx \boldsymbol{P}_{\mathcal{B}}^{h}(x-\Delta)$, leading to:

$$W_1\left(\boldsymbol{P}_{\mathcal{B}}^{h}, \boldsymbol{P}_{\mathcal{T}}^{h}\right) \approx \|\Delta\|. \tag{12}$$

Thus, in the context of ALPS, $s_h^{PAD} = W_1\left(\boldsymbol{P}_{\mathcal{B}}^{h}, \boldsymbol{P}_{\mathcal{T}}^{h}\right)$ directly reflects the magnitude of the parameter shift caused by task-specific fine-tuning. This linear relationship ensures that even small shifts in the attention head's parameters are captured in proportion to $\|\Delta\|$.

As for the KL divergence, the shifts between $\boldsymbol{P}_{\mathcal{T}}^{h}$ and $\boldsymbol{P}_{\mathcal{B}}^{h}$ is defined as:

$$D_{KL}(\boldsymbol{P}_{\mathcal{B}}^{h}\|\boldsymbol{P}_{\mathcal{T}}^{h}) = \sum_i \boldsymbol{P}_{\mathcal{B}}^{h}(i)\log\frac{\boldsymbol{P}_{\mathcal{B}}^{h}(i)}{\boldsymbol{P}_{\mathcal{T}}^{h}(i)}. \tag{13}$$

For small perturbations Δ, we can perform a second-order Taylor expansion. Under regularity conditions, this yields:

$$D_{KL}(\boldsymbol{P}_{\mathcal{B}}^{h}\|\boldsymbol{P}_{\mathcal{T}}^{h}) \approx \frac{1}{2}\mathcal{I}(\boldsymbol{P}_{\mathcal{B}}^{h})\,\|\Delta\|^2, \tag{14}$$

where $\mathcal{I}(\boldsymbol{P}_{\mathcal{B}}^{h})$ is the Fisher information associated with $\boldsymbol{P}_{\mathcal{B}}^{h}$.

Notice that while W_1 scales linearly with the magnitude of the shift $\|\Delta\|$, the KL divergence scales quadratically. This quadratic behavior means that for small but significant shifts, KL divergence may underemphasize these changes compared to W_1 (Arjovsky et al., 2017). Moreover, KL divergence is asymmetric and can become unstable or even infinite if $\boldsymbol{P}_T^h$ assigns near-zero probability where $\boldsymbol{P}_B^h$ does not.

F Heatmaps

Figure 8 presents heatmaps of the top 10% task-sensitive attention heads selected by our method for three Llama-3 models (Llama-3.2-1B, Llama-3.2-3B, and Llama-3.1-8B) across general, math, and code tasks. Each cell in the heatmap corresponds to a specific layer-head combination, and its color intensity reflects the s^{PAD} value. In the 1B and 3B models, the heatmaps for general and code tasks exhibit considerable overlap, whereas in the 8B model, the distributions for all three tasks are distinctly different. These observations indicate that as the model scale increases, the set of task-sensitive heads shifts, reflecting the diverse strategies that different models employ to handle task-specific requirements. Our method, ALPS, adaptively identifies these critical heads, demonstrating robust generalizability across various model scales and datasets.

G Full Tables

In this section, we present the complete baseline results corresponding to those summarized in Table 4 and Table 5.

In Table 1 we mentioned that each task employs a separately fine-tuned model trained on its own dataset. With six baselines, three datasets, and three model scales, this setup requires $6 \times 3 \times 3 = 54$ distinct models, making a full evaluation across ten benchmarks computationally prohibitive. Consequently, our main experiments report only per-task performance. To probe ALPS's generalization further, we conducted selective evaluations of the baselines on additional tasks. Table 14 summarizes these results. These findings confirm that ALPS consistently generalizes better across diverse tasks.

Model	Method	General	Math	Code	Avg.
Llama-3.2-1B	w/ full	14.28	5.87	22.28	14.14
	w/o attn	19.73	8.39	19.98	16.03 ↑1.89
	Random	21.80	7.91	19.63	16.45 ↑2.31
	LC	21.57	8.02	22.23	17.27 ↑3.13
	LoRA	22.98	9.23	23.27	18.49 ↑4.35
	ALPS	**23.43**	**11.51**	**24.11**	**19.68 ↑5.54**
Llama-3.2-3B	w/ full	21.57	15.23	38.55	25.12
	w/o attn	22.31	13.49	35.72	23.84 ↓1.28
	Random	22.38	17.80	**41.37**	27.18 ↑2.06
	LC	23.42	17.20	40.28	26.97 ↑1.85
	LoRA	24.01	17.98	40.57	27.52 ↑2.40
	ALPS	**24.65**	**18.70**	40.24	**27.86 ↑2.74**
Llama-3.1-8B	w/ full	27.21	28.76	42.34	32.77
	w/o attn	18.29	25.77	**46.11**	30.06 ↓3.38
	Random	27.93	**32.96**	40.43	33.77 ↑1.00
	LC	26.52	30.28	41.28	32.69 ↓0.08
	LoRA	27.87	31.02	42.27	33.72 ↑0.95
	ALPS	**28.21**	31.31	44.57	**34.67 ↑1.90**

Table 12: Full table of Table 4.

Model	Method	MMLU	ARC-C	Avg.
Llama-3.2-1B	vanilla	32.2	32.8	32.5
	w/ full	28.14	26.95	27.55
	w/o attn	27.80	23.18	25.49
	Random	27.86	26.27	27.07
	LC	26.83	27.08	26.96
	LoRA	28.79	27.98	28.39
	ALPS	**29.88**	**28.73**	**29.31**
Llama-3.2-3B	vanilla	58	69.1	63.55
	w/ full	44.22	50.82	47.52
	w/o attn	46.16	46.95	46.56
	Random	45.55	47.30	46.43
	LC	44.34	48.98	46.66
	LoRA	47.28	50.88	49.08
	ALPS	**48.83**	**52.37**	**50.60**
Llama-3.1-8B	vanilla	66.7	79.7	73.2
	w/ full	55.40	60.34	57.87
	w/o attn	22.95	49.01	35.98
	Random	54.59	62.92	58.76
	LC	53.82	61.89	57.86
	LoRA	56.28	63.01	59.65
	ALPS	**57.87**	**64.29**	**61.08**

Table 13: Full table of Table 5.

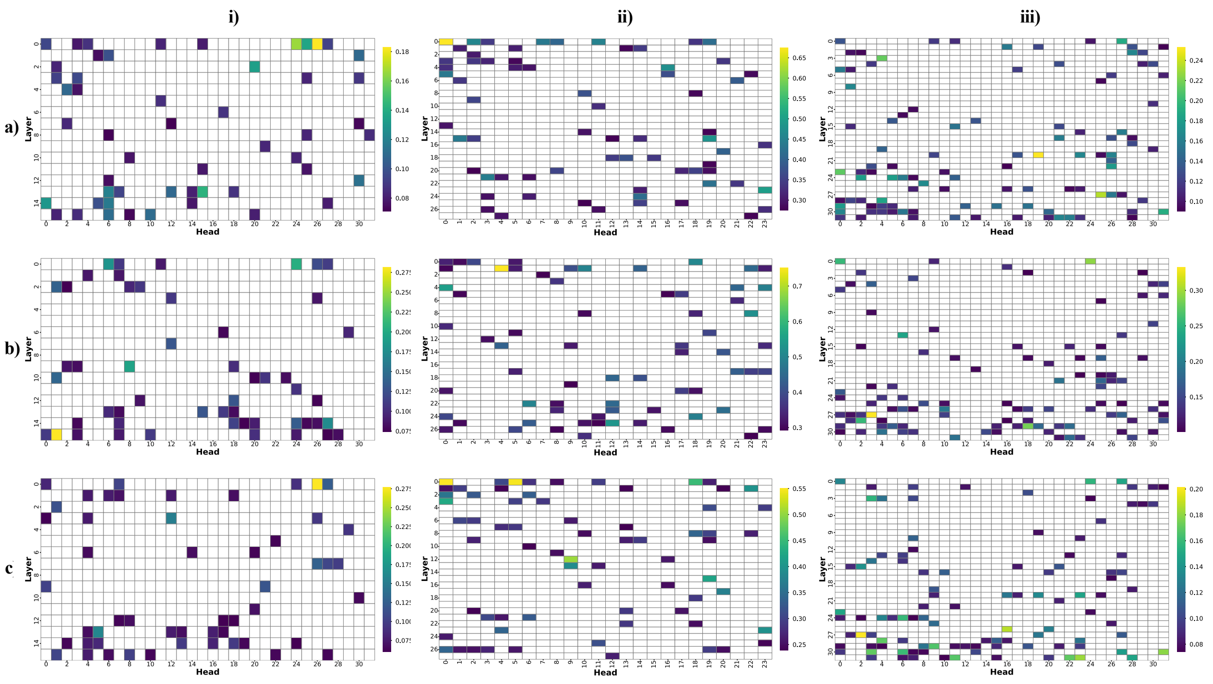


Figure 8: Heatmaps of task-sensitive attention heads (10%) selected by our method on Llama-3.2-1B, Llama-3.2-3B, and Llama-3.1-8B, for **a)** general, **b)** math, and **c)** code tasks. The color intensity indicates the value of s_h^{PAD} of each head.

Model	**Method**	**Task**	**Attn%**	**General**		**Math**		**Code**			
				IFEval	GPQA	GSM8K	MATH	HEval	HEval+	MBPP	MBPP+
	w/ full	code	100%	27.58	10.27	**7.88**	**2.50**	*45.12*	*39.08*	*28.63*	*23.78*
1B-MC	w/o attn	code	0%	28.30	11.16	7.28	2.10	*41.51*	*35.28*	***29.91***	***24.88***
	ALPS	code	10%	**29.38**	**12.27**	7.81	2.48	***46.89***	***40.22***	*29.13*	*24.21*
	w/ full	math	100%	31.06	8.26	*17.82*	*4.60*	11.63	10.42	16.91	**16.17**
1B-MI	w/o attn	math	0%	30.18	17.63	*23.35*	*5.20*	12.87	11.67	16.92	14.87
	ALPS	math	10%	**32.82**	**18.72**	***23.74***	***7.28***	**16.55**	**14.07**	**19.38**	15.68
	w/ full	general	100%	*34.53*	*18.75*	**6.93**	1.24	17.17	14.62	18.89	14.83
1B-UC	w/o attn	general	0%	*35.85*	*19.64*	5.84	2.52	**17.77**	**15.21**	17.75	12.72
	ALPS	general	10%	***36.69***	***26.16***	6.44	**3.32**	16.55	14.38	**19.32**	**15.67**
	w/ full	code	100%	**40.41**	**18.32**	28.73	8.72	*56.18*	*50.53*	*50.00*	*41.50*
3B-MC	w/o attn	code	0%	37.65	17.41	31.31	9.44	*55.52*	*50.62*	*48.71*	*39.72*
	ALPS	code	10%	37.65	18.08	**32.26**	**10.28**	***58.28***	***51.67***	***52.21***	***42.78***
	w/ full	math	100%	35.25	17.63	*31.77*	*16.68*	26.28	23.21	29.92	25.73
3B-MI	w/o attn	math	0%	**35.37**	17.41	*33.21*	*13.76*	28.27	25.63	31.72	27.79
	ALPS	math	10%	33.33	**18.33**	***34.27***	***17.12***	**29.93**	**26.82**	**34.11**	**28.82**
	w/ full	general	100%	*42.81*	*22.32*	6.82	8.16	29.92	27.43	33.33	29.13
3B-UC	w/o attn	general	0%	*40.29*	***24.33***	6.62	**8.22**	**32.94**	**28.49**	**34.97**	28.82
	ALPS	general	10%	***44.96***	***24.33***	5.98	7.92	29.33	26.82	34.43	**29.97**
	w/ full	code	100%	47.84	18.75	46.63	13.40	*69.50*	*64.28*	*63.21*	*52.38*
8B-MC	w/o attn	code	0%	47.48	19.20	**49.13**	10.22	*70.72*	***65.88***	*63.27*	*52.55*
	ALPS	code	10%	**49.96**	**19.64**	48.98	**15.48**	***72.68***	*65.03*	***63.67***	***52.88***
	w/ full	math	100%	36.69	20.76	*63.00*	*20.52*	30.53	28.78	28.23	23.80
8B-MI	w/o attn	math	0%	26.38	22.78	*64.06*	*15.48*	32.38	31.28	30.13	27.62
	ALPS	math	10%	**38.13**	**24.11**	***64.13***	***22.48***	**35.43**	**32.33**	**34.18**	**28.37**
	w/ full	general	100%	*51.20*	*25.22*	54.06	15.12	32.33	28.72	41.54	34.92
8B-UC	w/o attn	general	0%	*40.41*	*26.16*	53.28	14.28	33.28	28.62	41.58	34.78
	ALPS	general	10%	***51.33***	***27.29***	**54.28**	**15.48**	**35.42**	**29.37**	**42.33**	**36.21**

Table 14: Selective baseline evaluation across additional tasks. Italicized and correspond to those in Table 1