SEAP: Sparse Expert Activation Pruning Unlocks the Brainpower of Large Language Models

Xun Liang[1*], Hanyu Wang[1*], Huayi Lai[1], Simin Niu[1], Shichao Song[1], Jiawei Yang[1], Jihao Zhao[1], Feiyu Xiong[2,3], Bo Tang[2,3], Zhiyu Li[2,3†]

[1]Renmin University of China
[2]Institute for Advanced Algorithms Research (Shanghai)
[3]MemTensor (Shanghai) Technology Co., Ltd.
{xliang, hy.wang, laihuayi, niusimin, songshichao, j1aweiyang}@ruc.edu.cn
{xiongfy, tangb, lizy}@iaar.ac.cn

Abstract

Pruning is a promising approach to reduce the high inference cost of large language models (LLMs), but it often comes at the expense of performance. Motivated by the "functional localization" theory in neuroscience, we hypothesize that LLMs contain task-specific expert activation paths, where specific subsets of neurons are co-activated for particular tasks. This structure allows selective activation to preserve task performance while improving inference efficiency. We introduce **Sparse Expert Activation Pruning (SEAP)**, a training-free pruning method for large language models. SEAP identifies task-relevant activation paths by analyzing the clustering patterns of hidden states and neuron activations on a multi-task calibration dataset. Cross-task transfer evaluations confirm the existence of such expert activation structures. SEAP constructs task-aware pruning masks by leveraging a task-expert calibration dataset, which provides representative samples across diverse tasks to reveal their activation signatures. It then employs a lightweight task router to dynamically select relevant computation paths based on the input task. This design significantly reduces inference cost without compromising accuracy. Experimental results show that SEAP retains model performance with only a 1.5% drop on most tasks at 20% sparsity, and at 50% sparsity, it surpasses strong pruning baselines such as WandA and FLAP by over 20%. These results highlight SEAP as a scalable and effective solution for efficient LLM inference.

Code — https://github.com/IAAR-Shanghai/SEAP

Introduction

Large Language Models (LLMs) have achieved remarkable success across a wide spectrum of natural language processing (NLP) tasks (Zhao et al. 2024; Zheng et al. 2025), demonstrating their versatility and adaptability in diverse applications. However, their deployment in real-world scenarios remains a significant challenge due to the substantial computational demands during inference. The inference process of LLMs is constrained by memory bandwidth and hardware limitations (Chavan et al. 2024), making efficient deployment particularly difficult, especially in resource-constrained environments such as real-time systems and edge computing. As LLMs continue to scale, these challenges become even more pronounced, necessitating novel approaches to optimize computational efficiency while preserving model performance.

To mitigate the computational overhead of LLMs, several techniques have been explored. Quantization methods (Bai et al. 2021; Frantar et al. 2023) reduce weight precision, while Mixture of Experts (MoE) architectures (Shazeer et al. 2017; Lepikhin et al. 2020; Fedus, Zoph, and Shazeer 2022) dynamically activate only subsets of the network to improve efficiency. Another widely adopted approach is pruning (Frantar and Alistarh 2023; Ma, Fang, and Wang 2023; Liu et al. 2024), which removes redundant parameters, neurons, or connections to reduce inference costs and storage requirements. Despite the effectiveness of pruning in reducing model complexity, most existing methods are static, relying on activation distributions collected from general datasets such as WikiText-2 (Merity et al. 2016) and C4 (Raffel et al. 2020). Although recent work has begun to examine the impact of different calibration datasets (Bandari et al. 2024), these methods still adopt a uniform pruning strategy across all tasks, which may lead to suboptimal efficiency and fail to fully leverage task-specific knowledge requirements.

Motivation Inspired by the theory of functional localization in cognitive neuroscience—which suggests different brain regions are selectively activated based on task demands (Brett, Johnsrude, and Owen 2002; Mesulam 2000; Li et al. 2022)—we hypothesize that LLMs exhibit a similar mechanism: different tasks activate distinct neuron subsets, forming task-specific computational pathways. This suggests that pruning should be dynamic and task-aware, rather than static and uniform, as traditional methods often overlook task-specific parameter dependencies. We propose a task-aware sparsification strategy that dynamically selects relevant neurons based on input characteristics, enabling more efficient pruning with minimal performance loss.

To validate this hypothesis, we conduct a multi-task experiment on 12 representative benchmarks covering code generation, mathematical reasoning, language modeling, common-

*Equal contribution
†Corresponding author

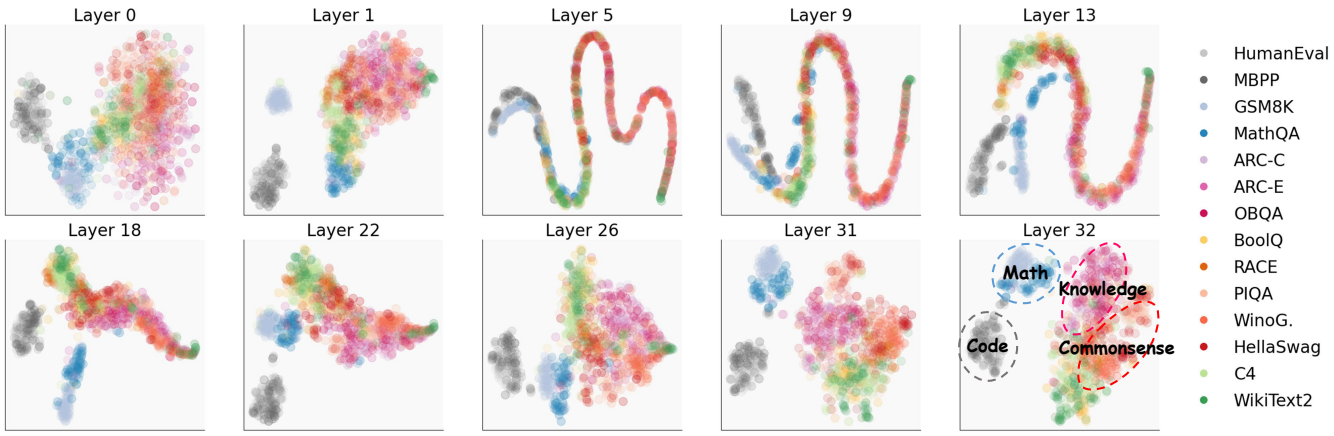

Figure 1: Visualization of hidden states $h(P)$ from different tasks. Each point represents the activation of a hidden state in the model for a specific task. The clustering patterns illustrate how tasks with similar requirements tend to activate similar regions in the model. Tasks of the same type are shown in similar colors, highlighting their representational proximity.

sense reasoning, textual inference, and QA. For each task, we input curated question-answer pairs into LLaMA2-7B and extract hidden states across layers. As shown in Figure 1, task representations are initially entangled but become increasingly clustered at deeper layers, revealing task-dependent activation partitioning.

Our analysis shows that tasks occupy distinct regions in the hidden state space, with activations encoding task-specific information—mirroring functional specialization in the human brain. We propose that leveraging these patterns for dynamic parameter selection during inference enables efficient computation with minimal loss in performance. This task-adaptive pruning paradigm offers a promising alternative to static sparsity, enabling more efficient and specialized LLM deployment.

Contributions Our key contributions are:

- We analyze task-specific activation patterns in LLMs. Visualization and cross-task transfer experiments reveal a strong correlation between task types and hidden state distributions, offering new insights into data-driven, task-aware sparsification.
- We propose SEAP, a training-free, task-adaptive pruning method. It identifies key activation paths using task-specific data and employs a lightweight router to dynamically select relevant parameters. A retention-based sparsity function distributes pruning ratios across layers based on their relative importance.
- Experiments show that SEAP consistently outperforms strong baselines in task accuracy, storage efficiency, and inference speed. It preserves full performance under low sparsity and remains robust at high sparsity, validating its effectiveness for efficient LLM deployment.

Method

Overview of SEAP

Building on the insights from Section Motivation, we propose **Sparse Expert Activation Pruning (SEAP)**, a training-free and task-adaptive pruning framework for LLMs. SEAP selectively activates task-relevant neurons during inference by leveraging activation patterns derived from expert calibration data, reducing computational cost without compromising performance. As illustrated in Figure 2, SEAP comprises the following five stages:

1. **Expert Calibration Dataset Construction**: For each task type (e.g., mathematical reasoning, code generation, commonsense QA), we construct a task-specific calibration dataset containing representative input formats to activate relevant model behavior.
2. **Activation Pattern Modeling**: We forward the expert calibration data through the LLM and extract hidden state activations across multiple layers. This reveals neuron response patterns that characterize task-specific activation.
3. **Neuron Importance Scoring**: Based on statistics such as mean, variance, and ℓ_2 norm of neuron activations, we compute task-aware importance scores that quantify the relevance of each neuron to the task.
4. **Dynamic Sparsity Allocation**: We introduce a retention-based metric to evaluate the impact of pruning each layer under varying sparsity levels. This informs a dynamic allocation of pruning ratios across layers to preserve task-critical neurons while maximizing overall sparsity.
5. **Expert Pruning and Routing**: Using the computed scores, we generate a pruning mask tailored to each task. During inference, a lightweight task router predicts the task type and selects the corresponding expert mask, enabling efficient and adaptive execution.

Activation Patterns Modeling

To validate the feasibility of task-specific expert pruning, we analyze whether the hidden state activations of LLMs exhibit two key properties: (1) intra-task consistency and (2) inter-task differentiation. The presence of both properties is essential for enabling effective pruning based on task-aware neuron selection.

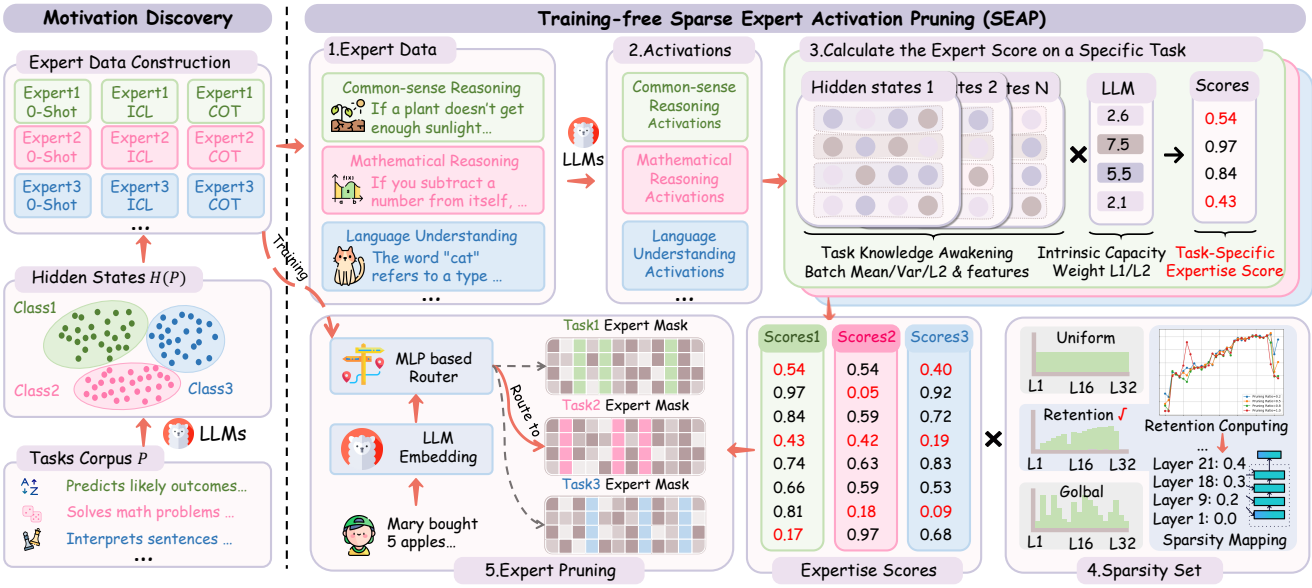

Figure 2: Framework of the SEAP approach. **Left: Motivation Discovery**, where task-specific activations are identified. **Right: Training-free Sparse Expert Activation Pruning** pipeline (see Section **Method**).

Formally, let τ denote a task type and p_i a specific prompt sampled from task τ. For each prompt, we extract its hidden state vector from a particular layer of the model, defined as:

$$\begin{aligned} \mathbf{h}(p_i)^\tau &= \left[h_1(p_i)^\tau, h_2(p_i)^\tau, \ldots, h_C(p_i)^\tau\right], \\ H^\tau &= \{\mathbf{h}(p_1)^\tau, \ldots, \mathbf{h}(p_{n_\tau})^\tau\}, \end{aligned} \tag{1}$$

where C is the dimensionality of the hidden state, and H^τ denotes the set of hidden states for all n_τ prompts under task τ. This formulation enables us to statistically analyze how each hidden dimension responds across different prompts and tasks, which forms the foundation for identifying task-relevant neurons and pruning patterns.

To better quantify how each hidden dimension (or neuron channel) responds under different tasks, we compute the following statistics from the activation vectors. These measures capture the central tendency, variability, and energy of activations across task-specific prompts.

Let $\mathbf{h}_j^\tau = [h_j(p_1)^\tau, h_j(p_2)^\tau, \ldots, h_j(p_{n_\tau})^\tau]^\top \in \mathbb{R}^{n_\tau}$ denote the activation values of dimension j under task τ. Then, the statistics are defined as:

$$\mu_j^\tau = \frac{1}{n_\tau}\sum_{i=1}^{n_\tau} h_j(p_i)^\tau = \frac{1}{n_\tau}\cdot \mathbf{1}^\top \mathbf{h}_j^\tau, \tag{2}$$

$$\sigma_j^\tau = \sqrt{\frac{\sum_{i=1}^{n_\tau}\left(h_j(p_i)^\tau - \mu_j^\tau\right)^2}{n_\tau}} = \frac{\left\|\mathbf{h}_j^\tau - \mu_j^\tau \mathbf{1}\right\|_2}{\sqrt{n_\tau}}, \tag{3}$$

$$\overline{\|\mathbf{h}_j^\tau\|_2} = \frac{1}{n_\tau}\cdot\left\|\mathbf{h}_j^\tau\right\|_2 = \frac{\sqrt{\sum_{i=1}^{n_\tau}\left(h_j(p_i)^\tau\right)^2}}{n_\tau}. \tag{4}$$

Figure 3 visualizes the dimension-wise normalized ℓ_2 activations $\overline{\|h_j^\tau\|_2}$ after the attention and normalization modules, which serve as inputs to the MLP layers. To ensure comparability across tasks, the activations are standardized using z-score normalization.

Columns represent hidden dimensions and rows denote model layers. Brighter regions indicate stronger ℓ_2 activations. Datasets from the same task show consistent high-activation dimensions, indicating stable intra-task activation patterns. Semantically different tasks activate distinct dimensions, suggesting reliance on different neural subspaces. These patterns support our view that similar tasks share activation dimensions, while dissimilar tasks

Furthermore, consider the weight matrix $W \in \mathbb{R}^{A\times C}$ that applies a linear transformation to the hidden state. We conceptualize W as consisting of C column "slices," where each slice $w_i \in \mathbb{R}^A$ corresponds to the i-th dimension in the input hidden state. If a particular dimension i is consistently unimportant for a given task τ, then its associated column w_i can be pruned—thereby reducing computation without sacrificing essential features.

Formally, for a prompt p_i from task τ, the output $\mathbf{o} \in \mathbb{R}^A$ of a linear layer is computed as:

$$\mathbf{o} = W\,\mathbf{h}(p_i)^\tau = \sum_{j=1}^{C} h_j(p_i)^\tau \cdot \mathbf{w}_j = \sum_{j=1}^{C} h_j(p_i)^\tau \begin{bmatrix} w_{1,j} \\ w_{2,j} \\ \vdots \\ w_{A,j} \end{bmatrix}, \tag{5}$$

where each $\mathbf{w}_j \in \mathbb{R}^A$ is the j-th column of the weight matrix W, representing how the j-th hidden dimension contributes to all output neurons. The scalar activation $h_j(p_i)^\tau$ scales the full column $\mathbf{w}_j$, and the total output is the sum of these contributions across all C dimensions. Here, $w_{a,i}$ is the weight linking the i-th hidden dimension to the a-th output unit.

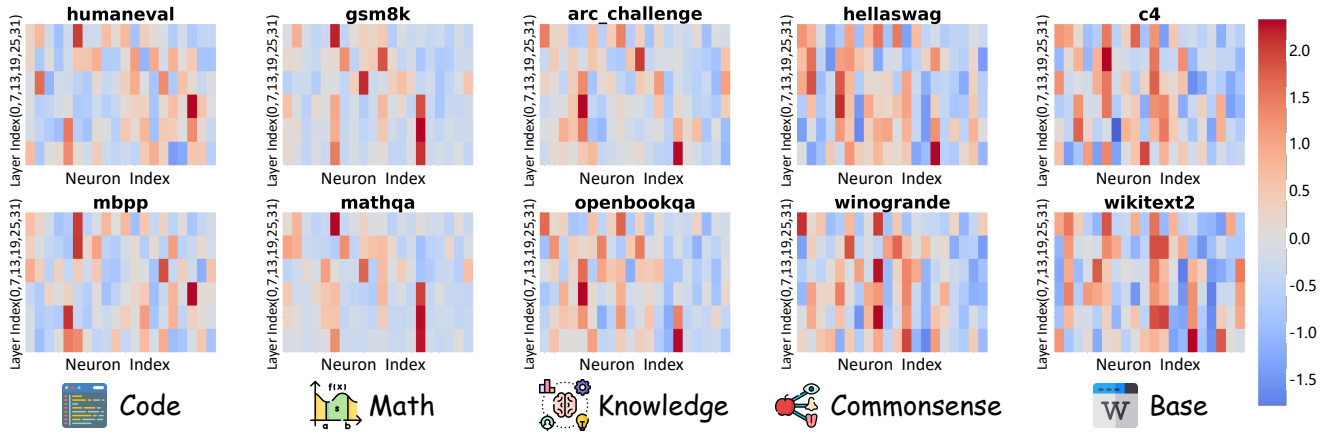

Figure 3: Heatmaps of normalized dimension-wise ℓ_2 activations after the attention and normalization modules, representing the inputs to the MLP layers. Each row corresponds to a layer, and each column represents a hidden dimension. The top and bottom parts show two datasets under the same task category. Despite differences in dataset sources, tasks of the same type exhibit consistent activation patterns, highlighting task-specific neuron selectivity and supporting the feasibility of expert-based pruning.

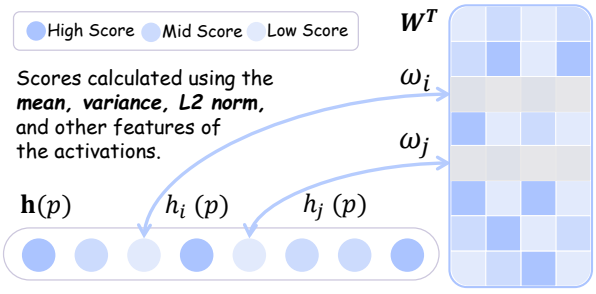

Figure 4: Illustration of how neurons are pruned based on importance scores.

If dimension i is deemed unimportant for task τ, we zero out the entire column $\{w_{1,i}, w_{2,i}, \ldots, w_{A,i}\}$ (see Figure 4), resulting in a form of structured sparsity that is hardware-friendly for inference acceleration.

In summary, by identifying and pruning inactive or low-importance dimensions on a per-task basis, we can achieve task-adaptive compression.

Task-Aware Pruning Procedure

As mentioned, the next step is to prune neurons based on their task-specific importance scores. For each neuron i in layer ℓ under task τ, we compute an importance score $s_i^{(\ell,\tau)}$ using a function $f(\cdot)$, which integrates activation statistics and the associated weight vector:

$$s_i^{(\ell,\tau)} = f\Big(\mu_i^{(\ell,\tau)},\ \sigma_i^{(\ell,\tau)},\ \overline{\|h_i^{(\ell,\tau)}\|_2},\ w_i^{(\ell)}\Big), \quad (6)$$

where $\mu_i^{(\ell,\tau)}$ and $\sigma_i^{(\ell,\tau)}$ denote the mean and standard deviation of activations for neuron i in layer ℓ under task τ; $\overline{\|h_i^{(\ell,\tau)}\|_2}$ represents the average ℓ_2 norm of its activations, and $\|w_i^{(\ell)}\|$ is the ℓ_2-norm of the corresponding weight vector.

The function f represents an abstract scoring space that can instantiate a variety of importance heuristics. In this work, we explore two representative training-free scoring strategies.

The first follows the design in FLAP (An et al. 2024), which defines the importance of a neuron as the product of its activation variance and the squared ℓ_2-norm of its weight vector:

$$s_{F,i}^{(\ell,\tau)} = \big(\sigma_i^{(\ell,\tau)}\big)^2 \cdot \big\|w_i^{(\ell)}\big\|_2^2. \quad (7)$$

This formulation favors neurons whose responses fluctuate more across different prompts and whose corresponding parameters are large in magnitude.

The second scoring strategy draws from WandA (Sun et al. 2024), which replaces the variance with the total activation energy (i.e., squared ℓ_2-norm of neuron activations), and scales it using the ℓ_1-norm of the associated weight vector:

$$s_{W,i}^{(\ell,\tau)} = \big\|h_i^{(\ell,\tau)}\big\|_2^2 \cdot \big\|w_i^{(\ell)}\big\|_1. \quad (8)$$

This variant emphasizes neurons that consistently exhibit strong responses and incorporates weight magnitude to guide sparsity selection.

In MLP layers, each neuron i corresponds to a single activation channel. In attention modules, we aggregate scores at the head level. If attention head h covers a dimension set $\mathcal{I}_h$, its overall importance is computed by summing the scores of the corresponding neurons: $s_h^{(\ell,\tau)} = \sum_{i\in\mathcal{I}_h} s_i^{(\ell,\tau)}$. In attention modules, pruning is performed at the head level by aggregating neuron scores across the dimensions that compose each head.

Once the absolute importance scores $|s_i^{(\ell,\tau)}|$ of all C neurons in layer ℓ are obtained, we sort them and identify the pruning threshold as follows:

$$\theta^{(\ell,\tau)} = \mathrm{Sort}\Big(\big|s_1^{(\ell,\tau)}\big|, \big|s_2^{(\ell,\tau)}\big|, \ldots, \big|s_C^{(\ell,\tau)}\big|\Big)\big[\lfloor \rho \cdot C \rfloor\big], \quad (9)$$

where $\rho \in [0,1]$ is the desired sparsity ratio, and $\lfloor\cdot\rfloor$ denotes the floor function.

All neurons whose importance is below or equal to this threshold are pruned by zeroing their associated weights:

$$w_i^{(\ell)} = \begin{cases} 0, & \text{if } \left|s_i^{(\ell,\tau)}\right| \leq \theta^{(\ell,\tau)}, \\ w_i^{(\ell)}, & \text{otherwise.} \end{cases} \tag{10}$$

Here, $w_i^{(\ell)}$ is the weight vector corresponding to neuron i in layer ℓ, and setting it to zero effectively disables the neuron.

Expert Pruning Strategy

We propose a task-driven pruning strategy, **Expert Pruning**, which leverages task-specific importance scores to dynamically sparsify neurons and attention heads. We organize tasks into five broad capability categories, inspired by the basic abilities of LLMs as outlined in (Zhao et al. 2024), and described in Section Motivation. *code generation* (HumanEval, MBPP), *math reasoning* (GSM8K, MathQA), *commonsense reasoning* (PIQA, Winogrande, HellaSwag), *textual reasoning* (BoolQ, RACE), and *knowledge-based QA* (ARC-C, ARC-E, OBQA). To better capture the semantic and structural diversity of each category, we expand their datasets with various prompting formats, including zero-shot, chain-of-thought (CoT), in-context learning (ICL), and QA pairs. These enhanced corpora are used for activation pattern extraction and importance score computation. Each neuron i in layer ℓ is assigned a task-specific importance score $s_i^{(\ell)} = s_i^{(\ell,\tau_{\text{chosen}})}$, where τ_{chosen} is the task type predicted by a lightweight router. This router is a 5-way MLP classifier trained on the calibration set to infer the task type from the input's initial hidden states. Importantly, downstream tasks are not required to explicitly fall into the five predefined categories. The router predicts the most suitable expert based on the input's internal representation, enabling automatic and flexible task-to-expert mapping without manual categorization.

Retention-Based Sparsity Strategy

To enable adaptive sparsification across layers, we propose a retention-based strategy that assigns layerwise sparsity based on functional importance. Rather than pruning uniformly, we simulate structured pruning at different sparsity levels to evaluate each layer's contribution.

For each layer ℓ, we prune it at multiple ratios $\mathcal{R} = \{0.1, 0.2, \ldots, 1.0\}$ and compare the resulting output $y_\ell^{(r)}$ to the unpruned output y_{ref} using cosine similarity, defined as $\text{sim}(y, y_{\text{ref}}) = \frac{y \cdot y_{\text{ref}}}{\|y\| \cdot \|y_{\text{ref}}\|}$. The *retention score* is then computed as $\text{Ret}_\ell = \frac{1}{|\mathcal{R}|} \sum_{r \in \mathcal{R}} \text{sim}(y_\ell^{(r)}, y_{\text{ref}})$, and the corresponding *importance score* as $\text{Imp}_\ell = 1 - \text{Ret}_\ell$.

We normalize Imp_ℓ across unprotected layers to produce the final sparsity profile $\{\rho_\ell\}$ under the global constraint $\frac{1}{L} \sum_{\ell=1}^{L} \rho_\ell = G$. As illustrated in Figure 5, early layers in LLaMA2-7B show lower retention scores under pruning, indicating higher sensitivity and functional importance.Notably, this finding aligns with conclusions from LLM-Pruner (Ma, Fang, and Wang 2023) and FLAP (An et al. 2024), both of which emphasize that layers near the output are crucial for maintaining language modeling capability.

Algorithm 1: Retention-Based Layerwise Sparsity Assignment

Require: model M, text set T, sparsity ratios $\mathcal{R}$, target sparsity G, total layers L

```
Encode T using M, cache {H_ℓ}_{ℓ=0}^{L}
Get reference output y^ref from M(T)
for each layer ℓ = 1 to L do
    for each r ∈ R do
        Prune layer ℓ at ratio r
        Forward from ℓ to L using H_ℓ, get y^(ℓ,r)
        Compute y^(ℓ,r) · y^ref / (||y^(ℓ,r)|| · ||y^ref||)
    end for
    S_ℓ = mean of similarities,   I_ℓ = 1 − S_ℓ
end for
Normalize {I_ℓ} to get {ρ_ℓ} s.t. (1/L) Σ ρ_ℓ = G
return {ρ_ℓ}_{ℓ=1}^{L}
```

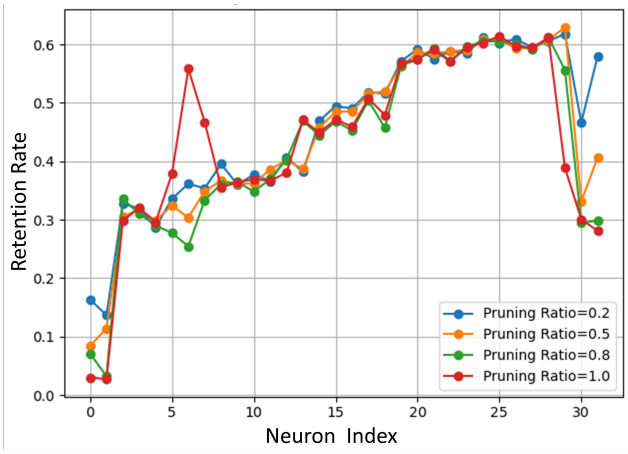

Figure 5: Retention rates of LLaMA2-7B layers under different pruning ratios. Each curve shows output similarity to the full model, indicating retention capacity for sparsity allocation.

Experiment and Results Analysis

Experimental Settings

LLMs and Tasks We evaluate our method on dense models such as LLaMA2-7B and 13B. Our evaluation covers ten downstream tasks. Multiple-choice benchmarks include **BoolQ**, **RACE**, **ARC-Easy**, **ARC-Challenge**, **HellaSwag**, **OBQA**, **PiQA**, **Winogrande**, and **MathQA**. Generation tasks include **GSM8K**, and **MBPP**. Accuracy and pass@1 are used for classification and code generation, respectively. All tasks are evaluated in the zero-shot setting using the EleutherAI LM Harness (Gao et al. 2024).

Baselines We compare our method to the original (dense) models and two training-free sparsification methods: **FLAP** and **WandA-sp**. WandA-sp is a structured variant of WandA that applies uniform layerwise pruning. The importance scores are computed using FLAP's s_F and WandA's s_W formulas, respectively.

Ratio	Method	Wino	HSwag	PIQA	OBQA	ARC-c	ARC-e	BoolQ	RACE	MathQA	Avg
						LLaMA2-7B					
0%	Dense	69.06	76.03	79.12	44.20	46.24	74.54	77.74	39.62	28.17	59.41
20%	WandA-sp (s_W)	65.04	68.26	74.21	39.60	37.71	65.57	66.67	32.82	23.58	52.61
	FLAP (s_F)	64.01	69.38	74.97	37.20	38.65	64.01	68.84	33.97	24.25	52.81
	SEAP (s_W)	67.88	74.84	78.12	**44.00**	44.36	71.97	76.87	**38.09**	28.41	57.88
	SEAP (s_F)	**68.76**	**75.17**	**78.18**	43.40	**44.88**	**72.62**	**77.22**	37.99	**28.48**	**58.52**
30%	WandA-sp (s_W)	55.72	58.93	70.24	33.80	31.91	52.86	50.58	26.31	22.75	44.79
	FLAP (s_F)	61.96	59.90	70.08	33.80	33.10	54.50	57.16	31.67	22.71	47.21
	SEAP (s_W)	66.30	69.86	76.28	**42.80**	**43.17**	67.80	**71.50**	**36.07**	**27.54**	**55.70**
	SEAP (s_F)	**66.93**	**70.75**	**76.33**	42.20	**43.17**	**68.06**	69.82	**36.07**	27.17	55.61
50%	WandA-sp (s_W)	50.04	28.89	53.48	26.20	23.63	28.99	40.58	23.64	20.40	32.87
	FLAP (s_F)	52.09	30.15	53.97	28.00	25.08	28.37	53.61	24.40	21.68	35.26
	SEAP (s_W)	53.04	**54.39**	**70.13**	35.80	36.95	**54.25**	59.17	**31.87**	24.79	46.71
	SEAP (s_F)	**54.14**	54.00	68.66	**38.20**	**37.20**	53.45	**60.24**	31.67	**25.23**	**46.98**
						LLaMA2-13B					
0%	Dense	72.22	79.37	80.52	45.20	49.06	77.53	80.55	40.48	31.79	61.86
20%	WandA-sp (s_W)	64.96	75.86	73.61	40.40	40.70	66.62	72.50	35.42	25.76	55.09
	FLAP (s_F)	65.75	72.54	74.32	41.20	39.76	64.35	72.01	36.27	27.30	54.83
	SEAP (s_W)	**69.85**	77.50	**79.92**	**44.60**	47.78	75.88	78.23	**37.58**	30.72	**60.23**
	SEAP (s_F)	69.38	**77.69**	78.56	44.40	**48.21**	**76.09**	**79.79**	36.36	**31.29**	60.17
30%	WandA-sp (s_W)	62.27	66.73	70.19	37.40	35.62	63.13	64.25	30.41	22.44	50.27
	FLAP (s_F)	62.50	68.14	69.86	36.50	34.71	62.71	63.67	29.59	22.41	50.01
	SEAP (s_W)	**67.32**	75.57	**78.07**	**41.80**	**46.75**	**71.80**	**71.68**	33.42	27.07	57.05
	SEAP (s_F)	66.77	**75.81**	77.74	41.60	47.26	71.68	71.62	**34.07**	**33.49**	**57.78**
50%	WandA-sp (s_W)	53.22	50.49	64.51	27.50	23.84	34.28	57.88	25.77	21.54	39.89
	FLAP (s_F)	54.41	51.28	59.60	29.60	27.75	30.31	55.62	26.85	21.70	39.68
	SEAP (s_W)	59.05	**68.64**	**71.62**	34.20	**37.46**	56.33	**66.73**	31.05	24.55	49.96
	SEAP (s_F)	**61.16**	68.03	70.31	**37.20**	37.20	**57.45**	65.44	**31.67**	**24.62**	**50.34**

Table 1: Task performance accuracy (%) on LLaMA2-7B and LLaMA2-13B under different pruning ratios. Bold and underlined entries highlight the top two scores.

Ratio	Method	GSM8K	MBPP	Avg
		LLaMA2-7B		
0%	Dense	14.10	23.20	18.65
20%	FLAP (s_F)	1.81	1.20	1.51
	SEAP (s_F)	**8.87**	**16.60**	**12.74**
		LLaMA2-13B		
0%	Dense	23.50	27.20	26.35
20%	FLAP (s_F)	5.91	10.60	8.26
	SEAP (s_F)	**16.75**	**13.60**	**15.18**

Table 2: Results on GSM8K (accuracy) and MBPP (pass@1) under 20% pruning. Rows with gray background denote our SEAP methods.

Results and Analysis

Multiple-Choice Tasks We evaluate SEAP on a suite of multiple-choice benchmarks, demonstrating its ability to reduce computational cost while retaining accuracy. As shown in Table 1, for both LLaMA2-7B and 13B, SEAP outperforms WandA-sp and FLAP under 20% pruning, with only a 1.5% drop from the dense model. At 50% pruning, SEAP's advantage grows further, outperforming both baselines by over 20% on average, showing robustness under high sparsity.

Generation Tasks We evaluate SEAP on GSM8K and MBPP, two challenging generation tasks requiring strong mathematical reasoning and code synthesis capabilities. While all methods experience noticeable performance degradation under pruning, SEAP retains 68.3% of the original performance on GSM8K with the 7B model—whereas FLAP almost completely loses its ability to handle math-related generation. These results underscore the importance of evaluating pruning methods beyond multiple-choice benchmarks, especially in scenarios requiring complex generation and reasoning, as shown in Table 2.

Cross-Task Transfer Evaluation We evaluate the transferability of calibration sources via a cross-task transfer experiment, where each of the 14 datasets (e.g., GSM8K, MBPP, PIQA) is used individually as the sole calibration source for pruning. The pruned model is evaluated on all other downstream tasks, excluding C4 and WikiText2 to avoid distribu-

tional overlap. For each target task, we report the relative performance drop compared to its best-performing calibration source.

We observe that calibration datasets from the same task category as the target generally lead to smaller performance degradation, indicating stronger intra-category transferability and supporting the expert pruning hypothesis. In several cases, pruning with a related dataset matches or even surpasses the performance achieved using the target dataset itself, suggesting that task type plays a more critical role than dataset identity in effective calibration.

Inference Speed SEAP completes task-specific importance scoring and mask generation for Llama-2-7B within 5 minutes on a single NVIDIA H800 (80GB) GPU. As shown in Table 3, SEAP delivers significantly faster inference than unstructured pruning baselines like WandA. While slightly slower than FLAP at 20% sparsity, SEAP achieves comparable speed at 50% sparsity with minimal overhead. These results confirm SEAP's efficiency and suitability for practical deployment across diverse hardware platforms.

Ratio	Method	LLaMA2-7B		LLaMA2-13B	
		Tok/s	Up	Tok/s	Up
0%	Dense	31.88		27.45	
20%	WandA	32.05	×1.01	28.01	×1.02
	FLAP	**38.90**	×1.22	**33.96**	×1.24
	SEAP	37.32	×1.17	33.02	×1.20
50%	WandA	31.24	×0.98	27.01	×0.98
	FLAP	**47.94**	×1.50	**43.45**	**×1.58**
	SEAP	47.10	×1.48	41.78	×1.52

Table 3: Inference speed (tokens/s) and speedup at different pruning ratios. Higher is better.

Sparsity Setting Comparison As shown in Table 4, we compare three pruning strategies: Uniform Layerwise (UL), Adaptive Layerwise (AL) from FLAP, and our proposed Retention-Based (RB) method. At both 20% and 50% pruning ratios, SEAP with RB achieves higher multiple-choice accuracy than WandA-sp and FLAP. This demonstrates that RB enables more balanced sparsity allocation across layers, better preserves critical computation paths, and improves overall performance.

Related Works

The computational cost and inference time of LLMs impact deployment. Researchers have addressed these challenges through model compression(Michel, Levy, and Neubig 2019; Yao et al. 2022; Lin et al. 2024), quantization(Bai et al. 2021; Frantar et al. 2023), structural modifications(Gu and Dao 2023; Peng et al. 2023), and optimized decoding. Sparsification has become a key technique, including Mixture of Experts (MoE) (Shazeer et al. 2017), which activates subsets of the network to improve efficiency while maintaining performance (Lewis et al. 2021; Lepikhin et al. 2020; Zhang et al. 2022).

Ratio	Method	Set.	Avg.	Set.	Avg.
		LLaMA2-7B			
0%	Dense	–	59.41	–	59.41
20%	WandA-sp	UL	52.61	RB	53.00
	FLAP	AL	52.81	RB	54.51
	SEAP (s_F)	UL	<u>56.54</u>	RB	**58.52**
50%	WandA-sp	UL	32.87	RB	34.50
	FLAP	AL	35.26	RB	38.88
	SEAP (s_F)	UL	<u>43.17</u>	RB	**46.98**

Table 4: Comparison of sparsity strategies on LLaMA2-7B. UL = Uniform, AL = Adaptive, RB = Retention-Based.

Pruning is another effective sparsification technique for reducing computational and memory costs, categorized into unstructured, structured, and activation pruning. **Unstructured Pruning**, which sparsifies individual weights but can hinder hardware efficiency. Examples include SparseGPT (Frantar and Alistarh 2023) and WandA (Sun et al. 2024). **Structured Pruning**, which prunes entire units like channels or attention heads for improved hardware efficiency and inference speed, with methods like Bonsai (Dery et al. 2024), QPruner (Zhou et al. 2024), LLM-Pruner (Ma, Fang, and Wang 2023), FLAP (An et al. 2024), and Depth2 (Li, Dong, and Lei 2024). **Activation Pruning** sparsifies network activations, reducing memory bandwidth during inference. Activation functions like SiLU and GeLU (Mirzadeh et al. 2023), and variants like dReLU (Song et al. 2024b), ReGLU (Raffel et al. 2020), and RELU^2 (So et al. 2021; Zhang et al. 2024) help reduce computational load. Methods like TEAL (Liu et al. 2024), CATS (Lee et al. 2024), SCAP (Chua, Pan, and Jain 2024), QSparse (Wang et al. 2024), and ProSparse (Song et al. 2024a) achieve pruning.

Conclusion

We propose **SEAP (Sparse Expert Activation Pruning)**, a training-free and task-adaptive pruning framework for large language models. Inspired by functional specialization in cognitive neuroscience, SEAP reveals that different tasks activate distinct neuron subsets in LLMs. By constructing expert calibration data and modeling task-specific activation patterns, SEAP identifies relevant neurons for each task and generates corresponding pruning masks. A lightweight router enables dynamic selection of expert paths at inference time, supporting efficient and adaptive computation. Moreover, we introduce a retention-based sparsity strategy to allocate pruning ratios across layers based on their functional importance. Experiments demonstrate that SEAP consistently outperforms strong baselines in both accuracy and efficiency. On multiple-choice benchmarks, it preserves nearly full performance at 20% sparsity and exceeds FLAP and WandA by over 20% on average at 50% sparsity. These results validate SEAP's effectiveness in enabling structured, hardware-friendly sparsification with minimal performance loss, making it a promising solution for scalable, real-world LLM deployment.

Acknowledgements

This work was supported by National Natural Science Foundation of China 62072463, Supported by the Fundamental Research Funds for the Central Universities, and the Research Funds of Renmin University of China 24XNKJ31, and Opening Project of State Key Laboratory of Digital Publishing Technology of Founder Group.

References

An, Y.; Zhao, X.; Yu, T.; Tang, M.; and Wang, J. 2024. Fluctuation-Based Adaptive Structured Pruning for Large Language Models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(10): 10865–10873.

Bai, H.; Zhang, W.; Hou, L.; Shang, L.; Jin, J.; Jiang, X.; Liu, Q.; Lyu, M.; and King, I. 2021. BinaryBERT: Pushing the Limit of BERT Quantization. In Zong, C.; Xia, F.; Li, W.; and Navigli, R., eds., *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 4334–4348. Online: Association for Computational Linguistics.

Bandari, A.; Yin, L.; Hsieh, C.-Y.; Jaiswal, A. K.; Chen, T.; Shen, L.; Krishna, R.; and Liu, S. 2024. Is C4 Dataset Optimal for Pruning? An Investigation of Calibration Data for LLM Pruning. In Al-Onaizan, Y.; Bansal, M.; and Chen, Y.-N., eds., *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 18089–18099. Miami, Florida, USA: Association for Computational Linguistics.

Brett, M.; Johnsrude, I. S.; and Owen, A. M. 2002. The problem of functional localization in the human brain. *Nature Reviews Neuroscience*, 3(3): 243–249.

Chavan, A.; Magazine, R.; Kushwaha, S.; Debbah, M.; and Gupta, D. 2024. Faster and lighter LLMs: a survey on current challenges and way forward. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, IJCAI '24. ISBN 978-1-956792-04-1.

Chua, V. S.; Pan, Y.; and Jain, N. 2024. Post-Training Statistical Calibration for Higher Activation Sparsity. arXiv:2412.07174.

Dery, L.; Kolawole, S.; Kagy, J.-F.; Smith, V.; Neubig, G.; and Talwalkar, A. 2024. Everybody Prune Now: Structured Pruning of LLMs with only Forward Passes. arXiv:2402.05406.

Fedus, W.; Zoph, B.; and Shazeer, N. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120): 1–39.

Frantar, E.; and Alistarh, D. 2023. SparseGPT: Massive Language Models Can Be Accurately Pruned in One-Shot. arXiv:2301.00774.

Frantar, E.; Ashkboos, S.; Hoefler, T.; and Alistarh, D. 2023. OPTQ: Accurate Quantization for Generative Pre-trained Transformers. In *The Eleventh International Conference on Learning Representations*.

Gao, L.; Tow, J.; Abbasi, B.; Biderman, S.; Black, S.; DiPofi, A.; Foster, C.; Golding, L.; Hsu, J.; Le Noac'h, A.; Li, H.; McDonell, K.; Muennighoff, N.; Ociepa, C.; Phang, J.; Reynolds, L.; Schoelkopf, H.; Skowron, A.; Sutawika, L.; Tang, E.; Thite, A.; Wang, B.; Wang, K.; and Zou, A. 2024. A framework for few-shot language model evaluation.

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

Lee, D.; Lee, J.-Y.; Zhang, G.; Tiwari, M.; and Mirhoseini, A. 2024. CATS: Contextually-Aware Thresholding for Sparsity in Large Language Models. arXiv:2404.08763.

Lepikhin, D.; Lee, H.; Xu, Y.; Chen, D.; Firat, O.; Huang, Y.; Krikun, M.; Shazeer, N.; and Chen, Z. 2020. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*.

Lewis, M.; Bhosale, S.; Dettmers, T.; Goyal, N.; and Zettlemoyer, L. 2021. Base layers: Simplifying training of large, sparse models. In *International Conference on Machine Learning*, 6265–6274. PMLR.

Li, J.; Dong, Y.; and Lei, Q. 2024. Greedy Output Approximation: Towards Efficient Structured Pruning for LLMs Without Retraining. arXiv:2407.19126.

Li, Y.; Liu, A.; Fu, X.; Mckeown, M. J.; Wang, Z. J.; and Chen, X. 2022. Atlas-guided parcellation: Individualized functionally-homogenous parcellation in cerebral cortex. *Computers in Biology and Medicine*, 150: 106078.

Lin, J.; Tang, J.; Tang, H.; Yang, S.; Chen, W.-M.; Wang, W.-C.; Xiao, G.; Dang, X.; Gan, C.; and Han, S. 2024. AWQ: Activation-aware Weight Quantization for On-Device LLM Compression and Acceleration. *Proceedings of Machine Learning and Systems*, 6: 87–100.

Liu, J.; Ponnusamy, P.; Cai, T.; Guo, H.; Kim, Y.; and Athiwaratkun, B. 2024. Training-Free Activation Sparsity in Large Language Models. arXiv:2408.14690.

Ma, X.; Fang, G.; and Wang, X. 2023. LLM-Pruner: On the Structural Pruning of Large Language Models. In *Advances in Neural Information Processing Systems*.

Merity, S.; Xiong, C.; Bradbury, J.; and Socher, R. 2016. Pointer Sentinel Mixture Models. arXiv:1609.07843.

Mesulam, M. M. 2000. *Principles of Behavioral and Cognitive Neurology*. Oxford University Press. ISBN 9780195134759.

Michel, P.; Levy, O.; and Neubig, G. 2019. Are Sixteen Heads Really Better than One? *Advances in neural information processing systems*, 32.

Mirzadeh, I.; Alizadeh, K.; Mehta, S.; Del Mundo, C. C.; Tuzel, O.; Samei, G.; Rastegari, M.; and Farajtabar, M. 2023. Relu strikes back: Exploiting activation sparsity in large language models. *arXiv preprint arXiv:2310.04564*.

Peng, B.; Alcaide, E.; Anthony, Q.; Albalak, A.; Arcadinho, S.; Biderman, S.; Cao, H.; Cheng, X.; Chung, M.; Derczynski, L.; Du, X.; Grella, M.; Gv, K.; He, X.; Hou, H.; Kazienko, P.; Kocon, J.; Kong, J.; Koptyra, B.; Lau, H.; Lin, J.; Mantri, K. S. I.; Mom, F.; Saito, A.; Song, G.; Tang, X.; Wind, J.; Woźniak, S.; Zhang, Z.; Zhou, Q.; Zhu, J.; and Zhu, R.-J. 2023. RWKV: Reinventing RNNs for the Transformer Era. In Bouamor, H.; Pino, J.; and Bali, K., eds., *Findings of the Association for Computational Linguistics: EMNLP 2023*,

14048–14077. Singapore: Association for Computational Linguistics.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 21(140): 1–67.

Shazeer, N.; Mirhoseini, A.; Maziarz, K.; Davis, A.; Le, Q.; Hinton, G.; and Dean, J. 2017. Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer. In *International Conference on Learning Representations*.

So, D.; Mańke, W.; Liu, H.; Dai, Z.; Shazeer, N.; and Le, Q. V. 2021. Searching for Efficient Transformers for Language Modeling. In Beygelzimer, A.; Dauphin, Y.; Liang, P.; and Vaughan, J. W., eds., *Advances in Neural Information Processing Systems*.

Song, C.; Han, X.; Zhang, Z.; Hu, S.; Shi, X.; Li, K.; Chen, C.; Liu, Z.; Li, G.; Yang, T.; et al. 2024a. ProSparse: Introducing and Enhancing Intrinsic Activation Sparsity within Large Language Models. *arXiv preprint arXiv:2402.13516*.

Song, Y.; Xie, H.; Zhang, Z.; Wen, B.; Ma, L.; Mi, Z.; and Chen, H. 2024b. Turbo Sparse: Achieving LLM SOTA Performance with Minimal Activated Parameters. *arXiv preprint arXiv:2406.05955*.

Sun, M.; Liu, Z.; Bair, A.; and Kolter, J. Z. 2024. A Simple and Effective Pruning Approach for Large Language Models. In *The Twelfth International Conference on Learning Representations*.

Wang, H.; Ma, S.; Wang, R.; and Wei, F. 2024. Q-sparse: All large language models can be fully sparsely-activated. *arXiv preprint arXiv:2407.10969*.

Yao, Z.; Yazdani Aminabadi, R.; Zhang, M.; Wu, X.; Li, C.; and He, Y. 2022. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. *Advances in Neural Information Processing Systems*, 35: 27168–27183.

Zhang, Z.; Lin, Y.; Liu, Z.; Li, P.; Sun, M.; and Zhou, J. 2022. MoEfication: Transformer Feed-forward Layers are Mixtures of Experts. In Muresan, S.; Nakov, P.; and Villavicencio, A., eds., *Findings of the Association for Computational Linguistics: ACL 2022*, 877–890. Dublin, Ireland: Association for Computational Linguistics.

Zhang, Z.; Song, Y.; Yu, G.; Han, X.; Lin, Y.; Xiao, C.; Song, C.; Liu, Z.; Mi, Z.; and Sun, M. 2024. ReLU^2 Wins: Discovering Efficient Activation Functions. *arXiv preprint arXiv:2402.03804*.

Zhao, W. X.; Zhou, K.; Li, J.; Tang, T.; Wang, X.; Hou, Y.; Min, Y.; Zhang, B.; Zhang, J.; Dong, Z.; Du, Y.; Yang, C.; Chen, Y.; Chen, Z.; Jiang, J.; Ren, R.; Li, Y.; Tang, X.; Liu, Z.; Liu, P.; Nie, J.-Y.; and Wen, J.-R. 2024. A Survey of Large Language Models. arXiv:2303.18223.

Zheng, Z.; Wang, Y.; Huang, Y.; Song, S.; Yang, M.; Tang, B.; Xiong, F.; and Li, Z. 2025. Attention heads of large language models. *Patterns*, 6(2).

Zhou, C.; Zhou, Y.; Han, S.; Qiao, Q.; and Li, H. 2024. QPruner: Probabilistic Decision Quantization for Structured Pruning in Large Language Models. arXiv:2412.11629.