SEAP: Training-free Sparse Expert Activation Pruning Unlock the Brainpower of Large Language Models

Anonymous ACL submission

Abstract

Large Language Models have achieved remarkable success across various natural language processing tasks, yet their high computational cost during inference remains a major bottleneck. This paper introduces Sparse Expert Activation Pruning (SEAP)[1], a training-free pruning method that selectively retains task-relevant parameters to reduce inference overhead. Inspired by the clustering patterns of hidden states and activations in LLMs, SEAP identifies task-specific expert activation patterns and prunes the model while preserving task performance and enhancing computational efficiency. Experimental results demonstrate that SEAP significantly reduces computational overhead while maintaining competitive accuracy. Notably, at 50% pruning, SEAP surpasses both WandA and FLAP by over 20%, and at 20% pruning, it incurs only a 2.2% performance drop compared to the dense model. These findings highlight SEAP's scalability and effectiveness, making it a promising approach for optimizing large-scale LLMs.

1 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 et al., 2022) dynamically activate only subsets of the network to improve efficiency. Another widely adopted approach is pruning (Frantar and Alistarh, 2023; Ma et al., 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., 2020a). These methods apply a uniform pruning strategy across all tasks, which may lead to suboptimal efficiency and fail to fully leverage task-specific knowledge requirements.

Inspired by cognitive neuroscience, where different brain regions are selectively activated based on task demands, we hypothesize that a similar mechanism exists in LLMs—where different tasks rely on distinct sets of neurons working collaboratively. This suggests that pruning strategies should be adaptive rather than static, dynamically selecting the most relevant parameters for each task. By leveraging task-specific activation patterns, we can develop a more effective sparsification technique that maintains task performance while significantly enhancing computational efficiency.

Motivation Discovery In cognitive neuroscience, the brain parcellation theory posits that different regions of the brain are selectively activated based on specific task demands, thereby optimizing cognitive efficiency (Mesulam, 2000; Li et al., 2022).

[1]Our code is available at https://anonymous.4open.science/status/SEAP-EF0F

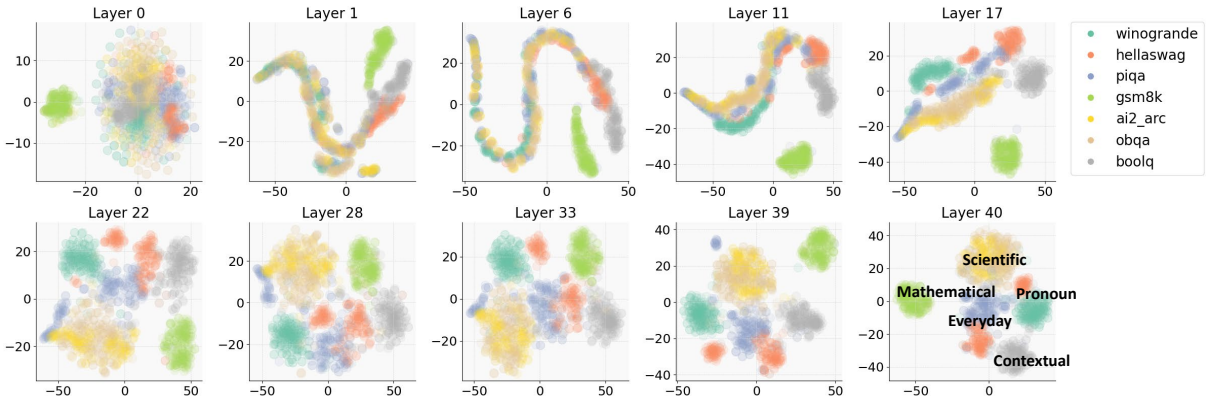

Figure 1: Visualization of hidden states $h(P)$ from different tasks. Each point represents the activation of a hidden state in the model for a specific task. The clustering patterns illustrate how tasks with similar requirements tend to activate similar regions in the model.

Inspired by this principle, we investigate whether a similar mechanism exists in LLMs — where distinct tasks may activate different sets of neurons, forming task-specific computational pathways. This perspective challenges conventional pruning approaches, which typically apply a uniform sparsity pattern across all tasks, potentially overlooking task-dependent knowledge representations.

We hypothesize that the knowledge requirements and activation patterns of different tasks are closely linked. If so, pruning should not be a one-size-fits-all process but rather a dynamic, task-aware strategy. By leveraging this relationship, pruning can adaptively retain the most relevant parameters based on each task's specific characteristics, thereby enhancing computational efficiency while preserving task performance.

To validate this hypothesis, we design a multi-task experiment to analyze whether different tasks induce partitioned representations in LLM hidden states. We select seven task categories, covering a broad spectrum of linguistic and reasoning challenges, including common sense reasoning, mathematical problem-solving, and scientific question answering. We construct task-specific knowledge corpora consisting of question-answer pairs and feed them into an LLM to extract hidden states across multiple layers. The details of the corpus construction process are provided in Section A. To visualize the structure of these hidden states, we project their high-dimensional representations onto a two-dimensional plane. As shown in Figure 1, embeddings are initially intermingled. However, as forward propagation progresses, the model refines the semantic features of the input, leading to increasingly distinct task-specific clusters.

For instance, in the final layer, GSM8K(Cobbe et al., 2021), a challenging mathematical reasoning task, exhibits a clear separation from common-sense reasoning tasks. Similarly, OBQA(Mihaylov et al., 2018) and ARC(Clark et al., 2018), both of which rely heavily on external scientific knowledge, form a closely related distribution. On the other hand, PIQA(Bisk et al., 2020) and HellaSwag(Zellers et al., 2019), though both categorized as common-sense reasoning tasks, emphasize everyday knowledge, positioning them below OBQA and ARC in the visualization. Interestingly, Winogrande(Sakaguchi et al., 2019), a pronoun resolution task, clusters with certain HellaSwag prompts, likely due to their shared reliance on pronoun-based reasoning. Lastly, BoolQ(Clark et al., 2019), a contextual reasoning task, forms a distinct grouping, indicating its unique reliance on contextual comprehension.

These findings suggest that each task occupies a distinct region within the hidden state space, with certain dimensions of activation corresponding to task-specific information. This observation draws an intriguing parallel to the functional specialization of the human brain, where different cognitive processes activate distinct neural circuits.

Building on this insight, we propose our central hypothesis: During inference, leveraging task-specific activation patterns and dynamically selecting the most relevant parameters can significantly reduce computational overhead while maintaining task performance. This task-adaptive pruning paradigm stands in contrast to traditional static

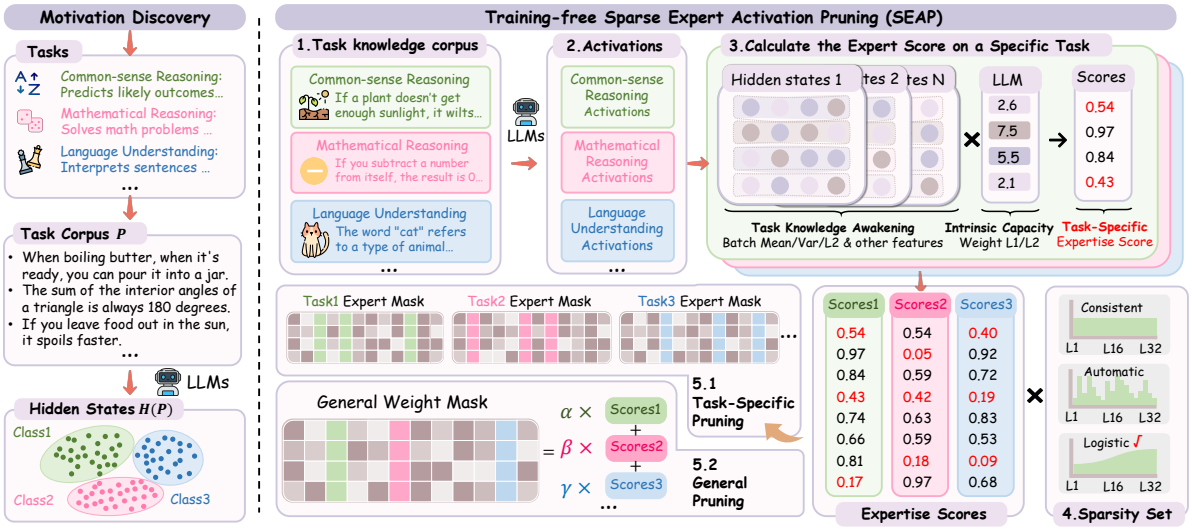

Figure 2: Framework of the SEAP approach. The **left** side shows the **Motivation Discovery** phase, where task-specific activation patterns are identified by analyzing hidden states and neuron activations extracted from the task corpus. The **right** side illustrates the **Training-free Sparse Expert Activation Pruning** process, consisting of five main steps described in Section 2.1.

pruning approaches, offering a promising path toward more efficient and specialized LLM deployment.

Contributions Our key contributions are:

- We analyze task-specific activation patterns in LLMs, revealing their correlation with hidden state distributions and providing new insights for adaptive sparsification.
- We propose SEAP, a training-free, task-adaptive pruning method that dynamically adjusts sparsity based on task type, improving efficiency while preserving performance.
- We demonstrate that SEAP outperforms existing baselines in task accuracy, storage efficiency, and inference speed, confirming its effectiveness for efficient LLM deployment.

2 Method

2.1 Overview of SEAP

Building on the insights from Section 1, we propose Sparse Expert Activation Pruning (SEAP), a training-free, task-adaptive pruning method that selectively activates task-relevant parameters during inference. By dynamically pruning based on task-specific activation patterns, SEAP reduces computational overhead while maintaining model performance. The SEAP Workflow (shown in Figure 2) is as follows,

1. **Task-Specific Knowledge Corpus Construction**: We compile datasets from various tasks, such as reasoning, mathematical problem-solving, and scientific question answering, to form task-specific knowledge corpora (details in Section A).
2. **Activation Patterns Modeling**: We feed the constructed corpora into an LLM and extract hidden state activations from multiple layers to analyze task-specific neural activity. This step lays the foundation for understanding how different tasks engage distinct parameter subsets.
3. **Compute Neuron Importance Scores**: We perform task knowledge awakening by computing features such as mean, variance, and $\ell 2$ norm from the collected activations, which are used to derive the task-specific expertise scores. These scores quantify the relevance of each neuron to the task and serve as the foundation for pruning decisions.
4. **Distribute Sparsity Dynamically**: We introduce a logistic-based sparsity function that dynamically adjusts pruning ratios across layers, retaining critical neurons while maximizing efficiency. This enables structured sparsification tailored to task complexity.
5. **Apply Task-Specific Pruning Strategies**: (5.1) Expert-Based Pruning: Task-specific

expert scores are used to generate pruning masks, allowing the model to dynamically select the most relevant parameters during inference. (5.2) General Pruning: A unified pruning mask is created by aggregating scores across multiple tasks, ensuring broad applicability.

2.2 Activation Patterns Modeling

To validate the feasibility of task-specific expert pruning, we analyze the consistency within task categories and the distinguishability between different task categories in the activations of LLMs. Only when both of these properties are present can the task-specific expert pruning method be effectively applied. To formalize our analysis, let τ denote a task type, and let p_i be a specific prompt within task τ. We denote by

$$\mathbf{h}(p_i)^\tau = \left[h_1(p_i)^\tau,\, h_2(p_i)^\tau,\, \ldots,\, h_C(p_i)^\tau\right] \quad (1)$$

the hidden state vector of dimension C extracted from a particular layer of the model. We further define

$$H^\tau = \left\{\mathbf{h}(p_1)^\tau,\, \mathbf{h}(p_2)^\tau,\, \ldots,\, \mathbf{h}(p_{n_\tau})^\tau\right\}, \quad (2)$$

where n_τ is the number of prompts for task τ.

To quantify how each dimension (often viewed as a "neuron channel") responds under different tasks, we define the following statistical measures that capture the mean activation level, variance, and $\ell 2$ norm of each dimension:

$$\mu_j^\tau = \frac{1}{n_\tau}\sum_{i=1}^{n_\tau} h_j(p_i)^\tau, \quad (3)$$

$$(\sigma_j^\tau)^2 = \frac{1}{n_\tau}\sum_{i=1}^{n_\tau}\left(h_j(p_i)^\tau - \mu_j^\tau\right)^2, \quad (4)$$

$$\overline{\left\|h_j^\tau\right\|_2} = \frac{\left\|h_j^\tau\right\|_2}{n_\tau} = \frac{1}{n_\tau}\sqrt{\sum_{i=1}^{n_\tau}\left(h_j(p_i)^\tau\right)^2}. \quad (5)$$

In these equations, μ_j^τ denotes the mean activation of dimension j for task τ, $(\sigma_j^\tau)^2$ represents its variance, and $\left\|h_j^\tau\right\|_2$ is the total $\ell 2$ norm of that dimension's activations. The normalized measure $\overline{\left\|h_j^\tau\right\|_2}$ divides the raw $\ell 2$ norm by n_τ.

Figure 3 visualizes $\overline{\left\|h_j^\tau\right\|_2}$ for multiple tasks in a given layer or module. Columns represent different dimensions, while rows correspond to either distinct layers or modules. Bright regions indicate higher $\ell 2$ norms, suggesting stronger activation in those dimensions. Within each task, two random subsets of prompts often reveal remarkably consistent high-activation dimensions, indicating internal stability. In contrast, tasks with very different objectives exhibit distinct "hot spots," implying that they engage disjoint sets of dimensions. Hence, these dimension-wise patterns reinforce our claim: tasks of the same type focus on overlapping dimensions, whereas tasks from different domains activate largely separate regions of the hidden state.

Furthermore, consider the weight matrix $W \in \mathbb{R}^{A\times C}$ that applies a linear transformation to the

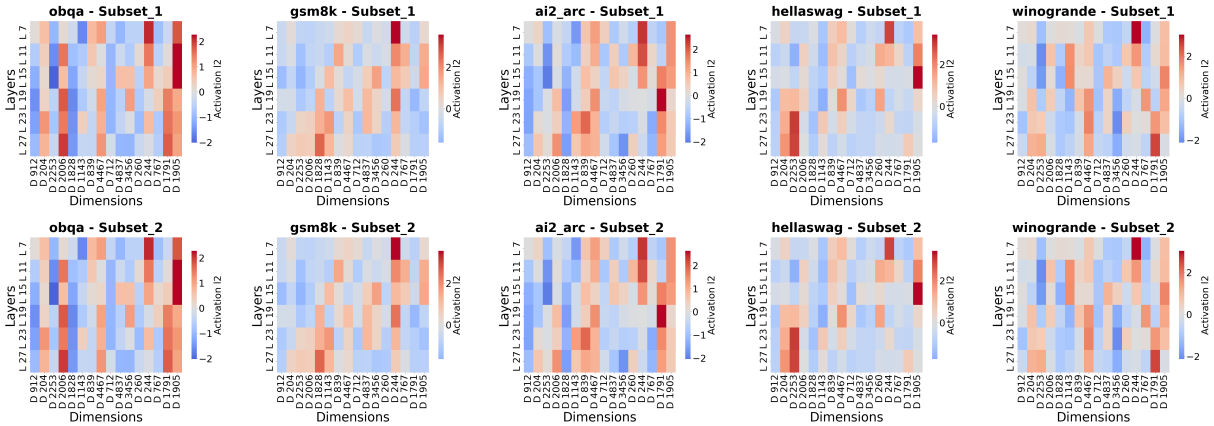

Figure 3: Heatmaps of dimension-wise average $\ell 2$ norms for different tasks. Each row corresponds to a layer or module, and each column represents a dimension in the hidden state space. The top and bottom parts of the figure show activation patterns from two randomly selected subsets of the same task. Consistent color patterns appear within tasks of the same type, while distinctly different tasks exhibit unique activation signatures, supporting our hypothesis that tasks selectively activate specific dimensions.

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 in task τ, the output $\mathbf{o} \in \mathbb{R}^A$ of the linear layer is

$$\mathbf{o} = W\,\mathbf{h}(p_i)^\tau = \begin{bmatrix} w_{1,1} & \cdots & w_{1,C} \\ \vdots & \ddots & \vdots \\ w_{A,1} & \cdots & w_{A,C} \end{bmatrix} \cdot \begin{bmatrix} h_1(p_i)^\tau \\ \vdots \\ h_C(p_i)^\tau \end{bmatrix}, \quad (6)$$

Here, $w_{a,i}$ is the weight linking the i-th hidden dimension to the a-th output unit. 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.

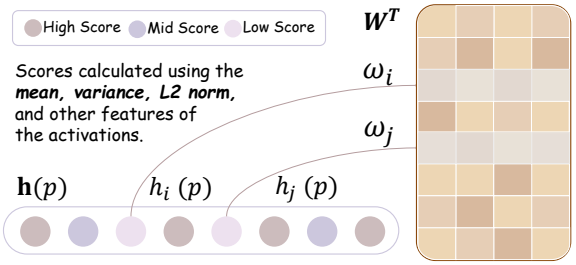

Figure 4: Illustration of how neurons are pruned based on importance scores.

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

2.3 Pruning Procedure

As mentioned, the next step is to prune neurons based on their importance scores for each task. Specifically, the neuron importance $s_i^{(\ell,\tau)}$ for each task is computed using a function $f(\cdot)$, which combines the aforementioned statistical measures with the weight norm of the corresponding neuron in layer ℓ. Formally,

$$s_i^{(\ell,\tau)} = f\Big(\mu_i^{(\ell,\tau)},\ \big(\sigma_i^{(\ell,\tau)}\big)^2,\ \overline{\|h_i^{(\ell,\tau)}\|}_2,\ w_i^{(\ell)}\Big), \quad (7)$$

where $\mu_i^{(\ell,\tau)}$ and $\big(\sigma_i^{(\ell,\tau)}\big)^2$ denote the mean and variance of the activations for neuron i in layer ℓ under task τ, $\overline{\|h_i^{(\ell,\tau)}\|}_2$ represents the average $\ell 2$ norm of its activations, and $w_i^{(\ell)}$ is the corresponding weight in layer ℓ.

Once these importance values are obtained for all C neurons in the chosen module of layer ℓ, we collect $\{|s_1^{(\ell,\tau)}|, \ldots, |s_C^{(\ell,\tau)}|\}$ and sort them in ascending order:

$$|s^{(\ell,\tau)}|_{\text{sorted}} = \text{Sort}\Big(|s_1^{(\ell,\tau)}|, \ldots, |s_C^{(\ell,\tau)}|\Big). \quad (8)$$

Given a desired sparsity ratio $\rho \in [0,1]$, we identify the ρ-th quantile in (8):

$$\theta^{(\ell,\tau)} = |s^{(\ell,\tau)}|_{\text{sorted}}\big(\lfloor \rho\, C \rfloor\big), \quad (9)$$

where $\lfloor \cdot \rfloor$ is the floor function. All neurons whose importance $|s_i^{(\ell,\tau)}|$ is less than or equal to $\theta^{(\ell,\tau)}$ are then pruned:

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

Here, $w_i^{(\ell)} = 0$ effectively disables neuron i.

2.4 Scores Calculating Strategy

The expert activation pruning framework we propose is highly flexible, capable of accommodating various neuron importance metrics. The importance score $s_i^{(\ell,\tau)}$ can be easily integrated with existing training-free methods, such as **FLAP**(An et al., 2024) and **WandA**(Sun et al., 2024), by simply adjusting their respective formulas.

In the framework above, each neuron i in layer ℓ under task τ is assigned an importance score $s_i^{(\ell,\tau)}$ by combining its activation statistics and weight information. Specifically, we rely on the definitions of mean activation, variance, and total activation energy (i.e., squared ℓ_2-norm) from equations (3), (4), and (5) in Section 2.2, respectively. Let $w_i^{(\ell)} \in \mathbb{R}^{D_\ell}$ be the weight vector corresponding to neuron i in layer ℓ, with $\|w_i^{(\ell)}\|_2$ and $\|w_i^{(\ell)}\|_1$ denoting its ℓ_2- and ℓ_1-norms.

Based on these quantities, we introduce two specific scoring functions, s_F and s_W. The first, s_F, follows the scoring method used in FLAP by multiplying the neuron's variance with the squared ℓ_2-norm of its weights:

$$s_{F,i}^{(\ell,\tau)} = (\sigma_i^{(\ell,\tau)})^2 \times \big\|w_i^{(\ell)}\big\|_2^2, \quad (11)$$

This gives higher importance to neurons whose activations vary significantly across prompts and whose weight magnitudes are relatively large.

The second, s_W, follows the scoring method used in WandA, replacing the variance with the total activation energy (i.e., the squared ℓ_2-norm of the neuron's activations) and weighting it by $\|w_i^{(\ell)}\|_1$:

$$s_{W,i}^{(\ell,\tau)} = \|h_i^{(\ell,\tau)}\|_2^2 \times \|w_i^{(\ell)}\|_1. \quad (12)$$

After computing s_F or s_W for all neurons, we apply the threshold-based pruning procedure described in Section 2.3 to remove those receiving lower scores.

In the MLP layers, each neuron index i directly corresponds to a single channel, so the scores s_F or s_W in (11)–(12) apply on a channel-by-channel basis. By contrast, in the attention layers, we aggregate neuron scores at the head level: suppose each attention head h spans a contiguous set of dimensions $\mathcal{I}_h$, then its overall importance score can be taken as

$$s_h^{(\ell,\tau)} = \sum_{i \in \mathcal{I}_h} s_i^{(\ell,\tau)}. \quad (13)$$

A threshold is then applied to $s_h^{(\ell,\tau)}$ to prune or retain the entire head.

2.5 Expert-Based vs General Pruning

We propose two pruning strategies to adapt to task-specific scenarios and general scenarios. The first strategy, **Expert-based Pruning**, uses task-specific importance scores to prune neurons or attention heads. The pruning process selects the importance score $s_i^{(\ell)}$ for each neuron i in layer ℓ based on the task type τ_{chosen} as follows:

$$s_i^{(\ell)} = s_i^{(\ell,\tau_{\text{chosen}})}, \quad (14)$$

where τ_{chosen} is the task selected for pruning, and $s_i^{(\ell,\tau_{\text{chosen}})}$ is the corresponding importance score. During inference, different pruning masks can be flexibly applied based on the task type.

The second strategy, **General Pruning**, integrates importance scores across multiple tasks to identify neurons or attention heads that are less important across all tasks. This general pruning approach forms a unified model, ensuring that important components are retained across a broader range of tasks. The score is computed as a weighted average of the importance scores from each task:

$$s_i^{(\ell)} = \sum_{\tau} \alpha_\tau s_i^{(\ell,\tau)}, \quad (15)$$

where α_τ is the weight assigned to task τ, and the sum is taken across all tasks.

2.6 Sparsity Setting

To determine appropriate sparsity levels for each layer in LLMs, we conduct a *remove test* on two tasks: MMLU(Hendrycks et al., 2021) and PIQA. This test prunes neurons across layers at varying sparsity levels and measures task performance. Figures 5 and 6 show that early layers are more sensitive to pruning, while deeper layers tolerate higher sparsity with minimal performance loss, consistent with our observations in Section 1.

Additionally, LLM-Pruner(Ma et al., 2023) and FLAP methods highlight that layers near the output are crucial for language modeling. Thus, we set the sparsity of the final n layers to zero and adjust the sparsity of other layers to maintain overall sparsity. For sparsity setting, we employ a differentiable

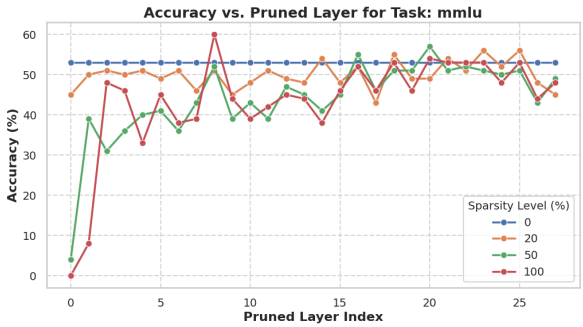

Figure 5: Impact of pruning on MMLU performance at different layers and sparsity levels. Early layers are more sensitive to pruning.

logistic function to ensure a smooth and continuous distribution of sparsity across layers. Each layer index ℓ is mapped to the interval $[0, 1]$ using $x_\ell = \frac{\ell-1}{L-1}$, where L is the total number of layers. The sparsity for layer ℓ is defined as:

$$\rho_\ell = \rho(x_\ell) = \Lambda \frac{1}{1 + \exp\big(-k(x_\ell - x_0)\big)}, \quad (16)$$

where k controls the steepness, x_0 sets the inflection point, and Λ represents the maximum sparsity. This ensures lower sparsity in early layers and progressively higher sparsity in deeper layers.

To meet a global sparsity target G, we adjust Λ so that the average sparsity satisfies:

$$\frac{1}{L} \sum_{\ell=1}^{L} \rho_\ell = G. \quad (17)$$

This is done via a numerical search for Λ. In our experiments, we use $(x_0, k) = (0.3, 1)$.

Pruning Ratio	Method	Llama-2-7B							
		WinoGrande	OBQA	HellaSwag	PIQA	ARC-c	ARC-e	BoolQ	Average
0%	Dense	69.14	44.20	76.01	79.11	46.33	76.26	77.71	66.97
20%	WandA-sp (s_W)	62.67	40.80	71.56	76.28	42.41	71.93	61.01	60.95
	SEAP (s_W)	66.77	**43.00**	72.53	77.80	**45.48**	**75.42**	71.77	64.68
	SEAP-gen (s_W)	67.80	41.00	73.77	77.58	44.71	74.33	71.44	64.38
	FLAP (s_F)	67.32	41.00	72.77	76.12	42.75	71.93	62.57	62.07
	SEAP (s_F)	**68.19**	42.60	74.07	78.07	45.39	**75.42**	**74.50**	**65.46**
	SEAP-gen (s_F)	67.72	41.20	**74.82**	**78.35**	45.39	74.12	71.68	64.75
50%	WandA-sp (s_W)	52.72	35.20	41.11	64.36	30.97	52.78	39.45	45.23
	SEAP (s_W)	56.12	37.20	58.07	73.83	**38.74**	**61.32**	**60.15**	55.06
	SEAP-gen (s_W)	54.70	38.20	56.97	71.76	35.24	57.15	57.25	53.04
	FLAP (s_F)	56.04	34.40	48.62	63.00	32.17	51.18	42.32	46.82
	SEAP (s_F)	**60.14**	38.80	**58.22**	**74.32**	38.14	60.56	59.94	**55.73**
	SEAP-gen (s_F)	59.91	**39.80**	58.17	73.39	37.97	55.72	57.98	54.71

Table 1: Task performance accuracy on Llama-2-7B under different pruning ratios. A higher ↑ score indicates better performance. The **bolded** entries represent the highest scoring methods, while the underlined entries represent the second highest scoring methods.

3 Experiment and Results Analysis

3.1 Experimental Settings

LLMs and Tasks We evaluate our method on the Llama2-7B and Llama2-13B models, assessing their performance across a range of downstream tasks. Zero-shot performance is evaluated on seven benchmarks—BoolQ, ARC Easy, ARC Challenge, HellaSwag, OBQA, PiQA, and Winogrande—using the EleutherAI LM Harness(Gao et al., 2024). In addition to accuracy, we also compare inference speed. More details can be found in Section A.

Baselines We compare our method to the original (dense) models and two established training-free sparsification methods: WandA and FLAP. The key difference between these baselines is in the importance score calculation. For the expert-based and general models, we use the scoring methods s_W from WandA and s_F from FLAP, respectively. All methods, including baselines, adopt the logistic sparsity setting proposed in this paper, enabling consistent comparison of knowledge corpus expert activation differences. A detailed comparison of sparsity settings is provided in Section 3.2.

3.2 Results and Analysis

Zero-shot Tasks Performance We evaluate SEAP's zero-shot performance across multiple benchmarks, demonstrating its ability to reduce computational overhead while maintaining competitive accuracy. For Llama-2-7B (see Table 1), at 20% pruning, SEAP outperforms both WandA and FLAP with minimal performance loss, showing only a 2.2% drop compared to the dense model, which is exceptional for structured pruning. At 50% pruning, SEAP's advantage over FLAP and WandA increases, with the average score surpassing both baselines by over 20%, indicating strong performance even at high sparsity levels.

Interestingly, the results do not always align with expectations for general versus expert models. In HellaSwag, the general model outperforms the expert model, likely due to richer knowledge corpora enhancing task-relevant activation distributions. A similar trend is observed in BoolQ with Llama-2-13B (see Table 4), where higher sparsity leads to a noticeable performance drop, possibly due to the simpler True/False nature of the task, which lacks a sufficiently rich knowledge corpus for task-specific pruning to be fully effective.

Overall, our results confirm that task-specific pruning improves efficiency without compromising performance.

Inference Speed Our pruning method completes pruning on Llama-2-7B in approximately 5–10 minutes on a single NVIDIA H800 80GB GPU. As shown in Table 2, SEAP significantly improves inference speed compared to non-structured pruning methods like WandA. At 20% pruning, SEAP is slightly slower than FLAP, but at 50% pruning, SEAP maintains high speed with only a minimal difference compared to FLAP. These results demonstrate that SEAP reduces computational resources while maintaining high inference speed, making it

suitable for real-world deployment across various hardware environments.

Ratio	Method	Llama-2-7B		Llama-2-13B	
		Tokens/s	Up	Tokens/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-gen	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-gen	47.10	×1.48	41.78	×1.52

Table 2: Inference speed (Tokens per second) and speedup under different pruning ratios.A higher↑ speed indicates better performance.

Sparsity Setting Comparison As shown in Table 3, we compare our Logistic-based (LB) sparsity setting with other strategies: Uniform Sparsity across Layers (UL) and Adaptive Sparsity across Layers and Modules (AL) from FLAP. At both 20% and 50% pruning ratios, our method (SEAP-gen with LB) consistently outperforms WandA-sp and FLAP in terms of performance, demonstrating that the LB setting leads to more efficient resource allocation and better performance.

Ratio	Method	Set.	Average	Set.	Average
0%	Dense	-	69.46	-	69.46
20%	WandA-sp	UL	61.47	LB	**65.57**
	FLAP	AL	63.03	LB	**66.76**
	SEAP-gen	UL	66.03	LB	**68.75**
50%	WandA-sp	UL	48.80	LB	**49.94**
	FLAP	AL	51.12	LB	**51.78**
	SEAP-gen	UL	59.03	LB	**60.89**

Table 3: Sparsity settings and average sparsity on the Llama-2-13B model. The table shows three sparsity strategies: "UL" (Uniform Layer Sparsity), "LB" (Logistic-based Sparsity), and "AL" (Adaptive Layer Sparsity).A higher↑ score indicates better performance.

4 Related Works

The computational cost and inference time of LLMs significantly impact deployment. Researchers have addressed these challenges through model compression(Michel et al., 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).

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 et al., 2023), FLAP (An et al., 2024), and Depth2 (Li et al., 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., 2020b), 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 et al., 2024), QSparse (Wang et al., 2024), and ProSparse (Song et al., 2024a) achieve training-free activation pruning.

5 Conclusion

We present **SEAP (Sparse Expert Activation Pruning)**, a training-free, task-adaptive pruning framework for LLMs, inspired by the clustering of hidden states and task-specific activation patterns. SEAP dynamically selects and activates the most relevant neurons for each task, reducing computational overhead while maintaining strong task performance. Extensive experiments demonstrate that SEAP significantly improves efficiency—outperforming baselines by over 20% at 50% pruning, while maintaining over 97.8% of the original performance at 20% pruning. These results highlight SEAP's ability to achieve substantial sparsification with minimal performance degradation. By leveraging insights from hidden state clustering and activation-driven pruning, SEAP optimizes LLMs for real-world deployment, enabling more efficient, scalable, and adaptive language models. This approach paves the way for future advancements in structured pruning and task-aware model compression, making LLMs more accessible and practical across diverse applications.

Ethical Considerations

This work introduces a pruning method for LLMs to improve efficiency, but it raises ethical concerns. Pruning decisions could unintentionally affect model performance or fairness on certain tasks. While our method aims to preserve task-specific performance, it is important to monitor its impact on fairness and utility, especially in critical applications. Furthermore, pruned models could have unintended consequences in domains requiring nuanced decision-making. Transparent deployment and ongoing evaluation are essential to address these concerns.

Limitations

While SEAP improves inference efficiency, it does have some limitations. (1) Compared to other methods, our approach may result in a slight increase in perplexity, as it preserves task-specific parameters at the cost of some efficiency. (2) The acquisition of task-specific activation values could also benefit from more diverse data, and incorporating data synthesis techniques could improve model generalization. (3) Lastly, pairing SEAP with a simple task classifier to route tasks to the pruned model could further enhance efficiency, making the approach more adaptable in practical applications.

References

Yongqi An, Xu Zhao, Tao Yu, Ming Tang, and Jinqiao Wang. 2024. Fluctuation-based adaptive structured pruning for large language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(10):10865–10873.

Haoli Bai, Wei Zhang, Lu Hou, Lifeng Shang, Jin Jin, Xin Jiang, Qun Liu, Michael Lyu, and Irwin King. 2021. BinaryBERT: Pushing the limit of BERT quantization. In *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)*, pages 4334–4348, Online. Association for Computational Linguistics.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*.

Arnav Chavan, Raghav Magazine, Shubham Kushwaha, Mérouane Debbah, and Deepak Gupta. 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.

Vui Seng Chua, Yujie Pan, and Nilesh Jain. 2024. Post-training statistical calibration for higher activation sparsity. *Preprint*, arXiv:2412.07174.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

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:1803.05457v1*.

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

Lucio Dery, Steven Kolawole, Jean-François Kagy, Virginia Smith, Graham Neubig, and Ameet Talwalkar. 2024. Everybody prune now: Structured pruning of llms with only forward passes. *Preprint*, arXiv:2402.05406.

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

Elias Frantar and Dan Alistarh. 2023. Sparsegpt: Massive language models can be accurately pruned in one-shot. *Preprint*, arXiv:2301.00774.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2023. OPTQ: Accurate quantization for generative pre-trained transformers. In *The Eleventh International Conference on Learning Representations*.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2024. A framework for few-shot language model evaluation.

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

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

Donghyun Lee, Je-Yong Lee, Genghan Zhang, Mo Tiwari, and Azalia Mirhoseini. 2024. Cats: Contextually-aware thresholding for sparsity in large language models. *Preprint*, arXiv:2404.08763.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2020. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*.

Mike Lewis, Shruti Bhosale, Tim Dettmers, Naman Goyal, and Luke Zettlemoyer. 2021. Base layers: Simplifying training of large, sparse models. In *International Conference on Machine Learning*, pages 6265–6274. PMLR.

Jianwei Li, Yijun Dong, and Qi Lei. 2024. Greedy output approximation: Towards efficient structured pruning for llms without retraining. *Preprint*, arXiv:2407.19126.

Yu Li, Aiping Liu, Xueyang Fu, Martin J. Mckeown, Z. Jane Wang, and Xun Chen. 2022. Atlas-guided parcellation: Individualized functionally-homogenous parcellation in cerebral cortex. *Computers in Biology and Medicine*, 150:106078.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2024. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of Machine Learning and Systems*, 6:87–100.

James Liu, Pragaash Ponnusamy, Tianle Cai, Han Guo, Yoon Kim, and Ben Athiwaratkun. 2024. Training-free activation sparsity in large language models. *Preprint*, arXiv:2408.14690.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. Llm-pruner: On the structural pruning of large language models. In *Advances in Neural Information Processing Systems*.

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

M Marsel Mesulam. 2000. *Principles of Behavioral and Cognitive Neurology*. Oxford University Press.

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

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

Iman Mirzadeh, Keivan Alizadeh, Sachin Mehta, Carlo C Del Mundo, Oncel Tuzel, Golnoosh Samei, Mohammad Rastegari, and Mehrdad Farajtabar. 2023. Relu strikes back: Exploiting activation sparsity in large language models. *arXiv preprint arXiv:2310.04564*.

Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Leon Derczynski, Xingjian Du, Matteo Grella, Kranthi Gv, Xuzheng He, Haowen Hou, Przemyslaw Kazienko, Jan Kocon, Jiaming Kong, Bartłomiej Koptyra, Hayden Lau, Jiaju Lin, Krishna Sri Ipsit Mantri, Ferdinand Mom, Atsushi Saito, Guangyu Song, Xiangru Tang, Johan Wind, Stanisław Woźniak, Zhenyuan Zhang, Qinghua Zhou, Jian Zhu, and Rui-Jie Zhu. 2023. RWKV: Reinventing RNNs for the transformer era. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 14048–14077, Singapore. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020a. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020b. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. Winogrande: An adversarial winograd schema challenge at scale. *Preprint*, arXiv:1907.10641.

Noam Shazeer, *Azalia Mirhoseini, *Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*.

David So, Wojciech Mańke, Hanxiao Liu, Zihang Dai, Noam Shazeer, and Quoc V Le. 2021. Searching for efficient transformers for language modeling. In *Advances in Neural Information Processing Systems*.

Chenyang Song, Xu Han, Zhengyan Zhang, Shengding Hu, Xiyu Shi, Kuai Li, Chen Chen, Zhiyuan Liu, Guangli Li, Tao Yang, et al. 2024a. Prosparse: Introducing and enhancing intrinsic activation sparsity within large language models. *arXiv preprint arXiv:2402.13516*.

Yixin Song, Haotong Xie, Zhengyan Zhang, Bo Wen, Li Ma, Zeyu Mi, and Haibo Chen. 2024b. Turbo sparse: Achieving llm sota performance with minimal activated parameters. *arXiv preprint arXiv:2406.05955*.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. 2024. A simple and effective pruning approach for large language models. In *The Twelfth International Conference on Learning Representations*.

Hongyu Wang, Shuming Ma, Ruiping Wang, and Furu Wei. 2024. Q-sparse: All large language models can be fully sparsely-activated. *arXiv preprint arXiv:2407.10969*.

Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. 2022. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. *Advances in Neural Information Processing Systems*, 35:27168–27183.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Zhengyan Zhang, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. 2022. MoEfication: Transformer feed-forward layers are mixtures of experts. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 877–890, Dublin, Ireland. Association for Computational Linguistics.

Zhengyan Zhang, Yixin Song, Guanghui Yu, Xu Han, Yankai Lin, Chaojun Xiao, Chenyang Song, Zhiyuan Liu, Zeyu Mi, and Maosong Sun. 2024. ReLU^2 wins: Discovering efficient activation functions. *arXiv preprint arXiv:2402.03804*.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2024. A survey of large language models. *Preprint*, arXiv:2303.18223.

Zifan Zheng, Yezhaohui Wang, Yuxin Huang, Shichao Song, Mingchuan Yang, Bo Tang, Feiyu Xiong, and Zhiyu Li. 2025. Attention heads of large language models. *Patterns*, 6(2).

Changhai Zhou, Yuhua Zhou, Shijie Han, Qian Qiao, and Hongguang Li. 2024. Qpruner: Probabilistic decision quantization for structured pruning in large language models. *Preprint*, arXiv:2412.11629.

Pruning Ratio	Method	Llama-2-13B							
		WinoGrande	OBQA	HellaSwag	PIQA	ARC-c	ARC-e	BoolQ	Average
0%	Dense	72.14	45.20	79.37	80.52	48.98	79.42	80.58	69.46
20%	WandA-sp (s_W)	67.40	42.80	74.52	78.40	48.64	76.73	70.49	65.57
	SEAP (s_W)	**71.98**	43.60	78.73	**80.69**	48.46	77.61	74.68	67.96
	SEAP-gen (s_W)	69.85	43.20	78.13	80.47	48.55	**78.58**	72.54	67.33
	FLAP (s_F)	69.14	44.00	75.05	76.71	48.04	77.19	77.22	66.76
	SEAP (s_F)	70.64	**44.80**	**79.12**	**80.69**	47.95	76.85	76.82	68.12
	SEAP-gen (s_F)	70.09	44.20	78.97	80.09	**50.17**	78.37	**79.36**	**68.75**
50%	WandA-sp (s_W)	53.51	37.20	46.77	66.97	35.24	60.14	49.76	49.94
	SEAP (s_W)	58.96	40.60	66.91	76.77	44.03	71.09	57.43	59.40
	SEAP-gen (s_W)	63.38	**44.40**	66.75	76.55	43.43	71.09	49.79	59.34
	FLAP (s_F)	55.17	38.20	53.82	67.41	33.11	58.42	56.36	51.78
	SEAP (s_F)	**64.56**	42.00	**68.75**	76.93	**45.05**	**71.84**	52.57	60.24
	SEAP-gen (s_F)	62.59	43.20	67.05	**77.15**	41.55	67.93	**66.79**	**60.89**

Table 4: Task performance accuracy on Llama-2-13B under different pruning ratios. A higher ↑ score indicates better performance. The **bolded** entries represent the highest scoring methods, while the underlined entries represent the second highest scoring methods.

A Experimental Settings

A.1 Task-Specific Corpus Construction

In this study, we constructed a standardized task-specific corpus by reformatting the questions and answers from evaluation tasks into knowledge-rich inputs.

For each task, we began by extracting relevant components from the raw training data, including the question, answer options, and correct answers. These components were then formatted into standardized input prompts. By combining the question, options, and correct answer into a unified input format, we provided the model with the full context of each task, as shown in Table5. This structured input allows the model to learn task-specific patterns and understand the relationship between the question and the correct answer, ultimately improving its ability to make accurate predictions.

A.2 Tasks

For evaluating downstream task performance, we use the lm-eval harness(Gao et al., 2024) to assess zero-shot performance across seven benchmark tasks. We ensured that all tools and datasets used are properly cited, comply with their licenses and intended uses, and meet ethical standards, including data privacy and documentation. These tasks test a wide range of natural language understanding challenges and include:

- **BoolQ** (Clark et al., 2019): Evaluates models' ability to answer yes/no questions based on context, testing comprehension and reasoning.
- **ARC Easy and ARC Challenge** (Clark et al., 2018): Benchmarks from the AI2 Reasoning Challenge assessing reasoning on multiple-choice science questions; Easy set for direct retrieval, Challenge set for complex reasoning.
- **HellaSwag** (Zellers et al., 2019): Tests commonsense reasoning by having models select the most plausible continuation of a given sentence.
- **OBQA** (Mihaylov et al., 2018): An open-book question answering task assessing models' ability to answer factual questions using a collection of documents.
- **PiQA** (Bisk et al., 2020): Focuses on physical commonsense reasoning, requiring models to select the correct solution from two choices for a given problem.
- **Winogrande** (Sakaguchi et al., 2019): A large-scale dataset designed to evaluate models' ability to resolve commonsense reasoning tasks in the style of the Winograd Schema Challenge.

These tasks cover a broad spectrum of natural language understanding, from reasoning and commonsense knowledge to factual and situational understanding.

Task	Example Prompt
HellaSwag	Then, the man writes over the snow covering the window of a car, and a woman wearing winter clothes smiles. Then, the man continues removing the snow on his car.
PIQA	How do I ready a guinea pig cage for its new occupants? Provide the guinea pig with a cage full of a few inches of bedding made of ripped paper strips, you will also need to supply it with a water bottle and a food dish.
OBQA	The sun is the source of energy for physical cycles on Earth: plants sprouting, blooming, and wilting.
WinoGrande	Katrina had the financial means to afford a new car while Monica did not, since Katrina had a high paying job.
ARC	One year, the oak trees in a park began producing more acorns than usual. The next year, the population of chipmunks in the park also increased. Which best explains why there were more chipmunks the next year? Food sources increased.
GSM8K	Natalia sold clips to 48 of her friends in April, and then she sold half as many clips in May. How many clips did Natalia sell altogether in April and May? Natalia sold 48/2 = 24 clips in May. Natalia sold 48 + 24 = 72 clips altogether in April and May.
BoolQ	All biomass goes through at least some of these steps: it needs to be grown, collected, dried, fermented, distilled, and burned... Does ethanol take more energy to make than it produces? False

Table 5: Example Prompts from Various Tasks in the Task-Specific Corpus

A.3 Baselines

In this study, we select two representative methods as baseline models for comparison: Wanda and FLAP. Below is a detailed introduction to these two methods.

Wanda (Sun et al., 2024) Wanda evaluates parameter importance by calculating the product of the weight magnitude and the ℓ_2-norm of the corresponding input activation. It adopts a local pruning strategy, pruning weights associated with each output feature within a linear layer. We extend Wanda to structured pruning by computing the ℓ_2-norm of weight groups within the linear layer, evaluating the importance of the entire group. This extended version, called Wanda-sp, enables structured pruning in large language models.

FLAP (An et al., 2024) FLAP (Fluctuation-based Adaptive Structured Pruning) is a novel structured pruning method for large language models, achieving compression without retraining. It uses a fluctuation pruning metric to assess the recoverability of the output feature map after removing a column of weights. By normalizing importance scores, FLAP adaptively determines the global structure of the compressed model.

A.4 Hyperparameters

The hyperparameters in this study involve the weighting of tasks and the sparsity setting across layers.

For general pruning, the importance score $s_i^{(\ell)}$ for each neuron in layer ℓ is calculated as a weighted sum of task-specific scores:

$$s_i^{(\ell)} = \sum_{\tau} \alpha_\tau s_i^{(\ell,\tau)},$$

where α_τ is the weight assigned to task τ. WikiText2 is assigned a weight of 3 as an expert activation for language modeling, while other tasks are assigned an equal weight of 2.

To achieve a global sparsity target G, we adjust the sparsity distribution across layers by tuning the parameter Λ such that the average sparsity satisfies:

$$\frac{1}{L}\sum_{\ell=1}^{L} \rho_\ell = G.$$

This is done through a numerical search for the optimal Λ. In our experiments, we use $(x_0, k) = (0.3, 1)$ for the logistic sparsity function.

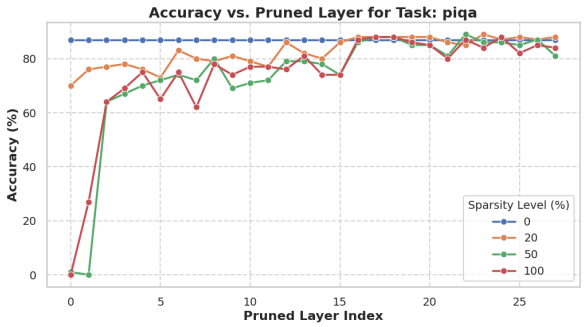

Figure 6: Impact of pruning on PIQA performance at different layers and sparsity levels. Deeper layers are more robust to pruning.

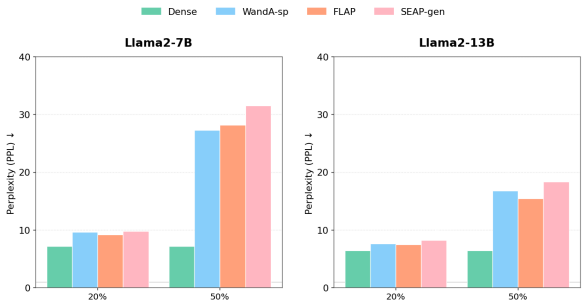

Figure 7: Perplexity (PPL) results under different pruning ratios. A lower↓ perplexity indicates better performance.

B Additional Experiments

In this section, we present two additional experiments to support our proposed method. These experiments are designed to assess key aspects of the model's performance: perplexity as a measure of language modeling quality and task classification for task-specific pruning.

B.1 Perplexity

We evaluate the impact of pruning on language modeling by assessing perplexity (PPL) on the WikiText2 dataset. Perplexity measures how well a model predicts the next word in a sequence, with lower values indicating better performance. This experiment helps determine whether pruning methods, including SEAP, can maintain language generation quality while achieving computational savings.

We use 128 random samples from the WikiText2 dataset (Merity et al., 2016), each with a 2048-token context and a 512-token evaluation window, following the FLAP setup (An et al., 2024). As shown in Figure 7, at 20% sparsity, SEAP leads to a slight increase in perplexity compared to WandA-sp and FLAP, reflecting a small trade-off in language modeling quality. At 50% sparsity, perplexity increases across all methods, with SEAP-gen showing the highest values. However, these increases remain within an acceptable range, especially considering the significant improvements in task-specific performance.

B.2 Task Classifier

A key feature of the task-specific expert activation pruning method is its ability to dynamically select pruning masks based on the task type, improving computational efficiency. The challenge lies in quickly identifying the task type with minimal overhead to ensure efficient mask selection.

To address this, we propose a lightweight task classification method. We extract a vector from the model's 0th-layer embedding and train a single-layer classifier to identify the task type, enabling the model to select the appropriate task-specific mask with minimal cost.

Class	Precision	Recall	F1-Score	Support
hellaswag	0.94	0.89	0.91	236
gsm8k	0.92	0.99	0.95	233
winogrande	0.98	0.99	0.98	218
piqa	0.88	0.91	0.89	201
mmlu	0.95	0.87	0.91	233
ai2_arc	0.91	0.95	0.93	222
Accuracy			0.93 (1343)	
Macro avg	0.93	0.93	0.93	1343
Weighted avg	0.93	0.93	0.93	1343

Table 6: Task classification performance metrics for the proposed task classifier. Precision, recall, and F1-score are reported for each task class, along with the overall accuracy, macro average, and weighted average.

As shown in Table 6, the task classifier performs effectively, distinguishing between different task types with minimal computational resources. This confirms that task classification can be done rapidly, allowing for efficient pruning and dynamic adjustment of model parameters.

C Generation Examples

In this section, we provide several examples of generated outputs to illustrate the effectiveness of our method. The following tables showcase two results of our model's generation capabilities using prompts from the FLAP: Table 7 and Table 8.

Model	Generated Sentences
LLaMA-7B 20% Pruned	AI can create a logo in seconds. But it's not just a logo that you need to build a brand. A brand is a collection of ideas, emotions, and associations that people have about a product, service, or company. It's the sum of all the things that make up a brand, and it's what differentiates one brand from another. The most important thing to remember about branding is that it's not just about the logo.
	What is McDonald's? McDonald's is a fast-food restaurant franchise that serves burgers, fries, chicken nuggets, soft drinks, milkshakes, sundaes, salads, wraps, breakfast items (in some restaurants), desserts, and other items. The menu varies by country and region, but there are over 30,000 locations worldwide.
LLaMA-7B 50% Pruned	AI can create a logo in seconds. The process of creating logotypes is called typestudy. A Log is a handwritten symbol or group of alphabets used in a script. In 1915, Eric Blake created the typeface Helvetica. It was a revival of handwriting from the Italian Renaissance. The first known example of handwriting is from a cave in Georgia, USA. It is dated 10,000 BCE.
	What is McDonald's? It is a small inn in an alley, a hundred yards or two from the gates. The tavern's walls are thick, and it has a steeply pitched roof. Above the door there is carved a dragon's flicked beak, with the words _Bow of Arrows_ carved in the sill. Inside, the tavern is large and well lit by daylight. There are three stairways leading to the north, all of which are occupied by men in armor.

Table 7: Generated Sentences by LLaMA-7B with Different Pruning Levels

Model	Generated Sentences
LLaMA-13B 20% Pruned	"AI can create a logo in seconds. But it's not that simple. There are pros and cons to AI-generated logos. On the one hand, they are quick to create and don't require a lot of time or resources. On the other hand, they can be generic and lack personality. In this article, we'll explore the pros and cons of AI-generated logos and how they compare to human-designed logos. "
	"What is McDonald's? McDonald's is a fast-food hamburger restaurant that serves burgers, fries, chicken nuggets, milkshakes, salads, and breakfast items. The menu is simple and affordable, and the restaurant is known for its cleanliness and friendliness. What is the history of McDonald's? In 1940, two brothers, Ray Kroc and Richard McDonald, opened the first McDonald's restaurant in Des Plaines, Illinois."
LLaMA-13B 50% Pruned	"AI can create a logo in seconds. It's a matter of fact that the time to create a logo has decreased from 10 days to 24 hours in the last decade. This is due to the development of computer graphics and digital technologies. In the 1990s, the world's first computer-animated film "Turtle Island" was released in 1990. It took three years to make the film and cost $40 million. The sequel of the franchise, Taz the Stone Age, was released in 1994. It made $ 24 million in box office and gross revenue of $402 million worldwide."
	"What is McDonald's? Founded in 1946 by Mac and his brother Dave McDonald in Aberdeen, Scotland as a bar for American servicemen and their friends, the Macdonald family took over the business in 1972. It was renamed The White Rose in 1974 and changed to its current name in 1986 due to the unfortunate similarity of McDonald's which was registered at the time some 30 miles away. Today the pub serves 1200 pints a week and has live music 6 nights a week with DJ's playing between breaks."

Table 8: Generated Sentences by LLaMA-13B with Different Pruning Levels