😮 STUN: Structured-Then-Unstructured Pruning for Scalable MoE Pruning

Jaeseong Lee[*], **Seung-won Hwang**[*], **Aurick Qiao**,
Daniel Campos, **Zhewei Yao**, **Yuxiong He**
Snowflake AI Research, Seoul National University[*]

Abstract

Mixture-of-experts (MoEs) have been adopted to reduce inference costs by sparsely activating experts in large language models (LLMs). Despite these reductions, the massive number of parameters in MoEs still makes them expensive to serve. Conventionally, unstructured or structured pruning has been considered to reduce the number of parameters. Our key contribution is exploring the interpolation between structured and unstructured pruning, to propose a novel structured-then-unstructured (STUN) approach outperforming both structured and unstructured pruning, especially for MoEs. In the first stage, we show a scalable expert pruning with O(1) forward pass, unlike existing work requiring $O(\frac{k^n}{\sqrt{n}})$ forward passes for n experts that cannot scale for recent MoEs with hundreds of experts. We then show our expert-pruned MoEs are robust to unstructured pruning to follow. Experiments on Snowflake Arctic and Mixtral show that our proposal is highly effective– For Snowflake Arctic, a 480B-sized MoE with 128 experts, our method needs only one H100 and two hours to achieve nearly no loss in performance with 40% sparsity, even in generative tasks such as GSM8K, where state-of-the-art structured or unstructured pruning methods fail. The code is publicly available.[1]

1 Introduction

Large language models (LLMs) have become state-of-the-art for various tasks (OpenAI, 2023; Touvron et al., 2023; Jiang et al., 2023; Lieber et al., 2024). However, their prohibitive inference cost is becoming a bottleneck to deployment (Kaddour et al., 2023), and detrimental to the environment (Strubell et al., 2019; Zeng et al., 2023).

Mixture-of-experts (MoE) presents a promising alternative, by sparsely activating a specific subset of parameters, named as experts, to reduce the inference cost. This architecture has been empirically proven effective, in training cost (Fedus et al., 2022), and inference cost (Du et al., 2022).

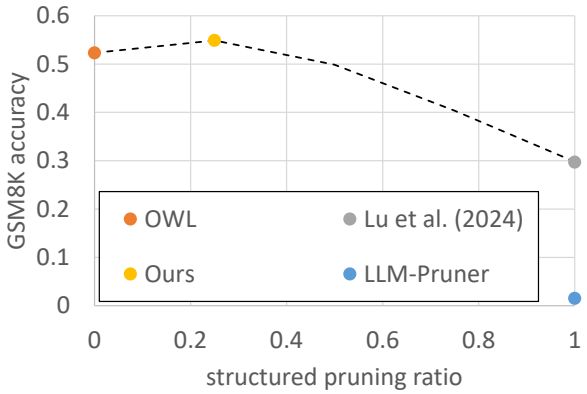


Figure 1: GSM8K 5-shot accuracy by pruning Mixtral-8x7B-Instruct by 50% of sparsity. We probe interpolation of structured and unstructured pruning, by varying the ratio of structured pruning.

Despite these reductions, the massive number of parameters remains unchanged, requiring significantly more GPU memory, which makes serving large MoE models challenging for many. Additionally, recent MoEs tend to increase the number of experts n, resulting in even larger MoEs. For instance, accommodating 56B parameters of Mixtral (Jiang et al., 2024) with 8 experts or 132B of DBRX (Databricks, 2024) with 16 experts, or 480B of Snowflake Arctic (Snowflake, 2024) with 128 experts, requires an ever-growing amount of memory and more GPUs to serve.

To reduce the number of parameters, unstructured (Frantar and Alistarh, 2023; Sun et al., 2024), or structured pruning (Ma et al., 2023) can be considered. Unstructured pruning allows weight tensors to be sparse anywhere, while structured pruning imposes patterns on sparsification, such as removing rows, entire weight tensors (Ma et al., 2023), or pruning experts in MoE (Lu et al., 2024a).

In this paper, we propose the interpolation of two,

[*]Work done while visiting Snowflake. Correspond to seungwonh@snu.ac.kr

[1]https://github.com/thnkinbtfly/STUN

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 13660–13676

Inter-Expert Sparsity	Intra-Expert Sparsity	Figure 1
✓	✗	Lu et al., 2024a (●)
✗	✓	LLM-Pruner (●), Wanda, OWL (●)
✓	✓	Ours (●)

Table 1: Comparison of existing pruning methods for MoEs.

Structured-Then-UNstructured pruning (STUN). Figure 1 motivates our interpolated method, where unstructured- or structured-only, $x = 0$ and $x = 1$, respectively, is outperformed by the peak, combining both.

Our first phase, expert pruning, by leveraging model-inherent expert for pruning, significantly outperforms row/column-level structured pruning (Figure 1 blue; Ma et al., 2023) However, existing expert-level pruning for MoE (Figure 1 grey; Lu et al., 2024a) often does not scale well over the solution space, requiring an exhaustive combination of experts, leading to $O(\frac{k^n}{\sqrt{n}})$ GPU calls, with $k = \frac{1}{\phi^\phi(1-\phi)^{1-\phi}}$, and $\phi < 1$ is sparsity (Lu et al., 2024a). While this was acceptable in an early MoE work with few experts, it does not scale to recent trends in MoEs with large n (Bai et al., 2023; Dai et al., 2024; Snowflake, 2024), or even infinity (He, 2024). Our distinction is drastically reducing the number of GPU calls to $O(1)$, without compromising the performance. The main intuition is leveraging a latent structure between experts, based on behavior similarity, such that the greedy decision of whether to prune closely captures the joint pruning effect.

The second contribution is allowing unstructured phase to follow, to consider both inter- and intra-expert sparsity (Table 1). STUN removes redundant experts by expert-level structured pruning first, then desires fine-grained sparsity within individual experts.[2]

We support STUN with the findings of Mason-Williams and Dahlqvist (2024), which show that higher kurtosis in the weight distribution (indicating many outliers) suggests more weights can be pruned while maintaining performance, highlighting the robustness of unstructured pruning. We argue that expert-level pruning does not reduce kurtosis, thereby preserving the network's resilience to unstructured pruning.

Our contributions can be summarized as follows:

- We propose STUN, the first method to combine structured and unstructured pruning, outperforming both approaches.
- **Scalable first phase:** We design an expert-level pruning method with $O(1)$ GPU calls, outperforming the $O(\frac{k^n}{\sqrt{n}})$ solution (Lu et al., 2024a).
- **Justifying the second phase:** We show the expert-pruned network remains robust to unstructured pruning to follow.
- **State-of-the-art efficiency and compression:** For Snowflake Arctic (480B, 128 experts), it requires just 1 H100 and two hours, with no backpropagation or fine-tuning needed. Compression reaches up to 40% sparsity without compromising performance, even in generative tasks like GSM8K, where unstructured pruning fails. We report consistent results for Mixtral models.

2 Related Work

2.1 LLM Pruning

LLM pruning can be classified into unstructured and structured pruning (Behnke and Heafield, 2021). Unstructured pruning involves finding mask tensors to sparsify weight tensors. Such masking leads to practical speedups in hardware such as CPU (NeuralMagic, 2021), and ongoing research is actively developing methods to achieve similar speedups on GPUs (Mishra et al., 2021; Zhao et al., 2024). SparseGPT (Frantar and Alistarh, 2023) uses the Hessian matrix for second-order Taylor approximation, while GBLM-Pruner (Das et al., 2024) and Pruner-Zero (Dong et al., 2024) leverage gradients to identify mask tensors. However, as these methods demand substantial GPU memory for LLMs, we focus on more memory-efficient approaches, using two recent baselines: Wanda (Sun et al., 2024) evaluates the *importance* of neurons in each layer by its weight multiplied by the activation value, removing those with low scores. While Wanda assumes a uniform sparsity across layers,

[2]From now on, we will define sparsity as the number of pruned parameters divided by the total number of parameters in the original model.

OWL (Yin et al., 2024) probes the optimal sparsity per layer, given the pruning budget.

Structured pruning, on the other hand, imposes constraints on the sparsification pattern, such as removing rows, columns, or even entire weight tensors. Early methods that involve pruning attention heads (Voita et al., 2019; Shim et al., 2021; Zhang et al., 2021), rows (Gong et al., 2022), entire dense layers (Liang et al., 2021), or whole transformer blocks (Fan et al., 2019; Li et al., 2020) fall under this category. Recent works have applied structured pruning for LLMs (Ma et al., 2023; Cheng et al., 2024; Gao et al., 2024; Zhang et al., 2024a; Dery et al., 2024), but without fine-tuning, these methods generally underperform when compared to unstructured pruning.

Our distinction is to introduce a new class of pruning– structured-then-unstructured pruning– and demonstrate its significant advantages for MoEs, surpassing the performance of either method alone. This approach differs from previous methods that combine structured and unstructured pruning (Kurtic et al., 2022), which failed to outperform unstructured pruning.

2.2 Expert Pruning

Early work on expert pruning was domain-specific (Kim et al., 2021; Koishekenov et al., 2023; Liu et al., 2024), such as in translation MoEs, by keeping most activated experts (Kim et al., 2021), or pruning based on gate statistics (Koishekenov et al., 2023). Lu et al. (2024a) introduced a domain-agnostic expert pruning, searching for the best combination of experts to reduce the reconstruction loss, and quantify their criticality in output prediction. Concurrently to our study, (Zhang et al., 2024b) proposed an efficient expert pruning method.

Our distinction is two-fold. First, we interpolate expert pruning with unstructured pruning to outperform either method alone. Second, for scalable expert-level structured pruning, we derive a scalable expert-level structured pruning method with $O(1)$ GPU calls, improving on the $O(\frac{k^n}{\sqrt{n}})$ solution enumerating combinatorial pruning.

2.3 Pruning Robustness

Robustness in post-hoc pruning is quantified by whether performance is maintained after pruning. Kurtosis of weights (Mason-Williams and Dahlqvist, 2024) has been used as a proxy for robustness, with networks showing higher weight kurtosis able to tolerate higher unstructured pruning ratios. Our contribution is demonstrating that an expert-pruned network remains robust to additional unstructured pruning, which naturally supports our design of unstructured pruning as the second phase.

3 Preliminaries: MoE

As a promising alternative to large language models, which incur prohibitive inference costs, MoE employs a multitude of specialized experts. In each forward pass, MoE selectively activates specific experts conditioned on input tokens, thereby reducing the train and inference costs.

We now formally describe the behavior of an MoE. An MoE layer M consists of experts $E_i = E(x;\theta_i)$, where θ_i represents the parameters of expert E_i, and a router layer r. Each expert E typically follows the same MLP architecture.

First, the router layer selects which experts to sparsely activate based on the current input token, and provides the coefficients $r(x) \in \mathbb{R}^n$ for linear combination of selected expert outputs. The coefficients $r(x)$ and the top-k indices of experts $\mathcal{T}$ are formulated as follows:

$$r(x) = \text{softmax}(Wx) \tag{1}$$

$$\mathcal{T} = \text{topk}(r(x)) \tag{2}$$

where W is the learnable weight matrix for router r.

Next, these coefficients are used for the linear combination of expert outputs:

$$M(x;\theta) = \sum_{i\in\mathcal{T}} r_i(x) E(x;\theta_i) \tag{3}$$

4 Proposed Method

4.1 Overview: STUN

Figure 2 overviews our two-phase approach, interpolating structured and unstructured pruning as motivated in Table 1. Section 4.2 describes ① how we remove redundant experts with expert-level structured pruning with high scalability, then Section 4.3 describes ② how we perform unstructured pruning inside individual experts.

4.2 Expert-level Structured Pruning with $O(1)$ GPU calls

Now, we describe our expert-level structured pruning with $O(1)$ GPU calls.[3] Previous solution (Lu

[3]We focus on GPU cost, since it dominates the CPU cost– For example, the accumulated CPU cost (including the hy-

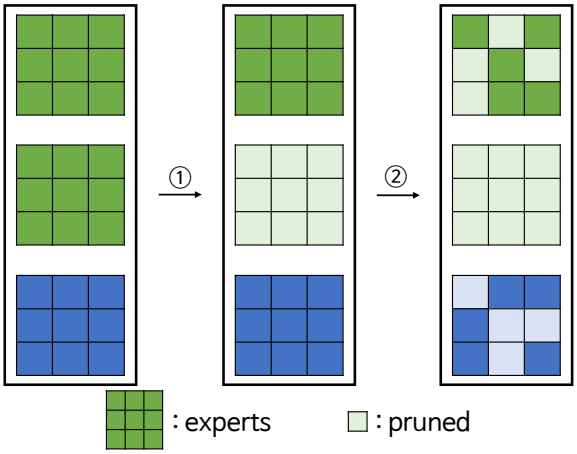


Figure 2: Overview of our proposed STUN. ① We first remove redundant experts with expert-level structured pruning, then ② perform unstructured pruning inside individual experts. Black box represents a layer in MoE, and different colors represent different behavioral similarities.

et al., 2024a) minimizes reconstruction loss (Section 4.2.1), requiring GPU call per combination of experts, that is $O(\frac{k^n}{\sqrt{n}})$. Our key contribution is to approximate this combinatorial reconstruction loss to reduce the number of GPU calls to $O(1)$, by leveraging *latent cluster structure* among experts, based on behavioral similarity.

Specifically, we find clusters of similar experts layer by layer, yielding a total of ϕnl clusters in the whole MoE, where ϕ is the sparsity, n is the number of experts in each layer, and l is the number of layers in MoE. Then we greedily prune every expert but one representative per each cluster.

Later sections show why our greedy pruning is as effective as its combinatorial counterpart.

4.2.1 $O(\frac{k^n}{\sqrt{n}})$: Combinatorial Reconstruction Loss

We start from the conventional goal of pruning– minimizing the reconstruction loss. Reconstruction loss has been employed to assess how closely the pruned model $\theta - \theta_S$ without expert set S mirrors the behavior of the unpruned θ (Lu et al., 2024a). Formally, this loss is quantified by the Frobenius norm of the difference between the original output $M(x;\theta)$ and the output of pruned layer $M(x;\theta - \theta_S)$, denoted as $\mathcal{E}_S$.

$$\mathcal{E}_S = \|M(x;\theta) - M(x;\theta - \theta_S)\|_F \quad (4)$$

perparameter search to meet desired sparsity) required by our algorithm is less than 1 minute, even on 480B Snowflake Arctic.

where x is the whole input we consider. The objective of pruning is to explore all possible combinations of experts, $\binom{n}{|S|}$, to determine the expert set S that minimizes $\mathcal{E}_S$.

While such an exhaustive search is feasible for smaller models like Mixtral (Jiang et al., 2024), which contains only 8 experts, it becomes prohibitive for recent MoEs featuring a massive number of experts.

To elaborate, deciding which experts to prune using Eq. 4 for $|S| = \phi n$ requires $\binom{n}{|S|} \approx O(\frac{k^n}{\sqrt{n}})$ forward passes according to Stirling's approximation, where $k = \frac{1}{\phi^\phi (1-\phi)^{1-\phi}}$, and $\phi < 1$ represents the sparsity. Our distinction is to lower the computation to $O(1)$, without compromising the performance– In fact, as we will elaborate later, we outperform the combinatorial objective.

4.2.2 Towards $O(n)$: Probabilistic Interpretation

As a stepping stone towards $O(1)$, we propose to rephrase the goal of finding θ_S to minimize $\mathcal{E}_S$ (Eq. 4) as:

$$\text{argmax}_S \prod_k P(X_k = s_k | X_1 = s_1, \cdots, X_{k-1} = s_{k-1}) \quad (5)$$

Our contribution is greedy optimization without compromise for Eq. 5– We decompose the multiplication of Eq. 5 at each step k, and obtain the distribution $P(X_k | s_1, \cdots, s_{k-1})$, to select X_k that maximizes the probability. To achieve it, we estimate the rank between the probabilities. Such rank estimation can benefit from the latent structure among experts, specifically, a cluster of similar experts in MoE. Given cluster mapping c which maps an expert to a set of similarly behaving experts, we assign the value $P(E_i | S_{k-1})$, as follows:

$$P(E_i|S_{k-1}) = \begin{cases} P(E_i) - p & c(E_i) \subseteq S_k \\ P(E_i) & \text{otherwise} \end{cases} \quad (6)$$

This enables the calculation of all $P(E_i|S_{k-1})$ in Eq. 5 from $P(E_i)$s, which needs only n forwards in total.

Clustering the Similar Experts Our remaining task is to obtain cluster information c: One signal is pairwise behavioral similarity $b_{i,j}$, from the pretrained router weights W at a minimal cost. Suppose two rows $W_i \approx W_j$ are similar; then

$r_i(x) \approx r_j(x)$, meaning E_i, E_j tend to be selected by similar inputs, implying similar expertise. Thus, the behavioral similarity $b_{i,j}$ between two experts E_i, E_j can be obtained as follows:

$$b_{i,j} = -\|W_i - W_j\|_F \tag{7}$$

which can be improved with coactivation statistics $a_{i,j}$, if we allow some inference cost. As a result, we illustrate our clustering algorithm in Alg 1, whose detailed derivation can be found in Appendix A.1.

Algorithm 1 Expert Clustering Algorithm

Require: $l \leftarrow$ Number of layers
Require: $n \leftarrow$ Number of experts per layer
Require: $\{a_{i,j}\}_{i,j} \leftarrow$ Coactivation statistics of E_i, E_j for every layer
Require: $\lambda_1, \lambda_2 \leftarrow$ Hyperparameter for behavioral similarity
Require: $t \leftarrow$ Threshold to determine sparsity
Ensure: $c \leftarrow$ The mapping from expert to cluster of the similar experts
  for m in $[1..l]$ **do**
    $W \leftarrow$ Router weight of layer m
    $\{a_{i,j}\}_{i,j} \leftarrow$ Coactivation statistics of layer m
    for i in $[1..n-1]$ **do**
      for j in $[i+1..n]$ **do**
        $b_{i,j} \leftarrow -\lambda_1\|W_i - W_j\|_F + \lambda_2 a_{i,j}$
      end for
    end for
    for i in $[1..n]$ **do**
      $c(E_i) \leftarrow \{E_i\}$
    end for
    while $\min_{i,j} b_{i,j} < t$ **do**
      $d, e \leftarrow \text{argmin}_{i,j} b_{i,j}$
      $m_d \leftarrow \max_{i \in c(E_e)} b_{d,i}$
      $m_e \leftarrow \max_{i \in c(E_d)} b_{i,e}$
      if $c(E_d) \neq c(E_e) \wedge \max(m_d, m_e) < t$ **then**
        $c(E_d) = c(E_e) \leftarrow c(E_d) \cup c(E_e)$
      end if
      $b_{d,e} \leftarrow \infty$ ▷ Mark as visited
    end while
  end for
  return c

4.2.3 Towards $O(1)$: Taylor Approximation and Selective Reconstruction

While the previous section immensely reduces the cost to obtain the probability distribution to $O(n)$ by requiring only $P(E_i)$s, we can further reduce the number of forward passes– We aim to remove the GPU calls for $P(E_i)$, which is needed as in Eq. 10.

The key idea is approximating E_i's reconstruction loss value $\mathcal{E}_i = \|M(x;\theta) - M(x;\theta - \theta_i)\|_F$. To address this, with 1st order Taylor approximation, we find the expert closest to $\bar{\theta}_i$ within each cluster has the highest priority to be retained. We assign ranks similarly, and the same greedy algorithm is applied to optimize Eq. 5. Additionally, we selectively reconstruct the expert. The final algorithm is summarized in Alg 2, which is described in detail in Appendix A.2.

Algorithm 2 Our $O(1)$ Expert Pruning

Require: $l \leftarrow$ Number of layers
Require: $n \leftarrow$ Number of experts per layer
Require: $c \leftarrow$ The mapping from expert to cluster of the similar experts
Require: $\kappa \leftarrow$ Threshold for selective reconstruction
  for m in $[1..l]$ **do**
    $r(m) = [\,]$
    $A \leftarrow \{c(E_1), \cdots, c(E_n)\}$
    for C in A **do**
      $\bar{\theta}_i \leftarrow \frac{1}{|C|}\sum_{i \in C} \theta_i$
      if $|A| < \kappa$ **then**
        $\theta_C \leftarrow \bar{\theta}_i$ ▷ Reconstruct
      else
        $\theta_C \leftarrow \min_{\theta_j \in C} \|\theta_j - \bar{\theta}_i\|_F$
      end if
      $r(m).append(\theta_C)$
    end for
  end for
  return r

4.3 Unstructured Pruning on Expert-pruned Model

Our main conjecture for STUN is intra-expert sparsity yet remains intact after the first phase. Specifically, we propose to pursue fine-grained sparsity within the remaining experts, by leveraging unstructured pruning methods designed for general LLM , such as OWL (Yin et al., 2024) or Wanda (Sun et al., 2024).

Now we theoretically verify our conjecture: intra-expert sparsity remains high, or, even higher, after the expert pruning phase. In other words, we explain why performing unstructured pruning after

model	sparsity	method	GSM8K	Avg(→)	ARC-c	ARC-e	HellaSwag	MMLU
Arctic	0%	unpruned	70.74	68.33	56.91	84.60	66.94	64.86
	40%	STUN (w/ OWL)	**70.28**	**67.66**	57.68	83.29	64.94	64.75
		OWL	63.76	67.35	56.74	84.13	65.08	63.47
		STUN (w/ Wanda)	**69.60**	**67.64**	57.25	83.63	64.86	64.81
		Wanda	64.59	67.54	57.00	84.64	65.19	63.32
	65%	STUN (w/ OWL)	**43.97**	**62.67**	51.54	80.01	59.91	59.24
		OWL	13.42	56.68	44.37	76.64	53.69	52.02
Mixtral-8x7B (Instruct)	65%	STUN (w/ OWL)	**25.09**	**60.34**	48.12	78.79	54.05	60.39
		OWL	1.29	45.20	24.15	49.79	49.27	57.60
		LLM-Pruner	1.29	31.74	22.78	45.96	35.09	23.13
Mixtral-8x22B (Instruct)	70%	STUN (w/ OWL)	**30.78**	**60.20**	47.95	77.86	55.41	59.56
		OWL	19.64	57.74	45.48	76.60	52.47	56.42

Table 2: Comparison between STUN and the baselines across various models.

expert pruning is better than continuously performing unstructured pruning only.

Intra-expert sparsity, formally speaking, robustness to unstructured pruning, can be estimated by the kurtosis of weights (Mason-Williams and Dahlqvist, 2024). Kurtosis is expressed as follows:

$$K(\theta) = E\left[\left(\frac{\theta - \mu}{\sigma}\right)^4\right] \tag{8}$$

Suppose the weight of experts θ follow a zero-meaned Gaussian distribution $\mathcal{N}$. Unstructured pruning (Sun et al., 2024; Yin et al., 2024; Das et al., 2024; Dong et al., 2024), which tends to remove near-zero weights,[4] would shift the distribution closer to a bimodal symmetric distribution, whose kurtosis is minimum (Darlington, 1970). As a result, unstructured pruning would lower the kurtosis value, leaving less margin for further unstructured pruning.

In contrast, coarse structured pruning, such as expert pruning, is less likely to decrease the kurtosis value, since the assumption $\theta \sim \mathcal{N}$ still holds for remaining experts. This implies that expert pruning preserves the robustness of unstructured pruning, unlike applying unstructured pruning with a similar sparsity.[5]

5 Experiments

5.1 Experimental Settings

We use Snowflake Arctic (Snowflake, 2024) as a representative large MoE, with a total of 480B parameters and 128 experts. To compare our method with previous works (Lu et al., 2024a), we also experiment with Mixtral (Jiang et al., 2024).

Tasks and Datasets In contrast to previous unstructured pruning studies (Sun et al., 2024; Yin et al., 2024), we also evaluate the NLG task, GSM8K (Cobbe et al., 2021), where maintaining performance proves to be much more challenging (see Table 2; Appendix F). We further assess performance on four NLU tasks– ARC-challenge and ARC-easy (Clark et al., 2018), HellaSwag (Zellers et al., 2019), and MMLU (Hendrycks et al., 2021). When comparing with expert pruning methods, following previous work (Lu et al., 2024a), we also conduct a zero-shot evaluation on BoolQ (Wang et al., 2019), OpenBookQA (Mihaylov et al., 2018), RTE (Wang et al., 2018), WinoGrande (Sakaguchi et al., 2021).

To provide some data for inference, we employ the C4 dataset (Raffel et al., 2020), following the baselines (Yin et al., 2024; Sun et al., 2024; Lu et al., 2024a).

Implementation Details We explore the values of $(\lambda_1, \lambda_2) \in \{(0,1),(1,0),(1,1)\}$, except for Snowflake Arctic, the largest MoE in our experiments, where we only consider $(\lambda_1, \lambda_2) = (1,0)$, meaning no GPU calls are required for expert pruning. The sparsity for expert-level structured pruning is set to 20%, 12.5%, and 10% for Snowflake Arctic, Mixtral-8x7B, and Mixtral-8x22B respectively. We use $\kappa = 3$ for selective reconstruction. More detailed implementation information and hyperparameter decisions are provided in Appendix.

[4]The importance score of unstructured pruning typically increases as the absolute value of the weight increases.

[5]In our experiments, the kurtosis increased from 14248 to 15623 after expert pruning.

model	sparsity	method	cost	Avg
Mixtral-8x7B (Instruct)	0%	unpruned		69.98
	25%	Ours	$O(1)$	**68.05**
		Lu et al. (2024a)	$O(\frac{k^n}{\sqrt{n}})$	67.45
Mixtral-8x7B	0%	unpruned		67.58
	25%	Ours	$O(1)$	**64.34**
		Lu et al. (2024a)	$O(\frac{k^n}{\sqrt{n}})$	64.22

Table 3: Comparing the average performance of 8 tasks of the proposed expert pruning, with other baselines.

	sparsity	ARC-C	BoolQ	HellaSwag	MMLU	RTE	WinoGrande	Avg
Unpruned	0%	59.4	84.2	84.0	67.9	70.4	75.6	71.5
Ours	25%	55.6	83.1	81.1	63.3	68.6	72.7	**70.7**
SEER-MoE (Muzio et al., 2024)	25%	-	-	-	56.7	-	-	-
Expert Drop (He et al., 2024)	25%	53.2	77.7	80.5	52.2	55.6	76.8	66.0
Layer Drop (He et al., 2024)	25%	47.7	85.3	75.2	67.3	69.7	74.6	70.0

Table 4: Comparison of efficient expert pruning methods on Mixtral-8x7B.

	sparsity	cost	GSM8K
unpruned	0%		63.46
Ours	25%	$O(1)$	**53.22**
Lu et al. (2024a)	25%	$O(\frac{k^n}{\sqrt{n}})$	48.52

Table 5: GSM8K accuracy comparison with baseline on Mixtral-8x7B-Instruct.

sparsity	$O(n)$	$O(1)$
25%	63.97	64.34
50%	59.90	59.58

Table 6: Comparing the average performance of 8 tasks of our $O(n)$ and $O(1)$ algorithms for expert pruning on Mixtral-8x7B.

5.2 Experimental Results

5.2.1 RQ1: STUN Outperforms Unstructured Pruning

Table 2 describes that our proposed (STUN) significantly outperforms the unstructured pruning methods. We emphasize that we use the same unstructured pruning approach for all for fair comparison.

For example, with 40% of sparsity for the Arctic, STUN neatly retains the original GSM8K performance, while unstructured pruning results in a noticeable performance drop. This is consistent for different unstructured pruning methods, Wanda, as well. As the sparsity increases, the table shows that STUN can maintain the original performance much better than the baselines– For 65% sparsity, STUN's GSM8K performance is nearly 30%p better than that of unstructured pruning. With different models, we observe different gaps but a similar trend– For 65% of sparsity for Mixtral-8x7B-Instruct, STUN's GSM8K performance is nearly 20 times better than that of unstructured pruning. In the ARC-challenge, the unstructured pruning performance falls below the random-guess accuracy of 25.02 (Clark et al., 2018), whereas STUN maintains a significantly higher performance, achieving twice the score. Mixtral-8x22B shows a similar trend.

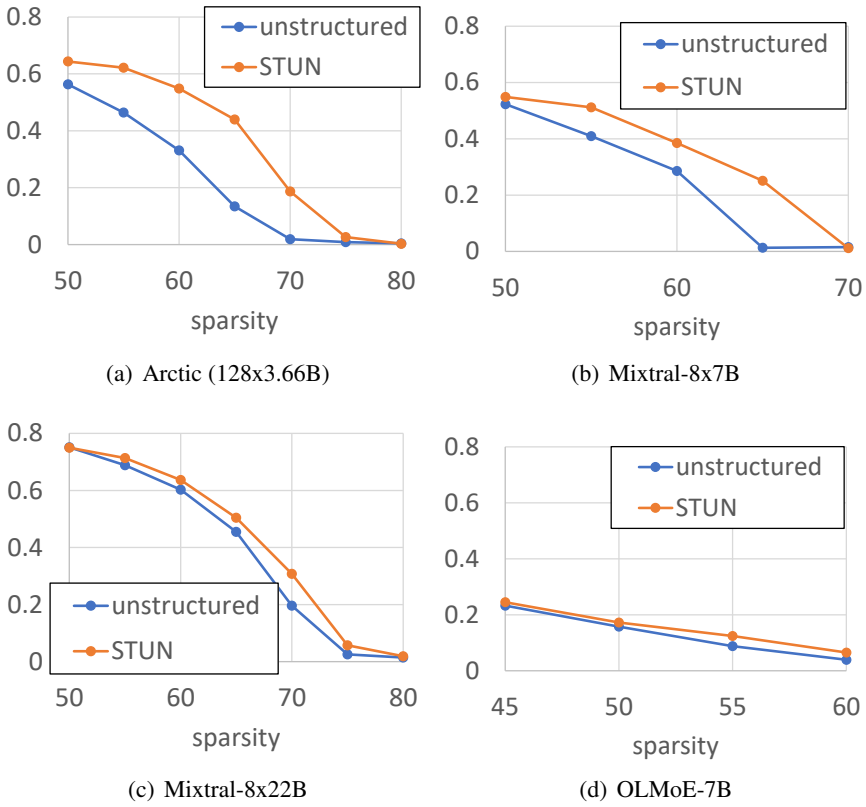
5.2.2 RQ2: Our $O(1)$ Expert Pruning Outperforms Existing Methods

Tables 3 and 5 show that our proposed $O(1)$ expert pruning method is highly effective, outperforming the previous $O(\frac{k^n}{\sqrt{n}})$ solution. This is because we derive the latent structure from the pretrained MoE, while the previous work (Lu et al., 2024a) relies solely on the given calibration data. This validates our design of $O(1)$ in section 4. The detailed results are in Appendix E.

Table 4 shows that our expert pruning outperforms other efficient pruning methods. Compared to SEER-MoE, our pruning clearly outperforms the performance in MMLU, by a substantial margin. Moreover, ours achieves a higher average performance compared to the results reported by He et al. (2024).[6]

Moreover, Table 6 shows that our $O(1)$ expert

[6]Note that SEER-MoE (Muzio et al., 2024) only reveals the performance of MMLU, and we used different metrics following He et al. (2024).



(a) Arctic (128x3.66B) (b) Mixtral-8x7B

(c) Mixtral-8x22B (d) OLMoE-7B

Figure 3: Comparing STUN and unstructured pruning for various MoEs.

	Train GPU cost		GPUs #
Lu et al. (2024a)	0	infeasible[7]	> 8
STUN (w/ OWL)	0	1.12h	1
OWL (Yin et al., 2024)	0	1.12h	1
STUN (w/ Wanda)	0	0.58h	1
Wanda (Sun et al., 2024)	0	0.58h	1

Table 7: Cost comparison of diverse pruning methods with Snowflake-Arctic. Train: training cost, GPU #: number of GPUs required for pruning.

pruning method achieves similar performance to our $O(n)$ method. This supports our choice to use the $O(1)$ method, which is more efficient.

5.2.3 RQ3: STUN Adapts Large Number of Small Experts

Figure 3 illustrates the trend of STUN in different MoEs. The performance gap between STUN and unstructured pruning increases as the MoE has more experts with small sizes (from (c) to (a)). This is because having more experts, rather than having fewer but larger ones, provides greater flexibility to our expert pruning. Notably, MoEs with a large number of small experts are favored in recent models (He, 2024).

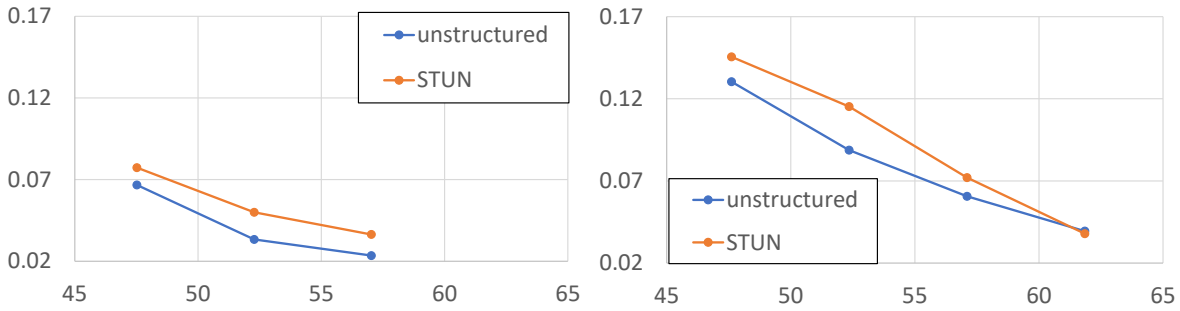
5.2.4 RQ4: STUN Outperforms Unstructured Pruning in non-MoEs

To investigate whether STUN is generalizable to non-MoE as well, we employ a state-of-the-art structured pruning algorithm for non-MoE models, namely, LLM-surgeon (van der Ouderaa et al., 2024) with 5% sparsity before performing unstructured pruning, which is OWL in our case. Figure 4 illustrates that such STUN outperforms unstructured pruning.

6 Cost Analysis

Table 7 shows the cost comparison between diverse pruning methods. While none of the pruning methods require training cost, Lu et al. (2024a) is infeasible due to its prohibitive number of forward passes in GPUs. Due to the efficiency of proposed expert pruning, STUN is as efficient as the unstructured pruning method it uses, making it a feasible pruning method even for large MoEs, such as Snowflake-Arctic.

[7] In detail, 23951146041928082866135587776380551750 forward passes per layer at minimum.



(a) Llama-2 7B (b) Llama-2 13B

Figure 4: GSM8K 5-shot accuracy comparing STUN and unstructured pruning for various non-MoEs.

7 Conclusion

In this paper, we proposed STUN– an interpolation between structured and unstructured pruning, leveraging both inter- and intra-expert sparsity.We provide both theoretical and empirical evidence demonstrating why designing expert pruning before unstructured pruning is beneficial.

Limitation

Since our method utilizes unstructured pruning in the second stage, we share the same disadvantages with unstructured pruning, that is, on some hardware, the acceleration may not be trivial. However, it is shown that some hardware, such as CPU, can successfully accelerate unstructure-pruned networks (NeuralMagic, 2021), and ongoing research is actively developing methods to achieve similar speedups on GPUs (Mishra et al., 2021; Zhao et al., 2024). With a substantial body of work dedicated to unstructured pruning (Frantar and Alistarh, 2023; Das et al., 2024; Dong et al., 2024; Sun et al., 2024; Yin et al., 2024; Li et al., 2024; Hu et al., 2024; Lu et al., 2024b), we believe this approach remains highly relevant and leads to practical performance improvements with ongoing hardware support.

We already have shown our method works in versatile extreme settings, such as non-MoE models, or high sparsity, etc. We leave more extensive evaluations, such as performance under highly imbalanced or skewed data distributions, as future work.

References

Daniel Aloise, Amit Deshpande, Pierre Hansen, and Preyas Popat. 2009. NP-hardness of Euclidean sum-of-squares clustering. *Machine Learning*, 75(2):245–248.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023. Qwen Technical Report. *Preprint*, arXiv:2309.16609.

Maximiliana Behnke and Kenneth Heafield. 2021. Pruning Neural Machine Translation for Speed Using Group Lasso. In *Proceedings of the Sixth Conference on Machine Translation*, pages 1074–1086, Online. Association for Computational Linguistics.

Daniel Brélaz. 1979. New methods to color the vertices of a graph. *Communications of The Acm*, 22(4):251–256.

Hongrong Cheng, Miao Zhang, and Javen Qinfeng Shi. 2024. MINI-LLM: Memory-Efficient Structured Pruning for Large Language Models. *Preprint*, arXiv:2407.11681.

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. *Preprint*, arXiv:1803.05457.

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. *Preprint*, arXiv:2110.14168.

Damai Dai, Chengqi Deng, Chenggang Zhao, R.x. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y. Wu, Zhenda Xie, Y.k. Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and Wenfeng Liang. 2024. DeepSeekMoE: Towards Ultimate Expert Specialization in Mixture-of-Experts

Language Models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1280–1297, Bangkok, Thailand. Association for Computational Linguistics.

Richard B. Darlington. 1970. Is Kurtosis Really "Peakedness?". *The American Statistician*, 24(2):19–22.

Rocktim Jyoti Das, Mingjie Sun, Liqun Ma, and Zhiqiang Shen. 2024. Beyond Size: How Gradients Shape Pruning Decisions in Large Language Models. *Preprint*, arXiv:2311.04902.

Sanjoy Dasgupta. 2008. The hardness of k-means clustering.

Databricks. 2024. Databricks/dbrx. Databricks.

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.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient finetuning of quantized LLMs. In *Thirty-Seventh Conference on Neural Information Processing Systems*.

Peijie Dong, Lujun Li, Zhenheng Tang, Xiang Liu, Xinglin Pan, Qiang Wang, and Xiaowen Chu. 2024. Pruner-Zero: Evolving Symbolic Pruning Metric From Scratch for Large Language Models. In *Forty-First International Conference on Machine Learning*.

Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, Barret Zoph, Liam Fedus, Maarten P Bosma, Zongwei Zhou, Tao Wang, Emma Wang, Kellie Webster, Marie Pellat, Kevin Robinson, Kathleen Meier-Hellstern, Toju Duke, Lucas Dixon, Kun Zhang, Quoc Le, Yonghui Wu, Zhifeng Chen, and Claire Cui. 2022. GLaM: Efficient scaling of language models with mixture-of-experts. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 5547–5569. PMLR.

Angela Fan, Edouard Grave, and Armand Joulin. 2019. Reducing Transformer Depth on Demand with Structured Dropout. In *International Conference on Learning Representations*.

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. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 10323–10337. PMLR.

Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, Jason Phang, Laria Reynolds, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2021. A framework for few-shot language model evaluation. Zenodo.

Yuan Gao, Zujing Liu, Weizhong Zhang, Bo Du, and Gui-Song Xia. 2024. Optimization-based Structural Pruning for Large Language Models without Back-Propagation. *Preprint*, arXiv:2406.10576.

Hongyu Gong, Xian Li, and Dmitriy Genzel. 2022. Adaptive Sparse Transformer for Multilingual Translation. *arXiv:2104.07358 [cs]*.

Babak Hassibi and David Stork. 1992. Second order derivatives for network pruning: Optimal Brain Surgeon. In *Advances in Neural Information Processing Systems*, volume 5. Morgan-Kaufmann.

Shwai He, Daize Dong, Liang Ding, and Ang Li. 2024. Demystifying the Compression of Mixture-of-Experts Through a Unified Framework. *Preprint*, arXiv:2406.02500.

Xu Owen He. 2024. Mixture of A Million Experts. *Preprint*, arXiv:2407.04153.

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

Yuezhou Hu, Jun Zhu, and Jianfei Chen. 2024. S-STE: Continuous Pruning Function for Efficient 2:4 Sparse Pre-training. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7B. *Preprint*, arXiv:2310.06825.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024. Mixtral of Experts. *Preprint*, arXiv:2401.04088.

Jean Kaddour, Joshua Harris, Maximilian Mozes, Herbie Bradley, Roberta Raileanu, and Robert McHardy. 2023. Challenges and Applications of Large Language Models. *Preprint*, arXiv:2307.10169.

Young Jin Kim, Ammar Ahmad Awan, Alexandre Muzio, Andres Felipe Cruz Salinas, Liyang Lu, Amr Hendy, Samyam Rajbhandari, Yuxiong He, and Hany Hassan Awadalla. 2021. Scalable and Efficient MoE Training for Multitask Multilingual Models. *Preprint*, arXiv:2109.10465.

Yeskendir Koishekenov, Alexandre Berard, and Vassilina Nikoulina. 2023. Memory-efficient NLLB-200: Language-specific Expert Pruning of a Massively Multilingual Machine Translation Model. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3567–3585, Toronto, Canada. Association for Computational Linguistics.

Eldar Kurtic, Daniel Campos, Tuan Nguyen, Elias Frantar, Mark Kurtz, Benjamin Fineran, Michael Goin, and Dan Alistarh. 2022. The Optimal BERT Surgeon: Scalable and Accurate Second-Order Pruning for Large Language Models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 4163–4181, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Lujun Li, Peijie Dong, Zhenheng Tang, Xiang Liu, Qiang Wang, Wenhan Luo, Wei Xue, Qifeng Liu, Xiaowen Chu, and Yike Guo. 2024. Discovering Sparsity Allocation for Layer-wise Pruning of Large Language Models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Xian Li, Asa Cooper Stickland, Yuqing Tang, and Xiang Kong. 2020. Deep transformers with latent depth. In *Advances in Neural Information Processing Systems*, volume 33, pages 1736–1746. Curran Associates, Inc.

Chen Liang, Simiao Zuo, Minshuo Chen, Haoming Jiang, Xiaodong Liu, Pengcheng He, Tuo Zhao, and Weizhu Chen. 2021. Super Tickets in Pre-Trained Language Models: From Model Compression to Improving Generalization. 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 6524–6538, Online. Association for Computational Linguistics.

Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, Omri Abend, Raz Alon, Tomer Asida, Amir Bergman, Roman Glozman, Michael Gokhman, Avashalom Manevich, Nir Ratner, Noam Rozen, Erez Shwartz, Mor Zusman, and Yoav Shoham. 2024. Jamba: A Hybrid Transformer-Mamba Language Model. *Preprint*, arXiv:2403.19887.

Enshu Liu, Junyi Zhu, Zinan Lin, Xuefei Ning, Matthew B. Blaschko, Shengen Yan, Guohao Dai, Huazhong Yang, and Yu Wang. 2024. Efficient Expert Pruning for Sparse Mixture-of-Experts Language Models: Enhancing Performance and Reducing Inference Costs. *Preprint*, arXiv:2407.00945.

Xudong Lu, Qi Liu, Yuhui Xu, Aojun Zhou, Siyuan Huang, Bo Zhang, Junchi Yan, and Hongsheng Li. 2024a. Not All Experts are Equal: Efficient Expert Pruning and Skipping for Mixture-of-Experts Large Language Models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6159–6172.

Xudong Lu, Aojun Zhou, Yuhui Xu, Renrui Zhang, Peng Gao, and Hongsheng Li. 2024b. SPP: Sparsity-Preserved Parameter-Efficient Fine-Tuning for Large Language Models. In *Forty-First International Conference on Machine Learning*.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. LLM-Pruner: On the structural pruning of large language models. In *Thirty-Seventh Conference on Neural Information Processing Systems*.

Gabryel Mason-Williams and Fredrik Dahlqvist. 2024. What makes a good prune? Maximal unstructured pruning for maximal cosine similarity. In *The Twelfth International Conference on Learning Representations*.

Nimrod Megiddo and Kenneth J. Supowit. 1984. On the Complexity of Some Common Geometric Location Problems. *SIAM Journal on Computing*, 13(1):182–196.

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 *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, Brussels, Belgium. Association for Computational Linguistics.

Asit Mishra, Jorge Albericio Latorre, Jeff Pool, Darko Stosic, Dusan Stosic, Ganesh Venkatesh, Chong Yu, and Paulius Micikevicius. 2021. Accelerating Sparse Deep Neural Networks. *Preprint*, arXiv:2104.08378.

Alexandre Muzio, Alex Sun, and Churan He. 2024. SEER-MoE: Sparse Expert Efficiency through Regularization for Mixture-of-Experts. *Preprint*, arXiv:2404.05089.

NeuralMagic. 2021. Neuralmagic/deepsparse: Sparsity-aware deep learning inference runtime for CPUs. https://github.com/neuralmagic/deepsparse.

OpenAI. 2023. GPT-4 Technical Report. *Preprint*, arXiv:2303.08774.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. 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. 2021. WinoGrande: An adversarial winograd schema challenge at scale. *Communications of The Acm*, 64(9):99–106.

Kyuhong Shim, Iksoo Choi, Wonyong Sung, and Jungwook Choi. 2021. Layer-wise Pruning of Transformer Attention Heads for Efficient Language Modeling. In *2021 18th International SoC Design Conference (ISOCC)*, pages 357–358.

Peter H. A. Sneath and Robert R. Sokal. 1973. Numerical taxonomy. The principles and practice of numerical classification.

Snowflake. 2024. Snowflake-Labs/snowflake-arctic. Snowflake Labs.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and Policy Considerations for Deep Learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy. Association for Computational Linguistics.

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*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. *Preprint*, arXiv:2307.09288.

Tycho F. A. van der Ouderaa, Markus Nagel, Mart Van Baalen, and Tijmen Blankevoort. 2024. The LLM surgeon. In *The Twelfth International Conference on Learning Representations*.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy. Association for Computational Linguistics.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. SuperGLUE: A stickier benchmark for general-purpose language understanding systems. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, 294, pages 3266–3280. Curran Associates Inc., Red Hook, NY, USA.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Lu Yin, You Wu, Zhenyu Zhang, Cheng-Yu Hsieh, Yaqing Wang, Yiling Jia, Gen Li, Ajay Kumar Jaiswal, Mykola Pechenizkiy, Yi Liang, Michael Bendersky, Zhangyang Wang, and Shiwei Liu. 2024. Outlier Weighed Layerwise Sparsity (OWL): A Missing Secret Sauce for Pruning LLMs to High Sparsity. In *Forty-First International Conference on Machine Learning*.

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*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

Qingcheng Zeng, Lucas Garay, Peilin Zhou, Dading Chong, Yining Hua, Jiageng Wu, Yikang Pan, Han Zhou, Rob Voigt, and Jie Yang. 2023. GreenPLM: Cross-Lingual Transfer of Monolingual Pre-Trained Language Models at Almost No Cost. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, pages 6290–6298, Macau, SAR China. International Joint Conferences on Artificial Intelligence Organization.

Honghe Zhang, XiaolongShi XiaolongShi, Jingwei Sun, and Guangzhong Sun. 2024a. Structured Pruning for Large Language Models Using Coupled Components Elimination and Minor Fine-tuning. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 1–12, Mexico City, Mexico. Association for Computational Linguistics.

Zeliang Zhang, Xiaodong Liu, Hao Cheng, Chenliang Xu, and Jianfeng Gao. 2024b. Diversifying the Expert Knowledge for Task-Agnostic Pruning in Sparse Mixture-of-Experts.

Zhengyan Zhang, Fanchao Qi, Zhiyuan Liu, Qun Liu, and Maosong Sun. 2021. Know what you don't need: Single-Shot Meta-Pruning for attention heads. *AI Open*, 2:36–42.

Kang Zhao, Tao Yuan, Han Bao, Zhenfeng Su, Chang Gao, Zhaofeng Sun, Zichen Liang, Liping Jing, and Jianfei Chen. 2024. Beyond 2:4: Exploring V:N:M sparsity for efficient transformer inference on GPUs. *Preprint*, arXiv:2410.16135.

A Derivation From $O(\frac{k^n}{\sqrt{n}})$ to $O(1)$

A.1 Towards $O(n)$: Probabilistic Interpretation

As a stepping stone towards $O(1)$, we propose to rephrase the goal of finding θ_S to minimize $\mathcal{E}_S$ (Eq. 4) as:

$$\text{argmax}_S P(X_1 = s_1, \cdots, X_{|S|} = s_{|S|}) \quad (9)$$

where s_is are the experts included in the expert set S, and $P(X_1 = s_1, \cdots, X_{|S|} = s_{|S|})$ is the joint probability of pruning S. One intuitive way to design P so that Eq. 9 yields the same S to minimize Eq. 4 is as follows:

$$P(X_1 = s_1, \cdots, X_{|S|} = s_{|S|}) = \frac{1}{Z} \cdot \frac{1}{\mathcal{E}_S} \quad (10)$$

where Z is the normalization factor. Each joint probability needs $O(1)$ GPU calls, since $\mathcal{E}_S$ needs the output of $M(x; \theta - \theta_S)$.

Section 4.2.1 corresponds to enumerating joint probability from all combinations, requiring $\binom{n}{|S|}$ different values, which is compute-intensive. When chain rule is applied, Eq. 9 can be reformulated as follows:

$$\text{argmax}_S \prod_k P(X_k = s_k | X_1 = s_1, \cdots, X_{k-1} = s_{k-1}) \quad (11)$$

Our contribution is greedy optimization without compromise for Eq. 11– We decompose the multiplication of Eq. 11 at each step k, and obtain the distribution $P(X_k | s_1, \cdots, s_{k-1})$, to select X_k that maximizes the probability. For simplicity, we will omit X_ks from this point on.

As our goal is finding the argmax of the probabilities as in Eq. 11, estimating the rank between them is sufficient, rather than evaluating exact values. Such rank estimation can benefit from the latent structure among experts, specifically, a cluster of similar experts in MoE, enabling $P(X_k | s_1, \cdots, s_{k-1})$ calculation without chain-rule multiplications in Eq. 11.

Assume we know oracle clusters, $c(E_i)$, where c is the mapping from an expert to a set of similarly behaving experts identified from the latent clusters. When we have knowledge of similar experts, for example, $c(E_i) = c(E_j) = \{E_i, E_j\}$ indicating E_i and E_j are highly similar, we will decide not to prune E_i if E_j is already pruned. That is, if $c(E_i) \subseteq S_k$ then $P(X_k = E_i | S_{k-1})$ should be lowered by some value p, to guide the model against pruning. Moreover, $P(E_i | S_{k-1})$ should be larger, or rank higher, otherwise.

To generalize, we will cluster similar experts. Once the cluster of similar experts is finalized, we assign the value $P(E_i | S_{k-1})$, as follows:

$$P(E_i | S_{k-1}) = \begin{cases} P(E_i) - p & c(E_i) \subseteq S_k \\ P(E_i) & \text{otherwise} \end{cases} \quad (12)$$

We set p as a constant for simplicity. This enables the calculation of all $P(E_i | S_{k-1})$ in Eq. 11 from $P(E_i)$s, which needs only n forwards in total.

Clustering the Similar Experts Our remaining task is to obtain cluster information c: One signal is pairwise behavioral similarity $b_{i,j}$, from the pretrained weights W at a minimal cost. Suppose two rows $W_i \approx W_j$ are similar; then $r_i(x) \approx r_j(x)$, meaning E_i, E_j tend to be selected by similar inputs, implying similar expertise. Thus, the behavioral similarity $b_{i,j}$ between two experts E_i, E_j can be obtained as follows:

$$b_{i,j} = -\|W_i - W_j\|_F \quad (13)$$

Next, we generalize pairwise similarity into clusters of experts, such that experts in each cluster C_l are highly similar to its representative μ_l. Formally, the objective of clustering is to minimize the sum of squared errors between μ_l and experts E_i in the cluster:

$$\sum_{i \in C_l} \sum_l \|W_i - \mu_l\|^2 \quad (14)$$

which is an NP-hard problem (Megiddo and Supowit, 1984; Dasgupta, 2008; Aloise et al., 2009).

Practically, we found that the agglomerative clustering algorithm (Sneath and Sokal, 1973) performs well.[8] Specifically, clusters are initialized as individual experts and then iteratively merged, with a termination condition that prevents the experts within each cluster from being too dissimilar. This condition is tuned based on the desired sparsity.

Lastly, if we allow inference on some data, we can improve Eq. 13 with coactivation statistics $a_{i,j}$, which measure the frequency with which E_i, E_j are selected simultaneously.[9] However, these coactivation statistics depend on the given data, whose

[8] We tried other clustering algorithms in the Appendix.

[9] We normalize $a_{i,j}$ by dividing it with the total coactivations in one layer.

distribution may differ from the test data. Therefore, we combine the two as follows:

$$b_{i,j} = -\lambda_1 \|W_i - W_j\|_F + \lambda_2 a_{i,j} \quad (15)$$

We recap the algorithm in the Appendix (Alg 1).

A.2 Towards $O(1)$: Taylor Approximation and Selective Reconstruction

1st-order Taylor Approximation While previous section immensely reduces the cost to obtain the probability distribution to $O(n)$ by requiring only $P(E_i)$s, we can further reduce the number of forward passes– We aim to remove the GPU calls for $P(E_i)$, which is needed as in Eq. 10.

The key idea is approximating E_i's reconstruction loss value $\mathcal{E}_i = \|M(x;\theta) - M(x;\theta - \theta_i)\|_F$, and assigning $P(E_i)$ as some high value L if the reconstruction loss $\mathcal{E}_i$ is lowest. This neatly estimates the rank between $P(E_i)$s.

Though $\mathcal{E}_i$ can be approximated via conventional 2nd-order reconstruction methods (Hassibi and Stork, 1992; Frantar and Alistarh, 2023), the size of the hessian matrix increases quadratically with the number of experts, which often yields out-of-memory errors.

To address this, we propose using a 1st order Taylor approximation. To rank the reconstruction loss values, we consider approximating the reconstruction loss when replacing the output from θ_i with some specific expert θ_C in $C = c(E_i)$ as follows:

$$\mathcal{E}_i = \|E'(\theta_i) \cdot (\theta_i - \theta_C)\|^2 \quad (16)$$

As the convention of 2nd order Taylor approximation (Hassibi and Stork, 1992; Frantar and Alistarh, 2023), we assume the parameters are near a local minimum. Thus, with a small constant γ, $\|E'(\theta_i)\| < \gamma$, leading to:

$$\sum_i \mathcal{E}_i < \sum_i \gamma \|\theta_i - \theta_C\|^2 \quad (17)$$

whose upper bound in the right-hand side can be minimized when $\theta_C = \overline{\theta_i}$, where $\bar{}$ denotes the average.

Therefore, the expert closest to $\bar{\theta_i}$ within each cluster has the highest priority to be retained. We assign $\mathcal{E}_i$ a large number $L > p$ if E_i is the closest to $\bar{\theta_i}$ from the corresponding cluster $c(E_i)$, and set it to zero otherwise. The same greedy algorithm is applied to optimize Eq. 11.

Selective Reconstruction of Experts While letting θ_C as the expert closest to $\bar{\theta_i}$ successfully minimizes $\sum_i \mathcal{E}_i$, sometimes we can minimize them further, by replacing the weight of the closest expert θ_C to $\bar{\theta_i}$. However, blindly doing so is suboptimal, as there is another kind of error to consider. The decision boundaries of the next layer are accustomed to the output of $\{E(x;\theta_i)\}_{i=1}^{|C|}$, but changing the output as $E(x;\theta_C) = E(x;\bar{\theta_i})$ could result in a distribution that the model is unfamiliar with. This potential error, which we denote as $\mathcal{E}_d$, would be minimized if $\theta_C \in \{\theta_i\}_{i=1}^{|C|}$.

To balance these two types of errors, we selectively decide whether to reconstruct. We observe that $\sum_i \mathcal{E}_i$ increases if the total number of clusters in a layer decreases, as this would introduce more $\|E'(\theta_i) \cdot (\theta_i - \theta_C)\|^2$ terms. Therefore, if the total number of clusters is below a threshold κ, we use $\theta_C = \overline{\theta_i}$ to minimize $\sum_i \mathcal{E}_i$. Otherwise, we set θ_C as the expert within the cluster $\{\theta_i\}_{i=1}^{|C|}$ closest to the $\overline{\theta_i}$, to minimize $\mathcal{E}_d$. The router weight reconstruction is done similarly, following its corresponding expert.

B Implementation Details

We probe $(\lambda_1, \lambda_2) \in \{(0,1),(1,0),(1,1)\}$, except for the Snowflake Arctic, which is the biggest MoE we deal with, where we only consider $(\lambda_1, \lambda_2) = (1,0)$, which means no GPU calls is needed for expert pruning. To get coactivation values $a_{i,j}$, we utilize 1000 samples from the C4 dataset, each of which has 2048 sequence length. We evaluate on LM-EVALUATION-HARNESS (Gao et al., 2021) We use 4bit quantization (Dettmers et al., 2023) for experiments with Mixtral-8x22B and Arctic, due to their model size. We use 20%, 12.5%, and 10% for Arctic, Mixtral-8x7B, and Mixtral-8x22B respectively, as the expert sparsity for STUN. These are the maximum values among 10, 12.5, 20, 25, and 35%, with minimum performance loss. We use $\kappa = 3$ for selective reconstruction. For Wanda and OWL, we use 128 C4 samples following the original papers (Yin et al., 2024; Sun et al., 2024), while we use 4096 for sequence length. For OWL, we use the default setting, $M = 5, \lambda = 0.08$.

All experiments are conducted on H100 80GB GPUs, with a maximum of 4. Each evaluation is done within 4 hours, and each unstructured pruning requires less than 2 hours on one GPU. Evaluation is done only once, since we introduce no randomness in our experiment.

Cluster	Reconstruct	LM-eval Avg
Ours	Ours	**59.58**
DSatur	Ours	58.59
Ours	Always	57.60
Ours	Never	59.22

Table 8: Ablation experiments for the first component of STUN, the proposed expert pruning.

C Other Clustering Algorithms

We also considered DSatur (Brélaz, 1979) as a clustering algorithm for Eq. 12, converting into clique-partitioning in a graph where edge $e_{i,j}$ connected if two experts are similar enough as follows,

$$e_{i,j} = \begin{cases} 1 & b_{i,j} >= t_{DSatur} \\ \infty & \text{otherwise} \end{cases} \quad (18)$$

where t_{DSatur} is some threshold to control the sparsity of MoE.

D Ablation Studies

To validate our design of expert pruning in sections A.1 and A.2, we evaluate alternative approaches to expert-prune Mixtral-8x7B at 50% sparsity. Table 8 confirms that our design choices are valid. Our agglomerative clustering algorithm outperforms the DSatur algorithm, an alternative clustering algorithm we discuss in the Appendix. Additionally, selective reconstruction proves superior to always or never reconstructing, as shown in the last two rows. Detailed per-task performance of ablation studies are described in Tables 9, 10. Detailed hyperparameter ablation studies are described in Tables 11, 12. Results with applying expert-pruning only are provided in Table 14.

E Detailed Results for RQ2

Table 13 describes the per-task performance of RQ2.

F Why GSM8K is Harder

The nature of the GSM8K task, which is a generation task, accounts for this discrepancy. A random generation baseline on GSM8K would achieve 0% accuracy, making it far more challenging to maintain performance. In contrast, ARC, HellaSwag, and MMLU are multiple-choice tasks where random baselines can achieve reasonable accuracy by comparing the perplexity of different completion options. As a result, maintaining performance on GSM8K is considerably harder, a challenge often overlooked in previous works. This explains the more pronounced difference in performance between STUN and the baselines on GSM8K.

	ARC-c	ARC-e	BoolQ	HellaSwag	MMLU	OBQA	RTE	WinoGrande	Avg
DSatur	45.99	75.38	80.76	54.62	45.06	29.00	66.79	71.11	58.59
Ours	45.73	75.13	83.46	54.55	53.29	31.20	62.45	70.80	**59.58**

Table 9: Our agglomerative clustering algorithm is better than the alternative.

	ARC-c	ARC-e	BoolQ	HellaSwag	MMLU	OBQA	RTE	WinoGrande	Avg
Always reconstruct ($\kappa = 8$)	42.92	74.20	82.42	54.66	50.09	28.40	55.96	72.14	57.60
No reconstruct ($\kappa = 0$)	45.22	75.21	82.45	54.53	52.31	30.00	62.82	71.19	59.22
Ours ($\kappa = 3$)	45.73	75.13	83.46	54.55	53.29	31.20	62.45	70.80	**59.58**

Table 10: Selective reconstruction outperforms the baselines.

	λ_1	λ_2	GSM8K
Mixtral-8x7B	1	0	**63.38**
	0	1	58.53
	1	1	60.42
Mixtral-8x22B	1	0	81.50
	0	1	**81.58**
	1	1	80.52

Table 11: Comparison of different λ_1, λ_2 configurations. For the Arctic, we only consider the cheapest, (λ_1, λ_2) = (1,0).

	sparsity	GSM8K
Arctic	10%	70.74
	20%	69.90
	35%	60.05
Mixtral-8x7B	12.50%	63.38
	25%	53.22
Mixtral-22B	10%	81.58
	12.50%	79.91
	25%	74.07

Table 12: Comparison of different sparsity of our proposed expert-level pruning. (λ_1, λ_2) is probed among (1,0),(0,1),(1,1), except for 10% and 20% of Arctic.

model	sparsity	method	cost	ARC-c	ARC-e	BoolQ	HellaSwag	MMLU	OBQA	RTE	WinoGrande	Avg
Mixtral-8x7B (Instruct)	0%	unpruned		62.20	87.04	88.50	67.59	68.87	36.60	72.20	76.87	69.98
	25%	Ours	$O(1)$	59.30	85.44	88.13	64.42	64.52	35.40	71.84	75.37	**68.05**
		Lu et al. (2024a)	$O(\frac{k^n}{\sqrt{n}})$	58.19	84.89	87.34	65.24	62.47	35.60	70.04	75.85	67.45
Mixtral-8x7B	0%	unpruned		57.17	84.01	85.35	64.88	67.88	35.00	70.40	75.93	67.58
	25%	Ours	$O(1)$	52.73	81.82	83.09	60.84	63.34	31.60	68.59	72.69	**64.34**
		Lu et al. (2024a)	$O(\frac{k^n}{\sqrt{n}})$	51.62	81.94	83.64	61.60	58.72	33.00	67.87	75.37	64.22

Table 13: Comparing the first component of STUN, the proposed expert pruning, with other baselines.

	sparsity	GSM8K	Avg (→)	ARC-c	ARC-e	HellaSwag	MMLU
Arctic	20%	69.90	67.95	57.00	83.71	65.70	65.39
Mixtral-8x7B	12.50%	63.38	70.65	61.18	86.66	66.80	67.97
Mixtral-8x22B	10%	81.58	71.78	61.18	85.82	66.62	73.52

Table 14: Performance when applying only our expert pruning.