DSMoE: Matrix-Partitioned Experts with Dynamic Routing for Computation-Efficient Dense LLMs

Anonymous ACL submission

Abstract

As large language models continue to scale, computational costs and resource consumption have emerged as significant challenges. While existing sparsification methods like pruning reduce computational overhead, they risk losing model knowledge through parameter removal. This paper proposes DSMoE (**D**ynamic **S**parse **M**ixture-**o**f-**E**xperts), a novel approach that achieves sparsification by partitioning pre-trained FFN layers into computational blocks. We implement adaptive expert routing using sigmoid activation and straight-through estimators, enabling tokens to flexibly access different aspects of model knowledge based on input complexity. Additionally, we introduce a sparsity loss term to balance performance and computational efficiency. Extensive experiments on LLaMA models demonstrate that under equivalent computational constraints, DSMoE achieves superior performance compared to existing pruning and MoE approaches across language modeling and downstream tasks, particularly excelling in generation tasks. Analysis reveals that DSMoE learns distinctive layer-wise activation patterns, providing new insights for future MoE architecture design.

1 Introduction

Large Language Models(LLM) have demonstrated remarkable performance across various downstream tasks(Touvron et al., 2023; Dai et al., 2022; Anil et al., 2023; Biderman et al., 2023). However, as model sizes continue to expand, computational costs and resource consumption grow exponentially. How to improve computational efficiency while maintaining model performance has become a pressing challenge(Cheng et al., 2024).

At the algorithmic level, approaches to model efficiency optimization generally follow two paradigms: post-training compression and acceleration of dense models, or training of Mixture of Experts (MoE) architectures. While compression methods like pruning achieve efficiency through permanent parameter removal(Ashkboos et al., 2024; Ma et al., 2023; Frantar and Alistarh, 2023), they may discard valuable knowledge and lack flexibility in handling inputs of varying complexity. Conversely, although effective, MoE approaches—whether trained from scratch(Fedus et al., 2022; Dai et al., 2024; Liu et al., 2024) or warm-started from dense models(Jiang et al., 2024)—cannot reduce the computational cost of the original dense model. Given that the most widely used and effective foundation models still maintain dense architectures (such as LLaMA(Touvron et al., 2023), Qwen(Bai et al., 2023)), there is a crucial need to optimize these widely-adopted dense models for better efficiency without sacrificing their performance

To address these limitations, we propose DSMoE, which achieves model sparsification by partitioning pre-trained FFN layers into smaller computational blocks. Unlike existing approaches, DSMoE preserves all model parameters, eliminating the risk of knowledge loss, while achieving lower computational overhead than the original dense model through dynamic routing mechanisms. This design maintains the model's complete knowledge while improving computational efficiency.

Extensive experiments conducted on LLaMA-1B and LLaMA-7B models demonstrate encouraging results. Under equivalent computational constraints, our method achieves significant improvements in language modeling perplexity and downstream task performance compared to existing pruning and MoE approaches. Notably superior performance is observed in reasoning and question-answering tasks, particularly in generation tasks.

The main contributions of this work include:

- proposing a novel approach that enables transition from dense to dynamically sparse mod-

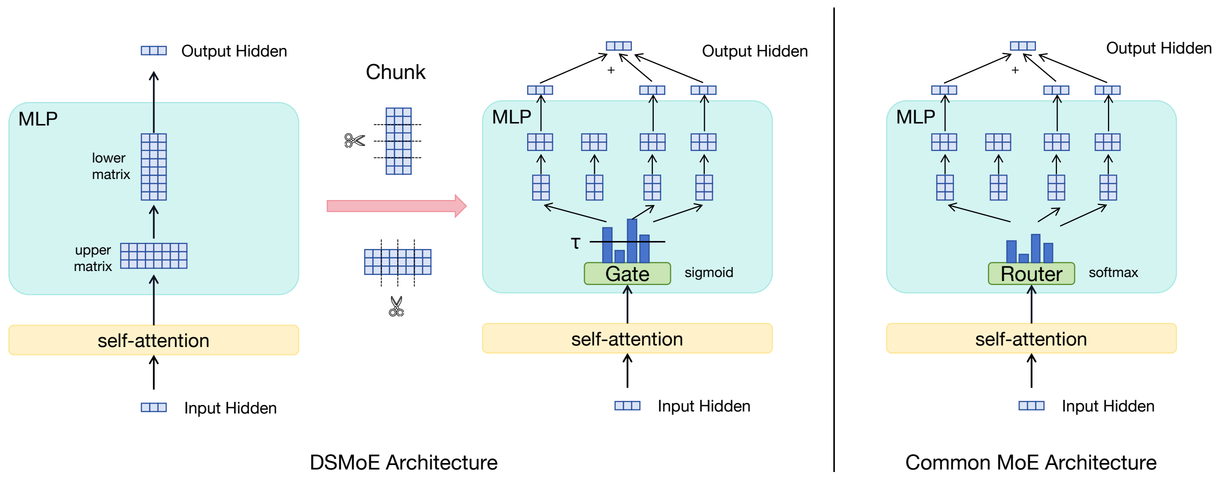


Figure 1: The Overview of DSMoE versus Traditional MoE Framework Architectures. The structure shown in the figure is a simplified representation of the transformer backbone. We have simplified the MLP layer structure here; the MLP layer also includes a gating matrix with dimensions matching the upper matrix, which performs Hadamard multiplication with the upper matrix without affecting our partitioning scheme. In the MLP layer, we partition matrices along the intermediate dimension, where portions corresponding to the original matrix multiplication form new expert MLP layers.

els by preserving and partitioning pre-trained knowledge, enabling different tokens to adaptively access varying portions of model knowledge.

- validating the method's effectiveness across multiple benchmarks through extensive experimentation, providing new insights for MoE large model optimization.

2 Related Work

Model pruning is an effective approach to achieving sparse LLMs while maintaining model functionality. Pruning methods can be categorized into two main types: unstructured and structured pruning. Unstructured pruning operates at the weight level, allowing for arbitrary weight removal (Lee et al., 2018). In large language models, pruned weights are set to zero (Frantar and Alistarh, 2023; Sun et al., 2023). However, this method requires specialized hardware and software support for acceleration(Han et al., 2015; Wen et al., 2016; Filters'Importance, 2016; Tang et al., 2021). Structured pruning takes a coarser-grained approach by removing complete structural units such as convolution kernels, channels, attention heads, or entire layers (You et al., 2019; Ashkboos et al., 2024; Liu et al., 2021; Ma et al., 2023; Men et al.). Its main advantage is the ability to directly produce regular, narrow model architectures that can achieve acceleration without specialized sparse computation libraries (Luo et al., 2017; Liu et al., 2021; Filters'Importance, 2016; Nonnenmacher et al., 2021). However, both approaches face a fundamental limitation: achieving efficiency through permanent parameter removal may discard valuable knowledge and lose the ability to adapt computation based on input complexity.

In recent years, there has been growing interest in exploring sparse computation in large language models. Mixture-of-Experts (MoE) represents a pioneering approach that demonstrates how sparse activation can effectively balance model capacity and computational efficiency. In MoE architectures, only a subset of FFN modules (experts) are activated for each input token (Fedus et al., 2022; Lepikhin et al., 2021; Huang et al., 2024). This foundational idea of conditional computation has inspired various innovations in expert activation strategies. Some works explore heterogeneous expert architectures (Sun et al., 2024) or introduce zero-computation experts (Jin et al., 2024) to further optimize computational efficiency. These advances in MoE architectures demonstrate the potential of sparse computation and motivate our exploration of applying similar principles within individual FFN layers.

3 Background

For simplicity, we focus on the prevalent architecture of generative large language models while maintaining a concise mathematical formulation.

In autoregressive generation tasks, given a sequence $X = (x_1, x_2, ..., x_T)$ of length T, the model iteratively produces a probability distribution over the vocabulary for each position conditioned on preceding tokens. This process can be formulated as:

$$\begin{aligned} P_{\cdot,t} &= \text{softmax}(EH^L_{\cdot,t}) \\ H^L &= \text{Transformer}(x_1, x_2, ..., x_{T-1}) \end{aligned} \tag{1}$$

Here, L denotes the number of layers in the Transformer architecture. For any position t, $P_{\cdot,t}$ represents the probability distribution over the vocabulary, derived from the t-th column of the hidden state matrix h^L. Specifically, $H^L = [h^L_1, h^L_2, ..., h^L_{T-1}]$ contains the hidden representations from the final layer, where h^L_t is the contextual embedding at position t. The probability of the ground-truth token x_{t+1} is denoted as $P_{x_{t+1},t}$ in the distribution $P\cdot, t$. The transformation from hidden states to probability distributions is achieved through a linear projection matrix E, followed by a softmax operation.

In typical scenarios, we employ cross-entropy loss for autoregressive learning, which can be expressed as:

$$\mathcal{L}_{\text{LM}} = -\sum_{t=1}^{T-1} \log P(x_{t+1}|x_{\leq t}) \tag{2}$$

The Transformer architecture consists of multiple layer-wise submodules, where each layer comprises a self-attention module and a Feed-Forward Network (FFN) module. The simplified mathematical formulation can be expressed as:

$$\hat{h}^l_t = \text{Attn}([h^{l-1}_1, h^{l-1}_2, ..., h^{l-1}_t]) \tag{3}$$

$$h^l_t = \text{FFN}(\hat{h}^l_t) \tag{4}$$

FFN modules typically consist of two matrix transformations with a non-linear activation function. In modern language models, the most prevalent FFN implementation uses SwiGLU activation, which involves three essential matrices: the up-projection matrix $\mathbf{U}_{\text{up}}$, the down-projection matrix $\mathbf{V}_{\text{down}}$, and the gate matrix $\mathbf{W}_{\text{gate}}$. The up-projection matrix transforms the input to a higher dimensional space for richer feature representation, the down-projection matrix compresses the information back to the original dimension, and the gate matrix controls information flow through adaptive feature weighting. The FFN output is computed through the following operation:

$$h^l_t = (act(\hat{h}^l_t W_{gate}) \odot (\hat{h}^l_t U_{up}))V_{down} \tag{5}$$

In this formulation, act$(\cdot)$ represents the activation function and $\odot$ denotes Hadamard product.

4 Method

Although our method is termed DSMoE, its training approach differs from traditional MoE methods such as Switch Transformer (Fedus et al., 2022) and DeepSeeKMoE (Dai et al., 2024). Our objective is to achieve sparsity through partitioning pre-trained models, where each expert inherits a distinct portion of the original model's knowledge. Our approach is based on the principle that the model should learn to selectively utilize different aspects of pre-trained knowledge based on input complexity, rather than routing tokens among independently trained experts. To implement this insight, we present our method in three modules.

4.1 MLP Partitioning

The widespread adoption of MoE architectures inspires our exploration of sparsity in FFN layers, suggesting that different parts of computation can be dynamically activated based on input patterns. Previous work has further revealed that FFN layers essentially operate as key-value memories, where different portions of the layer specialize in detecting and processing distinct input patterns(Geva et al., 2020). Building on these insights, we propose to directly partition pre-trained FFN layers. As shown in Equation 5, we partition the matrices $\mathbf{U}$, $\mathbf{V}$, and $\mathbf{W}$ into n groups along the intermediate dimension, where each group can be viewed as an "expert" that inherits a portion of the original transformation capabilities. When summing all expert outputs, this partitioned form yields identical results to the original MLP computation:

$$\begin{aligned} h^l_t = (act(\hat{h}^l_t &\begin{bmatrix} W_1 & \cdots & W_n \end{bmatrix}) \odot \\ (\hat{h}^l_t &\begin{bmatrix} U_1 & \cdots & U_n \end{bmatrix})) \begin{bmatrix} V_1 \\ \vdots \\ V_n \end{bmatrix} \\ = (act(\hat{h}^l_t W_1) &\odot \hat{h}^l_t U_1)V_1 + \cdots \\ &+ (act(\hat{h}^l_t W_n) \odot \hat{h}^l_t U_n)V_n \end{aligned} \tag{6}$$

We can structurally split the original MLP layer matrix into multiple small MLP matrices. To enable dynamic expert activation based on input, we employ a gating network that determines which experts should be activated. The expert's output is propagated to the subsequent layer only when the corresponding gating activation value exceeds a certain threshold τ. This can be formulated as:

$$\begin{gathered}
o_i = (act(\hat{h_t^l} W_i) \odot \hat{h_t^l} U_i) V_i \\
h_t^l = \sum_{i=1}^{n} o_i * G(\sigma(\hat{h_t^l} \mathbf{Y}_i)) \\
G(x) = \begin{cases} x & \text{if } x > \tau \\ 0 & \text{others} \end{cases}
\end{gathered} \tag{7}$$

where $\mathbf{Y} = [\mathbf{Y}_1, \dots, \mathbf{Y}_n] \in \mathbb{R}^{d \times n}$ represents the parameters of the gating network, and $\sigma(\cdot)$ denotes the sigmoid activation function.

To maintain consistent output norm regardless of the number of active experts, similar to dropout, we scale h_t^l by the ratio of total expert count n to the number of activated experts. This normalization can be expressed as:

$$h_t^l = \frac{n \cdot h_t^l}{\sum_{i=1}^{n} \mathbb{I}[\sigma(\hat{h}_t^l \mathbf{Y}_k) > \tau]} \tag{8}$$

4.2 Straight-Through Estimator

A key challenge in converting dense models to sparse ones is maintaining the learning capability of all experts. During the forward pass, experts with activation values below the threshold τ do not participate in computation, as defined by the gating function $G(x)$ in Equation 7. However, this thresholding operation creates a critical problem during backpropagation - experts that are not activated receive zero gradients:

$$\begin{gathered}
\frac{\partial h_t^l}{\partial \mathbf{V}_i} = \frac{\partial h_t^l}{\partial \mathbf{W}_i} = \frac{\partial h_t^l}{\partial \mathbf{U}_i} = \\
\frac{\partial h_t^l}{\partial \mathbf{Y}_i} = \mathbf{0}, \text{if } \sigma(\hat{h}_t^l \mathbf{Y}_i) \leq \tau
\end{gathered} \tag{9}$$

This gradient blocking prevents non-activated experts from receiving training signals, leading to a "dead expert" problem where these experts become permanently inactive. Unlike traditional MoE models that train experts from scratch, our experts inherit pre-trained knowledge that we wish to preserve and adapt. To address this issue, we employ the straight-through estimator technique, which allows gradient flow through non-activated experts while maintaining thresholded activation during the forward pass:

$$S(x) = sg(G(x)) + x - sg(x) \tag{10}$$

$$h_t^l = \sum_{i=1}^{n} o_i \cdot S(\sigma(\hat{h}_t^l \mathbf{Y}_k)) \tag{11}$$

where the operator "$sg(\cdot)$" is the "`stop gradient`" operator to prevent gradient back propagation. The partial derivatives for experts and their gates below the threshold are as follows. Let:

$$\begin{aligned}
a_i &= \text{act}(\hat{h}_t^l \mathbf{W}_i) \\
a_i' &= \text{act}'(\hat{h}_t^l \mathbf{W}_i) \\
g_i &= \sigma(\hat{h}_t^l \mathbf{Y}_i) \\
u_i &= \hat{h}_t^l \mathbf{U}_i
\end{aligned} \tag{12}$$

The gradients for expert parameters and their gates can be derived as:

$$\frac{\partial h_t^l}{\partial \mathbf{V}_i} = \begin{cases} (a_i \odot u_i)^\top \cdot g_i & \text{if } g_i > \tau \\ \mathbf{0} & \text{if } g_i \leq \tau \end{cases} \tag{13}$$

$$\frac{\partial h_t^l}{\partial \mathbf{W}_i} = \begin{cases} (\hat{h}_t^l)^\top \odot a_i' \cdot ((u_i \odot \mathbf{V}_i) \cdot g_i) & \text{if } g_i > \tau \\ \mathbf{0} & \text{if } g_i \leq \tau \end{cases} \tag{14}$$

$$\frac{\partial h_t^l}{\partial \mathbf{U}_i} = \begin{cases} (\hat{h}_t^l)^\top \cdot (a_i \odot \mathbf{V}_i \cdot g_i) & \text{if } g_i > \tau \\ \mathbf{0} & \text{if } g_i \leq \tau \end{cases} \tag{15}$$

$$\frac{\partial h_t^l}{\partial \mathbf{Y}_i} = (\hat{h}_t^l)^\top \cdot (o_i \cdot \sigma'(\hat{h}_t^l \mathbf{Y}_i)) \tag{16}$$

The gradient dynamics reveal an important property: since $\sigma'(\hat{h}_t^l \mathbf{Y}_i) > 0$, an expert that produces meaningful output o_i for an input $(\hat{h}_t^l)^\top$ will receive gradients that increase its activation probability for similar inputs in future iterations, regardless of its current activation status. This adaptive mechanism ensures that experts can learn to specialize in processing specific input patterns while maintaining their inherited knowledge from pre-training.

4.3 Sparse Loss

Since our experts inherit from a dense model, the model naturally tends to activate all experts to access complete knowledge. However, this conflicts with our goal of sparse computation. We introduce a sparsity loss term that creates an adversarial effect with expert gate gradients, encouraging the

Model	Configuration	Params	Activated Params	FLOPs	PPL (↓)
LLaMA-1B	d=2048, D=8192	1.24B	1.24B	2.53 T	5.67
LLaMA-7B	d=4096, D=11008	6.74B	6.74B	13.53 T	3.40
LLaMA-1B					
LLM-Pruner-channel	d=1215, D=8192	889M	889M	1.50 T	7.51
LLM-Pruner-block	d=2048, D=3896.4	735M	735M	1.50 T	7.46
SparseGPT	d=2048, D=8192	1.24B	1.24B	2.53 T	9.82
MoE	d=2048, D=1024 $\times$8, topK=3	1.24B	736M	1.50 T	7.45
DSMoE(ours)	d=2048, D=1024 $\times$8	1.24B	735M	1.50 T	**7.41**
LLaMA-7B					
LLM-Pruner-channel	d=2401, D=11008	3.95B	3.95B	7.93 T	4.01
LLM-Pruner-block	d=11008, D=6256.5	3.94B	3.94B	7.93 T	4.01
SparseGPT	d=4096, D=11008	6.74B	6.74B	13.53 T	3.96
MoE	d=2048, D=1376 $\times$8, topK=3	6.74B	3.98B	7.99 T	4.12
DSMoE(ours)	d=2048, D=1376 $\times$8	6.74B	3.93B	7.91 T	**3.91**

Table 1: Results of perplexity (PPL) across different language models. The **bold** values indicate the best-performing method among various acceleration approaches. The Configuration column describes the specific model architecture, where d represents the hidden dimension, D denotes the expansion dimension in MLP layers (for LLM-Pruner-block method, this represents the average value), $\times$ n indicates the use of n parallel MLP layers, and topK specifies the number of activated experts per layer in the MoE architecture. The Params column shows the total number of model parameters, while Activated Params indicates the average number of parameters activated during inference. FLOPs (Floating Point Operations) represents the average number of floating-point operations per sample.

model to learn which knowledge is truly necessary for different inputs.

$$\mathcal{L} = \mathcal{L}_{\text{LM}} + \lambda \mathcal{L}_{\text{sparse}} \tag{17}$$

where $\mathcal{L}_{\text{sparse}}$ denotes the sparsity loss term, which we abbreviate as $\mathcal{L}$s in subsequent equations. The hyperparameter λ controls the strength of sparsity regularization, with larger values encouraging sparser activation patterns.

$$\mathcal{L} = \mathcal{L}_{\text{LM}} + \frac{\lambda}{LN} \sum_{l=1}^{L} \sum_{n=1}^{N} \mathcal{L}_s(G(\sigma(\hat{h}_t^l \mathbf{Y}_n))) \tag{18}$$

We employ $L1$ norm as the sparsity function $\mathcal{L}_s$. Given that our activation function $\sigma(x) > 0$, our final loss function becomes:

$$\mathcal{L} = \mathcal{L}_{LM} + \frac{\lambda}{LN} \sum_{l=1}^{L} \sum_{n=1}^{N} G(\sigma(\hat{h}_t^l \mathbf{Y}_n)) \tag{19}$$

The gradients introduced by this sparse loss term create an adversarial effect with the gate gradients, encouraging the model to actively suppress the output of less important experts across different layers.

It is worth noting that our approach differs fundamentally from the MoE framework and therefore does not require auxiliary load balancing losses. While load balancing losses in MoE aim to ensure uniform training across experts, our objective is solely focused on learning sparse activation patterns. Furthermore, unlike MoE which typically enforces a fixed number of active experts, our method allows for flexible activation patterns determined by the learned gating mechanism.

5 Experiments

5.1 Dataset

We gathered datasets from various domains to continually pre-train the base model. For the general domain, we used the Fineweb-edu dataset, which consists of high-quality educational web pages filtered from the Fineweb dataset (Penedo et al., 2024). In the math and coding domains, we selected the OpenWebMath (Paster et al., 2024) and StarCoder (Li et al., 2023) datasets respectively. The OpenWebMath dataset contains high-quality mathematical text data extracted from web pages, while the StarCoder dataset offers a diverse range of code data and has been demonstrated to effectively pre-train well-behaved code models. Furthermore, it has been demonstrated that incorporating synthetic data enhances model pre-training performance (Abdin et al., 2024). Therefore, we introduced the Cosmopedia dataset to leverage this advantage(Ben Allal et al., 2024).

Furthermore, we mixed datasets from different

Model	Hellaswag	LAMBADA	PIQA	SIQA	StoryCloze	Wino	GSM8K	TriviaQA	WebQs	NatrualQs
LLaMA-1B	64.09	61.05	75.51	42.47	72.58	60.85	4.85	12.52	36.08	22.49
LLaMA-7B	76.39	72.34	79.05	44.67	79.15	70.87	14.70	26.28	61.89	32.82
LLaMA-1B										
LLM-Pruner-channel	53.44	45.04	71.43	40.94	68.67	**58.45**	1.44	6.98	17.46	14.56
LLM-Pruner-block	51.05	46.28	71.71	41.04	68.62	56.27	1.36	7.28	18.46	14.56
SparseGPT	**54.01**	**56.49**	71.10	40.68	68.05	57.30	1.51	5.29	14.44	11.61
MoE	49.06	44.84	70.02	41.05	65.47	55.64	1.62	5.76	13.49	11.27
DSMoE(ours)	50.92	48.12	**72.36**	**41.14**	**68.78**	56.35	**1.67**	**8.17**	**25.52**	**18.21**
LLaMA-7B										
LLM-Pruner-channel	66.41	61.63	74.97	43.19	75.30	66.85	4.85	12.63	36.02	20.57
LLM-Pruner-block	67.93	62.02	76.22	44.26	75.46	63.53	1.81	12.96	38.77	21.65
SparseGPT	**73.60**	67.43	77.36	44.21	**76.37**	**70.48**	**8.33**	17.61	47.83	24.90
MoE	63.89	60.49	74.10	43.29	72.90	61.17	3.26	11.58	31.25	19.09
DSMoE(ours)	70.22	**67.61**	**78.12**	**44.31**	**76.37**	66.77	6.41	**22.04**	**57.94**	**29.92**

Table 2: Performances of language models on downstream tasks. The best score is marked in **bold**.

domains. Due to computational resource limitations, we set the total amount of training data to 10 billion tokens. Finally, we used the tokenizers from LLaMA to segment the data, limiting the maximum sample length to 1024 tokens for each. We randomly sampled 5,000 non-overlapping instances from each dataset as the validation set, ensuring no intersection with the training set.

5.2 Experimental Setup

We evaluate DSMoE on two pre-trained models of different scales: Llama-7B[1] and Llama-1B[2]. For our method's hyperparameters, we simply set the activation threshold $\tau = 0.5$ and the sparsity regularization coefficient $\lambda = 1.0$.

We compare our approach with several baselines: the channel-wise and block-wise methods from LLM-Pruner (a structured pruning approach), and SparseGPT (an unstructured pruning method). To ensure fair comparison, we first measure the FLOPs of our trained model, then estimate the pruning ratio for baseline methods to maintain a slightly higher FLOPs than our method. The FLOPs metric directly corresponds to the number of parameters involved in computation, providing a standardized measure of computational efficiency.

Additionally, we explore an alternative approach by applying the same FFN partitioning scheme but training it as a traditional MoE architecture (with fixed expert selection and standard MoE training objectives) to investigate whether the conventional MoE framework better accommodates the warm-starting paradigm.

[1] `https://huggingface.co/meta-llama/Llama-2-7b`

[2] `https://huggingface.co/meta-llama/Llama-3.2-1B`

5.3 Main Results

We first present the model's perplexity on the validation set. Following previous work(Touvron et al., 2023; Brown et al., 2020; Su et al., 2024; Dai et al., 2024), we then evaluate the model's performance on downstream benchmarks, which includes zero-shot accuracy testing on HellaSwag(Zellers et al., 2019), LAMBADA(Paperno et al., 2016), SIQA(Sap et al., 2019), PIQA(Bisk et al., 2020), StoryCloze(Mostafazadeh et al., 2016), and Winogrande(Sakaguchi et al., 2021). Additionally, we conduct 5-shot evaluation measuring exact match performance on TriviaQA(Joshi et al., 2017), WebQuestions (WebQs)(Berant et al., 2013), GSM8K(Cobbe et al., 2021), and Natural Questions (NaturalQs)(Kwiatkowski et al., 2019).

5.3.1 Perplexity Results

Table 1 presents the perplexity results of the baseline dense model and its pruned, sparsified variants. The results demonstrate that DSMoE consistently outperforms baseline models under equivalent activation constraints. Since SparseGPT acceleration requires specific pruning ratios and hardware support, we conducted our comparative analysis only on models with equivalent parameter pruning levels. Our experimental results indicate that DSMoE achieves superior efficiency compared to static parameter pruning. Furthermore, DSMoE exhibits better performance than fixed-activation methods like MoE, which can be attributed to the fact that knowledge from all experts contributes to the model's learning process, enabling it to develop the ability to flexibly select activations based on input. Additionally, DSMoE exhibits distinctive feature processing capabilities, learning layer-specific activation patterns that naturally emerge from the input complexity. We will examine these emergent patterns in detail in the analysis section.

In conclusion, DSMoE demonstrates consistent superiority across models of two different scales, highlighting its robust advantages.

5.3.2 Benchmark Results

Table 2 presents the benchmark performance of various pruning methods, traditional MoE approaches, and DSMoE. DSMoE achieved the best performance in 7 out of 10 benchmarks for both LLaMA-1B and LLaMA-7B model architectures, demonstrating superior effectiveness over existing sparsification methods across most evaluation metrics.

Specifically, DSMoE exhibited excellent performance on inference tasks (i.e., the first 6 benchmarks), achieving the best results on PIQA, SIQA, and StoryCloze test sets. While not achieving top performance on Hellaswag, LAMBADA, and Wino test sets, DSMoE still ranked among the leading models. For generation tasks (i.e., the last 4 benchmarks), DSMoE demonstrated remarkable effectiveness. Apart from slightly lower performance on GSM8K with LLaMA-7B compared to SparseGPT, it significantly outperformed other sparse methods on all other test sets, with performance only a few points below the dense model. These results highlight DSMoE's potential, particularly in generation tasks.

Furthermore, we observed that the performance gap between DSMoE and other sparse approaches was more pronounced in LLaMA-7B compared to LLaMA-1B. This may be attributed to greater model redundancy at larger parameter scales, enabling DSMoE to more effectively prune unnecessary information. This observation suggests the potential scalability of DSMoE to models with larger parameter counts.

6 Analyses

6.1 Ablation Study: Removing Straight-Through Estimator

To validate the necessity of the straight-through estimator mechanism in DSMoE, we conduct an ablation study by removing this component. Specifically, instead of using Equation (11) for training, we employ Equation (7). We perform this comparative analysis on the LLaMA-1B model.

As shown in Table 3, the model without straight-through estimator significantly underperforms the complete model in terms of both perplexity and benchmark performance. This substantial degradation occurs because routing parameters for non-activated experts receive zero gradients during backpropagation, preventing these routes from being adjusted to utilize more of the pre-trained knowledge inherited from the dense model. Without the ability to adaptively modify routing decisions, potentially valuable knowledge encoded in these experts becomes permanently inaccessible, leading to significant performance loss.

Model	DSMoE	*w/o* $S(x)$
Hellaswag	50.92	32.29
LAMBADA	48.12	27.79
PIQA	72.36	62.73
SIQA	41.14	39.30
StoryCloze	68.67	57.14
Wino	56.35	50.83
GSM8K	1.67	0.38
TriviaQA	8.17	2.47
WebQs	25.52	2.95
NatrualQs	18.21	1.00
PPL	7.41	12.75

Table 3: Ablation study of DMoE against the model without direct estimation function S(x), where G(x) is employed in place of S(x).

6.2 Ablation Study: Training without Piecewise Function G(x)

To validate the necessity of incorporating piecewise function learning during training, we conduct an ablation study by removing the piecewise function G(x) and using the following formula for training:

$$h_t^l = \sum_{i=1}^{n} o_i * \sigma(\hat{h}_t^l Y_i) \tag{20}$$

Prior to inference, we determine the appropriate activation level by adjusting the threshold value on the validation set, with a step size of 0.05. Figure 2 illustrates the relationship between perplexity and the average number of activated experts on the validation set.

The results clearly demonstrate that as the threshold increases, perplexity rises rapidly while the average number of activated experts decreases correspondingly. This observation indicates that without the piecewise function G(x), all experts participate in computation and gradient updates. Under the constraint of sparsity loss, the model tends to distribute activation values uniformly across all experts rather than learning to distinctively identify more important experts. This leads to two conse-

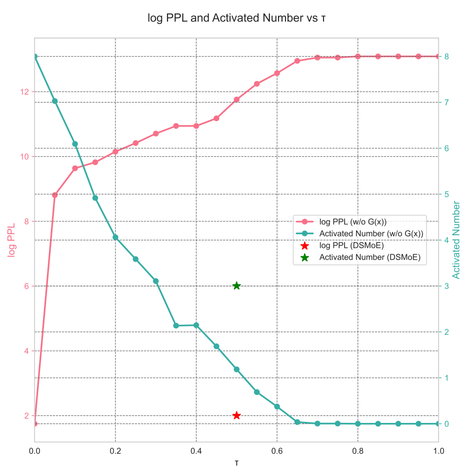
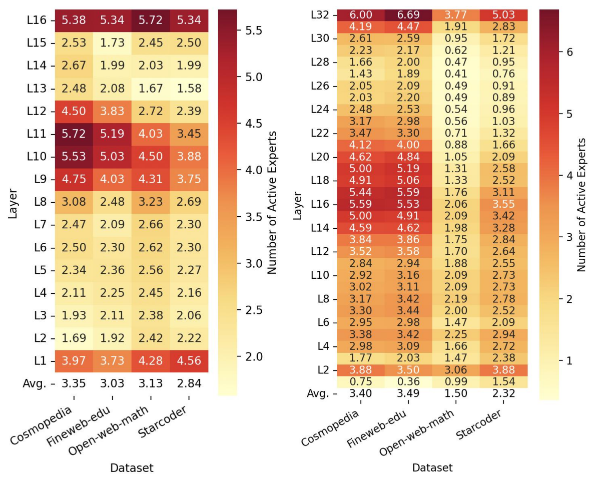


Figure 2: During the training phase, G(x) is not utilized. In the inference phase, G(x) is employed for activation. The model's perplexity and the number of activated experts vary with the threshold τ. The pentagram markers indicate the perplexity and number of activated experts achieved by DSMoE.

quences: first, the activation values for each expert are suppressed to a relatively low level, and second, the learned importance of each expert becomes relatively uniform. Under the same activation constraints as DSMoE, the approach without the piecewise function G(x) exhibits higher perplexity, highlighting how this training-inference inconsistency significantly degrades model performance.

6.3 Layer-wise Activation Patterns Analysis



(a) Heatmap for 1B model (b) Heatmap for 7B model

Figure 3: Heatmap visualization of expert activation counts across different layers and average expert activations for LLaMA-7B and LLaMA-1B models on various validation sets.

We evaluated DSMoE across different validation sets and generated heatmaps to visualize the distribution of activated experts across network layers. Both model sizes exhibit a distinctive activation pattern: higher activation counts at both input and output layers, elevated activation in middle layers, and lower activation in remaining layers - forming a "W-shaped" pattern.

The bottom layers, which typically encode fundamental features, demonstrate high expert activation. This suggests the model's tendency to activate multiple experts in parallel to process multi-dimensional input features, potentially serving as an "information preservation mechanism" to retain critical base-level information. The top layers, responsible for final decision-making and output generation, show increased expert activation to enhance output robustness by reducing individual expert bias through collective decision-making. The elevated activation in middle layers suggests these layers serve as critical zones for feature transformation, integration, and processing of long-range dependencies. This bottom-middle-top activation pattern forms a complete information processing pipeline: bottom layers for extensive collection and processing of basic features, middle layers for feature transformation and information integration, and top layers for comprehensive decision-making and output generation.

Furthermore, we observed significant variations in both the average number of activated experts and activation patterns across different test sets. This indicates that DSMoE implements dynamic regulation mechanisms specific to different inputs rather than converging to a homogeneous learning pattern.

These observations provide novel insights for future MoE architectures, suggesting that expert activation counts can be strategically varied across different layers of the network.

7 Conclusion

This paper presents DSMoE, a novel approach that achieves model sparsification by partitioning pretrained FFN layers into computational blocks. Experiments on LLaMA models demonstrate superior performance over existing pruning and MoE approaches under equivalent computational constraints, while revealing distinctive layerwise activation patterns for future MoE designs.

8 Limitations

Due to computational resource constraints, we were only able to evaluate DSMoE on language models up to 7B parameters. Future work with access to larger computational resources could explore the scalability and effectiveness of our approach on larger model architectures, which may reveal additional insights about the relationship between model scale and dynamic sparsification patterns.

References

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J Hewett, Mojan Javaheripi, Piero Kauffmann, et al. 2024. Phi-4 technical report. *arXiv preprint arXiv:2412.08905*.

Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. 2023. Gemini: A family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 1.

Saleh Ashkboos, Maximilian L Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. 2024. Slicegpt: Compress large language models by deleting rows and columns. *arXiv preprint arXiv:2401.15024*.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Loubna Ben Allal, Anton Lozhkov, Guilherme Penedo, Thomas Wolf, and Leandro von Werra. 2024. Cosmopedia.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1533–1544.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Hongrong Cheng, Miao Zhang, and Javen Qinfeng Shi. 2024. A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

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

Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y Wu, et al. 2024. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv preprint arXiv:2401.06066*.

Damai Dai, Li Dong, Shuming Ma, Bo Zheng, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Stablemoe: Stable routing strategy for mixture of experts. *arXiv preprint arXiv:2204.08396*.

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.

Determine Filters' Importance. 2016. Pruning filters for efficient convnets.

Elias Frantar and Dan Alistarh. 2023. Sparsegpt: Massive language models can be accurately pruned in one-shot.(2023). *URL https://arxiv. org/abs/2301.00774*.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2020. Transformer feed-forward layers are key-value memories. *arXiv preprint arXiv:2012.14913*.

Song Han, Huizi Mao, and William J Dally. 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*.

Quzhe Huang, Zhenwei An, Nan Zhuang, Mingxu Tao, Chen Zhang, Yang Jin, Kun Xu, Liwei Chen, Songfang Huang, and Yansong Feng. 2024. Harder tasks need more experts: Dynamic routing in moe models. *CoRR*, abs/2403.07652.

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, et al. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Peng Jin, Bo Zhu, Li Yuan, and Shuicheng Yan. 2024. Moe++: Accelerating mixture-of-experts methods with zero-computation experts. *arXiv preprint arXiv:2410.07348*.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. 2018. Snip: Single-shot network pruning based on connection sensitivity. *arXiv preprint arXiv:1810.02340*.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2021. Gshard: Scaling giant models with conditional computation and automatic sharding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, Qian Liu, Evgenii Zheltonozhskii, Terry Yue Zhuo, Thomas Wang, Olivier Dehaene, Mishig Davaadorj, Joel Lamy-Poirier, João Monteiro, Oleh Shliazhko, Nicolas Gontier, Nicholas Meade, Armel Zebaze, Ming-Ho Yee, Logesh Kumar Umapathi, Jian Zhu, Benjamin Lipkin, Muhtasham Oblokulov, Zhiruo Wang, Rudra Murthy V, Jason T. Stillerman, Siva Sankalp Patel, Dmitry Abulkhanov, Marco Zocca, Manan Dey, Zhihan Zhang, Nour Fahmy, Urvashi Bhattacharyya, Wenhao Yu, Swayam Singh, Sasha Luccioni, Paulo Villegas, Maxim Kunakov, Fedor Zhdanov, Manuel Romero, Tony Lee, Nadav Timor, Jennifer Ding, Claire Schlesinger, Hailey Schoelkopf, Jan Ebert, Tri Dao, Mayank Mishra, Alex Gu, Jennifer Robinson, Carolyn Jane Anderson, Brendan Dolan-Gavitt, Danish Contractor, Siva Reddy, Daniel Fried, Dzmitry Bahdanau, Yacine Jernite, Carlos Muñoz Ferrandis, Sean Hughes, Thomas Wolf, Arjun Guha, Leandro von Werra, and Harm de Vries. 2023. Starcoder: may the source be with you! *Trans. Mach. Learn. Res.*, 2023.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Liyang Liu, Shilong Zhang, Zhanghui Kuang, Aojun Zhou, Jing-Hao Xue, Xinjiang Wang, Yimin Chen, Wenming Yang, Qingmin Liao, and Wayne Zhang. 2021. Group fisher pruning for practical network compression. In *International Conference on Machine Learning*, pages 7021–7032. PMLR.

Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. 2017. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision*, pages 5058–5066.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. Llm-pruner: On the structural pruning of large language models. *Advances in neural information processing systems*, 36:21702–21720.

Xin Men, Mingyu Xu, Qingyu Zhang, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. Shortgpt: Layers in large language models are more redundant than you expect, 2024. *URL https://arxiv. org/abs/2403.03853*.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and evaluation framework for deeper understanding of commonsense stories. *arXiv preprint arXiv:1604.01696*.

Manuel Nonnenmacher, Thomas Pfeil, Ingo Steinwart, and David Reeb. 2021. Sosp: Efficiently capturing global correlations by second-order structured pruning. *arXiv preprint arXiv:2110.11395*.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The lambada dataset: Word prediction requiring a broad discourse context. *arXiv preprint arXiv:1606.06031*.

Keiran Paster, Marco Dos Santos, Zhangir Azerbayev, and Jimmy Ba. 2024. Openwebmath: An open dataset of high-quality mathematical web text. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, Thomas Wolf, et al. 2024. The fineweb datasets: Decanting the web for the finest text data at scale. *arXiv preprint arXiv:2406.17557*.

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.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. 2019. Socialiqa: Commonsense reasoning about social interactions. *arXiv preprint arXiv:1904.09728*.

Zhenpeng Su, Xing Wu, Zijia Lin, Yizhe Xiong, Minxuan Lv, Guangyuan Ma, Hui Chen, Songlin Hu, and Guiguang Ding. 2024. Cartesianmoe: Boosting knowledge sharing among experts via cartesian product routing in mixture-of-experts. *arXiv preprint arXiv:2410.16077*.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. 2023. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*.

Xingwu Sun, Yanfeng Chen, Yiqing Huang, Ruobing Xie, Jiaqi Zhu, Kai Zhang, Shuaipeng Li, Zhen Yang, Jonny Han, Xiaobo Shu, et al. 2024. Hunyuan-large: An open-source moe model with 52 billion activated parameters by tencent. *arXiv preprint arXiv:2411.02265*.

Yehui Tang, Yunhe Wang, Yixing Xu, Yiping Deng, Chao Xu, Dacheng Tao, and Chang Xu. 2021. Manifold regularized dynamic network pruning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5018–5028.

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

Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. 2016. Learning structured sparsity in deep neural networks. *Advances in neural information processing systems*, 29.

Zhonghui You, Kun Yan, Jinmian Ye, Meng Ma, and Ping Wang. 2019. Gate decorator: Global filter pruning method for accelerating deep convolutional neural networks. *Advances in neural information processing systems*, 32.

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