Self-Adaptive Graph Mixture of Models

Mohit Meena, Yash Punjabi, Abhishek A, Vishal Sharma*, Mahesh Chandran

Fujitsu Research of India, Bangalore
{mohitkumar.meena, yash.punjabi, abhishek.a, mahesh.chandran}@fujitsu.com, i.vishal1990@gmail.com

Abstract

Graph Neural Networks (GNNs) have emerged as powerful tools for learning over graph-structured data, yet recent studies have shown that their performance gains are beginning to plateau. In many cases, well-established models such as GCN and GAT, when appropriately tuned, can match or even exceed the performance of more complex, state-of-the-art architectures. This trend highlights a key limitation in the current landscape: the difficulty of selecting the most suitable model for a given graph task or dataset. To address this, we propose Self-Adaptive Graph Mixture of Models (SAGMM), a modular and practical framework that learns to automatically select and combine the most appropriate GNN models from a diverse pool of architectures. Unlike prior mixture-of-experts approaches that rely on variations of a single base model, SAGMM leverages architectural diversity and a topology-aware attention gating mechanism to adaptively assign experts to each node based on the structure of the input graph. To improve efficiency, SAGMM includes a pruning mechanism that reduces the number of active experts during training and inference without compromising performance. We also explore a training-efficient variant in which expert models are pretrained and frozen, and only the gating and task-specific layers are trained. We evaluate SAGMM on 16 benchmark datasets covering node classification, graph classification, regression, and link prediction tasks, and demonstrate that it consistently outperforms or matches leading GNN baselines and prior mixture-based methods, offering a robust and adaptive solution for real-world graph learning.

Code — https://github.com/ast-fri/SAGMM

1 Introduction

Graph Neural Network (GNN) architectures have seen substantial progress since their early formulation in seminal work on graph-based semi-supervised learning (Kipf and Welling 2016). Though developments in the neural architecture have led to an improved performance (on an average), there is still no consensus on distinct advantage of one model over others. This is reflected in two important observations in recent times: (a) the performance of GNN models have plateaued with no significant performance gain reported with complex architecture. (b) slight tuning of hyperparameters of classical GNN models like GCN (Kipf and Welling 2016), GAT (Veličković et al. 2017), and GraphSAGE (Hamilton, Ying, and Leskovec 2017a) have shown state-of-the-art (SOTA) performance in node classification tasks, matching or even surpassing latest Graph Transformers (GTs) across diverse datasets (Luo, Shi, and Wu 2024). This suggests different models tend to learn different (overlapping) regions of the representation space, but each falling short of covering the space to capture diverse patterns and features of the datasets. Moreover, selecting the right model for a dataset often involves trial-and-error and computationally expensive hyperparameter tuning, with many models discarded after underperforming. However, each model may still capture unique structural patterns or inductive biases (model assumptions) that, while insufficient alone, could contribute meaningfully when combined. In machine learning (ML), ensemble learning approach is a proven method to pool a set of (weak) models to create a strong model (Li, Paffenroth, and Berthiaume 2021). In recent years, mixture of experts (MoE) approach has become the *de facto* choice for ensemble learning, though it differs from the classical approach by the gating (routing) network which is trained to select expert(s) based on the input context. The MoE and its variants have been successfully applied to large language models (LLMs) to increase model capacity without proportionate increase in computational cost (Fedus, Zoph, and Shazeer 2022). In the graph domain, MoE-based approaches are still in the early stages of exploration but have demonstrated significant potential. Existing methods such as GMoE (Wang et al. 2023) typically exhibit limited architectural diversity, often employing variations of a single base model (e.g., GCNs with MoE layers inserted at intermediate depths). Furthermore, their gating mechanisms are generally not designed to capture complex graph topologies, and expert pruning strategies, if present, are either ad hoc or inefficiently integrated. These limitations underscore the need for a more flexible and topology-aware MoE framework tailored to the unique challenges of graph-structured data.

In this work, we propose Self-Adaptive Graph Mixture of Models (SAGMM), a modular and practical framework that learns to automatically select and combine the most appropriate GNN models for each part of the graph. SAGMM

*Work done while at Fujitsu Research of India, Bangalore. Now at Microsoft.

leverages a diverse pool of GNN architectures, each serving as an expert with a fixed view of the data, and employs a topology-aware sparse attention gate that dynamically routes input to the most relevant experts. The framework is designed to be trained end-to-end but as a practical variation, we also explore a training strategy, where experts are pre-trained independently and only the routing mechanism and task head are trained, reducing training time. The core idea of SAGMM is that different GNNs offer complementary perspectives of the graph, but their architectural rigidity limits their individual expressiveness. By selectively combining them at the representation level, SAGMM achieves greater adaptability and structural awareness. This formulation also aligns with the broader model merging paradigm (Yang et al. 2024), though our merging occurs in representation space rather than parameter space. The salient features of the SAGMM are as follows:

1. SAGMM leverages the strength of diverse models such as GCN, GAT, and GraphSAGE, *etc.* each contributing unique inductive biases to capture varied structural properties.
2. SAGMM incorporates a sparse, topology-aware attention gating that adaptively selects the most relevant experts for each node by evaluating attention scores between the node and the expert models.
3. Pruning of expert pool using importance score which improves efficiency while preserving performance.
4. Pooling pre-trained models that are otherwise discarded during the model selection phase of standard real-world pipelines unlocks additional value within the SAGMM architecture and enables flexible integration of newer models at minimal cost.

Extensive experiments on node classification, graph classification, graph regression, and link prediction tasks demonstrate that SAGMM consistently outperforms GNN models and prior graph-based MoE frameworks.

2 Background and Related Work

2.1 Preliminaries and Notations

Let $G(A, X)$ be an undirected graph with an adjacency matrix $A \in \{0,1\}^{n\times n}$ and a node feature matrix $X \in \mathbb{R}^{n\times d}$, where n is the number of nodes and d is the feature dimension, D represents the degree matrix of graph. The batch size is denoted as b. In a GNN with L layers, let $C^{(l)}$ denote the convolution matrix and $W^{(l)}$ the learnable parameters at layer l.

We define a pool of experts $e = \{e_1, e_2, \ldots, e_{N_0}\}$, where N_0 is the initial number of experts and N is the number of experts after pruning, such that $N = N_0$ at the start of training. For a given node u, let k_u denote the subset of experts it activates. Notably, in conventional models, the expert pool remains fixed ($N = N_0$).

2.2 Graph Neural Networks

Several GNN architectures based on encoder-decoder framework (Hamilton, Ying, and Leskovec 2017b) with encoding performed by message-passing mechanism have been proposed for graph ML tasks. According to (Balcilar et al. 2021), the general update rule for the node embeddings $H^{(l+1)}$ at layer $l+1$ is represented by:

$$H^{(l+1)} = \sigma\left(\sum_q C^{(l,q)} H^{(l)} W^{(l,q)}\right) \tag{1}$$

where $\sigma(\cdot)$ is a nonlinear activation function, $W^{(l,q)}$ are learnable parameters associated with the filter q, and $C^{(l,q)}$ are model-specific convolution matrices. Different GNNs define $C^{(l,q)}$ differently. For instance, GCN (Kipf and Welling 2016) uses a normalized adjacency matrix fixed across layers, GraphSAGE (Hamilton, Ying, and Leskovec 2017a), referred to as SAGE, employs identity and mean aggregation. GAT (Veličković et al. 2017) introduces attention-based weighting, while GIN (Xu et al. 2018a) approximates the Weisfeiler-Lehman (WL) test via learnable aggregation (Ding et al. 2022). JKNet (Xu et al. 2018b) improves depth flexibility by aggregating information from multiple GNN layers via jump connections. Variants such as SGC (Wu et al. 2019) eliminate nonlinearities for efficiency, MixHop (Abu-El-Haija et al. 2019) aggregates multi-hop neighborhoods, and GraphCNN (Defferrard, Bresson, and Vandergheynst 2016) applies spectral filtering via Chebyshev polynomials.

2.3 Mixture of Experts in Graph Learning

The MoE framework (Jacobs et al. 1991), when first introduced enables routing inputs to specialized models via a trainable dense gating network. Modern sparse activation MoE architectures, widely used in large-scale language and vision models, leverage thousands of experts to enhance model capacity (Zhang et al. 2025). Methods such as DynMoE (Guo et al. 2024) introduces top-any gating and pruning strategy to dynamically activate and prunes expert in LLM domain. While M32 (Wu et al. 2024b) introduces an attention-based routing mechanism, its quadratic complexity limits scalability for large graphs. Recent surveys (Cai et al. 2025; Zhang et al. 2025) provide a comprehensive overview of trends and challenges in MoE models for LLMs.

For graph data, GMoE (Wang et al. 2023) extends MoE to GNNs, enabling multi-hop information aggregation. GraphMETRO (Wu et al. 2024a) leverages MoE to tackle distribution shifts in GNNs. TopExpert (Kim et al. 2023) employs domain-specific linear experts for graph classification but lacks adaptability to diverse tasks. DA-MoE (Yao et al. 2024) uses GNN layers as experts to address depth sensitivity but does not support dynamic expert selection. GraphMoRE (Guo et al. 2025) leverages a Mixture of Riemannian Experts, where each expert operates on a learned manifold to capture diverse local structures. Link-MoE (Ma et al. 2024) relies on pre-trained GNN experts for link prediction, limiting its flexibility for broader applications.

3 Methodology

We now introduce the SAGMM in detail, with the subsections focusing its key components. Figure 1 illustrates the SAGMM architecture. Input features are first enhanced with

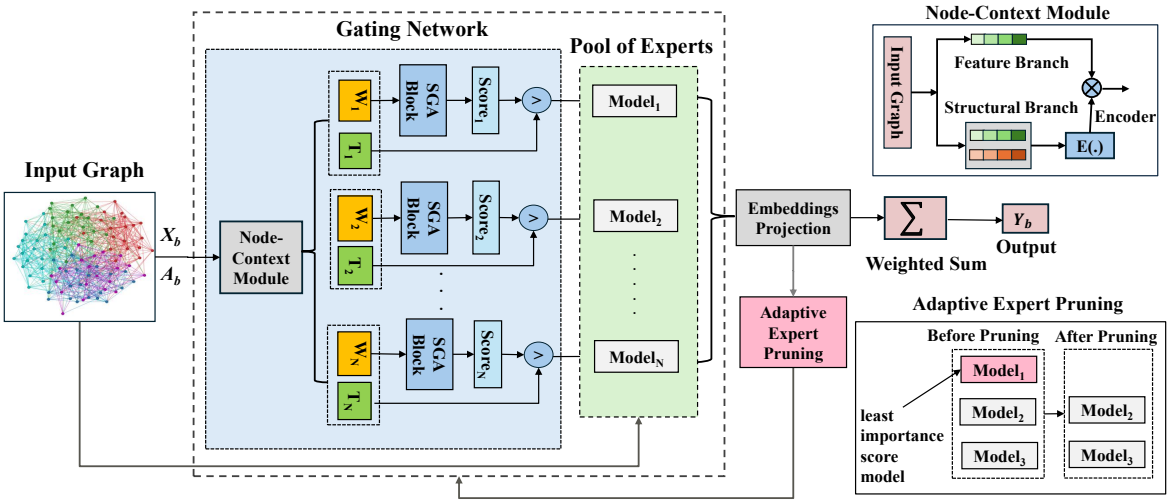

Figure 1: The overall illustration of the SAGMM framework. The key components includes the gating network, a pool of experts, and an adaptive expert pruning. W_i denotes W_{Q_i}, W_{K_i}, and W_{V_i} per expert i.

structural context, and expert selection is performed by a gating network using Simple Global Attention (SGA) (Wu et al. 2023) and learnable thresholds. The selected expert outputs are projected into a shared space, weighted by gate scores, and aggregated, while low-importance experts are pruned based on an adaptive threshold η. The training phase of SAGMM in outlined in Algorithm 1.

3.1 Pool of Experts

A distinctive feature of the SAGMM framework lies in its use of a structurally diverse set of expert models. This choice is motivated by the aim of maximizing architectural heterogeneity, thereby enhancing the expressiveness and representational capacity of the framework.

Recent surveys (Chen et al. 2020; Ju et al. 2024) categorize GNNs along three key dimensions: propagation strategy (spectral vs. spatial), aggregation mechanism (mean vs. attention), and training setup (transductive vs. inductive). Guided by these dimensions, we include GNNs like GCN (Kipf and Welling 2016) for its spectral filtering and suitability for homophilic graphs, GAT (Veličković et al. 2017) for attention-based neighbor weighting, and SAGE (Hamilton, Ying, and Leskovec 2017a) for its inductive capability through neighborhood sampling, along with additional models in the expert pool. Such heterogeneity enables the router to select experts that align with the structural and statistical properties of individual instances.

Formally, the embedding update for an expert e_i at layer $l+1$ is given by:

$$H_{e_i}^{(l+1)} = f_{e_i}(H_{e_i}^{(l)}, A), \tag{2}$$

Here f_{e_i} denotes the message-passing function of expert e_i. This design is also motivated by the bias-variance trade-off (Yang et al. 2020), where over-parameterized models can improve generalization by reducing variance. By incorporating multiple GNNs, our framework benefits from this property. Moreover, in line with the No Free Lunch theorem (Goldblum et al. 2023) which states that no single model performs optimally across all tasks, our approach accounts for variations in network topology, homophily, and heterophily that affects GNN performance. We show empirically that by pooling experts, each with an unique message-passing mechanism, within MoE framework, our approach dynamically adapts to diverse graph structures, overcoming limitations of a single model.

3.2 Gating Network

In this subsection, we introduce *Topology-Aware Attention Gating (TAAG)*, a novel gating mechanism for graph data. We begin by highlighting the limitations of existing methods before presenting our proposed approach.

Limitations of traditional gating mechanism. Several approaches have been proposed for designing gating networks in MoE architectures (Jordan and Jacobs 1994; Clark et al. 2022; Zhou et al. 2022). A widely adopted sparse gating strategy is the noisy top-k gating mechanism (Shazeer et al. 2017), which applies a softmax over the top-k logits produced by a linear transformation of the input. To encourage expert diversity and prevent load imbalance, this approach injects learnable Gaussian noise into the logits, a modification later adapted for graph data in GMoE model.

Despite considerable success of the noisy top-k gating in improving training and inference efficiency, our analysis reveals several key limitations still remain:

1. The model's performance is highly dependent on the choice of k, as different datasets exhibit varying optimal values as shown in Figure 2(a) for GMoE-GCN (Wang et al. 2023), highlighting the need for extensive hyperparameter tuning. Moreover, using a fixed k forces all nodes to activate the same number of experts, which may not be needed considering structural and feature characteristics. As shown in Figure 2(b), different nodes acti-

vate different number of experts as per their requirement, eventually leading to performance gains.

2. While prior works on gating strategies select multiple experts, thus implicitly including correlation between them, it is essential to ensure diversity among them for better coverage of the representation space.
3. Existing methods often rely on simple linear projections of node features to compute gating scores, neglecting structural dependencies within the graph. While DA-MoE (Yao et al. 2024) and GraphMoRE (Guo et al. 2025) attempt to overcome this limitation by leveraging GNN-based gating networks, a fundamental drawback remains: these approaches are still unable to effectively capture global graph structural information, which is vital for informed expert selection.

To address these limitations, for each node, TAAG selects experts based on the local and global structural information while dynamically adjusting the number of activated experts to improve efficiency and robustness. We leverage the SGA block from SGFormer (Wu et al. 2023) to compute attention scores efficiently. While standard attention mechanisms have a quadratic complexity of $O(n^2)$, SGA operates in linear time $O(n)$ w.r.t. n nodes, making it scalable for medium to large-scale graphs. To incorporate global structural information, we construct the matrix $X^{(g)}$ using the p smallest eigenvectors of the normalized graph Laplacian L (Eq. (4)), providing positional encodings for each node (Rampášek et al. 2022). We do not use diffusion-based methods (Gasteiger, Weißenberger, and Günnemann 2019; Chamberlain et al. 2021) as they involve expensive preprocessing, assumes homophily and perform poorly in tasks like link prediction. To further strengthen the structural information, we also add local structural features by aggregating 1-hop and 2-hop (approximate) features (Hamilton, Ying, and Leskovec 2017a; Wang et al. 2020). The detailed formulations are provided in Eq. (3) and Eq. (4).

$$\mathbf{X}^{(1)} = D^{-1}A\mathbf{X}, \quad \mathbf{X}^{(2)} = (D^{-1}A)^2\mathbf{X}. \tag{3}$$

$$L = I - D^{-1/2}AD^{-1/2}. \tag{4}$$

Hence, we define the final input features as:

$$\mathbf{X}' = \underbrace{\frac{1}{3}(\mathbf{X} + \mathbf{X}^{(1)} + \mathbf{X}^{(2)})}_{\text{Local Term}} \;||\; \underbrace{\mathbf{X}^{(g)}}_{\text{Global Positioning of Nodes}}. \tag{5}$$

where $X' \in \mathbb{R}^{n\times(d+p)}$. The local term aggregates multi-hop neighborhood information, while $X^{(g)}$ denotes the global positional encodings of the nodes of the graph. To compute gating scores, we first define the Query (Q), Key (K), and Value (V) representations as follows:

$$Q = X'W_Q, \quad K = X'W_K, \quad V = X'W_V \tag{6}$$

where $W_Q, W_K, W_V \in \mathbb{R}^{(d+p)\times N}$ are learnable projection matrices for query, key, and value, respectively. We compute the attention-based gating scores using the Simple Global Attention (SGA) mechanism (Wu et al. 2023):

$$Q_{\text{norm}} = \frac{Q}{\|Q\|_F}, \quad K_{\text{norm}} = \frac{K}{\|K\|_F} \tag{7}$$

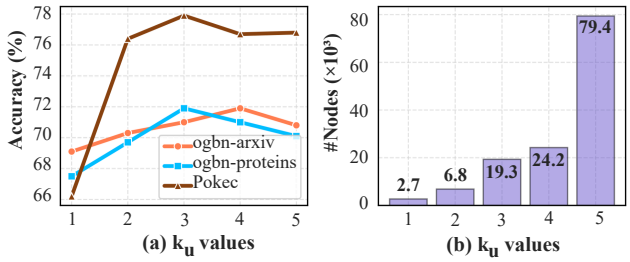

Figure 2: (a) Performance variation across different Top-k values in GMoE-GCN for various datasets. (b) Distribution of expert activation counts by SAGMM for ogbn-proteins dataset, which contains 132,534 nodes.

$$D_g = \text{diag}\left(1 + \frac{1}{N}Q_{\text{norm}}(K_{\text{norm}}^{\top}\mathbf{1})\right). \tag{8}$$

$$Z = \beta D_g^{-1}\left(V + \frac{1}{N}Q_{\text{norm}}(K_{\text{norm}}^{\top}V)\right) + (1-\beta)X. \tag{9}$$

where β is a learnable residual weight controlling the contribution of the original input, and Z is the attention scores for each node and experts in the pool.

To enforce sparsity in expert selection, we introduce a learnable gating threshold $T \in [0,1]$, which only activates experts which exceeds scores above T. It is initialized as $T_{\text{init}} \sim \{\mathbf{0}, \mathcal{U}(0,1)\}$, where $\mathcal{U}(0,1)$ denotes a uniform distribution over (0,1). Both T_{init} and expert scores Z are transformed via a sigmoid function to ensure values in $[0,1]$. A ReLU-based gating mask is applied for threshold comparison, the final gating scores (G) in Eq. (13) uses a binary selection mask (M) allowing only experts above the threshold to contribute.

$$M = \text{sign}(\text{ReLU}(Z' - T)). \tag{10}$$

$$\text{where, } T = \sigma(T_{\text{init}}), \quad Z' = \sigma(Z) \tag{11}$$

Since the sign function is non-differentiable, we customize the backpropagation by directly passing the gradient of the binary selection mask to $\text{ReLU}(Z' - T)$, effectively bypassing the sign function. The number of activated experts (k_u) per node u is defined in Eq. (12).

$$k_u = |\{j \mid M_{u,j} > 0\}|. \tag{12}$$

where $M_{u,j}$ is value for node u and expert j. For nodes where no expert is selected ($k_u = 0$), we assign the expert with the highest attention score to ensure each node is allocated at least one expert: $M_{u,\text{argmax}_j Z'_{u,j}} = 1$.

$$G = Z' \odot M. \tag{13}$$

Once experts are selected, the valid experts process the input (X, A), producing expert outputs H_{e_j} for expert e_j, these outputs are aggregated using the gating scores in Eq (13). Our ablation study (Section 4.2) shows that the proposed gating mechanism outperforms existing alternatives, validating its effectiveness.

Algorithm 1: SAGMM Framework (Training Phase)

Input: Graph (X, A) with labels Y, expert pool $e = \{e_1, \ldots, e_{N_0}\}$, pruning threshold η, pruning interval t.
Output: Updated expert pool e', gating weights G.

Initialize: $\forall\, e_i \in e,\ I(e_i) = 0$.
Divide (X, A) and Y into mini-batches $\{(X_b, A_b),\ Y_b\}$.
for each batch $\big(X_b, A_b, Y_b\big)$ at epoch q **do**
  $X_b' \leftarrow (X_b, A_b)$ Using Eq. (5).
  Obtain M, k_u, G, and via Eqs. (10), (12), (13), .
  Determine active experts: $S \leftarrow \bigcup_u \{\, j \mid M_{u,j} > 0\}$.
  for each expert $e_j \in S$ **do**
    Compute expert output $H_{e_j} \leftarrow e_j\big(X_b, A_b\big)$.
  end for
  Aggregate expert outputs:
    $\bar{Y}_b \leftarrow \sum_{j=1}^{N} G_j\, \mathrm{MLP}(H_j)$.
  Evaluate loss $\mathcal{L} \leftarrow \mathcal{L}\big(Y_b, \bar{Y}_b\big)$ and update weights for $e_j \in S$.
  Update expert importance $I_t(e_i)$ via Eq. (14).
  if $(q \bmod t) == 0$ **then**
    Prune experts using threshold η:
      $e \leftarrow \{e_i \mid I(e_i) \geq \eta\}$.
  end if
end for
return Updated expert pool e' and gating weights G.

3.3 Adaptive Expert Pruning

For efficient training and optimizing memory utilization, we also introduce an adaptive expert pruning mechanism that dynamically evaluates and updates the importance of each expert in the pool over time. The importance score $I(e_i)$ at pruning interval t for an expert e_i is updated based on its contribution in previous intervals as follows:

$$I_t(e_i) = (1 - \alpha) I_{t-1}(e_i) + \alpha \gamma(e_i). \tag{14}$$

Here, $\gamma(e_i)$ represents the contribution score of expert $e_i \in e$, computed as:

$$\gamma(e_i) = \left\| \sum_{u=1}^{n} \left(G_{e_i,u} H_{e_i,u,:} \right) \right\|. \tag{15}$$

where $G_{e_i,u}$ are gating weights and $H_{e_i,u,:}$ represents representation of expert e_i and node u. Above equation captures the cumulative weighted contribution of the expert to the overall model output for the selected batch of data. The smoothing factor $\alpha \in [0, 1]$ controls the balance between previous importance scores and the current contribution, thereby preventing the premature removal of experts that may become useful later in training. The importance scores for all experts are calculated at every epoch, and experts with importance scores below the threshold are removed at prune interval:

$$e \leftarrow \{e_i \mid I(e_i) \geq \eta\}. \tag{16}$$

The threshold factor η, which controls the pruning aggressiveness, is dynamically adjusted based on validation performance. As shown in Figure 3(a), the average k_u values differ from the final number of experts N post-pruning, highlighting the dynamic and input-dependent nature of expert selection. The TAAG gating mechanism selectively activates and weighs relevant experts, while the pruning strategy removes persistently underutilized ones. This combination ensures that SAGMM maintains both specialization and computational efficiency.

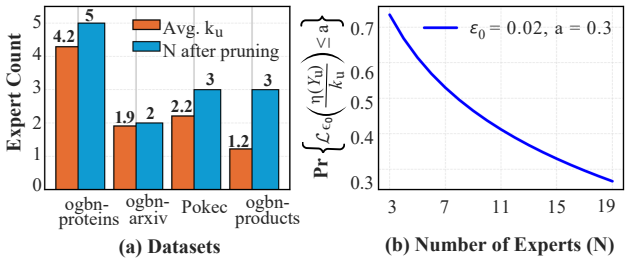

Figure 3: (a) Average number of experts activated per node u (k_u) and experts remaining post-pruning (N). (b) Visualization of $\Pr\left\{\mathcal{L}_{\epsilon_0}\left(\frac{\eta(Y_u)}{k_u}\right) \leq a\right\}$ (Theorem 1) for $\epsilon_0 = 0.02, a = 0.3$.

Auxiliary Loss Functions. To ensure balanced expert utilization, we incorporate importance loss and diversity loss from prior works (Wang et al. 2023; Guo et al. 2024). These losses prevent over-reliance on specific experts by encouraging more even and orthogonal expert activation patterns.

3.4 Pre-trained Models as Experts

One variant of SAGMM we experimented is SAGMM Pre-trained Experts (SAGMM-PE), where the expert GNNs are pre-trained independently and kept fixed during the downstream task with only the router and task-specific head being trained. This design reduces computational overhead and allows efficient reuse of specialized or weak models that may not perform well individually but contribute valuable structural signals when combined. As shown in Table 1, SAGMM-PE achieves competitive or superior performance on node classification task. Furthermore, the results in the pre-training ratios table show that comparable accuracy is achieved even when the experts and the router are pre-trained on only 50% to 70% of the training data. This highlights the data efficiency of our framework and the robustness of its model-agnostic expert design.

3.5 Theoretical Analysis

We provide a theoretical justification for using expert mixtures and pruning in SAGMM by analyzing how sparsely activating a subset of experts affects model performance.

Consider a simplified setting where (1) The GNN has a single-layer architecture with an update rule defined in Eq. 1 for binary node classification. (2) Input features $X = [x_1, x_2, \ldots, x_n]$, where $x_i \sim \mathcal{N}(0_d, \sigma^2 I_{d\times d})$ with $k_u > 2$ experts for each node u. (3) The aggregated output at layer 1 for node u is given by $\frac{\eta(Y_u)}{k_u}$, where $Y_u =$

Model	Deezer	YelpChi	ogbn-proteins	ogbn-arxiv	Pokec	ogbn-products	ogbn-papers100M
GCN	$57.70_{\pm 0.44}$	$85.62_{\pm 0.26}$	$69.74_{\pm 3.79}$	$71.74_{\pm 0.29}$	$76.52_{\pm 0.54}$	$75.54_{\pm 0.06}$	$61.42_{\pm 0.25}$
GAT	$58.59_{\pm 0.21}$	$85.42_{\pm 0.26}$	$69.56_{\pm 3.82}$	$71.42_{\pm 0.09}$	$78.87_{\pm 0.60}$	$76.77_{\pm 0.15}$	–
SAGE	$64.40_{\pm 0.79}$	$89.23_{\pm 0.57}$	$73.21_{\pm 1.88}$	$71.46_{\pm 0.23}$	$79.82_{\pm 1.62}$	$78.29_{\pm 0.11}$	$63.54_{\pm 0.18}$
SGC	$57.66_{\pm 0.92}$	$85.59_{\pm 0.25}$	$68.75_{\pm 3.55}$	$71.81_{\pm 0.20}$	$76.81_{\pm 0.52}$	$75.48_{\pm 0.14}$	$57.50_{\pm 0.28}$
JKNet	$64.44_{\pm 0.45}$	$89.75_{\pm 0.32}$	$75.67_{\pm 2.06}$	$71.72_{\pm 0.19}$	$78.84_{\pm 0.19}$	–	$62.98_{\pm 0.26}$
Graph CNN	$63.65_{\pm 0.60}$	$89.25_{\pm 0.26}$	$77.54_{\pm 1.09}$	$72.04_{\pm 0.19}$	$80.21_{\pm 0.14}$	–	–
GIN	$59.71_{\pm 1.25}$	$85.52_{\pm 0.26}$	$54.46_{\pm 2.63}$	$68.16_{\pm 0.27}$	$72.53_{\pm 1.45}$	–	–
MixHop	$57.83_{\pm 1.83}$	$85.74_{\pm 0.38}$	$73.64_{\pm 2.34}$	$71.94_{\pm 0.40}$	$81.24_{\pm 0.27}$	–	–
GMoE-GCN	$61.11_{\pm 1.19}$	$85.75_{\pm 0.31}$	$74.48_{\pm 0.58}$	$71.88_{\pm 0.32}$	$76.04_{\pm 0.14}$	$64.18_{\pm 0.25}$	$60.17_{\pm 0.20}$
DA-MoE	$62.15_{\pm 0.46}$	$85.53_{\pm 0.29}$	$75.22_{\pm 0.16}$	$71.96_{\pm 0.16}$	$64.87_{\pm 5.68}$	$68.77_{\pm 0.19}$	$53.63_{\pm 1.68}$
SAGMM	$64.73_{\pm 0.72}$	$91.06_{\pm 0.60}$	$\mathbf{78.15_{\pm 1.38}}$	$\mathbf{72.80_{\pm 0.43}}$	$\mathbf{81.76_{\pm 0.80}}$	$\mathbf{82.91_{\pm 0.53}}$	$\mathbf{64.40_{\pm 0.10}}$
SAGMM-PE	$\mathbf{65.51_{\pm 0.57}}$	$\mathbf{91.33_{\pm 0.26}}$	$76.28_{\pm 1.28}$	$72.08_{\pm 0.30}$	$80.57_{\pm 0.95}$	$82.17_{\pm 0.47}$	$62.54_{\pm 0.15}$

Table 1: Node classification results. For the ogbn-proteins dataset, the metric used is ROC-AUC, while testing accuracy is used for the other datasets. **SAGMM-PE**: Experts are pretrained and kept frozen, while only the router and task heads are trained. A dash (–) indicates that the corresponding model is not included in the expert pool.

Model	Graph Classification				Graph Regression	
	molbbbp	moltox21	moltoxcast	molhiv	molfreesolv	molesol
GCN	$68.87_{\pm 1.51}$	$75.29_{\pm 0.69}$	$63.54_{\pm 0.42}$	$76.06_{\pm 1.75}$	$2.64_{\pm 0.24}$	$1.11_{\pm 0.04}$
GIN	$65.50_{\pm 1.80}$	$74.30_{\pm 0.50}$	$63.30_{\pm 1.50}$	$75.40_{\pm 1.50}$	$2.76_{\pm 0.35}$	$1.17_{\pm 0.06}$
SAGE	$63.07_{\pm 1.83}$	$75.64_{\pm 0.73}$	$65.69_{\pm 0.29}$	$75.54_{\pm 1.28}$	$2.43_{\pm 0.30}$	$1.05_{\pm 0.08}$
GAT	$67.82_{\pm 1.75}$	$74.36_{\pm 0.70}$	$65.50_{\pm 0.89}$	$73.40_{\pm 1.87}$	$2.23_{\pm 0.11}$	$1.02_{\pm 0.05}$
GMoE-GCN	$\mathbf{70.04_{\pm 1.12}}$	$75.45_{\pm 0.58}$	$64.12_{\pm 0.61}$	$77.35_{\pm 0.63}$	$2.50_{\pm 0.19}$	$1.09_{\pm 0.04}$
DA-MoE	$69.62_{\pm 0.98}$	$75.59_{\pm 0.69}$	$65.18_{\pm 0.48}$	$\mathbf{77.62_{\pm 1.58}}$	$2.19_{\pm 0.07}$	$1.13_{\pm 0.04}$
SAGMM	$69.98_{\pm 0.18}$	$\mathbf{76.58_{\pm 0.88}}$	$\mathbf{66.63_{\pm 0.53}}$	$77.48_{\pm 1.10}$	$\mathbf{2.15_{\pm 0.12}}$	$\mathbf{0.93_{\pm 0.03}}$

Table 2: Graph classification and regression results. Metric used is testing ROC-AUC for classification and RMSE for regression. Best model per column is shown in bold.

$\sum_{m=1}^{k_u} \hat{H}^{(1)}_{e_i,u}$. Here, $\hat{H}^{(1)}_{e_i,u}$ represents the first-order approximation of $H^{(1)}_{e_i,u}$ (Details in Technical Appendix). The function $\eta(x)$ is defined as $\eta(x) = \frac{\exp(\mathbf{1}^T x)}{1+\exp(\mathbf{1}^T x)}$. (4) The loss function $\mathcal{L}$ is the numerically stable binary cross-entropy loss:

$$\mathcal{L}_{\epsilon_0}(x, y_u) = -y_u \log(x + \epsilon_0) - (1 - y_u) \log(1 - x + \epsilon_0).$$

Theorem 1 *Under the design conditions above, the following inequality holds:*

$$\Pr\left\{\mathcal{L}_{\epsilon_0}\left(\frac{\eta(Y_u)}{k_u}\right) \leq a\right\} \leq \frac{U - f(k_u, \epsilon_0)}{U - a}, \qquad (17)$$

where $U = -\log(\epsilon_0)$ *and* $f(k_u, \epsilon_0) = \log(2k_u^2) - \log(2k_u(1 + \epsilon_0) - 1)$.

This bound reveals that increasing the number of activated experts k_u generally improves the tightness of the bound, lowering expected error. However, as shown in Figure 3(b), pruning a small number of low-importance experts only marginally impacts the bound while substantially reducing computation. This result supports our adaptive pruning mechanism in SAGMM.

4 Experiments

Initial Number of Experts (N_0). For our primary task of node classification, we construct the expert pool by selecting four and eight widely recognized GNN architectures: GCN, JKNet, GraphCNN, MixHop, GAT, SGC, GIN, and SAGE, all of which are established standards for this application (Hu et al. 2020). In the case of other tasks, the expert pool is composed of four GNN models. The decision to use either eight or four experts is guided by the need to balance architectural diversity with computational feasibility. Notably, this choice is consistent with recent findings in the literature (Yun et al. 2024; He et al. 2025), which indicate that configuring MoE models with four or eight experts enables efficient inference and strong model performance.

Baselines. For each of the tasks, we compare against two types of baselines: 1) existing SOTA graph-based MoE architectures: GMoE and DA-MoE, and 2) each of the individual expert in the pool.

Settings. We use a common set of hyperparameters across all methods to ensure consistency. For model-specific settings, we adopt the default configurations provided in the

Model	ogbl-ddi	ogbl-collab	ogbl-ppa
SAGE	$53.90_{\pm 4.74}$	$48.10_{\pm 0.81}$	$16.55_{\pm 2.40}$
GCN	$37.07_{\pm 5.07}$	$44.75_{\pm 1.07}$	$18.67_{\pm 1.32}$
JKNet	$60.56_{\pm 8.69}$	$51.47_{\pm 0.59}$	$21.58_{\pm 2.59}$
SGC	$35.00_{\pm 4.65}$	$48.99_{\pm 0.60}$	$9.64_{\pm 1.60}$
GMoE	$37.96_{\pm 8.20}$	$32.61_{\pm 2.59}$	$19.25_{\pm 1.67}$
DA-MoE	$45.58_{\pm 6.93}$	$47.64_{\pm 3.77}$	$21.46_{\pm 2.53}$
SAGMM	$\mathbf{74.20}_{\pm 11.71}$	$\mathbf{52.39}_{\pm \mathbf{0.40}}$	$\mathbf{26.11}_{\pm \mathbf{1.12}}$

Table 3: Link prediction results. Metric used is HITS@20 for ogbl-ddi, HITS@100 for ogbl-ppa and HITS@50 for ogbl-collab as per OGB benchmark protocol.

respective original papers. All reported results are averaged over ten independent runs to ensure a fair comparison.

4.1 Performance Evaluation

We evaluate our framework across three categories of tasks: node classification, graph-level prediction, and link prediction, using a broad set of benchmark datasets.

Node Classification. We evaluate on seven standard datasets ranging from small to large-scale, including four homophilic graphs: *ogbn-arxiv*, *ogbn-proteins*, *ogbn-products*, and *ogbn-papers100M* from the OGB benchmark (Hu et al. 2020), and three heterophilic graphs: *Pokec* (Jure 2014), *Deezer* (Rozemberczki and Sarkar 2020), and *YelpChi* (Mukherjee et al. 2013). As shown in Table 1, SAGMM outperforms all baselines, achieving notable gains on medium to large-scale datasets such as a 5.90% improvement on *ogbn-products* and 1.57% on *ogbn-papers100M*, along with consistent accuracy improvements across the remaining benchmarks. Our analysis reveals that SAGMM consistently prunes weaker experts such as GIN across datasets, demonstrating its ability to adaptively select the most effective experts.

Graph-Level Prediction. We use six molecular property prediction datasets from the OGB benchmark: *molhiv*, *moltox21*, *moltoxcast*, *molbbbp*, *molesol*, and *molfreesolv*. As shown in Table 2, SAGMM achieves strong performance, with notable gains on *moltox21*, *moltoxcast*, and *molesol*. It incurs minor drops on *molhiv* and *molbbbp*, but remains competitive overall.

Link Prediction. We evaluate SAGMM on three standard OGB link prediction datasets: *ogbl-ppa*, *ogbl-ddi*, and *ogbl-collab*. As shown in Table 3, SAGMM achieves substantial performance improvements, with gains of 22.52% on *ogbl-ddi*, 20.90% on *ogbl-ppa*, and 1.78% on *ogbl-collab*, outperforming all baselines across the board.

Inference Time and Memory Usage. As shown in Table 4, SAGMM maintains competitive inference time and GPU memory usage compared to existing methods. Additionally, aggressive expert pruning can further reduce inference cost, with only a minor trade-off in performance, enabling a controllable balance between efficiency and effectiveness.

Model	ogbn-arxiv		ogbl-ddi	
	Time (ms)	GPU Mem (GB)	Time (ms)	GPU Mem (GB)
GCN	126	1.74	235	1.50
GMoE	200	3.73	298	1.50
DA-MoE	251	1.90	361	2.15
SAGMM	227	5.89	334	1.95

Table 4: Inference time and GPU memory usage comparison across models on ogbn-arxiv and ogbl-ddi datasets.

Method	ogbn-proteins	ogbl-collab
SAGMM	$\mathbf{78.15}_{\pm \mathbf{1.38}}$	$52.39_{\pm 0.40}$
SAGMM *(w/o diverse)*	$69.89_{\pm 1.84}$	$40.08_{\pm 4.73}$
SAGMM *(top-k gating)*	$67.29_{\pm 3.94}$	$48.67_{\pm 0.94}$
SAGMM *(top-any gating)*	$77.31_{\pm 2.31}$	$50.42_{\pm 0.99}$
SAGMM *(w/o pruning)*	$75.24_{\pm 1.49}$	$\mathbf{52.44}_{\pm \mathbf{0.57}}$

Table 5: Results of ablation study. Metrics are ROC-AUC for ogbn-proteins and HITS@50 for ogbl-collab.

4.2 Ablation Study

We conduct an ablation study to evaluate the contribution of each component in SAGMM by testing four variants: (i) using identical GNN architectures for all experts (*SAGMM (w/o diverse)*), (ii) replacing our gating module with noisy Top-k gating (*SAGMM (top-k gating)*), (iii) using Top-Any gating (Guo et al. 2024) having dynamic expert selection (*SAGMM (top-any gating)*), and (iv) disabling the adaptive expert pruning mechanism (*SAGMM (w/o pruning)*). As shown in Table 5, the largest drop is observed when expert diversity is removed, confirming its critical role. Substituting our gating mechanism with top-k or top-any variants impairs expert selection quality, while disabling pruning retains performance in the case of *ogbl-collab* but always leads to higher memory usage. These results underscore the importance of expert diversity, TAAG gating, and pruning in SAGMM's effectiveness.

5 Conclusion

This paper presents SAGMM, a novel framework that employs multiple GNN models as experts within a MoE paradigm. It features a topology-aware attention gating mechanism tailored for graph data and an adaptive pruning strategy. Extensive experiments demonstrate that SAGMM consistently outperforms traditional approaches across diverse downstream tasks. While SAGMM currently focuses on GNN-based experts, in principle, it can integrate any black-box model.

Future work will explore extending SAGMM through expert distillation, where the knowledge from a diverse pool of experts and the TAAG gating mechanism is transferred into a compact, unified student model. Also, SAGMM for dynamic graphs is another promising direction for further research.

References

Abu-El-Haija, S.; Perozzi, B.; Kapoor, A.; Alipourfard, N.; Lerman, K.; Harutyunyan, H.; Ver Steeg, G.; and Galstyan, A. 2019. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *ICML*, 21–29. PMLR.

Balcilar, M.; Renton, G.; Héroux, P.; Gaüzère, B.; Adam, S.; and Honeine, P. 2021. Analyzing the Expressive Power of Graph Neural Networks in a Spectral Perspective. In *ICLR*.

Cai, W.; Jiang, J.; Wang, F.; Tang, J.; Kim, S.; and Huang, J. 2025. A survey on mixture of experts in large language models. *IEEE Transactions on Knowledge and Data Engineering*.

Chamberlain, B.; Rowbottom, J.; Gorinova, M. I.; Bronstein, M.; Webb, S.; and Rossi, E. 2021. Grand: Graph neural diffusion. In *International conference on machine learning*, 1407–1418. PMLR.

Chen, Z.; Chen, F.; Zhang, L.; Ji, T.; Fu, K.; Zhao, L.; Chen, F.; Wu, L.; Aggarwal, C.; and Lu, C.-T. 2020. Bridging the gap between spatial and spectral domains: A survey on graph neural networks. *arXiv preprint arXiv:2002.11867*.

Clark, A.; de Las Casas, D.; Guy, A.; Mensch, A.; Paganini, M.; Hoffmann, J.; Damoc, B.; Hechtman, B.; Cai, T.; Borgeaud, S.; et al. 2022. Unified scaling laws for routed language models. In *International conference on machine learning*, 4057–4086. PMLR.

Defferrard, M.; Bresson, X.; and Vandergheynst, P. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. *NeurIPS*, 29.

Ding, M.; Rabbani, T.; An, B.; Wang, E. Z.; and Huang, F. 2022. Sketch-GNN: Scalable Graph Neural Networks with Sublinear Training Complexity. In Oh, A. H.; Agarwal, A.; Belgrave, D.; and Cho, K., eds., *NeurIPS*.

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

Gasteiger, J.; Weißenberger, S.; and Günnemann, S. 2019. Diffusion improves graph learning. *Advances in neural information processing systems*, 32.

Goldblum, M.; Finzi, M.; Rowan, K.; and Wilson, A. G. 2023. The no free lunch theorem, kolmogorov complexity, and the role of inductive biases in machine learning. *arXiv preprint arXiv:2304.05366*.

Guo, Y.; Cheng, Z.; Tang, X.; Tu, Z.; and Lin, T. 2024. Dynamic mixture of experts: An auto-tuning approach for efficient transformer models.

Guo, Z.; Sun, Q.; Yuan, H.; Fu, X.; Zhou, M.; Gao, Y.; and Li, J. 2025. GraphMoRE: Mitigating Topological Heterogeneity via Mixture of Riemannian Experts. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 11754–11762.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017a. Inductive representation learning on large graphs. *NeurIPS*, 30.

Hamilton, W. L.; Ying, R.; and Leskovec, J. 2017b. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*.

He, Y.; Liu, Y.; Liang, C.; and Awadalla, H. H. 2025. Efficiently Editing Mixture-of-Experts Models with Compressed Experts. *arXiv preprint arXiv:2503.00634*.

Hu, W.; Fey, M.; Zitnik, M.; Dong, Y.; Ren, H.; Liu, B.; Catasta, M.; and Leskovec, J. 2020. Open graph benchmark: Datasets for machine learning on graphs. *NeurIPS*, 33: 22118–22133.

Jacobs, R. A.; Jordan, M. I.; Nowlan, S. J.; and Hinton, G. E. 1991. Adaptive mixtures of local experts. *Neural Computation*.

Jordan, M. I.; and Jacobs, R. A. 1994. Hierarchical mixtures of experts and the EM algorithm. *Neural computation*, 6(2): 181–214.

Ju, W.; Fang, Z.; Gu, Y.; Liu, Z.; Long, Q.; Qiao, Z.; Qin, Y.; Shen, J.; Sun, F.; Xiao, Z.; et al. 2024. A comprehensive survey on deep graph representation learning. *Neural Networks*, 173: 106207.

Jure, L. 2014. SNAP Datasets: Stanford large network dataset collection. *Retrieved December 2021 from http://snap. stanford. edu/data*.

Kim, S.; Lee, D.; Kang, S.; Lee, S.; and Yu, H. 2023. Learning Topology-Specific Experts for Molecular Property Prediction. arXiv:2302.13693.

Kipf, T. N.; and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Li, W.; Paffenroth, R. C.; and Berthiaume, D. 2021. Neural network ensembles: theory, training, and the importance of explicit diversity. *arXiv preprint arXiv:2109.14117*.

Luo, Y.; Shi, L.; and Wu, X.-M. 2024. Classic GNNs are Strong Baselines: Reassessing GNNs for Node Classification. *Advances in Neural Information Processing Systems*.

Ma, L.; Han, H.; Li, J.; Shomer, H.; Liu, H.; Gao, X.; and Tang, J. 2024. Mixture of Link Predictors. *CoRR*.

Mukherjee, A.; Venkataraman, V.; Liu, B.; and Glance, N. 2013. What yelp fake review filter might be doing? In *AAAI*, 409–418.

Rampášek, L.; Galkin, M.; Dwivedi, V. P.; Luu, A. T.; Wolf, G.; and Beaini, D. 2022. Recipe for a General, Powerful, Scalable Graph Transformer. *NeurIPS*, 35.

Rozemberczki, B.; and Sarkar, R. 2020. Characteristic functions on graphs: Birds of a feather, from statistical descriptors to parametric models. In *29th ACM international conference on information & knowledge management*, 1325–1334.

Shazeer, N.; Mirhoseini, A.; Maziarz, K.; Davis, A.; Le, Q.; Hinton, G.; and Dean, J. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Wang, G.; Ying, R.; Huang, J.; and Leskovec, J. 2020. Multi-hop attention graph neural network. *arXiv preprint arXiv:2009.14332*.

Wang, H.; Jiang, Z.; You, Y.; Han, Y.; Liu, G.; Srinivasa, J.; Kompella, R. R.; and Wang, Z. 2023. Graph Mixture of Experts: Learning on Large-Scale Graphs with Explicit Diversity Modeling. arXiv:2304.02806.

Wu, F.; Zhang, T.; de Souza Jr. au2, A. H.; Fifty, C.; Yu, T.; and Weinberger, K. Q. 2019. Simplifying Graph Convolutional Networks. arXiv:1902.07153.

Wu, Q.; Zhao, W.; Yang, C.; Zhang, H.; Nie, F.; Jiang, H.; Bian, Y.; and Yan, J. 2023. Sgformer: Simplifying and empowering transformers for large-graph representations. *NeurIPS*, 36: 64753–64773.

Wu, S.; Cao, K.; Ribeiro, B.; Zou, J.; and Leskovec, J. 2024a. GraphMETRO: Mitigating Complex Graph Distribution Shifts via Mixture of Aligned Experts. In *NeurIPS*.

Wu, S.; Luo, J.; Chen, X.; Li, L.; Zhao, X.; Yu, T.; Wang, C.; Wang, Y.; Wang, F.; Qiao, W.; et al. 2024b. Yuan 2.0-m32: Mixture of experts with attention router. *arXiv preprint arXiv:2405.17976*.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2018a. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*.

Xu, K.; Li, C.; Tian, Y.; Sonobe, T.; Kawarabayashi, K.-i.; and Jegelka, S. 2018b. Representation learning on graphs with jumping knowledge networks. In *ICML*, 5453–5462. PMLR.

Yang, E.; Shen, L.; Guo, G.; Wang, X.; Cao, X.; Zhang, J.; and Tao, D. 2024. Model merging in llms, mllms, and beyond: Methods, theories, applications and opportunities. *arXiv preprint arXiv:2408.07666*.

Yang, Z.; Yu, Y.; You, C.; Steinhardt, J.; and Ma, Y. 2020. Rethinking bias-variance trade-off for generalization of neural networks. In *ICML*, 10767–10777. PMLR.

Yao, Z.; Liu, C.; Meng, X.; Zhan, Y.; Wu, J.; Pan, S.; and Hu, W. 2024. DA-MoE: Addressing Depth-Sensitivity in Graph-Level Analysis through Mixture of Experts. *arXiv preprint arXiv:2411.03025*.

Yun, L.; Zhuang, Y.; Fu, Y.; Xing, E. P.; and Zhang, H. 2024. Toward inference-optimal mixture-of-expert large language models. *arXiv preprint arXiv:2404.02852*.

Zhang, D.; Song, J.; Bi, Z.; Yuan, Y.; Wang, T.; Yeong, J.; and Hao, J. 2025. Mixture of Experts in Large Language Models. *arXiv preprint arXiv:2507.11181*.

Zhou, Y.; Lei, T.; Liu, H.; Du, N.; Huang, Y.; Zhao, V.; Dai, A.; Chen, Z.; Le, Q.; and Laudon, J. 2022. Mixture-of-experts with expert choice routing, 2022. *URL https://arxiv.org/abs/2202.09368*.