UNCERTAINTY-GUIDED ADAPTIVE EXPERT SELECTION FOR GRAPH NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph Mixture-of-Experts (MoE) models use a router to assign node-specific probabilities to expert networks and combine selected expert outputs for node classification. The router typically varies which experts are selected while fixing their number across nodes. Our observations show that the benefit of additional experts varies with predictive uncertainty, motivating node-specific control of expert participation. We propose D^{2}MoE, which uses entropy from cached predictions to specify how much of the router's probability distribution over experts should be retained. At each layer, experts are selected in descending probability order until this coverage target is reached. Predictive uncertainty thus determines the requested coverage, while the router's preferences determine which experts participate and how many are needed. Our analysis identifies a weighted participation cost in load balancing. In a controlled multiclass comparison with fixed routing probabilities, expert responses, and costs, the preferred expert count and retained probability are nondecreasing with reference entropy when the reference prediction is correct. Across 13 homophilous and heterophilous benchmarks, D^{2}MoE variants collectively achieve the highest performance among the evaluated methods, with relative gains of up to 7.92% on heterophilous graphs. End-to-end comparisons on two large graphs further show reductions reaching 73.07% in peak GPU memory and 46.53% in training time to early stopping.

1 INTRODUCTION

Node classification aims to infer node labels from node attributes and graph structure. However, nodes within the same graph may have substantially different local contexts. Some nodes lie in homophilous regions, where information from neighboring nodes reinforces their semantics. Others lie near class boundaries or in heterophilous or noisy neighborhoods, where neighborhood aggregation may introduce conflicting signals. Different nodes may therefore require different ways of processing graph information. Existing studies have expanded the range of graph transformations through higher-order message passing (Zhu et al., 2020), multi-channel graph filtering (Bo et al., 2021; Luan et al., 2022), and global attention (Chen et al., 2023; Zhang et al., 2022; Wu et al., 2023a; Shirzad et al., 2023). As these transformations become increasingly diverse, an important question is how to organize and apply them according to the context of each node.

Mixture-of-Experts (MoE) models offer a way to address this problem. An MoE model consists of multiple expert networks and a learnable router that assigns probabilities to the experts and selects a subset of them for representation learning (Shazeer et al., 2017). In graph learning, the experts may implement different aggregation ranges, propagation rules, or graph filters, allowing different graph transformations to be assigned to different nodes (Wang et al., 2023; Yao et al., 2025; Han et al., 2024; Chen et al., 2025; Zeng et al., 2024; Deng et al., 2025). However, many existing Graph MoE models rely on fixed top-k routing. Although the router can assign different experts to different nodes, the predefined value of k enforces the same expert count across the graph. This constraint overlooks the shape of each node's routing distribution. A concentrated distribution allows a few top-ranked experts to capture most of the probability mass, whereas a diffuse distribution requires more experts to achieve comparable coverage. Consequently, fixed top-k retains different proportions of the routing distribution for different nodes despite selecting the same number of experts. Recent routing methods have begun to vary the number of selected experts across inputs (Huang et al., 2024; Zhao et al.,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

2026; Meena et al., 2026), highlighting the need for a node-specific criterion that determines how much routing probability should be retained in Graph MoEs.

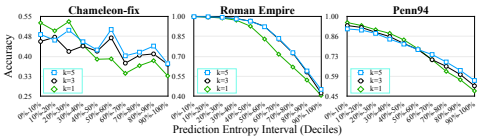

Figure 1: Node-classification accuracy of fixed top-k Graph MoE variants across predictive-entropy deciles. All variants use the same architecture, with six independently parameterized two-layer GCN experts, and are evaluated on common deciles defined by a separately trained two-layer GCN.

To examine how fixed expert counts affect nodes with different predictive states, we separately train three otherwise matched Graph MoE variants with $k \in \{1, 3, 5\}$. An independently trained GCN ranks the test nodes by predictive entropy and defines the same node groups for all variants. Figure 1 shows that these variants differ in relative performance across datasets and predictive-entropy intervals. Increasing the expert count does not consistently improve performance across all node groups. In several low-entropy intervals, fewer experts achieve similar performance to larger expert sets. In some high-entropy intervals, a larger expert set provides clearer gains. These results motivate replacing a uniform expert count with node-specific routing coverage.

To determine this node-specific routing coverage, we consider both the predictive state of the node and the router's preference over individual experts. Predictive entropy measures the uncertainty of the current node prediction, while the routing distribution represents the relative preference for each expert. We therefore formulate expert selection in terms of probability coverage. Predictive uncertainty determines how much routing probability mass should be retained, while the routing distribution determines which experts are selected and how many participate.

Accordingly, we propose **D**ifficulty-Aware **D**ynamic **M**ixture-**o**f-**E**xperts (D^{2}MoE), which links node predictive uncertainty to routing coverage. D^{2}MoE estimates node uncertainty from cached model predictions and maps it to a node-specific probability-coverage threshold. At each Graph MoE layer, dynamic top-p routing selects the smallest expert set whose cumulative routing probability reaches the threshold. The selected expert outputs are then aggregated to construct the node representation. New predictions update the node uncertainty estimates and guide subsequent routing decisions. In this way, predictive uncertainty controls the amount of routing probability mass retained for each node, while the learned routing distribution determines the identities and number of experts that contribute to its representation. Our contributions are as follows.

- We propose D^{2}MoE, which uses predictive entropy to set node-specific probability-coverage targets. Dynamic top-p routing determines which experts participate and how many are selected.
- We identify the participation cost in load balancing. In a controlled multiclass model with fixed routing, expert responses, and costs, we prove that preferred expert count and coverage are nondecreasing with reference entropy when the reference prediction is correct.
- We evaluate D^{2}MoE against 20 baselines on 13 benchmarks. Its variants collectively achieve the highest mean scores, with relative gains of up to 7.92% on heterophilous graphs. Ablation and routing analyses further support the proposed design.

2 Preliminaries

Notation. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote a graph with $N = |\mathcal{V}|$ nodes, adjacency matrix $\boldsymbol{A}$, and node feature matrix $\boldsymbol{X} \in \mathbb{R}^{N \times d}$. Each node $v \in \mathcal{V}$ is associated with a feature vector $\boldsymbol{x}_v$ and a label $y_v \in \{1, \ldots, C\}$, where C is the number of classes. $\mathcal{N}(v)$ denotes the neighborhood of node v.

Task. We consider transductive node classification, where the graph structure and node features are available for all nodes during training, while labels are observed only for a training subset $\mathcal{V}_{\text{train}} \subset \mathcal{V}$. Given $\mathcal{G}$ and the labels of the training nodes, the objective is to learn a model that predicts the labels of the remaining nodes. For each node v, the model produces a class probability vector $\boldsymbol{P}_v \in [0, 1]^C$.

Graph Message Passing. A GNN layer updates node v by aggregating neighboring representations.

$$\boldsymbol{h}_v^{(\ell+1)} = \mathrm{UP}\left(\boldsymbol{h}_v^{(\ell)}, \mathrm{AGG}\left(\left\{\boldsymbol{h}_u^{(\ell)} \mid u \in \mathcal{N}(v)\right\}\right)\right), \tag{1}$$

where $\boldsymbol{h}_v^{(0)} = \boldsymbol{x}_v$, AGG collects information from neighboring nodes, and UP combines the aggregated information with the current node representation. When expert routing is incorporated

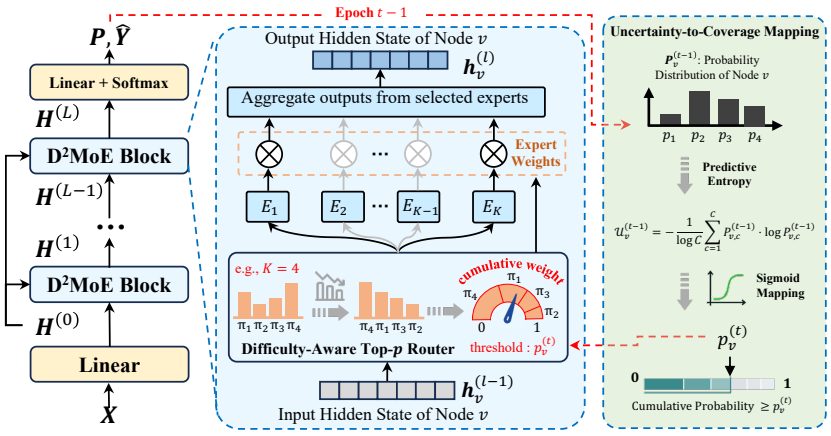

Figure 2: Overview of D^{2}MoE. Cached predictive entropy determines node-specific probability coverage. Each Graph MoE block uses its routing distribution to select and aggregate experts under this coverage. The predictions refresh the predictive-entropy cache for subsequent routing decisions.

into graph layers, the selected experts directly affect the representation produced for each node. Subsequent message-passing operations can propagate this change to neighboring nodes, coupling the effects of routing decisions across layers.

3 Related Work

Graph Mixtures-of-Experts. Graph MoE models extend input-dependent expert routing (Shazeer et al., 2017) to variations in node features and local structure. Existing designs diversify experts through aggregation ranges and propagation depths (Wang et al., 2023; Yao et al., 2025; Deng et al., 2025), spectral responses (Han et al., 2024), or combinations of propagation and transformation operators (Chen et al., 2025). Mowst coordinates weak and strong predictors through a confidence-based gate (Zeng et al., 2024). These methods select a transformation or predictor for each node, while participation counts are generally governed by fixed budgets or predefined gating outcomes.

Dynamic Expert Routing. Motivated by differences in input difficulty, Huang et al. (2024) propose confidence-based dynamic routing for Transformer MoEs that activates the smallest expert subset whose cumulative routing probability exceeds a globally fixed threshold. DTop-p MoE adjusts a global threshold with a proportional-integral controller to meet target sparsity and learns layer-specific routing normalization (Jin et al., 2026). ProbMoE formulates routing as probabilistic inference over expert subsets with fixed or variable cardinality (Zhao et al., 2026). In graph learning, SAGMM combines topology-aware attention gating with expert pruning to adapt expert participation (Meena et al., 2026). These studies adapt routing cardinality through router confidence, global sparsity control, probabilistic subset modeling, or topology-aware gating. Using node-level predictive signals to set probability coverage under graph message passing remains less explored.

4 Methodology

D^{2}MoE combines a Graph MoE backbone with uncertainty-conditioned expert routing. Figure 2 illustrates the resulting architecture and the interaction between representation learning and cached predictions. We first define the backbone and training objective, then analyze how predictive uncertainty affects the preferred participation scale in a controlled comparison. We then present the coverage rule based on cached entropy and the optimization procedure.

4.1 Backbone Architecture of D^{2}MoE

The backbone defines the complete forward path from the input features to the node predictions. It consists of an input projection, L stacked Graph MoE blocks, and a linear prediction head. For clarity,

we omit the epoch index from backbone representations, expert outputs, and routing probabilities, while retaining it for routing quantities that explicitly depend on the predictive-uncertainty cache.

Initial Embedding. The input node features $\boldsymbol{X} \in \mathbb{R}^{N\times d}$ are first projected into the h-dimensional hidden space. We apply a learnable linear projection followed by ReLU and dropout, yielding

$$\boldsymbol{H}^{(0)} = \mathrm{Dropout}\big(\mathrm{ReLU}(\boldsymbol{X}\boldsymbol{W}_0)\big), \tag{2}$$

where $\boldsymbol{W}_0 \in \mathbb{R}^{d\times h}$ is the projection matrix and $\boldsymbol{H}^{(0)} \in \mathbb{R}^{N\times h}$ stores initial node representations.

Stacked Graph MoE Blocks. The initial representations $\boldsymbol{H}^{(0)}$ are refined through L stacked Graph MoE blocks. At block $\ell \in \{1,\dots,L\}$, the input $\boldsymbol{H}^{(\ell-1)}$ is processed by an expert bank $\mathcal{E}^{(\ell)} = \{f_1^{(\ell)},\dots,f_K^{(\ell)}\}$ together with an MLP router $\mathrm{R}^{(\ell)}$. All K experts within a model use the same GNN architecture and propagation depth but maintain independent parameters, allowing them to learn distinct graph transformations. Expert i produces

$$\boldsymbol{Z}_i^{(\ell)} = f_i^{(\ell)}\big(\boldsymbol{H}^{(\ell-1)}, \boldsymbol{A}\big). \tag{3}$$

The K expert outputs provide candidate transformations for each node. Since nodes may benefit from different experts, the router assigns a node-specific probability distribution over experts

$$\boldsymbol{\pi}_v^{(\ell)} = \mathrm{Softmax}\big(\mathrm{R}^{(\ell)}(\boldsymbol{h}_v^{(\ell-1)})\big) = \mathrm{Softmax}\big(\boldsymbol{W}_2^{(\ell)}\sigma(\boldsymbol{W}_1^{(\ell)}\boldsymbol{h}_v^{(\ell-1)})\big), \tag{4}$$

where σ is the hidden activation and $\boldsymbol{W}_1^{(\ell)}$ and $\boldsymbol{W}_2^{(\ell)}$ are learnable parameters. The routing probability $\pi_{v,i}^{(\ell)}$ expresses the router's preference for assigning expert i to node v. These block-specific probabilities are used to construct the expert mixtures defined in Section 4.4.

Routed Representation Update. Let $\mathcal{S}_v^{(\ell,t)}$ and $\bar{\pi}_{v,i}^{(\ell,t)}$ denote the participating expert set and normalized routing weights from Section 4.4. Using these weights, block ℓ aggregates the selected expert outputs for node v and adds the resulting mixture to the shared initial representation

$$\boldsymbol{u}_v^{(\ell)} = \boldsymbol{h}_v^{(0)} + \sum_{i\in\mathcal{S}_v^{(\ell,t)}} \bar{\pi}_{v,i}^{(\ell,t)}[\boldsymbol{Z}_i^{(\ell)}]_v. \tag{5}$$

The shared residual gives every block direct access to the initial projected features. The matrix $\boldsymbol{U}^{(\ell)}$ formed by the node-wise updates is then normalized and transformed as

$$\boldsymbol{H}^{(\ell)} = \mathrm{Dropout}\left(\mathrm{ReLU}\left(\mathrm{Norm}^{(\ell)}(\boldsymbol{U}^{(\ell)})\right)\right), \tag{6}$$

where $\mathrm{Norm}^{(\ell)}$ is the normalization layer. Dropout is used in training and disabled in evaluation.

Output Layer. After L Graph MoE blocks, the final node representations are passed through a linear classification head to obtain class probabilities and predicted labels

$$\boldsymbol{P} = \mathrm{Softmax}\big(\mathrm{Linear}(\boldsymbol{H}^{(L)})\big), \qquad \widehat{Y}_v = \arg\max_c P_{v,c}. \tag{7}$$

These probabilities are used to compute the classification loss and predictive entropy. We introduce the training objective, whose dependence on participation underlies the routing analysis.

4.2 Training Objectives

Given the participating sets and normalized mixture weights, we train the backbone using the node-classification loss together with routing-entropy and load-balancing regularization. These terms specify both the predictive objective and how expert participation enters training.

Classification Objective. The node-classification objective is the cross-entropy loss

$$\mathcal{L}_{\text{task}} = -\frac{1}{|\mathcal{V}_{\text{train}}|}\sum_{v\in\mathcal{V}_{\text{train}}} \boldsymbol{Y}_v \cdot \log \boldsymbol{P}_v, \tag{8}$$

where $\mathcal{V}_{\text{train}}$ is the labeled training set and $\boldsymbol{Y}_v$ is the one-hot label of node v.

Routing Regularization. We use two complementary router regularizers to shape the distributions that determine expert participation. Routing-entropy regularization encourages each node's router to concentrate probability on its preferred experts, while load balancing discourages the same experts from dominating selections across the graph.

- **Routing entropy.** Following confidence-based dynamic routing (Huang et al., 2024), we minimize the entropy of the routing distribution. This encourages the router to concentrate probability on its preferred experts, so that a few experts can retain a large probability share. The loss is

$$\mathcal{L}_{\text{RE}} = -\frac{1}{N \cdot L} \sum_{\ell=1}^{L} \sum_{v=1}^{N} \sum_{i=1}^{K} \pi_{v,i}^{(\ell)} \cdot \log \pi_{v,i}^{(\ell)}. \tag{9}$$

- **Load balancing.** We use a graph-level load-balancing term to discourage repeated use of a small subset of experts (Shazeer et al., 2017)

$$\mathcal{L}_{\text{LB}} = \sum_{\ell=1}^{L} \left(K \sum_{i=1}^{K} T_i^{(\ell)} Q_i^{(\ell)} \right), \quad T_i^{(\ell)} = \frac{1}{N} \sum_{v=1}^{N} \mathbb{I}\{i \in \mathcal{S}_v^{(\ell,t)}\}, \quad Q_i^{(\ell)} = \frac{1}{N} \sum_{v=1}^{N} \pi_{v,i}^{(\ell)}. \tag{10}$$

  Here, $T_i^{(\ell)}$ is the fraction of nodes for which expert i participates, $Q_i^{(\ell)}$ is its mean routing probability, and $\mathbb{I}\{\cdot\}$ is the indicator function. The epoch dependence of $T_i^{(\ell)}$ and $Q_i^{(\ell)}$ is omitted for readability. The discrete selection indicators are treated as constants during backpropagation, while gradients pass through $Q_i^{(\ell)}$.

Total Objective. The overall training objective is

$$\mathcal{L} = \mathcal{L}_{\text{task}} + \lambda_1 \cdot \mathcal{L}_{\text{RE}} + \lambda_2 \cdot \mathcal{L}_{\text{LB}}, \tag{11}$$

where λ_1 and λ_2 weight the routing-entropy and load-balancing terms, respectively.

4.3 Theoretical Rationale for Uncertainty-Conditioned Routing

The observations in Section 1 motivate allocating participation according to predictive state. We examine this choice through the classification and load-balancing terms in Eq. (11). Substituting the selection frequencies into the latter gives

$$\mathcal{L}_{\text{LB}}^{(\ell)} = \frac{1}{N} \sum_{v \in \mathcal{V}} \sum_{i \in \mathcal{S}_v^{(\ell,t)}} K Q_i^{(\ell)}, \tag{12}$$

where $Q_i^{(\ell)}$ is the graph-wide mean routing probability. Thus each participating expert incurs a positive cost. Under uniform mean probabilities $Q_i^{(\ell)} = 1/K$, it equals the average expert count.

Fix all positive routing probabilities and consider one labeled node and block with $K \geq 1$ experts. Suppressing node and block indices, let ρ order experts by decreasing probability, with fixed tie breaking, and write $S_m = \{\rho(1), \ldots, \rho(m)\}$. Its retained probability is $s_m = \sum_{j=1}^{m} \pi_{\rho(j)}$ and its participation cost is $c_m = K \sum_{j=1}^{m} Q_{\rho(j)}$. The prefix uses normalized weights π_i / s_m. Both quantities increase strictly with m because the probabilities are positive.

At a fixed model state, let $\boldsymbol{b}_m \in \mathbb{R}^C$ be the finite classifier logits produced by this normalized prefix, including downstream graph operations and normalization layers. To isolate the role of existing class evidence while preserving these expert responses, consider

$$\boldsymbol{o}_m(a) = a\boldsymbol{e}_y + \boldsymbol{b}_m, \qquad a \in \mathbb{R}, \tag{13}$$

where $\boldsymbol{e}_y$ is the unit vector for true class y and $C \geq 2$. Varying a changes shared true-class evidence while all $\boldsymbol{b}_m$, routing probabilities, and costs remain fixed. Successive prefixes need not improve classification, so the comparison allows harmful extensions. Holding other nodes' predictions and other participation sets fixed yields the local comparison

$$J_a(m) = \ell_m(a) + \lambda c_m, \qquad m^*(a) = \min \arg\min_{1 \leq m \leq K} J_a(m), \tag{14}$$

where $\ell_m(a) = -\log \text{Softmax}(\boldsymbol{o}_m(a))_y$, $\lambda > 0$ weights participation cost, and ties favor the shorter prefix. Appendix A.1 derives its correspondence to the training objective.

Use the first prefix as a common reference, $\boldsymbol{q}^{\text{ref}}(a) = \text{Softmax}(\boldsymbol{o}_1(a))$, with normalized entropy $\mathcal{U}_{\text{ref}}(a) = -\sum_{c=1}^{C} q_c^{\text{ref}}(a) \log q_c^{\text{ref}}(a) / \log C$. This reference measures uncertainty independently of the candidate count. It separates changes in class evidence from changes in the expert mixture.

Theorem 1 (Uncertainty and preferred expert participation)**.** *Under the fixed routing probabilities, costs, and output family above, $m^*(a)$ is nonincreasing in a. Where $q_y^{\mathrm{ref}}(a) \geq q_c^{\mathrm{ref}}(a)$ for every class c, both $m^*(a)$ and $s_{m^*(a)}$ are nondecreasing functions of $\mathcal{U}_{\mathrm{ref}}(a)$.*

Proof sketch. A longer prefix with no classification benefit is dominated by its higher cost. For a beneficial extension, cross-entropy curvature makes its loss reduction decrease as shared evidence increases, while its additional cost stays fixed. Pairwise optimality therefore excludes an increase in preferred count. On the stated reference range, entropy strictly decreases with evidence, yielding the coverage conclusion. Appendix A.3 gives the full proof. □

For a coverage target $p \in (0, 1]$, probability ordering gives

$$k(p) = \min\{m \in \{1, \ldots, K\} : s_m \geq p\}, \tag{15}$$

the fewest experts attaining that target. Choosing $p = s_{m^*(a)}$ realizes the preferred prefix in the controlled family. Thus the preferred prefixes define an uncertainty-conditioned coverage direction. Section 4.4 implements this direction with cached predictive entropy and an increasing sigmoid map. The cache supplies the available prediction before routing, and each block's probability distribution determines the selected prefix. Expert count therefore also depends on the concentration of this distribution. Appendix A.4 details this connection.

4.4 Uncertainty-Conditioned Expert Routing

The analysis in Section 4.3 motivates adjusting routing coverage according to predictive uncertainty. To obtain this signal before selecting experts, D^{2}MoE caches the entropy of its preceding predictions. A sigmoid maps this entropy to a node-specific coverage target, and the block-specific routing probabilities determine the selected experts and their normalized weights.

Cached Predictive Uncertainty. Routing precedes the predictions of the same forward pass. The uncertainty used for routing is therefore carried over from the preceding epoch. After epoch $t-1$, we compute each node's predictive entropy from the updated model and cache it for routing in epoch t. Let $\boldsymbol{P}_v^{(t-1)} \in [0, 1]^C$ denote the class-probability vector of node v obtained at the end of epoch $t-1$. Its normalized predictive entropy is

$$\mathcal{U}_v^{(t-1)} = -\frac{1}{\log C} \sum_{c=1}^{C} P_{v,c}^{(t-1)} \log P_{v,c}^{(t-1)}, \tag{16}$$

where C is the number of classes. The normalization by $\log C$ places $\mathcal{U}_v^{(t-1)}$ in $[0, 1]$, where a larger value indicates higher predictive uncertainty. The resulting value is stored for node v and used to determine its routing coverage in epoch t.

Uncertainty-to-Coverage Mapping. We next map the cached entropy to a node-specific routing coverage threshold. Mapping uncertainty directly to an expert count would fix the cardinality without accounting for the distribution of routing probability across experts. Instead, D^{2}MoE uses uncertainty to specify the probability mass to be retained, while the block-specific routing distribution determines how many experts are required to reach this coverage. For $t \geq 2$, node v's coverage threshold is

$$p_v^{(t)} = \mathrm{Sigmoid}\left(\gamma\left(\mathcal{U}_v^{(t-1)} - \bar{\mathcal{U}}^{(t-1)}\right)\right), \qquad t \geq 2, \tag{17}$$

where $\bar{\mathcal{U}}^{(t-1)}$ is the mean cached entropy over all nodes and $\gamma > 0$ controls the sensitivity of the threshold to relative uncertainty. The mapping compares node v's uncertainty with the graph-wide mean. Accordingly, $p_v^{(t)} = 0.5$ when the two are equal, and $p_v^{(t)} > 0.5$ when the node uncertainty exceeds the graph-wide mean. For each node v, D^{2}MoE computes a node-specific coverage threshold $p_v^{(t)}$ shared across all Graph MoE blocks during epoch t. Because each block has its own routing distribution, the same threshold can yield different expert sets across blocks. Before cached uncertainty becomes available, we set $p_v^{(1)} = 1$ and retain all K experts during the first epoch.

Cumulative-Probability Expert Selection. We construct these sets using dynamic top-p expert selection. For node v at block ℓ, the cumulative-probability rule defines the participating expert set as

$$\mathcal{S}_v^{(\ell,t)} \in \underset{\mathcal{S} \subseteq \{1,\ldots,K\}}{\arg\min} |\mathcal{S}| \quad \text{s.t.} \quad \sum_{i \in \mathcal{S}} \pi_{v,i}^{(\ell)} \geq p_v^{(t)}. \tag{18}$$

Table 1: Node-classification test performance (%), reported as mean and standard deviation over ten runs. The metric is accuracy except on Tolokers, where it is ROC-AUC. Bold and underlined values indicate the best and second-best means, respectively.

Method \ Dataset	Computers	Photo	Coauthor CS	Coauthor Physics	Wiki-CS	Facebook	Ogbn-arxiv	Actor	Chameleon -fix	Squirrel -fix	Tolokers	Roman -empire	Penn94
MLP	85.01 ± 0.84	92.00 ± 0.56	94.80 ± 0.35	96.11 ± 0.14	79.57 ± 0.81	76.86 ± 0.34	53.46 ± 0.35	37.14 ± 1.06	33.31 ± 2.32	34.47 ± 3.09	53.18 ± 6.35	65.98 ± 0.43	75.18 ± 0.35
GCN	91.17 ± 0.54	94.26 ± 0.59	93.40 ± 0.45	96.37 ± 0.20	83.80 ± 0.66	93.98 ± 0.34	69.71 ± 0.18	30.65 ± 1.06	41.85 ± 3.22	33.89 ± 2.61	70.34 ± 1.64	50.76 ± 0.46	80.45 ± 0.27
GAT	91.44 ± 0.43	94.42 ± 0.61	93.20 ± 0.64	96.28 ± 0.31	83.99 ± 0.73	94.03 ± 0.36	70.03 ± 0.42	30.58 ± 1.18	43.31 ± 3.42	36.27 ± 2.12	79.93 ± 0.77	57.34 ± 1.81	78.10 ± 1.28
SAGE	90.94 ± 0.56	95.41 ± 0.45	94.17 ± 0.46	96.69 ± 0.23	84.75 ± 0.64	93.72 ± 0.35	69.15 ± 0.18	37.60 ± 0.95	44.94 ± 3.67	36.61 ± 3.06	82.37 ± 0.64	77.77 ± 0.49	OOM
H2GCN	91.69 ± 0.33	95.59 ± 0.48	95.62 ± 0.27	97.00 ± 0.16	84.62 ± 0.66	94.36 ± 0.32	OOM	37.27 ± 1.27	43.09 ± 3.85	40.07 ± 2.73	81.34 ± 1.16	79.47 ± 0.43	75.91 ± 0.44
GPRGNN	91.80 ± 0.55	95.44 ± 0.33	95.17 ± 0.34	96.94 ± 0.20	84.84 ± 0.54	94.84 ± 0.24	69.95 ± 0.19	36.89 ± 0.83	44.27 ± 5.23	40.58 ± 2.00	73.84 ± 1.40	67.72 ± 0.63	84.34 ± 0.29
FAGCN	89.54 ± 0.75	94.44 ± 0.62	94.93 ± 0.22	96.91 ± 0.27	84.47 ± 0.75	91.90 ± 1.95	66.87 ± 1.48	37.59 ± 0.95	45.28 ± 4.33	41.05 ± 2.67	81.38 ± 1.34	75.83 ± 0.35	79.01 ± 1.09
ACMGCN	91.66 ± 0.78	95.42 ± 0.39	95.47 ± 0.33	97.00 ± 0.27	85.10 ± 0.77	94.27 ± 0.33	69.98 ± 0.11	36.89 ± 1.13	43.99 ± 2.02	36.58 ± 2.75	83.52 ± 0.87	81.57 ± 0.35	83.01 ± 0.46
FSGNN	91.03 ± 0.56	95.50 ± 0.41	95.51 ± 0.32	96.98 ± 0.20	85.10 ± 0.58	94.32 ± 0.32	71.09 ± 0.21	37.14 ± 1.06	45.79 ± 3.31	38.25 ± 2.62	83.87 ± 0.98	79.76 ± 0.41	83.87 ± 0.98
Vanilla GT	84.41 ± 0.72	91.58 ± 0.73	94.61 ± 0.30	OOM	79.05 ± 0.97	OOM	OOM	37.08 ± 1.08	44.27 ± 3.98	39.55 ± 3.10	72.24 ± 1.17	OOM	OOM
ANS-GT	90.01 ± 0.38	94.51 ± 0.24	93.93 ± 0.23	96.28 ± 0.19	83.27 ± 0.49	92.61 ± 0.16	OOM	37.80 ± 0.95	40.74 ± 2.26	36.65 ± 0.80	76.91 ± 0.85	80.36 ± 0.71	OOM
NAGphormer	90.22 ± 0.42	94.95 ± 0.52	94.96 ± 0.25	96.43 ± 0.20	84.31 ± 0.72	93.35 ± 0.28	70.25 ± 0.13	36.99 ± 1.39	46.12 ± 2.25	38.31 ± 2.43	66.73 ± 1.18	75.92 ± 0.69	73.98 ± 0.53
Exphormer	91.46 ± 0.51	95.42 ± 0.26	95.62 ± 0.29	96.89 ± 0.20	84.57 ± 0.58	93.88 ± 0.40	71.59 ± 0.24	36.83 ± 1.10	42.58 ± 3.24	36.19 ± 3.20	82.26 ± 0.41	87.55 ± 1.13	OOM
DIFFormer	91.52 ± 0.55	95.41 ± 0.38	95.49 ± 0.26	96.98 ± 0.22	83.54 ± 0.70	94.23 ± 0.47	OOM	36.73 ± 1.27	44.44 ± 3.20	40.45 ± 2.51	81.04 ± 4.16	78.97 ± 0.54	OOM
SGFormer	90.70 ± 0.59	94.46 ± 0.49	95.21 ± 0.20	96.87 ± 0.18	82.67 ± 0.58	86.66 ± 0.54	65.84 ± 0.24	36.59 ± 0.90	44.27 ± 3.68	38.83 ± 2.19	80.46 ± 0.91	76.41 ± 0.50	76.65 ± 0.49
GMoE	91.37 ± 0.49	94.51 ± 0.68	93.18 ± 0.58	96.48 ± 0.23	83.65 ± 0.61	94.90 ± 0.25	71.66 ± 0.29	33.78 ± 1.32	46.69 ± 3.55	42.24 ± 2.45	85.21 ± 0.40	84.78 ± 0.76	79.03 ± 0.78
NodeMoE	91.87 ± 0.33	95.53 ± 0.41	OOM	OOM	85.08 ± 0.62	94.84 ± 0.28	70.51 ± 0.29	36.28 ± 1.39	45.67 ± 4.54	40.49 ± 2.01	OOM	74.31 ± 0.87	OOM
Mowst	92.00 ± 0.43	95.49 ± 0.36	94.51 ± 0.43	96.73 ± 0.18	85.13 ± 0.84	95.12 ± 0.24	71.34 ± 0.24	37.68 ± 0.73	44.16 ± 3.55	38.52 ± 2.07	79.43 ± 0.66	79.56 ± 0.51	78.84 ± 1.99
DAMoE	91.57 ± 0.64	94.39 ± 0.53	93.42 ± 0.50	96.42 ± 0.28	84.04 ± 0.80	94.96 ± 0.21	71.50 ± 0.22	28.76 ± 1.01	45.51 ± 2.80	41.08 ± 2.08	51.45 ± 1.07	81.92 ± 0.52	78.04 ± 0.58
Moscat	91.65 ± 0.59	95.48 ± 0.47	94.78 ± 0.40	97.11 ± 0.16	81.40 ± 0.64	94.46 ± 0.53	70.37 ± 0.24	35.50 ± 1.67	43.37 ± 2.60	42.25 ± 2.66	81.79 ± 1.07	77.96 ± 0.37	83.91 ± 0.63
D²MoE (GCN)	**92.23 ± 0.33**	95.59 ± 0.52	**95.82 ± 0.15**	**97.14 ± 0.21**	84.81 ± 0.65	95.36 ± 0.26	**71.72 ± 0.25**	38.01 ± 1.15	**50.39 ± 4.71**	**44.11 ± 3.53**	**85.74 ± 0.51**	84.39 ± 0.55	**84.55 ± 0.51**
D²MoE (SAGE)	92.01 ± 0.38	95.64 ± 0.28	95.42 ± 0.17	96.89 ± 0.21	85.23 ± 0.65	**95.47 ± 0.37**	71.36 ± 0.17	**38.26 ± 1.06**	49.16 ± 5.39	40.11 ± 2.94	84.63 ± 0.81	**87.61 ± 0.26**	81.02 ± 0.63
D²MoE (GAT)	92.14 ± 0.39	**95.79 ± 0.47**	95.71 ± 0.25	97.07 ± 0.26	**85.45 ± 0.74**	95.44 ± 0.41	71.41 ± 0.24	37.53 ± 0.88	48.31 ± 4.74	42.40 ± 2.38	85.26 ± 0.81	84.09 ± 0.34	82.86 ± 0.47

We order the routing probabilities in decreasing order and take the shortest prefix that satisfies Eq. (18), with a fixed ordering used to resolve ties. Let $s_v^{(\ell,t)}$ denote the routing probability retained by the selected experts. Their probabilities are normalized as

$$s_v^{(\ell,t)} = \sum_{j \in \mathcal{S}_v^{(\ell,t)}} \pi_{v,j}^{(\ell)}, \quad \bar{\pi}_{v,i}^{(\ell,t)} = \begin{cases} \dfrac{\pi_{v,i}^{(\ell)}}{s_v^{(\ell,t)}}, & i \in \mathcal{S}_v^{(\ell,t)}, \\ 0, & i \notin \mathcal{S}_v^{(\ell,t)}. \end{cases} \tag{19}$$

The resulting set and normalized weights enter the backbone update in Eq. (5). In the notation of Eq. (15), this set is $S_{k(p_v^{(t)})}$ and its retained probability is $s_v^{(\ell,t)} = s_{k(p_v^{(t)})}$. Thus the coverage target determines the required probability mass, while the learned router specifies the participating prefix.

Training alternates parameter updates and entropy-cache refreshes (Algorithm 1, Appendix B).

5 Experiments

5.1 Experimental Setup

Datasets. We evaluate D^{2}MoE on 13 node-classification benchmarks. The seven homophilous graphs are Computers, Photo (McAuley et al., 2015), Coauthor CS, Coauthor Physics (Shchur et al., 2018), Wiki-CS, Facebook (Rozemberczki et al., 2021), and Ogbn-arxiv (Hu et al., 2020). The six heterophilous graphs are Actor (Tang et al., 2009), Chameleon-filtered, Squirrel-filtered, Roman-empire (Platonov et al., 2023), Tolokers, and Penn94 (Lim et al., 2021). We use a 48%/32%/20% training/validation/test split, except on Ogbn-arxiv, where we use the official OGB split. We report accuracy, except ROC-AUC on Tolokers.

Baselines. We compare D^{2}MoE with 20 baselines in four groups. Standard neural models include MLP, GCN (Kipf & Welling, 2017), GraphSAGE (Hamilton et al., 2017), and GAT (Veličković et al., 2018). Heterophily-oriented GNNs include H2GCN (Zhu et al., 2020), GPRGNN (Chien et al., 2021), FAGCN (Bo et al., 2021), ACMGCN (Luan et al., 2022), and FSGNN (Maurya et al., 2022). Graph transformers include Vanilla GT, ANS-GT (Zhang et al., 2022), NAGphormer (Chen et al., 2023), SGFormer (Wu et al., 2023b), Exphormer (Shirzad et al., 2023), and DIFFormer (Wu et al., 2023a). The Graph MoE baselines are GMoE (Wang et al., 2023), DAMoE (Yao et al., 2025), NodeMoE (Han et al., 2024), Mowst (Zeng et al., 2024), and Moscat (Deng et al., 2025).

Implementation Details. We report the mean and standard deviation over ten runs with different random seeds. We train with AdamW for at most 500 epochs, stopping after 100 epochs without validation improvement. Appendix C lists the datasets, baselines, and hyperparameter search ranges.

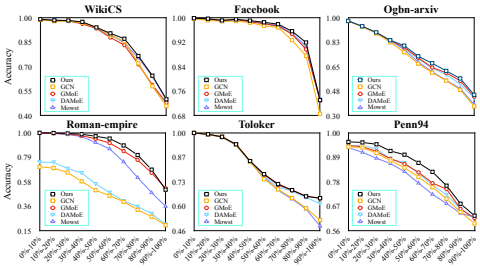

Fig. 3: Accuracy across entropy deciles.

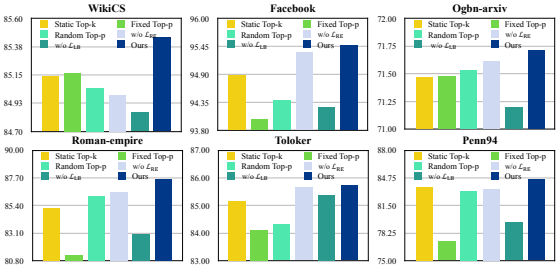

Fig. 4: Ablation analysis on D^{2}MoE.

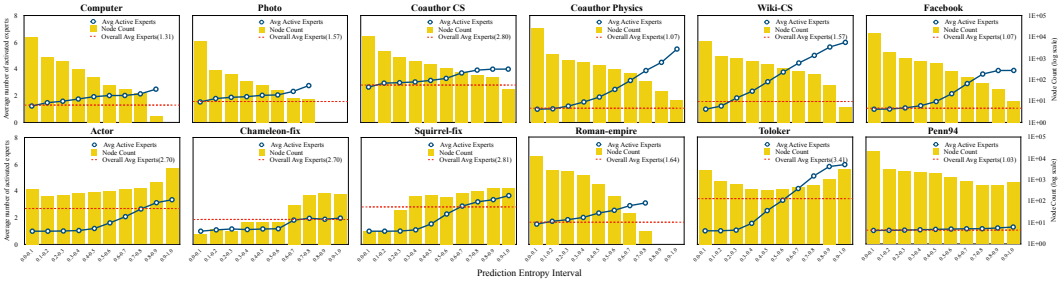

Figure 5: Node counts and mean selected expert counts across predictive-entropy intervals. Coverage thresholds increase with cached entropy, while routing probabilities determine the expert count.

5.2 Results and Analysis

Overall Performance. Table 1 shows that D^{2}MoE variants collectively achieve the best performance on all 13 datasets. Comparing the best D^{2}MoE variant with the strongest baseline on each dataset gives average relative performance gains of 0.22% and 2.41% across the seven homophilous and six heterophilous graphs, respectively. The larger average gains on heterophilous graphs align with the motivation to adapt expert participation to nodes receiving conflicting neighborhood cues.

Accuracy Across Predictive-Entropy Levels. Figure 3 uses common entropy deciles from an independent proxy, keeping the evaluated node subsets fixed across methods. D^{2}MoE remains competitive in low-entropy groups and often shows larger gains in the middle- and high-entropy deciles, where baseline accuracy declines. The larger gains on high-entropy nodes are consistent with our routing rule, which assigns higher coverage targets to nodes with less confident cached predictions. Appendix D.1 provides the full results.

Ablation Analysis. Static top-k and fixed top-p both lower performance in Figure 4. Fixed top-p already allows node-dependent counts, so its deficit supports adapting the coverage target beyond varying the count. Random top-p also varies coverage but underperforms entropy guidance, indicating that the allocation signal matters. Removing either regularizer reduces performance, consistent with their complementary roles. Routing entropy concentrates each node's preferences, while load balancing regulates graph-wide expert use and participation cost. Together, these comparisons support learning the routing distribution jointly with uncertainty-conditioned coverage. Appendix D.2 provides the settings and full results.

Expert Participation Across Entropy Intervals. Figure 5 shows an overall increase in mean expert count with predictive entropy across all datasets. The resulting allocation combines smaller mixtures for lower-entropy groups with broader participation for higher-entropy groups, consistent with the direction motivated by Section 4.3.

Expert Counts in the Embedding Space. The upper row of Figure 6 colors projected node representations by ground-truth class, and the lower row colors the same points by selected expert count. Regions where class labels overlap often align with brighter areas in the lower row, indicating larger expert counts, while clusters dominated by one class tend to use fewer experts. Together, the

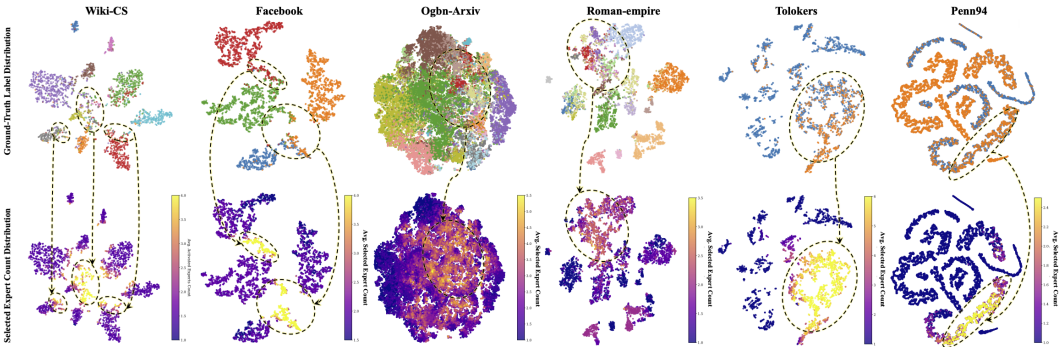

Figure 6: t-SNE of final-layer node representations on six datasets. The upper and lower rows show the same embeddings colored by ground-truth class and selected expert count, respectively.

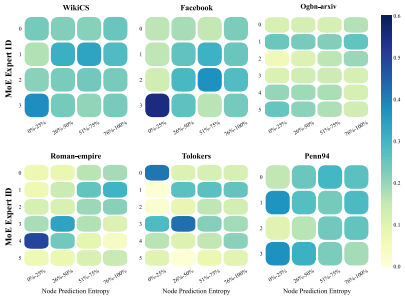

Fig. 7: Mean routing probabilities, shown by color. Rows denote experts and columns entropy quartiles.

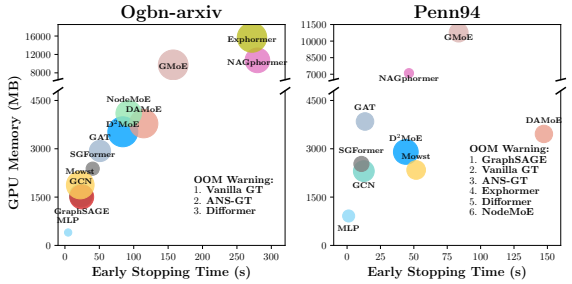

Fig. 8: Training efficiency on large graphs. Bubble area represents accuracy. "OOM Warning" denotes runs exceeding the memory limit.

two figures relate expert participation to both predictive entropy and local class overlap, supporting the motivation to allocate broader mixtures to ambiguous nodes. Appendix D.3 shows all 12 datasets.

Mean Routing Probabilities Across Entropy Levels. Figure 7 reports mean routing probabilities within predictive-entropy quartiles. In several datasets, lower-entropy groups favor a few experts, while higher-entropy groups exhibit more evenly distributed mean routing probabilities. For an individual node, a higher coverage target requests more probability mass, and a less concentrated routing distribution can require more experts to supply it. These two factors connect learned routing preferences with the expert counts in Figure 5. Appendix D.4 provides the full heatmaps.

Accuracy–Resource Tradeoff. In Figure 8, D^{2}MoE achieves higher mean accuracy than GMoE and DAMoE on both graphs with lower peak GPU memory and shorter time to early stopping, including reductions relative to GMoE of 73.07% in memory on Penn94 and 46.53% in training time on Ogbn-arxiv. Mowst uses less memory but attains lower accuracy on both graphs, illustrating the tradeoff between predictive performance and memory use.

6 Conclusion

This work presents D^{2}MoE, a Graph MoE method that adapts expert participation to predictive uncertainty through probability coverage. Experiments show that this design improves node classification across homophilous and heterophilous graphs, with favorable accuracy–resource tradeoffs on two large graphs. The effectiveness of this adaptation nevertheless depends on the reliability of cached predictions, since entropy may underestimate difficulty when predictions are confidently wrong. The current implementation also evaluates all experts before aggregation, so reduced participation does not directly save expert computation. These limitations motivate more reliable uncertainty estimation and selective execution that preserves graph dependencies.

AI USE STATEMENT

In this work, we used generative AI tools solely for language editing and improving the readability of the manuscript. The authors take full responsibility for the final content of this work.

ETHICS STATEMENT

We evaluate D^2MoE on the existing node-classification benchmarks described in Section 5. When applied to social graphs, predictions and uncertainty estimates may reflect biases in observed relationships and attributes, leading to uneven performance across groups. Deployment involving personal graph data should therefore include privacy safeguards and evaluation of prediction quality and expert allocation across relevant groups.

REPRODUCIBILITY STATEMENT

Section 4 specifies the backbone, routing rule, and training objectives. Appendix A provides the assumptions and derivations supporting the theoretical result, and Appendix B gives the training algorithm and entropy-cache update procedure. Section 5 and Appendix C describe the datasets, data splits, metrics, baselines, hyperparameter search ranges, optimization settings, and reporting over ten runs with different random seeds. Additional experimental results are provided in Appendix D. We have submitted the code artifacts as supplementary material for anonymous review and will make the code publicly available upon acceptance.

REFERENCES

Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. Beyond low-frequency information in graph convolutional networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 3950–3957, 2021.

Jinsong Chen, Kaiyuan Gao, Gaichao Li, and Kun He. Nagphormer: A tokenized graph transformer for node classification in large graphs. In *Proceedings of the Eleventh International Conference on Learning Representations*, pp. 1–18, 2023.

Xuanze Chen, Jiajun Zhou, Shanqing Yu, and Qi Xuan. Mixture of experts meets decoupled message passing: Towards general and adaptive node classification. In *Companion Proceedings of the ACM on Web Conference 2025*, pp. 907–910, 2025.

Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive universal generalized pagerank graph neural network. In *Proceedings of the Ninth International Conference on Learning Representations*, pp. 1–24, 2021.

Gangda Deng, Hongkuan Zhou, Rajgopal Kannan, and Viktor Prasanna. Mixture of scope experts at test: Generalizing deeper graph neural networks with shallow variants. *Advances in Neural Information Processing Systems*, 38:36155–36198, 2025.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, volume 30, pp. 1–11, 2017.

Haoyu Han, Juanhui Li, Wei Huang, Xianfeng Tang, Hanqing Lu, Chen Luo, Hui Liu, and Jiliang Tang. Node-wise filtering in graph neural networks: A mixture of experts approach. *arXiv preprint arXiv:2406.03464*, 2024.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020.

Quzhe Huang, Zhenwei An, Nan Zhuang, Mingxu Tao, Chen Zhang, Yang Jin, Kun Xu, Liwei Chen, Songfang Huang, and Yansong Feng. Harder task needs more experts: Dynamic routing in moe models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pp. 12883–12895, 2024.

Can Jin, Hongwu Peng, Mingcan Xiang, Qixin Zhang, Xiangchi Yuan, Amit Hasan, Ohi Dibua, Yifan Gong, Yan Kang, and Dimitris N. Metaxas. DTop-p moe: Sparsity-controlled dynamic top-p moe for foundation model pre-training. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*. PMLR, 2026.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *Proceedings of the Fifth International Conference on Learning Representations*, pp. 1–14, 2017.

Derek Lim, Felix Hohne, Xiuyu Li, Sijia Linda Huang, Vaishnavi Gupta, Omkar Bhalerao, and Ser Nam Lim. Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods. In *Advances in Neural Information Processing Systems*, volume 34, pp. 20887–20902, 2021.

Sitao Luan, Chenqing Hua, Qincheng Lu, Jiaqi Zhu, Mingde Zhao, Shuyuan Zhang, Xiao-Wen Chang, and Doina Precup. Revisiting heterophily for graph neural networks. *Advances in neural information processing systems*, 35:1362–1375, 2022.

Sunil Kumar Maurya, Xin Liu, and Tsuyoshi Murata. Simplifying approach to node classification in graph neural networks. *Journal of Computational Science*, 62:101695, 2022. ISSN 1877-7503. doi: https://doi.org/10.1016/j.jocs.2022.101695.

Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pp. 43–52, 2015.

Mohit Meena, Yash Punjabi, Vishal Sharma, Mahesh Chandran, et al. Self-adaptive graph mixture of models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 24344–24352, 2026.

Oleg Platonov, Denis Kuznedelev, Michael Diskin, Artem Babenko, and Liudmila Prokhorenkova. A critical look at the evaluation of GNNs under heterophily: Are we really making progress? In *Proceedings of the Eleventh International Conference on Learning Representations*, pp. 1–15, 2023.

Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. *Journal of Complex Networks*, 9(2):cnab014, 2021.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *Proceedings of the Fifth International Conference on Learning Representations*, pp. 1–19, 2017.

Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *Relational Representation Learning Workshop (R2L 2018), NeurIPS 2018*, 2018.

Hamed Shirzad, Ameya Velingker, Balaji Venkatachalam, Danica J Sutherland, and Ali Kemal Sinop. Exphormer: Sparse transformers for graphs. In *International conference on machine learning*, pp. 31613–31632. PMLR, 2023.

Jie Tang, Jimeng Sun, Chi Wang, and Zi Yang. Social influence analysis in large-scale networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 807–816, 2009.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *Proceedings of the Sixth International Conference on Learning Representations*, pp. 1–12, 2018.

Haotao Wang, Ziyu Jiang, Yuning You, Yan Han, Gaowen Liu, Jayanth Srinivasa, Ramana Kompella, Zhangyang Wang, et al. Graph mixture of experts: Learning on large-scale graphs with explicit diversity modeling. *Advances in Neural Information Processing Systems*, 36:50825–50837, 2023.

Qitian Wu, Chenxiao Yang, Wentao Zhao, Yixuan He, David Wipf, and Junchi Yan. DIFFormer: Scalable (graph) transformers induced by energy constrained diffusion. In *Proceedings of the Eleventh International Conference on Learning Representations*, pp. 1–26, 2023a.

Qitian Wu, Wentao Zhao, Chenxiao Yang, Hengrui Zhang, Fan Nie, Haitian Jiang, Yatao Bian, and Junchi Yan. Sgformer: Simplifying and empowering transformers for large-graph representations. *Advances in neural information processing systems*, 36:64753–64773, 2023b.

Zelin Yao, Mukun Chen, Chuang Liu, Xianke Meng, Yibing Zhan, Jia Wu, Shirui Pan, Huiting Xu, and Wenbin Hu. Da-moe: addressing depth-sensitivity in graph-level analysis through mixture of experts. *Neural Networks*, pp. 108064, 2025.

Hanqing Zeng, Hanjia Lyu, Diyi Hu, Yinglong Xia, and Jiebo Luo. Mixture of weak and strong experts on graphs. In *Proceedings of the Twelfth International Conference on Learning Representations*, volume 2024, pp. 24462–24502, 2024.

Zaixi Zhang, Qi Liu, Qingyong Hu, and Chee-Kong Lee. Hierarchical graph transformer with adaptive node sampling. *Advances in Neural Information Processing Systems*, 35:21171–21183, 2022.

Heng Zhao, Zilei Shao, Guy Van den Broeck, and Zhe Zeng. Probmoe: Differentiable probabilistic routing for mixture-of-experts. In *Forty-third International Conference on Machine Learning*, pp. 1–21, 2026.

Jiajun Zhou, Shengbo Gong, Xuanze Chen, Chenxuan Xie, Shanqing Yu, Qi Xuan, and Xiaoniu Yang. Clarify confused nodes via separated learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(4):2882–2896, 2025. doi: 10.1109/TPAMI.2025.3528738. URL https://doi.org/10.1109/TPAMI.2025.3528738.

Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in neural information processing systems*, 33:7793–7804, 2020.

APPENDIX

A DETAILS OF THE UNCERTAINTY-CONDITIONED PARTICIPATION PRINCIPLE

We prove the participation-cost identity and monotonicity result in Section 4.3, then relate the preferred prefixes to cumulative-probability routing. The analysis uses $C \geq 2$ classes, $K \geq 1$ experts, finite logits, and strictly positive routing probabilities in a fixed order. All logarithms are natural.

A.1 PARTICIPATION COST AND THE LOCAL COMPARISON OBJECTIVE

Proposition 2 (Weighted participation cost)**.** *The contribution of block ℓ to Eq.* (10) *satisfies Eq.* (12)*. In particular, if $Q_i^{(\ell)} = 1/K$ for all experts, then $\mathcal{L}_{\mathrm{LB}}^{(\ell)} = N^{-1} \sum_v |\mathcal{S}_v^{(\ell,t)}|$.*

For the ordered prefixes defined in Section 4.3, the retained probability and participation cost are

$$s_m = \sum_{j=1}^{m} \pi_{\rho(j)}, \qquad c_m = K \sum_{j=1}^{m} Q_{\rho(j)}. \tag{20}$$

Substituting the definition of $T_i^{(\ell)}$ from Eq. (10) gives

$$\mathcal{L}_{\mathrm{LB}}^{(\ell)} = K \sum_{i=1}^{K} Q_i^{(\ell)} \left(\frac{1}{N} \sum_{v \in \mathcal{V}} \mathbb{I}\{i \in \mathcal{S}_v^{(\ell,t)}\} \right) = \frac{1}{N} \sum_{v \in \mathcal{V}} \sum_{i \in \mathcal{S}_v^{(\ell,t)}} K Q_i^{(\ell)}. \tag{21}$$

The identity follows directly from the regularizer's definition. Under uniform mean routing probabilities, each summand $KQ_i^{(\ell)}$ equals one, proving the second statement of Proposition 2. The total regularizer then sums the average participating counts across blocks. More generally, fixed routing probabilities give fixed $Q_i^{(\ell)}$, so adding expert i at a node increases the regularizer by $KQ_i^{(\ell)}/N$. The prefix costs therefore satisfy

$$c_{m+1} - c_m = K Q_{\rho(m+1)} > 0. \tag{22}$$

Here c_m is the cost before node averaging, and ρ is the fixed local routing order from Section 4.3.

To relate the local comparison to Eq. (11), take a labeled node $v \in \mathcal{V}_{\mathrm{train}}$ and let $n_{\mathrm{tr}} = |\mathcal{V}_{\mathrm{train}}|$. Vary its prediction and prefix at one block while holding all routing probabilities, other participation sets, and other nodes' predictions fixed. Under this controlled change, the routing-entropy term and all other contributions remain constant across candidate prefixes. Writing $\ell_m(a)$ for this node's classification loss, the total objective becomes

$$\mathcal{L}_m(a) = A(a) + \frac{1}{n_{\mathrm{tr}}} \ell_m(a) + \frac{\lambda_2}{N} c_m, \tag{23}$$

where $A(a)$ collects all terms independent of m. Multiplying by n_{tr} and removing this constant yields Eq. (14) with $\lambda = \lambda_2 n_{\mathrm{tr}}/N > 0$. This correspondence applies to the isolated supervised comparison. A full graph intervention can also change other nodes' predictions, and unlabeled nodes have no individual terms in the supervised training loss.

The identity compares objective values across participation choices. During the implemented backward pass, $T_i^{(\ell)}$ is treated as constant, giving

$$\nabla_\theta \mathcal{L}_{\mathrm{LB}}^{(\ell)} = K \sum_i T_i^{(\ell)} \nabla_\theta Q_i^{(\ell)}, \tag{24}$$

where θ is a model parameter. The coverage rule supplies the masks, and this gradient updates the mean routing probabilities for those masks. Equation (21) measures regularization cost. Actual runtime depends on the expert execution strategy described in Section 5.

A.2 CLASSIFICATION LOSS AND REFERENCE ENTROPY

The family $\boldsymbol{o}_m(a) = a\boldsymbol{e}_y + \boldsymbol{b}_m$ changes the same true-class logit component for every candidate. The reference vectors can be constructed by recording the output logits from each normalized expert-prefix

intervention in a fixed model, including the resulting graph and normalization operations, as $\boldsymbol{b}_m$. Adding $a\boldsymbol{e}_y$ to these recorded outputs leaves all prefix differences $\boldsymbol{o}_n(a) - \boldsymbol{o}_m(a) = \boldsymbol{b}_n - \boldsymbol{b}_m$ fixed. Varying a thus changes classifier evidence while preserving the recorded expert responses.

Define the prefix's relative true-class evidence and its cross-entropy loss by

$$\begin{aligned} t_m &= b_{m,y} - \log \sum_{c \neq y} \exp(b_{m,c}), \\ \ell_m(a) &= -\log \mathrm{Softmax}(\boldsymbol{o}_m(a))_y = \log\big(1 + \exp[-a - t_m]\big). \end{aligned} \tag{25}$$

Expanding the true-class softmax probability gives

$$P_{m,y}(a) = \frac{e^{a+b_{m,y}}}{e^{a+b_{m,y}} + \sum_{c \neq y} e^{b_{m,c}}} = \frac{1}{1 + \exp[-a - t_m]}, \tag{26}$$

where $t_m = b_{m,y} - \log \sum_{c \neq y} e^{b_{m,c}}$. Taking the negative logarithm proves Eq. (25). For $\ell(r) = \log(1 + e^{-r})$, the derivatives are

$$\ell'(r) = -\frac{1}{1 + e^r}, \qquad \ell''(r) = \frac{e^r}{(1 + e^r)^2} > 0. \tag{27}$$

Increasing t_m therefore reduces classification loss, and the gain from a fixed increase in t_m decreases as the shared evidence a grows. The gain relation follows from the curvature of cross-entropy.

For the common first-prefix reference, let $\boldsymbol{q}^{\mathrm{ref}}(a) = \mathrm{Softmax}(\boldsymbol{o}_1(a))$ with normalized entropy

$$\mathcal{U}_{\mathrm{ref}}(a) = -\frac{1}{\log C} \sum_{c=1}^{C} q_c^{\mathrm{ref}}(a) \log q_c^{\mathrm{ref}}(a). \tag{28}$$

To prove the entropy relation for the first-prefix reference, define

$$u(a) = 1 - q_y^{\mathrm{ref}}(a), \qquad r_c = \frac{e^{b_{1,c}}}{\sum_{d \neq y} e^{b_{1,d}}}, \quad c \neq y. \tag{29}$$

Here $u(a) \in (0, 1)$ is the total probability assigned to competing classes, and $\boldsymbol{r}$ is their conditional probability distribution. The vector $\boldsymbol{r}$ remains independent of a, and the competing probabilities satisfy $q_c^{\mathrm{ref}}(a) = u(a) r_c$ for $c \neq y$. Let $H(\boldsymbol{r}) = -\sum_{c \neq y} r_c \log r_c$ and $h_2(u) = -u \log u - (1 - u) \log(1 - u)$. Expanding the entropy gives

$$\mathcal{U}_{\mathrm{ref}}(a) = \frac{h_2(u(a)) + u(a) H(\boldsymbol{r})}{\log C}. \tag{30}$$

Differentiating with respect to u gives

$$\frac{d\mathcal{U}_{\mathrm{ref}}}{du} = \frac{\log((1 - u)/u) + H(\boldsymbol{r})}{\log C}. \tag{31}$$

Write $r_{\max} = \max_{c \neq y} r_c$. Where the true class has highest reference probability, $1 - u \geq u r_{\max}$ and hence $\log((1 - u)/u) \geq \log r_{\max}$. Also, $H(\boldsymbol{r}) \geq -\log r_{\max}$ because $-\log r_c \geq -\log r_{\max}$ for every competitor and the r_c sum to one. Equation (31) is therefore nonnegative on this range. Equation (26) also gives

$$u'(a) = -u(a)(1 - u(a)) < 0. \tag{32}$$

The reference range is the interval $[a_0, \infty)$, where $a_0 = \max_{c \neq y} b_{1,c} - b_{1,y}$. For every $a > a_0$, the true-class probability strictly exceeds the competing probabilities, making Eq. (31) strictly positive. Thus $\mathcal{U}_{\mathrm{ref}}(a)$ is strictly decreasing on $[a_0, \infty)$. In the uniform case, the derivative is zero only at the endpoint, so strict monotonicity holds between distinct values of a. The proof includes $C = 2$, where $\boldsymbol{r}$ has one entry and $H(\boldsymbol{r}) = 0$. This establishes the entropy ordering on the range in Theorem 1.

A.3 Proof of the Preferred-Participation Theorem

For every finite a, the finite candidate set has at least one minimizer of $J_a(m)$. We prove that the shortest minimizer $m^*(a)$ is nonincreasing in a for an arbitrary sequence $t_1, \ldots, t_K$.

For a useful extension $n > m$ with $t_n > t_m$, the loss curvature in Eq. (27) gives

$$\frac{d}{da}\big[\ell(a + t_m) - \ell(a + t_n)\big] = \ell'(a + t_m) - \ell'(a + t_n) < 0. \tag{33}$$

Take two prefixes $n > m$. If $t_n \le t_m$, the loss is decreasing in its argument, and $c_n > c_m$, so

$$J_a(n) - J_a(m) = \ell(a + t_n) - \ell(a + t_m) + \lambda(c_n - c_m) > 0 \tag{34}$$

for every a, ruling out the longer prefix as a minimizer. If $t_n > t_m$, define the objective difference $D_{n,m}(a) = J_a(n) - J_a(m)$. Equation (27) gives

$$D'_{n,m}(a) = \ell'(a + t_n) - \ell'(a + t_m) > 0. \tag{35}$$

The longer prefix becomes strictly less favorable relative to the shorter one as evidence grows.

Suppose for contradiction that $a_1 < a_2$ but $m_1 = m^*(a_1) < m^*(a_2) = m_2$. Since m_2 is optimal at a_2, the dominance argument implies $t_{m_2} > t_{m_1}$. Optimality at the two evidence values gives

$$D_{m_2,m_1}(a_1) \ge 0, \qquad D_{m_2,m_1}(a_2) \le 0. \tag{36}$$

These inequalities contradict the strict increase in Eq. (35), so $m^*(a)$ is nonincreasing. Ties are allowed, with the shortest minimizer specifying a single choice. Since the comparison evaluates complete prefixes, a temporarily harmful extension can still form part of a later useful prefix. Adjacent t_m values may increase or decrease.

On the interval $[a_0, \infty)$, Appendix A.2 establishes that $\mathcal{U}_{\mathrm{ref}}(a)$ is strictly decreasing. Thus $m^*(a)$ is nondecreasing as a function of this entropy. All candidates share the same positive routing distribution, so s_m strictly increases with m and $s_{m^*(a)}$ is also nondecreasing in entropy. For $K = 1$, both conclusions hold trivially. Weak monotonicity allows the preferred count to remain unchanged over an interval of entropy values, completing the proof of Theorem 1.

A.4 Coverage Interpretation and Connection to Cached Predictions

For $p \in (0, 1]$, positive routing probabilities and $s_K = 1$ ensure that $k(p)$ in Eq. (15) exists. Set $s_0 = 0$. The prefix S_m is selected exactly when $p \in (s_{m-1}, s_m]$. Every candidate in the controlled comparison therefore corresponds to a coverage threshold. In particular, $p = s_{m^*(a)}$ realizes the preferred prefix within that comparison.

For any subset S of j experts, descending probability order gives $\sum_{i \in S} \pi_i \le s_j$. If $j < k(p)$, then $s_j < p$, so no such subset reaches the target. The prefix $S_{k(p)}$ reaches it with minimum cardinality, satisfying the criterion in Eq. (18). With nonuniform Q_i, cardinality and weighted participation cost are distinct criteria. Theorem 1 compares the weighted objective among the given prefixes. Under uniform Q_i, these costs equal prefix lengths.

The theorem measures reference entropy using a common first prefix with fixed expert corrections. The method uses the preceding cache-refresh prediction, available for every node before the next routing pass. Within an epoch, the sigmoid in Eq. (17) increases the requested coverage with this signal. The graph-wide mean sets the reference level, and γ controls sensitivity. Each block's routing distribution converts the shared coverage target into a count, which can differ across blocks. This implements the allocation direction studied in the controlled analysis, and the selected masks enter the parameter updates in Algorithm 1.

A diagnostic protocol could assess this connection by freezing a checkpoint and its existing entropy cache, enumerating the normalized prefixes at a node–block pair, and recomputing the predictions. Holding the cache fixed would test whether the signal available to the router identifies worthwhile extensions. For a full graph intervention, task losses should include the affected evaluated nodes because message passing and normalization can couple their predictions. The isolated-node correspondence in Eq. (23) applies when other predictions and routing probabilities remain fixed. Separate comparisons for reference-correct and reference-incorrect groups would identify the theorem's evidence regime. Matching average participating counts across adaptive, fixed-count, and shuffled-entropy policies would then assess the allocation signal's value in the full model.

B TRAINING ALGORITHM

Optimization Procedure. We optimize D^{2}MoE end to end using the procedure in Algorithm 1. Each epoch first obtains the coverage targets from the existing entropy cache, then performs the routed forward pass and parameter update using Eq. (11). The discrete selection indicators are held fixed during backpropagation, as specified in Section 4.2. A cache-refresh pass then uses the updated parameters to compute the predictive entropy for the next epoch. Because the node set remains fixed in the transductive setting, we maintain one cache entry per node and update it after each epoch.

Algorithm 1 Training Procedure of D^{2}MoE

Input: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{X}, \boldsymbol{Y})$, maximum epochs T, and hyperparameters $\gamma, \lambda_1, \lambda_2$.
Output: Trained model parameters $\boldsymbol{\Theta}$.
Initialization: Initialize $\boldsymbol{\Theta}$ and allocate one predictive-entropy cache entry for each node.
for epoch $t = 1$ to T **do**
  // Phase 1: Uncertainty-to-Coverage Mapping
  if $t = 1$ **then**
    Set $p_v^{(1)} \leftarrow 1$ and retain all K experts for every $v \in \mathcal{V}$.
  else
    Set $\bar{\mathcal{U}}^{(t-1)} \leftarrow |\mathcal{V}|^{-1} \sum_v \mathcal{U}_v^{(t-1)}$.
    for each node $v \in \mathcal{V}$ **do**
      Compute $p_v^{(t)}$ from $\mathcal{U}_v^{(t-1)}$ using Eq. (17).
    end for
  end if
  // Phase 2: Forward Pass with Dynamic Top-p Routing
  Compute initial representations $\boldsymbol{H}^{(0)}$ using Eq. (2).
  for layer $\ell = 1$ to L **do**
    Compute expert outputs $\boldsymbol{Z}_i^{(\ell)}$ for $i = 1, \ldots, K$.
    for each node $v \in \mathcal{V}$ **do**
      Compute routing probabilities $\boldsymbol{\pi}_v^{(\ell)}$ using Eq. (4).
      Select the minimum-cardinality expert set $\mathcal{S}_v^{(\ell,t)}$ using dynamic top-p routing in Eq. (18).
      Compute the normalized weights $\bar{\pi}_{v,i}^{(\ell,t)}$ using Eq. (19).
      Aggregate the selected expert outputs to obtain $\boldsymbol{u}_v^{(\ell)}$ using Eq. (5).
    end for
    Update $\boldsymbol{H}^{(\ell)}$ using Eq. (6).
  end for
  Obtain training predictions $\boldsymbol{P}_{\mathrm{tr}}^{(t)}$ using Eq. (7).
  // Phase 3: Parameter Optimization
  Compute $\mathcal{L}_{\mathrm{task}}$ using Eq. (8).
  Compute $\mathcal{L}_{\mathrm{RE}}$ and $\mathcal{L}_{\mathrm{LB}}$ using Eqs. (9) and (10).
  Form $\mathcal{L}$ using Eq. (11) and update $\boldsymbol{\Theta}$.
  // Phase 4: Predictive-Entropy Cache Update
  Using the updated parameters, perform a cache-refresh forward pass under $p_v^{(t)}$.
  for each node $v \in \mathcal{V}$ **do**
    Compute and cache $\mathcal{U}_v^{(t)}$ from the resulting prediction probabilities using Eq. (16).
  end for
end for
return optimized parameters $\boldsymbol{\Theta}$.

C Experimental Setup Details

Datasets. We evaluate D^{2}MoE on 13 node-classification benchmarks. The homophilous datasets are Computers, Photo (McAuley et al., 2015), Coauthor CS, Coauthor Physics (Shchur et al., 2018), Wiki-CS, Facebook (Rozemberczki et al., 2021), and Ogbn-arxiv (Hu et al., 2020). The heterophilous datasets are Actor (Tang et al., 2009), Chameleon-filtered, Squirrel-filtered, Roman-empire (Platonov et al., 2023), Tolokers, and Penn94 (Lim et al., 2021). We use a 48%/32%/20% training/validation/test split, except on Ogbn-arxiv, where we use the official OGB (Hu et al., 2020) split.

Baselines. We compare D^{2}MoE with 20 baselines from four model families. The standard neural models are MLP, GCN (Kipf & Welling, 2017), GraphSAGE (Hamilton et al., 2017), and GAT (Veličković et al., 2018). The heterophily-oriented GNNs are H2GCN (Zhu et al., 2020), GPRGNN (Chien et al., 2021), FAGCN (Bo et al., 2021), ACMGCN (Luan et al., 2022), and FSGNN (Maurya et al., 2022). The graph transformers are Vanilla GT, ANS-GT (Zhang et al., 2022), NAGphormer (Chen et al., 2023), SGFormer (Wu et al., 2023b), Exphormer (Shirzad et al., 2023), and DIFFormer (Wu et al., 2023a). The Graph MoE baselines are GMoE (Wang et al., 2023), DAMoE (Yao et al., 2025), NodeMoE (Han et al., 2024), Mowst (Zeng et al., 2024), and Moscat (Deng et al., 2025).

Implementation Details. We report the mean and standard deviation over ten runs with different random seeds. The common search space fixes the hidden dimension at $h = 64$ and searches learning rates in $\{0.005, 0.01, 0.05, 0.1\}$ and dropout rates in $\{0.1, 0.3, 0.5, 0.7, 0.9\}$. We train with AdamW for at most 500 epochs, stopping after 100 epochs without validation improvement. For D^{2}MoE, we use $\lambda_1 = 10^{-4}$, $\lambda_2 \in \{10^{-3}, 10^{-1}\}$, $\gamma \in \{1, 5, 10\}$, $K \in \{4, 6\}$, and $L \in \{1, 2\}$.

Efficiency Evaluation. Time to early stopping in Figure 8 combines epoch cost and stopping behavior. The measured D^{2}MoE pipeline computes all expert outputs before applying aggregation weights.

D Additional Experimental Results

Figures 9, 10, 11, and 12 present predictive-entropy, ablation, embedding, and routing-probability results on the 12 datasets other than Ogbn-arxiv. The Ogbn-arxiv results are included in the six-dataset comparisons in Figures 3, 4, 6, and 7.

D.1 Predictive Performance Across Entropy Levels

To examine how performance varies with predictive uncertainty, we evaluate all methods on common subsets defined by a fixed, independently trained proxy model, following the observation study. We sort test nodes by proxy entropy and divide them into ten equal-sized deciles, from the lowest-entropy $[0\%, 10\%]$ group to the highest-entropy $[90\%, 100\%]$ group. Figure 9 reports the accuracy of D^{2}MoE, GCN, GMoE, DAMoE, and Mowst within each decile on 12 datasets.

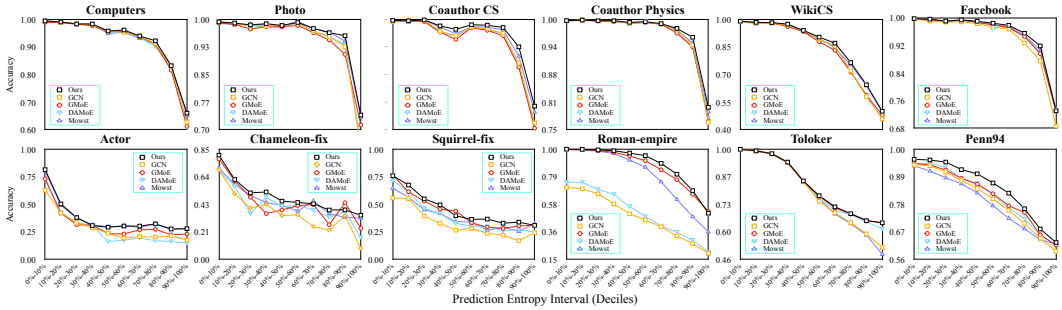

Figure 9: Classification accuracy across predictive-entropy deciles on 12 datasets. A fixed, independently trained proxy model defines a common partition for all methods. Deciles run from lower to higher entropy along the x-axis, and the y-axis gives accuracy within each decile.

Higher-entropy groups generally have lower accuracy, although the trend varies across methods and datasets. This association supports using proxy entropy to characterize differences in predictive uncertainty. D^{2}MoE remains competitive in low-entropy groups and often shows larger gains in the middle- and high-entropy deciles. We next examine the contributions of the routing components.

D.2 Routing and Regularization Ablations

We compare D^{2}MoE with five variants to assess its routing rule, uncertainty signal, and regularizers. Static top-k fixes the selected expert count with $k \in \{1, 2, 3, 4, 5, 6\}$, while fixed top-$p$ uses one graph-wide threshold from $p \in \{0.3, 0.5, 0.7\}$. Random top-p replaces cached entropy with a randomly generated node-level signal. The remaining variants remove routing-entropy regularization ($\mathcal{L}_{\mathrm{RE}}$) or load balancing ($\mathcal{L}_{\mathrm{LB}}$). Figure 10 reports their test scores on 12 datasets. The metric is accuracy except on Tolokers, where it is ROC-AUC.

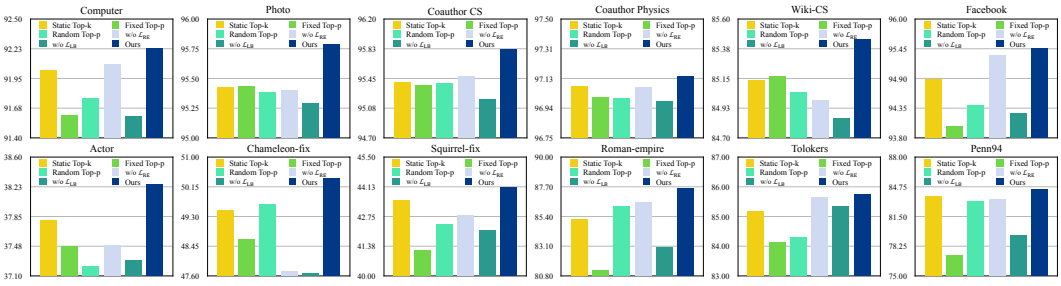

Figure 10: Ablation of the uncertainty signal, routing rule, and router regularizers in D^{2}MoE. Static top-k fixes the expert count, fixed top-p shares a threshold across nodes, and random top-p replaces cached entropy with a random node-level signal.

Adaptive Threshold versus Fixed Routing. Static top-k constrains every node to the same expert count. Fixed top-p follows the fixed coverage threshold of Huang et al. (2024), sharing one coverage target across the graph while allowing the count to depend on the router. On the heterophilous datasets in Figure 10, both variants have lower mean test scores than D^{2}MoE, with a larger decrease under fixed top-p. The comparison supports adapting both the participating count and the coverage target under the evaluated settings.

Contribution of Cached Uncertainty. Replacing cached-entropy guidance with random top-p lowers the mean test score on the datasets shown in Figure 10. Although both variants vary the selected expert count, the entropy-guided variant performs better in this comparison, supporting the use of predictive information to set node-level coverage targets.

Contribution of Router Regularization. Removing either $\mathcal{L}_{\mathrm{LB}}$ or $\mathcal{L}_{\mathrm{RE}}$ lowers the mean test score on the 12 datasets in Figure 10, with a larger decrease when load balancing is removed. The two terms shape routing at different scales. Routing-entropy regularization encourages each node's probabilities to concentrate, while load balancing discourages a small subset of experts from dominating across the graph. Both contribute to the predictive performance of D^{2}MoE.

D.3 Expert Counts in the Embedding Space

Figure 11 provides a qualitative view of expert participation in the final-layer representation space.

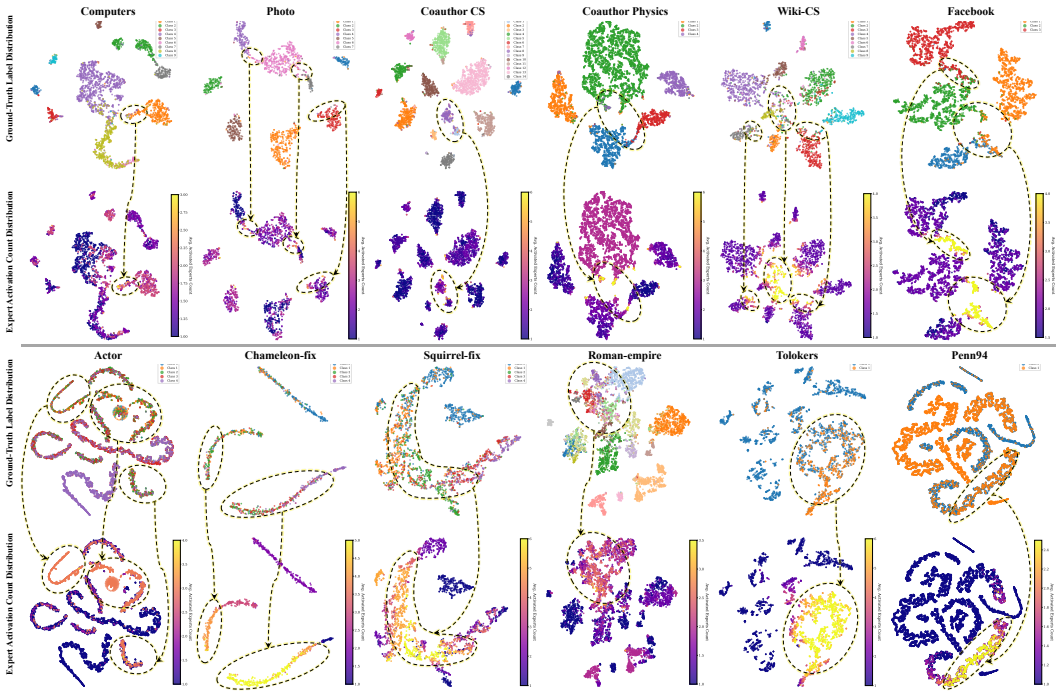

Figure 11: t-SNE of final-layer node representations on 12 datasets. For each dataset, the same embeddings are colored by ground-truth class (upper) and mean selected expert count (lower).

On several homophilous datasets, nodes selecting fewer experts often lie in compact same-class regions, while nodes selecting more experts appear more often in visually mixed regions. On heterophilous datasets, the class and expert-count patterns are less spatially distinct. These visual patterns complement the mean expert counts across entropy intervals.

D.4 Mean Routing Probabilities Across Entropy Levels

Figure 12 shows mean expert routing probabilities across predictive-entropy quartiles on 12 datasets.

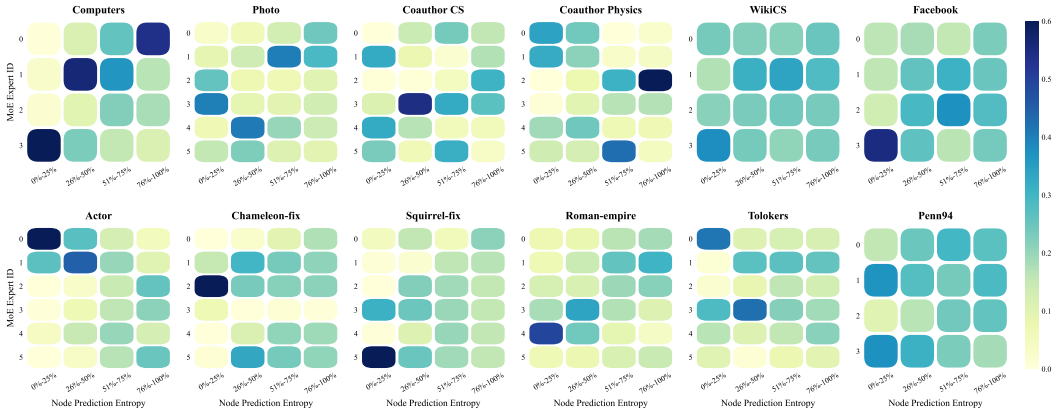

Figure 12: Mean routing probabilities across predictive-entropy quartiles on 12 datasets. Columns correspond to entropy quartiles, rows to experts, and color to mean routing probability.

Across the datasets, the mean routing distributions vary in both their dominant experts and concentration. Higher-entropy groups are more balanced in several cases, whereas Computers and Coauthor Physics retain a pronounced expert preference in the highest-entropy quartile. These group means summarize how expert preferences vary across node populations. At the node level, the prefix needed to reach a coverage target depends on that node's cumulative routing probabilities, linking the router's preferences to expert participation in Figure 5.

D.5 Comparison of Uncertainty Signals

The random-signal ablation in Appendix D.2 examines whether coverage benefits from predictive guidance. We further compare predictive entropy with Gini impurity and neighborhood confusion (NC) (Zhou et al., 2025) on Actor and Computers. Within each dataset, the backbone, total number of experts, hidden dimension, data split, and training procedure are held fixed across the three variants. Table 2 reports test accuracy and the mean number of selected experts, averaged over test nodes.

Table 2: Comparison of uncertainty signals. Accuracy (%) is reported as mean and standard deviation. Mean experts denotes the average selected expert count.

	Actor		Computers	
Signal	Accuracy	Mean experts	Accuracy	Mean experts
Gini	36.76 ± 0.79	1.5	92.11 ± 0.44	1.7
NC	37.62 ± 1.03	2.1	92.00 ± 0.49	1.6
Entropy	$\mathbf{38.26 \pm 1.06}$	2.3	$\mathbf{92.23 \pm 0.33}$	1.8

Entropy achieves the highest mean accuracy on both graphs. On Actor, its gains over NC and Gini accompany greater expert participation, whereas the three signals yield similar accuracy and counts on Computers. The choice of signal thus affects both predictive performance and mixture size. These results complement the random-signal ablation by showing how alternative node-level signals translate into different routing outcomes.

D.6 Sensitivity to Routing Regularization

The regularizer ablations in Appendix D.2 remove one term at a time. We further examine their joint effect on Computers by varying $\lambda_1 \in \{0, 10^{-4}, 10^{-2}, 1\}$ and $\lambda_2 \in \{0, 10^{-3}, 10^{-1}, 1\}$. Figure 13 reports accuracy, mean selected expert count, and routing entropy for each pair.

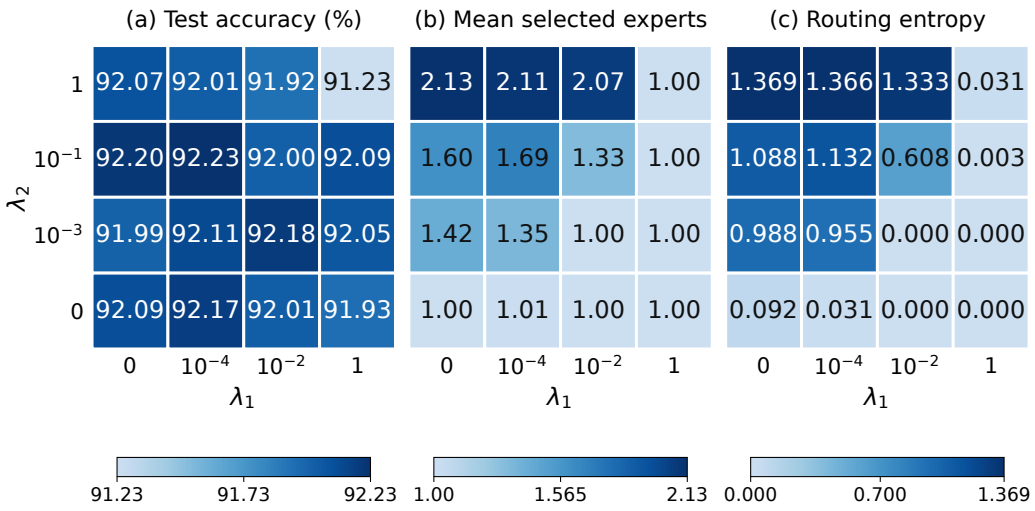

Figure 13: Sensitivity to the routing-entropy weight λ_1 and load-balancing weight λ_2 on Computers. Cells show the reported values. Each panel uses its own color scale.

For $\lambda_1 \leq 10^{-2}$, accuracy ranges from 91.92% to 92.23% across the tested combinations, while expert counts and routing entropy vary more substantially. With $\lambda_1 = 1$, the mean selected count is 1 for every tested λ_2. With $\lambda_1 = 10^{-4}$, increasing λ_2 from 0 to 10^{-1} raises the mean count from 1.01 to 1.69 and routing entropy from 0.031 to 1.132. The joint measurements reveal changes in expert participation that small accuracy differences alone would obscure.

D.7 Detailed Training Efficiency

We expand the resource comparison in Figure 8 by reporting parameter counts and actual training epochs alongside total training time and peak GPU memory in Table 3. The detailed benchmark also includes GNNMoE (Chen et al., 2025). Measurements were obtained on an NVIDIA A100 GPU with 40 GB of memory. The time-per-epoch column is calculated by dividing the reported total time by the training epochs.

Table 3: Detailed training cost on two large graphs. Time per epoch is the quotient of total time and epochs. The epoch limit is 500, and OOM denotes an out-of-memory run.

Method	Parameters	Epochs	Time (s)	Time/epoch (s)	Peak memory (MB)
Ogbn-arxiv					
GMoE	46,696	446	157.36	0.353	9,739.95
Mowst	22,353	390	22.25	0.057	1,874.93
DAMoE	65,956	345	114.47	0.332	3,779.95
NodeMoE	59,368	500	92.68	0.185	4,098.05
GNNMoE	104,252	420	95.53	0.227	3,987.97
D^2MoE	78,580	360	84.14	0.234	3,527.09
Penn94					
GMoE	344,195	223	83.53	0.375	10,774.71
Mowst	617,221	174	51.73	0.297	2,341.68
DAMoE	674,600	385	147.40	0.383	3,461.01
NodeMoE	1,554,078	OOM	OOM	OOM	OOM
GNNMoE	401,686	500	60.45	0.121	2,467.11
D^2MoE	658,826	205	49.45	0.241	2,902.03

Relative to GMoE, D^2MoE uses fewer training epochs and has a lower time-per-epoch quotient on both graphs. The comparison with GNNMoE illustrates a different source of the total-time advantage. On Penn94, D^2MoE takes 0.241 seconds per epoch compared with 0.121 seconds for GNNMoE, but runs for 205 epochs rather than 500, yielding a lower total time of 49.45 seconds versus 60.45 seconds. Total training time reflects both epoch cost and stopping behavior.

Memory use also differs from parameter count. Although D^2MoE has more parameters than GMoE on both graphs, its peak memory is lower. Relative to GNNMoE, peak memory is lower on Ogbn-arxiv and higher on Penn94. Together with the accuracy comparison in Figure 8, these measurements characterize the performance–resource tradeoff of the implemented pipeline, which computes all expert outputs before applying the selected aggregation weights.