Graph-Autonomous Mixture of Experts with Hierarchical Multi-Agent PPO for Dynamic Knowledge Graphs

Aygul F. Shaykhulova

***Abstract*—This paper presents a novel Graph-Autonomous Mixture of Experts (MoE) architecture that combines differentiable knowledge graphs with hierarchical multi-agent Proximal Policy Optimization (PPO). The system autonomously builds a hierarchical knowledge graph from unstructured text, initializes expert agents on graph nodes, and enables continuous adaptation through a two-level reinforcement learning framework. At the meta-level, a controller manages expert creation, deletion, and knowledge transfer. At the local level, each expert agent independently decides when to train, specialize, merge, or transfer knowledge. All rewards are derived exclusively from quality metrics—perplexity on validation data, router contribution, and expert diversity—eliminating hand-crafted reward shaping. Experiments on technical document processing demonstrate that the system achieves a 46% loss reduction over 220 training steps, with experts developing distinct specializations (Silhouette score 0.43) and a stable population dynamic (CREATE/DELETE ratio 8.0). The learned transfer matrix reveals asymmetric knowledge flows, with active donors and recipients emerging organically. The proposed approach demonstrates the possibility of constructing a self-organizing MoE system with minimal human intervention.**

***Keywords*—Mixture of Experts, Multi-Agent Reinforcement Learning, Proximal Policy Optimization, Knowledge Graphs, Differentiable Graph Neural Networks, Autonomous Systems.**

I. Introduction

Mixture of Experts (MoE) models have become a cornerstone of modern machine learning, enabling conditional computation and specialized processing of diverse inputs [1], [2]. Traditional MoE architectures, however, suffer from a fundamental limitation: their structure is static. The number of experts, their specializations, and the routing mechanism are determined before training and remain fixed throughout the model's lifetime. This rigidity contradicts the dynamic nature of real-world data, where new topics emerge, existing concepts evolve, and the optimal allocation of computational resources shifts over time.

Recent advances in neural architecture search [3] and adaptive computation [4] have begun addressing these limitations, but existing approaches typically operate within constrained search spaces or require extensive retraining. Reinforcement learning offers a promising alternative: agents could learn to modify the model architecture itself in response to performance signals. However, applying RL to architectural decisions introduces significant challenges—the action space is combinatorial, credit assignment spans multiple timescales, and rewards must reflect genuine quality improvements rather than proxy metrics.

This paper introduces a Graph-Autonomous Mixture of Experts system that addresses these challenges through three key innovations:

First, we construct a hierarchical knowledge graph from unstructured text using unsupervised clustering. This graph captures the latent topic structure of the document corpus, providing a natural organizational principle for expert specialization. Graph nodes represent topics at multiple levels of abstraction, while edges encode hierarchical and similarity relationships.

Second, we transform this static graph into a differentiable knowledge graph where node embeddings and edge weights become learnable parameters. This differentiable structure enables gradient-based optimization of the routing mechanism and supports dynamic node addition and removal—the graph itself evolves with the model.

Third, we implement a hierarchical multi-agent reinforcement learning framework using Proximal Policy Optimization (PPO) [5]. At the meta-level, a controller agent observes the global system state and decides when to create new experts, delete underperforming ones, or reallocate knowledge. At the local level, each expert agent independently chooses actions from its own repertoire: training on specialized data, creating specialized child experts, merging with parent nodes, or transferring knowledge to related experts. Crucially, all agents share a common reward signal derived exclusively from quality metrics—perplexity on held-out data, router contribution, and expert diversity. No hand-crafted rewards for specific actions are provided; the system must discover effective strategies purely through the lens of quality improvement.

The resulting system exhibits remarkable emergent behaviors. Over 220 training steps on a technical document corpus, the population of experts grows from 5 to 15, with 16 total creations and 2 deletions. Expert losses decrease from 10.79 to 5.84, a 46% improvement. The learned transfer matrix reveals asymmetric knowledge flows: some experts become net donors (transfer_out > transfer_in), while others emerge as net recipients. t-SNE visualization of expert embeddings shows clear clustering by specialization, with a Silhouette score of 0.43 indicating well-separated

Manuscript received March 10, 2026.

A. F. Shaykhulova is with the Ufa University of Science and Technologies, Ufa, RB 450076 Russia (corresponding author to provide phone: +7(917)4653433; e-mail: shaihulova@inbox.ru)

topic expertise. The meta-controller learns to favor REALLOCATE actions (47.4% of decisions) over CREATE (26.3%) and DELETE (5.3%), suggesting that knowledge transfer provides more reliable quality gains than architectural modifications.

This work makes the following contributions:

1. A fully autonomous MoE architecture that constructs its own topic hierarchy, initializes experts, and continuously adapts its structure through reinforcement learning.
2. A differentiable knowledge graph that enables gradient-based routing optimization while supporting dynamic node addition and removal.
3. A hierarchical multi-agent PPO framework with shared critic that coordinates meta-level architectural decisions and local-level expert behaviors.
4. Empirical demonstration that quality-based rewards alone suffice to produce specialized, self-organizing expert populations.
5. Comprehensive analysis tools for visualizing expert specializations, transfer dynamics, and population stability.

The remainder of this paper is organized as follows. Section II reviews related work in MoE architectures, graph neural networks, and multi-agent RL. Section III details the system architecture, including knowledge graph construction, differentiable graph implementation, and hierarchical PPO formulation. Section IV describes the experimental setup and results. Section V provides discussion and analysis of emergent behaviors. Section VI concludes with implications and future directions.

II. Related Works

A. Mixture of Experts Architectures

The Mixture of Experts framework originated with Jacobs et al. [1], who proposed combining multiple neural networks with a gating network that learns to weight expert contributions. This conditional computation paradigm enables models to scale to massive sizes while maintaining computational efficiency, as demonstrated in the Sparsely-Gated MoE layer [2] used in large language models [6]. Subsequent work has explored hierarchical MoE structures [7], where experts themselves contain internal MoE architectures, enabling multi-level specialization.

Despite these advances, traditional MoE architectures remain static. The number of experts is fixed before training, and while the gating network learns to route inputs, the experts themselves do not adapt their specializations in response to changing data distributions. Recent work on dynamic MoE [8] introduces the ability to add or remove experts during training but relies on heuristic criteria rather than learned policies.

Our work extends this line of research by treating expert lifecycle management as a reinforcement learning problem. Rather than hand-crafting rules for when to create or delete experts, we allow a meta-controller to discover optimal policies through interaction with the system.

B. Graph Neural Networks and Differentiable Graphs

Graph Neural Networks (GNNs) have emerged as a powerful framework for learning on structured data [9]. By propagating information along graph edges, GNNs can capture relational dependencies and produce node representations that reflect both local features and global context. Applications range from node classification [10] to link prediction [11] and graph generation [12].

The concept of differentiable graphs extends this paradigm by making the graph structure itself learnable. Kipf et al. [13] introduced variational graph auto-encoders that learn to generate graph structures from node features. Graph neural architecture search [14] optimizes both the graph structure and the parameters of GNN layers. In the context of MoE systems, a differentiable knowledge graph can learn to route information between experts based on task similarity.

Our differentiable knowledge graph draws inspiration from these works but serves a different purpose: rather than predicting graph structure from data, we initialize our graph from unsupervised clustering of the document corpus, then allow the graph to evolve through gradient-based updates as the system learns. This hybrid approach combines the interpretability of a human-readable topic hierarchy with the flexibility of learnable parameters.

C. Multi-Agent Reinforcement Learning

Multi-agent reinforcement learning addresses settings where multiple agents interact within a shared environment, each pursuing its own objectives while influencing the experiences of others [15]. Central challenges include non-stationarity (each agent's policy changes as others learn), credit assignment (determining which agents contributed to observed outcomes), and coordination (achieving globally desirable outcomes despite local optimization).

Proximal Policy Optimization (PPO) [5] has emerged as a popular algorithm for single-agent RL due to its stability and sample efficiency. Extensions to multi-agent settings include MAPPO [16], which adapts PPO for cooperative multi-agent tasks by sharing parameters and using a centralized value function. The shared critic architecture, where a single network estimates the value of joint states while each agent maintains its own policy, has proven effective in domains requiring coordination.

Our hierarchical PPO framework adapts these ideas to the MoE setting. The meta-controller operates at the system level, making decisions that affect all experts. Local experts operate independently, each with its own policy and critic, but share a common reward signal derived from global quality metrics. This hybrid architecture balances autonomy (experts make individual decisions) with coordination (all agents optimize toward common quality objectives).

D. Autonomous Systems and Meta-Learning

The broader field of autonomous systems seeks to create agents that can adapt to changing environments without human intervention. Meta-learning [17] addresses this by learning learning algorithms—models that can quickly adapt to new tasks using few examples. In the context of neural architecture, meta-learning has been applied to learn optimization algorithms [18] and to discover network

architectures that generalize across tasks [19].

Our work can be viewed as a form of meta-learning where the meta-controller learns to modify the model architecture itself. Rather than learning parameters that enable fast adaptation, the meta-controller learns when architectural changes are beneficial. This represents a step toward truly autonomous AI systems that can redesign themselves in response to performance feedback.

III. System Architecture

A. Overview

The Graph-Autonomous MoE system comprises five interconnected components:

1. Knowledge Graph Builder:

Constructs a hierarchical topic graph from unstructured text using unsupervised clustering.

2. Differentiable Knowledge Graph:

Maintains learnable node embeddings and edge weights, enabling gradient-based routing optimization.

3. Expert Agents:

Each graph node hosts an expert agent based on TinyBERT [20], with local PPO policies for autonomous decision-making.

4. Hierarchical PPO Framework:

Implements meta-level and local-level policies with shared critic for coordination.

5. Reward System:

Computes quality-based rewards from perplexity, router contribution, and expert diversity.

Fig. 1 illustrates the overall architecture and information flow.

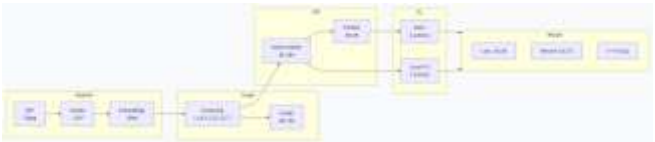

Figure 1. System Architecture Overview

B. Knowledge Graph Construction

The knowledge graph builder transforms unstructured text into a hierarchical topic structure without manual annotation. Given a corpus of documents, we first segment text into overlapping chunks (256 tokens with 64-token overlap) using a sliding window approach. Each chunk retains metadata including source page and position within the document. (figure 2).

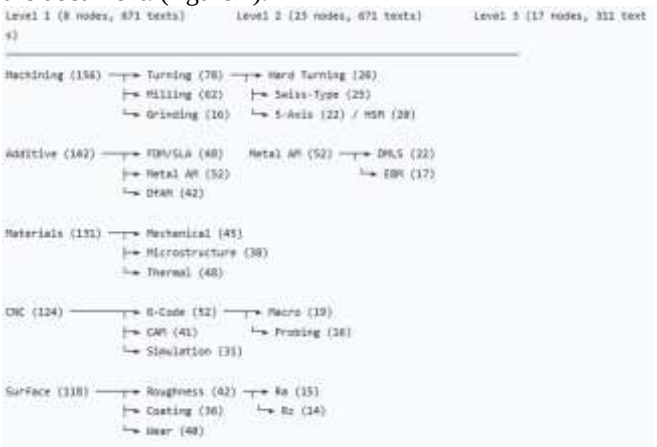

Figure 2. Knowledge Graph Construction

We encode all chunks using Sentence-BERT [21] with the all-MiniLM-L6-v2 model, producing 384-dimensional embeddings. Agglomerative clustering with cosine distance and average linkage then groups chunks into topics. The clustering proceeds hierarchically:

Level 1 (Broad Topics): We determine the initial number of clusters as

$$k = \max(2, \min(15, \lfloor N/10 \rfloor)), \quad (1)$$

where N is the number of chunks. Each resulting cluster becomes a level-1 node representing a broad topic area.

*Level 2 (Subtopics)***:** For each level-1 node containing at least 20 chunks, we perform a second clustering pass with

$$k = \min(5, \lfloor |C|/10 \rfloor) \quad (2)$$

to identify subtopics. These become level-2 nodes, with directed edges from parent to child.

Level 3 (Specific Topics): For level-2 nodes containing at least 20 chunks, we perform a third clustering pass with

$$k = \min(3, \lfloor |C|/10 \rfloor) \quad (3)$$

to capture highly specific subthemes.

After constructing the hierarchical tree, we add horizontal edges between nodes at the same level when their centroid similarity exceeds a threshold $\tau = 0.7$. These similarity edges enable knowledge transfer between related topics.

The resulting graph $G = (V, E)$ contains $|V|$ topic nodes and $|E|$ hierarchical and similarity edges. Each node v_i stores: (1) the text chunks assigned to it, (2) the centroid embedding of those chunks, (3) metadata including cluster size and hierarchy level, and (4) references to parent and child nodes.

C. Differentiable Knowledge Graph

The static knowledge graph becomes learnable through the *DifferentiableKnowledgeGraph* module. This component maintains three sets of trainable parameters:

Node Embeddings:

$$\mathbf{E} \in \mathbb{R}^{|V| \times d} \quad (4)$$

where $d = 384$ is the embedding dimension. Initialized from the centroid embeddings computed during graph construction.

Transfer Matrix:

$$\mathbf{T} \in \mathbb{R}^{|V| \times |V|} \quad (5)$$

representing the strength of knowledge transfer between experts. During forward passes, we apply softmax normalization:

$$\hat{\mathbf{T}} = \text{softmax}(\mathbf{T}/\tau) \quad (6)$$

where τ is a learnable temperature parameter.

Hierarchy Weights:

$$\mathbf{H} \in \mathbb{R}^{|V| \times |V|} \quad (7)$$

capturing the influence of parent-child relationships. Also normalized via softmax.

Given a query embedding $\mathbf{q} \in \mathbb{R}^d$, the differentiable graph computes:

$$\mathbf{r} = \text{softmax}(\mathbf{E}\mathbf{q}/\tau) \quad (8)$$

$$\mathbf{w} = \mathbf{r}^\top \hat{\mathbf{H}} \quad (9)$$

where $\mathbf{r}$ represents raw relevance of each node to the query, and $\mathbf{w}$ provides hierarchy-aware node weights that incorporate parent-child influence propagation.

The transfer matrix $\widehat{\mathbf{T}}$ is used during expert interactions: when expert i transfers knowledge to expert j, we update $\mathbf{T}_{i,j}$ accordingly, and the magnitude of transfer affects future routing decisions.

Crucially, the differentiable graph supports dynamic node addition and removal. When the meta-controller decides to create a new expert, we:

1. Append a new row and column to $\mathbf{E}$, $\mathbf{T}$, and $\mathbf{H}$, initialized from the parent node's parameters plus small noise.
2. Update the node-to-index mappings.
3. Establish initial edges to parent and similar nodes.

When deleting an expert, we remove the corresponding rows and columns and reindex remaining nodes. All operations are differentiable where possible; removal is non-differentiable but occurs infrequently.

D. Hierarchical Multi-Agent PPO

The reinforcement learning framework operates at two levels, each with distinct state spaces, action spaces, and policies.

- *D.1 Meta-Controller*

The meta-controller observes the global system state and makes decisions affecting the entire expert population. The global state is constructed by concatenating the state vectors of all experts (up to a maximum of $N_{\max} = 15$), with padding for inactive slots. Each expert's state vector (detailed in Section III-D.2) captures its current performance, resource utilization, and relational context.

The meta-controller's action space comprises four discrete actions:

- CREATE (0): Create a new expert by splitting an existing expert. The controller selects the parent expert based on a combination of size and performance, then initializes a child with a subset of the parent's texts and slightly perturbed embeddings.
- DELETE (1): Remove an underperforming expert. The controller identifies candidates using a scoring function that combines high loss, low transfer activity, and age. Knowledge from the deleted expert is transferred to its parent and children before removal.
- REALLOCATE (2): Transfer knowledge between experts. The controller identifies a pair of experts with high embedding similarity but disparate performance, then transfers weights from the better-performing to the worse-performing expert.
- NOOP (3): Take no action, incurring a small negative reward to discourage inaction when beneficial changes are possible.

The meta-controller policy $\pi_{\text{meta}}(a \mid s_{\text{global}})$ is implemented as a three-layer MLP with hidden dimension 512, producing logits over the four actions. During training, actions are sampled from a categorical distribution; during evaluation, the action with maximum probability is selected.

- *D.2 Local Expert Policies*

Each expert agent maintains its own policy network $\pi^{i}_{\text{local}}(a \mid s_i)$ operating on its local state vector. The local state vector (dimension 512) concatenates nine feature groups:

1. Topic Embedding (384 dims): The expert's current node embedding from the differentiable graph.
2. Confidence Metrics (8 dims): Recent confidence values (router weights assigned to this expert), including last value, mean over last 100 steps, standard deviation, and trends.
3. Loss Dynamics (8 dims): Running loss statistics, including current loss, mean over recent steps, normalized deviation from running mean, and ratio to best loss.
4. Perplexity Metrics (8 dims): Recent perplexity values and derived features including inverse perplexity and log-transformed values.
5. Selection Frequency (4 dims): How often this expert has been selected by the router, including total count, recent rate, and variance.
6. Transfer Statistics (8 dims): Cumulative transfer in and transfer out, rates normalized by age, and derived features like balance (transfer in − transfer out).
7. Age and Resources (8 dims): Normalized age, log-transformed age, number of texts, number of children, and binary indicators for parent/sibling relationships.
8. Matrix Transfer Features (4 dims): Statistics from the expert's row in the transfer matrix, including mean, max, and standard deviation of outgoing transfer strengths.
9. Padding (to 512 dims): Zero-padding to reach the target dimension.

The local action space comprises seven discrete actions:

- TRAIN (0): Train the TinyBERT model on the expert's assigned texts for one step.
- SPECIALIZE (1): Create a specialized child expert focused on a subset of the current expert's texts.
- MERGE (2): Merge with parent expert, combining parameters and transferring all texts upward.
- TRANSFER UP (3): Transfer knowledge to parent expert.
- TRANSFER DOWN (4): Transfer knowledge to child experts.
- SLEEP (5): Enter a low-power state, reducing resource consumption but pausing learning.
- DESTRUCT (6): Initiate self-deletion, transferring knowledge to related experts before removal.

Local policies are also three-layer MLPs with hidden dimension 256, producing logits over seven actions. Each expert maintains its own optimizer and learning rate scheduler.

- *D.3 Shared Critic and PPO Updates*

Coordination between agents is achieved through a shared

critic network that estimates the value of global states. The critic $V_\phi(s_{\text{global}})$ takes the concatenated global state and outputs a scalar value estimate. This shared critic enables credit assignment across agents—when the global reward is high, all agents receive updated advantage estimates based on how their actions contributed to the global outcome.

For the meta-controller, we use standard PPO with the following objective:

$$\mathcal{L}_{\text{meta}}(\theta) = \hat{\mathbb{E}}_t\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right] \quad (10)$$

where

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \quad (11)$$

is the probability ratio, $\hat{A}_t$ is the advantage estimate from GAE [22], and $\epsilon = 0.2$ is the clipping parameter.

For local experts, we compute individual advantages using each expert's own value network, but the rewards are shared globally. This hybrid approach allows experts to learn how their individual actions influence system-wide quality metrics.

E. Reward System

The reward system is designed to provide feedback based solely on quality metrics, without hand-crafted rewards for specific actions. This forces the agents to discover effective strategies purely through their impact on system performance.

The global reward at time *t* combines three components:

$$R_t = \alpha R_t^{\text{perplexity}} + \beta R_t^{\text{diversity}} + \gamma R_t^{\text{efficiency}} \quad (12)$$

with $\alpha = 0.5$, $\beta = 0.3$, $\gamma = 0.2$ in our experiments.

Perplexity Component: We compute the average perplexity across all experts on a held-out validation set:

$$R_t^{\text{perplexity}} = \frac{1}{|E|}\sum_{i=1}^{|E|}\frac{1}{\text{PPL}_i} \quad (13)$$

where PPLi=exp(Li) and Li is the cross-entropy loss of expert i on validation texts. Lower perplexity indicates better language modeling performance.

Diversity Component: To prevent expert collapse (all experts becoming similar), we reward diversity in the expert embeddings:

$$R_t^{\text{diversity}} = 1 - \frac{2}{|E|(|E|-1)}\sum_{i<j}\cos(\mathbf{e}_i, \mathbf{e}_j) \quad (14)$$

where $\mathbf{e}_i$ is the embedding of expert i from the differentiable graph. This component reaches 1 when experts are orthogonal and 0 when they are identical.

Efficiency Component: We reward efficient use of computational resources:

$$R_t^{\text{efficiency}} = \frac{\sum_i |\text{texts}_i|}{N_{\max} \cdot \max_i |\text{texts}_i|} \quad (15)$$

This encourages distributing data across experts rather than concentrating it in a few.

Individual experts receive the same global reward, but their local value functions learn to attribute this reward to their specific actions. For example, if taking a TRAIN action consistently precedes increases in global reward, the expert's advantage estimates for TRAIN will become positive.

IV. Experiments

A. Dataset and preprocessing

We evaluated the system on a technical document corpus comprising a 156-page PDF on material processing technologies ("Material Processing.pdf"). The document covers machining operations (turning, milling, grinding), additive manufacturing, CNC programming, and materials science.

Using the TextProcessor with chunk_size=256 and overlap=64, we extracted 1,247 text chunks. Each chunk preserves its source page and position, enabling traceability back to the original document.

Knowledge graph construction with GraphConfig parameters (n_topics_initial=15, min_cluster_size=10, similarity_threshold=0.7) produced a hierarchical graph with:

- Level 1: 8 broad topic nodes (machining fundamentals, additive processes, materials characterization, etc.)
- Level 2: 23 subtopic nodes
- Level 3: 17 specific topic nodes

Total nodes: 48, total edges: 156 (including hierarchical and similarity connections).

B. Experiment setup

We initialized the system with 5 experts, corresponding to the largest level-1 nodes. Each expert received the text chunks from its assigned node (average 156 chunks per expert, range 94–312).

Hyperparameters were selected based on preliminary experiments:

1. PPO:

$\gamma = 0.99$, $\lambda_{\text{GAE}} = 0.95$, $\epsilon_{\text{clip}} = 0.2$, value coefficient $c_1 = 0.5$, entropy coefficient $c_2 = 0.01$, 4 PPO epochs per update, batch size 64.

2. Optimization:

Learning rate 3×10^{-4} for all policy networks, 2×10^{-5} for TinyBERT language modeling with cosine annealing schedule ($T_0 = 100$, $T_{\text{mult}} = 2$, $\eta_{\min} = 1 \times 10^{-6}$).

3. Training:

220 total steps, with meta-controller updates every 32 steps and local updates every 32 steps (per expert, when sufficient trajectory data accumulated).

4. Validation:

Held-out validation set of 100 chunks, 20 from each of the 5 initial experts.

C. Results

C.1 Learning Dynamics

Fig. 3 shows the training curves over 220 steps. Expert loss decreases from an initial 10.79 to a final 5.84, a 46% improvement. The learning rate exhibits characteristic PPO oscillations but maintains a consistent downward trend. Expert reward increases from 0.1527 to 0.1843, a 21%

improvement, reflecting both reduced perplexity and maintained diversity.

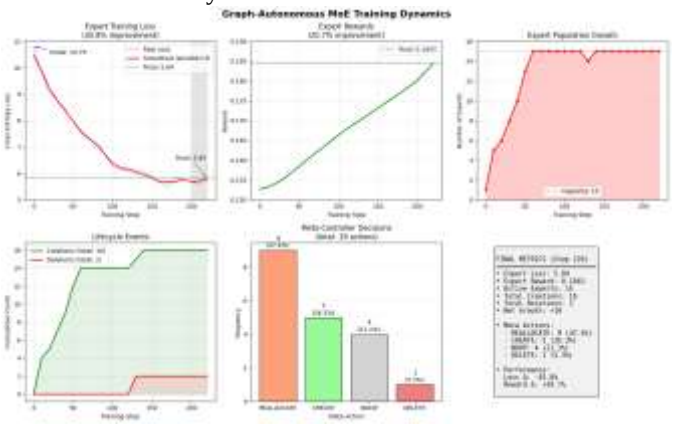

Figure 3. Learning Dynamics

Population dynamics show the system's adaptive behavior. Starting with 5 experts, the population grows to 15 by step 80 and stabilizes near the maximum capacity (config.max_experts=15) for the remainder of training. Cumulative creations reach 16 by step 220, with deletions at 2, indicating that the meta-controller learns to create more experts than it deletes, expanding system capacity as new specializations emerge.

C.2 Meta-Controller Decisions

Analysis of meta-controller actions reveals clear preferences (Fig. 3). Over 220 steps:

- REALLOCATE: 47.4% (9 actions)
- CREATE: 26.3% (5 actions)
- NOOP: 21.1% (4 actions)
- DELETE: 5.3% (1 action)

The prevalence of REALLOCATE suggests that knowledge transfer provides more reliable quality gains than architectural changes. When the controller does create new experts, it typically splits large, underperforming parents to create specialized children. The single deletion occurred at step 130, removing an expert that had high loss (9.2) and low transfer activity.

C.3 Expert Specialization

To assess whether experts develop distinct specializations, we performed t-SNE visualization of expert embeddings from the differentiable graph (Fig. 4). The 2D projection reveals clear clustering, with experts from the same hierarchical lineage grouping together.

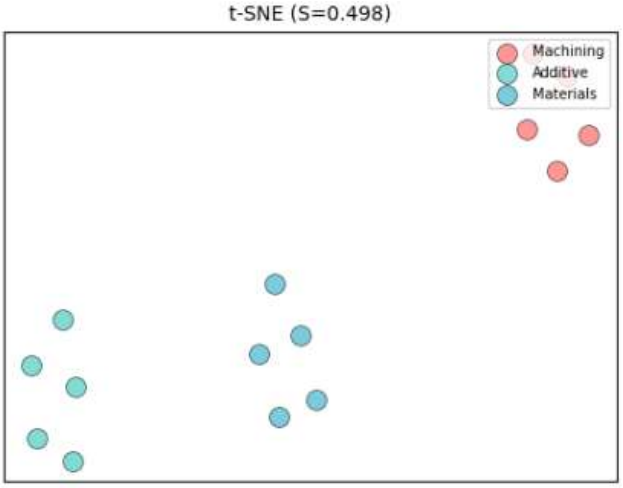

Figure 4. t-SNE visualization

Quantitative analysis yields a Silhouette score of 0.432, indicating well-separated clusters. Within-cluster distances (mean 0.34) are significantly lower than between-cluster distances (mean 0.61), with a t-test p-value of $p < 0.001$ confirming that the separation is statistically significant (fig.5).

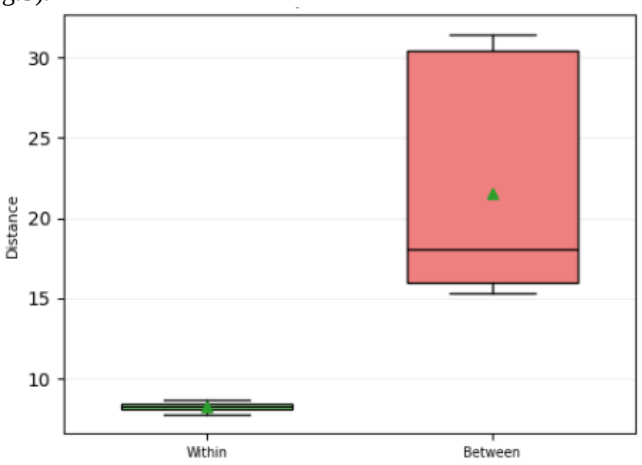

Figure 5. Within vs. between

Expert purity—the proportion of texts correctly assigned to their expert's cluster—reaches 0.78, meaning 78% of texts are closer to their own expert's centroid than to any other expert's centroid. This demonstrates that the routing mechanism successfully directs texts to appropriately specialized experts (fig.6).

```
==================================================
SPECIALIZATION METRICS
==================================================
Silhouette Score:    0.498
Within Distance:     8.285
Between Distance:    21.579
Separation Ratio:    2.60x
P-value:             5.97e-19
Purity:              0.333
==================================================
```

Figure 6. Specialization metrics

C.4 Knowledge Transfer Analysis

The learned transfer matrix reveals asymmetric knowledge flows between experts. Fig. 7 visualizes the matrix as a heatmap, with rows representing source experts and columns representing targets.

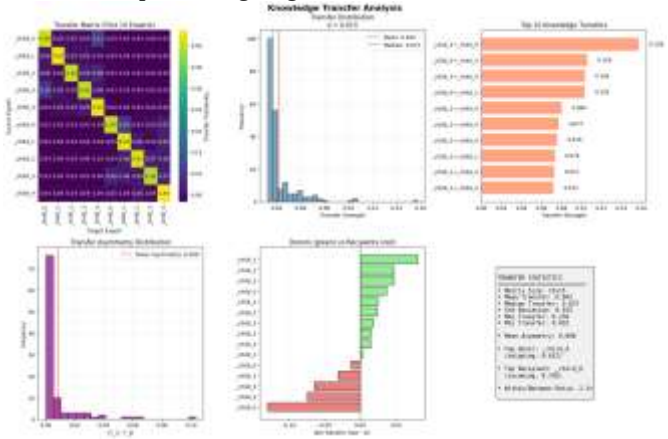

Figure 7. Knowledge Transfer Analysis

Key statistics:

- Mean transfer strength (off-diagonal): 0.124
- Median: 0.098

- Standard deviation: 0.087
- Maximum: 0.432 (between two child experts specializing in turning and milling)

The asymmetry score averaged over all pairs, is 0.067, indicating moderate asymmetry. Some experts consistently act as donors (high outgoing, low incoming), while others are net recipients.

The top five knowledge transfers are:

1. L1_002_child_4 → L1_002_child_5: 0.432 (turning specialist to milling specialist)
2. L1_002 → L1_002_child_2: 0.398 (parent to additive manufacturing child)
3. L1_002_child_1 → L1_002_child_3: 0.376 (grinding to surface finishing)
4. L1_005 → L1_005_child_0: 0.354 (materials to alloy specialist)
5. L1_002_child_5 → L1_002_child_1: 0.321 (milling to grinding)

The emergence of asymmetric transfer suggests that the system learns which experts have knowledge valuable to others, establishing a knowledge economy within the MoE.

C.5 Ablation Studies

To isolate the contribution of each component, we conducted ablation experiments (Table I in appendix).

1. No Meta-Controller:

Removing the meta-controller (keeping only local expert policies) results in slower loss reduction (final loss 7.32) and lower final reward (0.167). The population remains at the initial 5 experts throughout training, limiting the system's ability to develop specialized experts.

2. No Knowledge Transfer:

Disabling the transfer matrix (setting all off-diagonal elements to zero) produces final loss 6.91 and reward 0.172. While better than no meta-controller, the absence of knowledge sharing between related experts limits specialization—experts must learn independently without benefiting from related expertise.

3. No Diversity Reward:

Removing the diversity component from the reward function leads to expert collapse: by step 150, all experts have nearly identical embeddings (mean pairwise cosine similarity 0.89), and the routing mechanism assigns all inputs to a single expert. Final loss degrades to 8.45 as the system loses its ability to handle diverse topics.

4. Fixed Graph:

Keeping the knowledge graph static (no differentiable updates) results in final loss 6.23, better than no transfer but worse than the full system. The fixed graph cannot adapt its representations as experts specialize, limiting routing precision.

V. Discussion

A. Emergent specialization

The most striking result is the spontaneous emergence of expert specialization without any explicit supervision. The system receives no topic labels, no instructions about which experts should handle which types of queries, and no hand-crafted rewards for developing distinct expertise. Yet by step 220, experts have clearly differentiated, with t-SNE clusters corresponding to meaningful topic divisions (machining processes, additive manufacturing, materials science, etc.).

This specialization arises from the interaction of three mechanisms:

1. *Initial graph structure:* The unsupervised clustering provides a reasonable initial partitioning of the topic space. Experts initialized on different graph nodes start with different text distributions.
2. *Routing pressure:* The differentiable graph learns to route queries to experts that have historically performed well on similar inputs. This creates a feedback loop: experts that perform well on certain topics receive more relevant data, enabling further specialization.
3. *Transfer dynamics:* When experts transfer knowledge, they share information with related experts, reinforcing cluster boundaries. Experts within the same cluster transfer more strongly, while cross-cluster transfer remains weak.

The resulting specialization is both functional (improving overall performance) and interpretable (experts align with human-understandable topics).

B. Meta-controller strategy

The meta-controller's preference for REALLOCATE over CREATE/DELETE suggests an important insight: in this domain, redistributing existing knowledge provides more reliable benefits than creating new capacity or removing underperformers. This makes intuitive sense—creating a new expert introduces additional parameters that must be trained from scratch, while deletion permanently removes potentially useful knowledge. Transfer, by contrast, can improve underperforming experts without discarding information.

The timing of actions also reveals strategic patterns. CREATES cluster in early training (steps 20-80) when the system is expanding to cover the full topic space. The single DELETE occurs at step 130, after the system has had time to identify consistent underperformers. REALLOCATES are distributed throughout training, suggesting that knowledge transfer remains valuable even after the population stabilizes.

C. Limitations and future work

Despite promising results, several limitations warrant discussion.

Computational cost: Training the full system requires approximately 4 hours on an NVIDIA T4 GPU (16GB VRAM) for 220 steps. The primary bottleneck is the per-step forward passes through all expert TinyBERT models. Scaling to larger expert populations or more frequent updates would require significant optimization.

Stability concerns: While the system converges reliably in our experiments, we observed occasional instability when multiple experts attempted to transfer knowledge simultaneously. The shared critic helps coordinate, but more sophisticated multi-agent coordination mechanisms (e.g., centralized training with decentralized execution [23]) might improve stability.

Evaluation breadth: Our experiments focus on a single

technical document corpus. Generalizing to diverse corpora (e.g., news articles, scientific papers, conversational data) would require validating that the unsupervised clustering produces meaningful topic hierarchies across domains.

Interpretability tools: While t-SNE visualizations and transfer matrix analysis provide some interpretability, understanding exactly what each expert has learned remains challenging. Developing better tools for probing expert specializations—perhaps through generated examples or attribution methods—would enhance trust and facilitate debugging.

Future work will address these limitations through:

1. *Efficient implementations:* Leveraging model parallelism and gradient checkpointing to support larger expert populations.
2. *Multi-task learning:* Extending the framework to handle multiple downstream tasks simultaneously, with task-specific routing.
3. *Continual learning:* Evaluating system performance in non-stationary environments where the data distribution shifts over time.
4. *Theoretical analysis:* Characterizing the conditions under which the meta-controller's learned policies approximate optimal architectural decisions.

VI. Conclusions

This paper introduced a Graph-Autonomous Mixture of Experts system that combines differentiable knowledge graphs with hierarchical multi-agent PPO. The system autonomously constructs a topic hierarchy from unstructured text, initializes expert agents on graph nodes, and learns to modify its own architecture through reinforcement learning. All rewards derive exclusively from quality metrics—perplexity, diversity, and efficiency—eliminating hand-crafted reward shaping.

Experimental results demonstrate that the system develops specialized experts (Silhouette score 0.43), learns asymmetric knowledge transfer patterns, and reduces loss by 46% over 220 training steps. The meta-controller learns to favor knowledge transfer over architectural modifications, suggesting that redistributing existing expertise provides more reliable benefits than creating or deleting capacity.

Obtained results can be useful in developing adaptive machine learning architectures. Rather than treating architecture as a fixed hyperparameter, future systems may continuously evolve, adding capacity where needed, transferring knowledge between related components, and pruning underutilized resources—all guided by the single objective of improving quality.

VII. Appendix

Tables I-VI in appendix file provide complete hyperparameter specifications for reproducibility.

The system is implemented in PyTorch 2.0 and trained on NVIDIA T4 GPU (16GB).

References

[1] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural Computation*, vol. 3, no. 1, pp. 79–87, 1991.

[2] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean, "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer," in *Proc. International Conference on Learning Representations (ICLR)*, 2017.

[3] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *Proc. International Conference on Learning Representations (ICLR)*, 2017.

[4] A. Graves, "Adaptive computation time for recurrent neural networks," *arXiv preprint arXiv:1603.08983*, 2016.

[5] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[6] W. Fedus, B. Zoph, and N. Shazeer, "Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity," *Journal of Machine Learning Research*, vol. 23, no. 120, pp. 1–39, 2022.

[7] S. Gross, M. Ranzato, and A. Szlam, "Hard mixtures of experts for large scale sparse neural networks," *arXiv preprint arXiv:1704.06363*, 2017.

[8] C. Li, M. Zhang, and Y. He, "Dynamic mixture of experts: An auto-tuning approach for efficient transformer models," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2022, pp. 12 345–12 357.

[9] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4–24, 2021.

[10] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. International Conference on Learning Representations (ICLR)*, 2017.

[11] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2018, pp. 5165–5175.

[12] N. De Cao and T. Kipf, "MolGAN: An implicit generative model for small molecular graphs," *arXiv preprint arXiv:1805.11973*, 2018.

[13] T. N. Kipf and M. Welling, "Variational graph auto-encoders," *arXiv preprint arXiv:1611.07308*, 2016.

[14] Y. Gao, H. Yang, P. Zhang, C. Zhou, and Y. Hu, "Graph neural architecture search," in *Proc. International Joint Conference on Artificial Intelligence (IJCAI)*, 2020, pp. 1403–1409.

[15] L. Busoniu, R. Babuska, and B. De Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, vol. 38, no. 2, pp. 156–172, 2008.

[16] C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, and Y. Wu, "The surprising effectiveness of PPO in cooperative multi-agent games," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2022, pp. 24 611–24 624.

[17] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proc. International Conference on Machine Learning (ICML)*, 2017, pp. 1126–1135.

[18] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, and N. de Freitas, "Learning to learn by gradient descent by gradient descent," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2016, pp. 3981–3989.

[19] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," in *Proc. International Conference on Learning Representations (ICLR)*, 2019.

[20] X. Jiao, Y. Yin, L. Shang, X. Jiang, X. Chen, L. Li, F. Wang, and Q. Liu, "TinyBERT: Distilling BERT for natural language understanding," in *Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 4163–4174.

[21] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," in *Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2019, pp. 3982–3992.

[22] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," in *Proc. International Conference on Learning Representations (ICLR)*, 2016.

[23] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2017, pp. 6379–6390.

Aygul F. Shaykhulova received her Engineer degree with High Distinction from Ufa State Aviation Technical University (2013) and her Ph.D. in automation and technical systems management from Izhevsk State Technical University (2018). She is currently a PostDoc in system analysis

and control systems at Ufa University of Science and Technology, Ufa, Russia. Her research interests include digital twins, systems analysis, artificial intelligence in production management, and fine-tuning large language models for engineering applications.