Grenze International Journal of Engineering and Technology, June Issue

Study Cram Scheduling and Learning Assistant using Mixture-of-Recursions (MoR) in Agentic AI

Grace Gnanam J[1], Paul Immanuel J[2] and Anitha J[3]
[1-3]Karunya Institute of Technology and Sciences, Coimbatore, India
Email: gracegnanam@karunya.edu.in, paulimmanuel@karunya.edu.in, anitha_j@karunya.edu

***Abstract*—The digital learning industry has become a rapidly growing sector since early 2020. Traditionally, before the expansive presence of Artificial Intelligence (AI), browsing has been the primary source of information for learning or attending quizzes on educational websites. However, recently, Large Language Models (LLMs) have been utilized, which has led to efficiency competition among multiple AI models. LLMs have handled multiple workloads, which has reduced time efficiency. Agentic AI has introduced a divide-and-conquer strategy to improve time efficiency. This approach has supported the learning domain by enabling tasks to be completed effectively while reducing overall computation. Advancements in Transformer architecture is presented at the token level computation as well as in other areas resulting in the development and have been introduced as Mixture-of-Recursions (MoR). This approach has aimed to dynamically change the number of routing operations based on the importance assigned to tokens. MoR also represents an advancement of existing recursion mechanisms. The MoR architecture has been compared with Vanilla and other recursion architectures to evaluate improvements in accuracy. Considering the pretrained checkpoints of MoR, Vanilla, and Recursion as foundation, fine-tuning has been conducted, with each architecture incorporating a quiz model, a scheduler model, a pedagogy model, and an orchestrator model. The orchestrator models of MoR, Vanilla, and Recursion have been evaluated using metrics such as routing confidence margin (RCM), decision stability, token efficiency, and router entropy. This advancement has examined the possibility of maintaining higher accuracy when deployed as an orchestrator for effectively conveying commands to agents on an academic platform. The RCM of the MoR model has an almost 40-fold increase in RCM over Vanilla, demonstrating agents are being routed with greater decisiveness and reliability.**

***Index Terms*— vanilla, agentic AI, digital learning, recursion architectures, Mixture-of-Recursions, few-shot, adaptive.**

I. INTRODUCTION

The advent of digital education in the early 2020's dramatically changed how teachers teach students and students learn by facilitating real-time student participation. While it offers many advantages over traditional classroom learning; one major disadvantage of digital education is that it does not offer the same level of human assistance as traditional classroom settings. This lack of human interaction with the instructor creates feelings of loneliness or isolation and can lead to time management problems for the learner. In order to assist with some of the disadvantages associated with digital education, a Study Cram Scheduling and Learning Assistant (SCSLA) is proposed with flexible scheduling options, quizzes, and enhanced few-shot adaptability through the MoR

Grenze ID: 01.GIJET.12.2.502_26

optimization technique. The main contributions of this paper are integrating MoR into an educational agentic AI system as well as developing an adaptive orchestration method for combining pedagogical, scheduling and quiz generation agents. This research provides a different approach from current educational systems which rely solely on static transformer inference and conventional orchestration methods. The proposed architecture uses dynamic token-level recursion to enhance the routing confidence while at the same time reduce the computing resources required to select an appropriate agent. Additionally, three architectures including vanilla, recursion, and MoR were compared based on RCM (Routing Confidence Margin), DSS (Decision Stability Score), TE (Token Efficiency), and RE (Router Entropy) on various few-shot educational tasks. The feasibility of MoR-based orchestration in resource-constrained educational systems is also explored.

II. Literature Review

A. Transformer and Recursive Architectures

The transformer architecture introduced by Ashish Vaswani significantly advanced NLP through self-attention mechanisms capable of capturing long-range dependencies using parallel computation [1]. Since then, most modern LLMs have been built upon some type of transformer-based architecture. However, one major limitation of such models is that they can be quite large and deep. Recursive transformer architectures were developed to increase effective depth while reducing model size and memory requirements [2]. More recently, researchers have looked at how to make use of adaptive computation methods, such as Mixture-of-Experts (MoE) or sparse routing and token routing [3], [4] to selectively turn on/off different computation paths. While other authors have adapted these ideas into their own work [5], MoR represents a new way to implement adaptive token-level recursion using lightweight routing modules similar to those found in MoE. In contrast to traditional recursion methods where all tokens are recursively processed the same number of times, MoR allows for each token to determine its own recursion depth based on its level of importance. Therefore, MoR has been shown to improve both few-shot learning and validation loss over baseline methods and reduce computational costs. Recently, research on MoR-ViT has demonstrated that MoR's token-level recursive routing can be applied effectively across multiple modalities [6]. Additionally, recent studies on chain-of-thought reasoning have emphasized the importance of adaptive computation and token specialization in order to enhance the performance of few-shot models [7].

B. Agentic AI and Multi-Agent Orchestration

Agentic AI is an important milestone in Artificial Intelligence, allowing autonomous systems to reason about goals, decompose tasks and dynamically decide what to do next [8],[9]. Unlike traditional single-task systems, agentic architectures integrate specialized agents into complex workflows. Frameworks such as AutoGen [10], AutoGen Studio [11], LangGraph, and MetaGPT [12] demonstrate the growing importance of multi-agent orchestration. Orchestrators manage communication, routing, and task assignment. The success of these systems relies upon routing reliability, decision consistency and the ability to efficiently orchestrate. In addition to these characteristics, trust aware orchestration frameworks [13] also place emphasis upon the development of routing and retrieval-grounded reasoning that uses confidence calibrated routing. The improvement of distributed orchestration, convergence, and resource efficiency while maintaining user data privacy using Federated Learning (FL), in conjunction with Large Language Model (LLM)-based automation has provided new insights into improving distributed LLM orchestration and agentic workflows. The results from recent studies show that coordinating agents can provide increased specialization for tasks, decrease interaction complexity and increase adaptive execution. As such, there exists a need to explore adaptive recursive architectures such as MoR within educational AI orchestrators. Additionally, most of the current frameworks being used today rarely if ever utilize token-level adaptive recursion to increase routing confidence and/or computational efficiency.

C. AI-Based Educational Assistance Systems

The growth of artificial intelligence in digital education is evident with the use of intelligent tutoring systems, automated quizzing systems, smart study planners, and AI-empowered learning assistants. Current applications of AI in digital education utilize large language models (LLM), retrieval-augmented generative (RAG), reinforcement learning, and vector databases to generate assessments automatically, create a study schedule, provide pedagogical assistance etc. [17]. These technologies improve scalability while enabling personalized study plans based on learner goals, ability, and available time. [18]. Genetic algorithms, reinforcement learning, clustering, and neural networks can be used in conjunction with scheduling systems to optimize both hard constraints (classroom availability) and soft constraints (student preferences regarding class scheduling) [19]

allowing for increased flexibility compared to prior methods (paper/pencil, spreadsheet). studies show AI-assisted pedagogical systems improve engagement, language acquisition, critical thinking, and problem-solving abilities [20]. Narrative learning and adaptive explanations further improve comprehension [21]. Open-source educational AI platforms also allow for the creation of auto-generated quizzes and grading as well as immediate feedback to students [22]. Localized LLMs and retrieval-based assistants improve contextual understanding over traditional educational search systems [23].

D. Research Gap

Although agentic AI, recursive transformers, and educational AI systems have been widely studied, limited work has investigated integrating MoR within educational orchestration frameworks. Current systems mainly rely on traditional LLMs and static routing methods. The primary disadvantage of these methods is their low adaptability in few shot settings and high computational costs. Most prior studies have investigated automated education and recursive architecture separately; therefore, there has been minimal research investigating token level recursion to improve routing reliability and reduce computational cost. This paper will explore MoR as an architectural framework for orchestrating educational AI systems through evaluation of routing decisiveness', token efficiency, and few shot adaptabilities via Routing Confidence Metric (RCM); Decision Stability Score (DSS); Token Efficiency (TE); and Router Entropy (RE).

III. METHODOLOGY

A. Overview of the experimentation

The primary objective of is to evaluate MoR's learning behavior and its few-shot efficiency. Dynamic recursion adapts the iteration count at runtime; whereas traditional recursive methods employ a predetermined number of iterations. This provides for better computational performance than the non-recursive Vanilla model. Each agent was trained from a dataset specifically designed to be used as training data for specific tasks, with the agents' task-specific actions being directed by the Orchestrator. The The Orchestrator is the central component and is trained using dedicated datasets. The overall structure of the system consists of Pedagogy Agent, Questionnaire Agent, Scheduler Agent, and Orchestrator Agents illustrated in Figure 1, all of which were trained and tested using the dynamic recursion approach employing the MoR checkpoint.

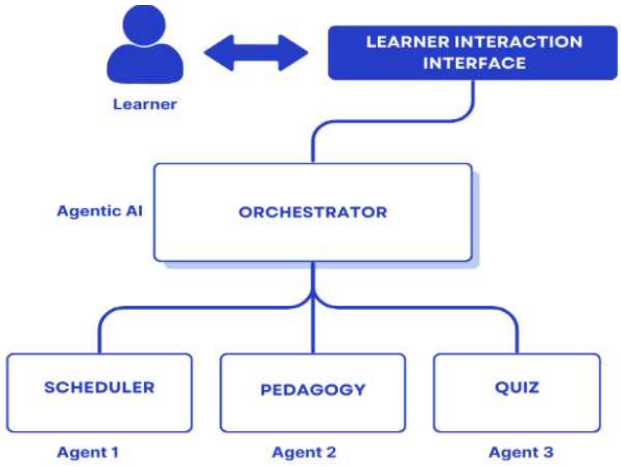

Figure 1. Generic workflow of Agentic architecture of one base model for the proposed system

In order to provide for fairness in comparisons, both Vanilla and Recursive models employed the same pre-trained baseline fineweb-edu. Model adaptability is measured based on how many few-shot samples were presented to each model during testing. Following initialization, scheduling, quiz, and orchestration agents are jointly trained using application-specific datasets to evaluate adaptability and few-shot efficiency. This approach enables deeper architectures with fewer parameters and lower memory usage through iterative refinement [24]. As recursive approaches employ a uniform, predetermined amount of 'attention', a measure of how much each input contributes to the output, over all tokens when determining relevance to the final output, MoR introduces the ability to adaptively perform recursion at the level of the individual tokens by allowing for learnable routing mechanisms, as depicted in Figure 2.

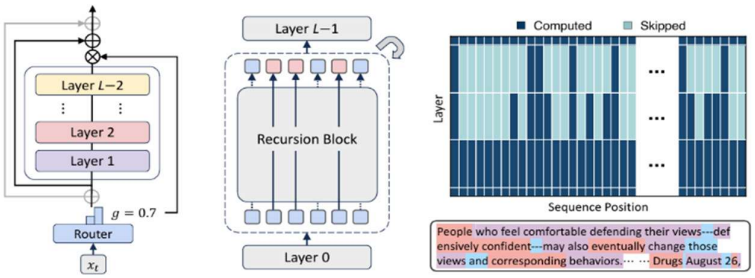

Figure 2. Outline of MoR [5]

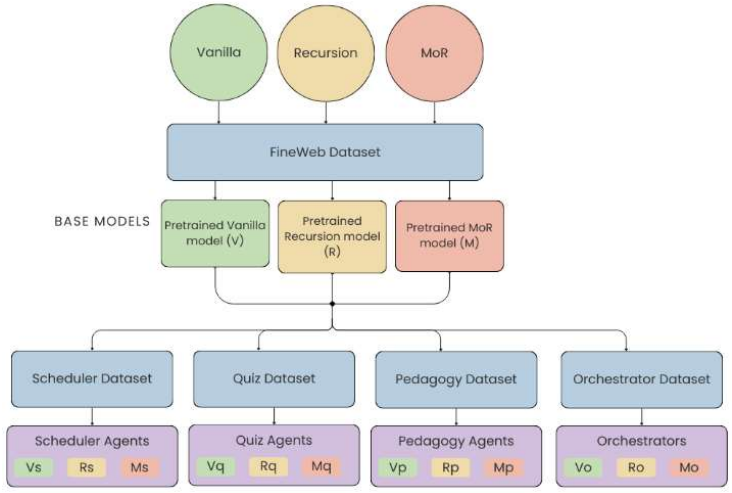

Figure 3. Overview of producing Trained Models

Unlike conventional recursion, MoR dynamically assigns recursion depth based on token importance, enabling efficient token-level processing as shown in Figure. 3.

B. Training Processes and Datasets

The model uses a pretrained checkpoint and configuration file containing training specifications. The Scheduling Agent generates optimized schedules using task timing and priority information. Similar Learning and Scheduling Datasets [25] and learner-specific datasets shown in Table I are used for effective time management and goal achievement.

TABLE I. GOAL-ALIGNED SAMPLES OF TRAINING DATASETS FOR EACH AGENT

Agent Type	Sample Data
Scheduling Agent	{"text": "User schedule: Breakfast with family at 8am, online class at 10am, and football practice at 5pm. Commute is 15 mins. Generated schedule: 8:00–9:00 am breakfast with family. 10:00–12:00 pm online class. 4:45–5:00 pm commute to football. 5:00–6:30 pm football practice."}
Quiz Agent	{"text": "Question:\nIn the desert, a hawk may enjoy an occasional\n\n Choices:\nA) coyote\nB) reptile\nC) bat\nD) scorpion\n\n Answer: B"}
Pedagogy Agent	{"text": "Topic: Time Complexity\nLevel: beginner\nStyle: intuitive\nExplanation: Time complexity describes how the execution time of an algorithm grows as the size of the input increases."}
Orchestrator	{"goal": "Short focused study session", "context": {"topic": "Boolean Algebra", "time_available":"45 minutes"}, "orchestration_plan" :[{"agent":"scheduler","input": "Create a 45-minute study plan"}, {"agent": "pedagogy", "input": "Explain basic Boolean laws"}], "final_success": true}

The Scheduling Agent will produce Study Plans that take into consideration Deadlines and Priorities; The Pedagogy Agent will modify Explanations based on learner Level and Preferences; and The Quiz Agent will assist learners in Self-Assessing their knowledge through Multiple Choice Questions. The Orchestrator will utilize customized datasets to facilitate coordination among Agents and select those best suited to accomplish the User's Goals.

C. Evaluation

The orchestrator is the primary decision-making component responsible for task routing and agent selection, with agent selection accuracy serving as a key metric. Decisiveness is provided by the RCM,

$$RCM = p(1) - p(2) \quad (1)$$

where the $p = softmax(z)$ is the probability distribution over tokens from router logits and p(1) is the largest probability and p(2) the second-largest probability. The softmax is calculated using the commonly used formula,

$$p_i = \frac{e^{z_i}}{\sum_j e^{z_j}} \quad (2)$$

Determining the measure of determinism and robustness of routing is based on the number of repeated runs and is the agent predicted in run and is the agent from the first run,

$$\mathrm{DSS} = \frac{1}{\mathrm{N}} \sum_{\mathrm{i}=1}^{\mathrm{N}} \mathrm{I}(\mathrm{a_i} = \mathrm{a_1}) \quad (3)$$

where I (•) is the indicator function. The token efficiency provides evaluation of the efficiency of orchestration, where the lower the values, the more efficient the orchestration is with less unnecessary generation,

$$\mathrm{TE} = \mathrm{L_{out}} - \mathrm{L_{in}} \quad (4)$$

where Lout is the generated sequence length and L_{in} is the input prompt length. The router entropy (RE) measures routing uncertainty which specifies that the lower entropy is sharper decisions, where p is the token probability distribution, number of generated steps (T), and probability of token I at step t ($p_{t,i}$),

$$\mathrm{RE} = -\frac{1}{\mathrm{T}} \sum_{\mathrm{t}=1}^{\mathrm{T}} \sum_{\mathrm{i}} \mathrm{p_{t,i}} \log\left(\mathrm{p_{t,i}}\right) \quad (5)$$

RCM, DSS, TE, and RE collectively evaluate decisiveness, consistency, efficiency, and uncertainty. The orchestrator model trained over the base architecture of MoR has a greater confidence for routing, but the decision stability remains the same. The routing entropy is calculated by (1) which computes the decisiveness of the orchestrator routing of the tasks to one agent over others.

TABLE II. GOAL-ALIGNED SAMPLES OF TRAINING DATASETS FOR EACH AGENT

Base Architecture	Routing Confidence Margin	Decision Stability	Token Efficiency (avg)	Router Entropy
MoR	0.0286	1.0000	64.0	4.9390
Recursion	0.0193	1.0000	59.21	4.2481
Vanilla	0.0007	1.0000	64.0	4.7315

The data in Table II illustrates how MoR has a larger Routing Confidence Margin (RCM) as well as greater improvements in Token Efficiency (TE) and Router Entropy (RE) than the original. Confidence margin, entropy and stability is used to measure the three important aspects of our approach. As a result, our results indicate an improvement of over 39 times better routing confidence for MoR over its previous version, therefore the agents make decisions with more certainty and consistency.

IV. RESULTS AND DISCUSSION

The architecture models in [5] were tested for benchmark performance through six tests of different types: Lambda (LD), Physical Interaction Question Answering (PQ), Harder Endings Longer Contexts and Low-shot Activities (HS), Massive Multitask Language Understanding (MMLU), Abstraction and Reasoning Corpus (ARC), and WinoGrande (WG). Scheduler, quiz, and auxiliary agents were evaluated individually before orchestration. Although trained on identical datasets, MoR, Recursion, and Vanilla exhibited different task behaviors and success rates. Figure 4 displays how the Vanilla model is expected to follow a normal warm up learning curve. The learning rate will be at its peak (approximately 3e−3) then it will begin to slow down.

Likewise, loss initially drops quickly then after several iterations reaches a new level that represents stable convergence with minimal oscillation. The loss curve shown in Figure 5 illustrates how the Recursion-based model has the same type of learning rate as the Vanilla model; however, it also demonstrates some moderate fluctuation during training. The loss curve for the Recursion-based model begins to converge at approximately the same point as the Vanilla model's loss curve; however, there are temporary oscillations before the loss stabilizes. Despite fluctuations, the Recursion model converged to a stable low-loss state. The MoR-based

architecture depicted in Figure 6 demonstrates an initial steep decline in loss similar to the other architectures; however, MoR has a larger amount of fluctuation and sometimes spikes in the last parts of the training process due to both routing specialization and expert selection. Despite loss variance, MoR and Recursive models converged successfully. The training behavior of all the pedagogic agents is shown in Figure. 7, Figure. 8 and Figure. 9 for each architecture using a common learning rate. Three general trends can be observed from these figures: Rapid initial convergence (warm up), followed by stability throughout the remainder of training; Distinctive optimization behaviors are exhibited based on the specific architecture.

Figure. 7. shows that vanilla training exhibits sharp decrease in loss initially and then becomes a smoother and generally stable function. The lack of variability in the loss function after it has converged indicates that the gradients have been consistently updated. This is indicative of stable optimization behavior. Although this method is successful, the monotone nature of the loss function does suggest a potential limitation of the representational flexibility of the model, when it is in a local optimum regime.

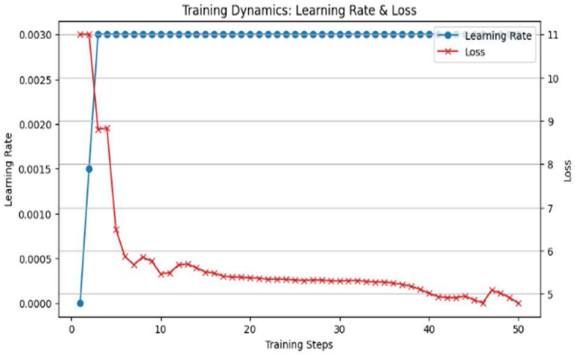

Figure 4. Vanilla Quiz Agent Training – Learning Rate and Loss

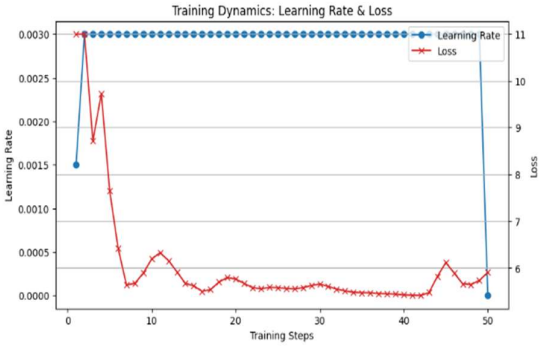

Figure 5. Recursion Quiz Agent Training – Learning Rate and Loss

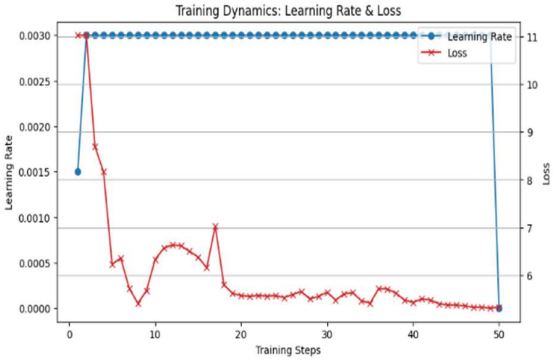

Figure 6. MoR Quiz Agent Training – Learning Rate and Loss

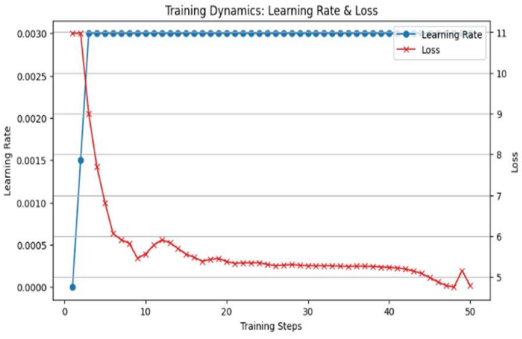

Figure 7. Vanilla Pedagogy Agent Training – Learning Rate and Loss

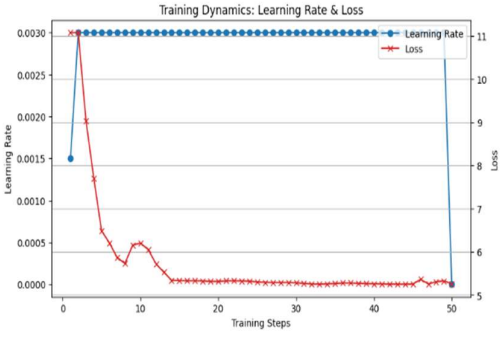

Figure 8. Recursion Pedagogy Agent Training – Learning Rate and Loss

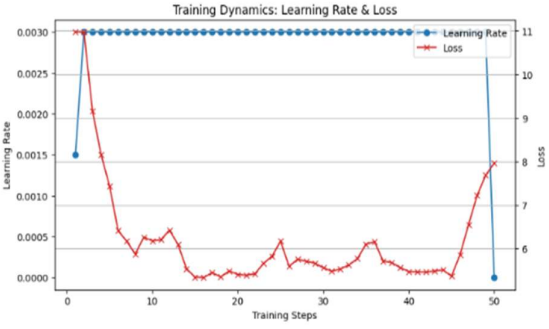

Figure 9. MoR Pedagogy Agent Training – Learning Rate and Loss

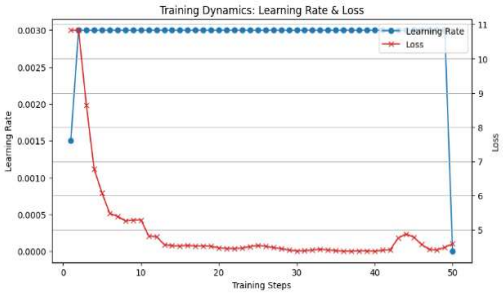

Figure 10. MoR Orchestrator Agent Training – Learning Rate and Loss

Similar to the Vanilla agent, Recursion converges rapidly during initial training and stabilizes at a lower loss point. Its consistent gradient flow results in predictable convergence, while lower variance indicates structural stability in the optimization path. MoR exhibits greater loss variability during early and intermediate stages before stabilization, likely due to expert specialization and adaptive routing for processing diverse task patterns.

TABLE III. GOAL-ALIGNED SAMPLES OF TRAINING DATASETS FOR EACH AGENT

Base Architecture	Time taken for training agents (in HH:MM:SS)			
	Pedagogy	Quiz	Scheduler	Orchestrator
MoR	1:53:19	2:15:26	2:13:31	1:25:22
Vanilla	4:53:25	5:09:18	4:05:12	5:00:18
Recursion	34:36	0:32:51	2:51:28	0:34:05

In addition, as demonstrated in Figure 10, MoR has an exploration profile for optimization that shows more exploratory characteristics compared to Vanilla and Recursion, whose profiles show less variation. Furthermore, the longer optimization phase and extended training time of Vanilla demonstrates that it has lower levels of adaptation at fewer shots of data, while Table III provides an overview of the computation costs associated with each architecture.

V. CONCLUSION

Evidences of the efficiency of the MoR architecture is presented, based on the point of view of training orchestrator in the realm of Agentic AI. Learning behavior was analyzed using training loss and convergence characteristics. The loss in the learning rate plots displays that the performance of MoR architecture on few-shot examples is significantly better when compared to Vanilla and Recursion architectures. The challenging part is performing training of agents with GPU-safe adjustments on a non-data centre system with a few local hardware constraints. The MoR checkpoint is trained on high graphics system hardware devices like four H100 or A100 GPUs, which is massive. The training for the task specific agents is performed on single NVIDIA RTX A400 GPU. Future training may benefit from higher-end GPUs such as A100 systems.

ACKNOWLEDGMENT

We extend our thanks to the researchers of the architecture of MoR, for the profound contribution to the open-source community. We would also like to thank the Microsoft Centre of Data Science and AI Laboratory, Division of Computer Science and Engineering, Karunya Institute of Technology and Sciences for the provision of hardware equipment to perform training and evaluation of the models for the purpose of this research.

REFERENCES

[1] Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).
[2] Gong, Zixuan, Yong Liu, and Jiaye Teng. "What Makes Looped Transformers Perform Better Than Non-Recursive Ones." arXiv preprint arXiv:2510.10089 (2025).
[3] Shazeer, N., et al., “Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer,” arXiv preprint arXiv:1701.06538, 2017.

[4] Fedus, William, Barret Zoph, and Noam Shazeer. "Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity." Journal of Machine Learning Research 23.120 (2022): 1-39.
[5] Bae, Sangmin, et al. "Mixture-of-recursions: Learning dynamic recursive depths for adaptive token-level computation." arXiv preprint arXiv:2507.10524 (2025).
[6] Li, YiZhou. "MOR-VIT: Efficient Vision Transformer with Mixture-of-Recursions." arXiv e-prints (2025): arXiv-2507.
[7] Liang, Yuanyuan, et al. "Prompting large language models with chain-of-thought for few-shot knowledge base question generation." arXiv preprint arXiv:2310.08395 (2023).
[8] Acharya, Deepak Bhaskar, Karthigeyan Kuppan, and B. Divya. "Agentic ai: Autonomous intelligence for complex goals–a comprehensive survey." IEEe Access (2025).
[9] Murugesan, San. "The rise of agentic AI: implications, concerns, and the path forward." IEEE Intelligent Systems 40.2 (2025): 8-14.
[10] Wu, Qingyun, et al. "Autogen: Enabling next-gen LLM applications via multi-agent conversations." First conference on language modeling. 2024.
[11] Dibia, Victor, et al. "Autogen studio: A no-code developer tool for building and debugging multi-agent systems." Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations. 2024.
[12] Hong, Sirui, et al. "MetaGPT: Meta programming for a multi-agent collaborative framework." International Conference on Learning Representations. Vol. 2024. 2024.
[13] Zhang, Jiayi, et al. "Aflow: Automating agentic workflow generation." International Conference on Learning Representations. Vol. 2025. 2025.
[14] Khamis, Alaa. "Agentic AI systems: architecture and evaluation using a frictionless parking scenario." IEEE Access (2025).
[15] Roumeliotis, Konstantinos I., et al. "Orchestrator-Agent Trust: A Modular Agentic AI Visual Classification System with Trust-Aware Orchestration and RAG-Based Reasoning." arXiv e-prints (2025): arXiv-2507.
[16] Mawela, Chamith, Chaouki Ben Issaid, and Mehdi Bennis. "A web-based solution for federated learning with LLM-based automation." IEEE Internet of Things Journal (2025).
[17] Singh, Aditi, et al. "Enhancing ai systems with agentic workflows patterns in large language model." 2024 IEEE World AI IoT Congress (AIIoT). IEEE, 2024.
[18] Pati, Ashis Kumar. "Agentic AI: A Comprehensive Survey of Technologies, Applications, and Societal Implications." IEEE Access (2025).
[19] Xu, Hanyi, et al. "Large language models for education: A survey." arXiv preprint arXiv:2405.13001 (2024).
[20] Mohanraj K R, Abinayasankar M, Balaji G B. “Automated Learning and Scheduling Assistant using LLM”. Journal of Ubiquitous Computing and Communication Technologies (2024)
[21] Lingshetti, Gagana, et al. "AI-Powered Smart Study Planner: Enhancing Personalized Learning Through Intelligent Scheduling." International Journal of Scientific Research in Science, Engineering and Technology 12.3 (2025): 148-155.
[22] Saradha, P., et al. "Integration of AI tools in learning pedagogy for L2 Learners in Engineering Education: Impacts, Challenges and Possibilities." 2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT). IEEE, 2024.
[23] Runceanu, Adrian, et al. "Enhancing the Learning Experience with AI." Information 16.5 (2025): 410.
[24] Ingram, William A., Bipasha Banerjee, and Edward A. Fox. "Agentic AI for Improving Precision in Identifying Contributions to Sustainable Development Goals." 2024 IEEE International Conference on Big Data (BigData). IEEE, 2024.
[25] Penedo, Guilherme, et al. "The fineweb datasets: Decanting the web for the finest text data at scale." Advances in Neural Information Processing Systems 37 (2024): 30811-30849.