LayerSkip for Mixture of Experts (MoE) Architecture: Project Check-in Report

Nicholas Papciak Tom Jeong
Georgia Institute of Technology

March 14, 2025

Abstract

Large language models (LLMs) have demonstrated remarkable capabilities but remain computationally expensive to deploy and operate. Mixture of Experts (MoE) architectures have emerged as a promising approach for scaling LLMs efficiently by selectively activating only a subset of expert parameters for each forward pass. While MoE provides width-wise sparsity (activating only a portion of the network horizontally), we identify an opportunity to integrate LayerSkip to provide complementary depth-wise sparsity, enabling dynamic computation paths based on input complexity.

1 Introduction

Recent advances in large language models (LLMs) have demonstrated incredible capabilities and revolutionary computational abilities. However, to support the large increase in the accuracy and capability of these models, the size of the models has had to increase drastically. This has underscored a clear need for efficient, scalable, sparsity techniques that can allow models to get larger without as much computational overhead. In recent years, Mixture of Experts (MoE) models have made themselves known as such a promising solution. They make LLMs more efficient by routing data through the network so that only a sparse subset of "experts" is activated at each step. Thus far, the approach has seen some great success. Deepseek-V3 (1) and GPT4 (2) which both heavily rely on the MoE architecture have achieved state-of-the-art results using this technique, with gargantuanly sized models. However, despite the massive model sizes, they are able to keep inference costs feasible because of (1) the inherent scalability of MoE and transformer architectures, and (2) the sparsity introduced by the MoE models. They are starting to reach a potential limit, however. The recent Deepseek-V3 model has a staggering 256 experts (1), for example, which is wildly larger from the 16 experts of the precursor GPT4 model. This represents a dramatic shift in the way MoE architecture will be used, and emphasizes the importance of model size when trying to improve LLMs. This is especially worrying as model sizes have been increasing faster than computational resources to support them have. Thus, there is a desperate need for further sparsity techniques to make inference cheaper and more efficient.

We propose implementing LayerSkip (6) as such a sparsity technique. LayerSkip allows the selective bypassing of expert layers based on input complexity, to add even more sparsity to an MoE model.

LayerSkip has been previously only been implemented in traditional dense LLama-7B models. The LayerSkip architecture works in three main stages.

1. Layer Dropout: Layer dropout is applied during training with one of two different curriculums and increasing rates as the network deepens.

2. Early Exit: The model learns to leave certain layers early during inference. The model is trained to try to "make up its mind" early in inference rather than going back and forth (as traditional models currently demonstrate) (6). Each layer shares a single exit layer (called the LM head), which is in contrast to previous approaches which tried to use separate exit layers for each individual layer (3).

3. Self-Speculative Decoding

   Because we exit at early layers, we can verify the results using the layer layers of the network.

2 Related Work

Our approach builds upon several key research areas:

Mixture of Experts Architectures

MoE has evolved significantly since its introduction. Shazeer et al. (12) proposed sparsely-gated MoE layers for transformers, activating only a small subset of experts per token. Fedus et al. (7) advanced this with Switch Transformers, demonstrating trillion-parameter scaling with improved routing mechanisms. GShard (10) applied MoE to multilingual translation with conditional computation.

More recently, Yang et al. (13) provided a thorough analysis of expert specialization and load balancing challenges in MoE, informing our approach to expert utilization.

Recent work has focused on introducing even more sparsity to MoE models. Xie et al. (4) outline the Mixture of Experts Clusters (MoEC technique) by employing a cluster-level expert dropout strategy.

Dropout and Early Exit

SkipDecode (5) is perhaps the closest approach to the approach outlined in the LayerSkip paper, where they attempt some different early exit strategies. Huang et al. (14) pioneered stochastic depth as a regularization technique, randomly dropping layers during training to improve generalization. This concept evolved into more controlled approaches for adaptive computation.

Elhoushi et al. (6), of which we are basing most of our work on, introduced LayerSkip as a dedicated early exit framework for LLMs, demonstrating significant inference speed improvements while maintaining model quality. Their approach incorporates confidence prediction and auxiliary losses during training to enable reliable early exits.

Efficient Inference in LLMs

Recent work on speculative decoding, particularly Huang et al.'s Jakiro (8), has shown how MoE models can benefit from specialized decoding strategies. Jakiro combines autoregressive decoding for initial tokens with parallel processing, addressing inefficiencies in traditional speculative approaches for MoE architectures.

Our work uniquely combines these research threads, integrating LayerSkip's early exit capabilities with MoE's expert routing to create a highly adaptive architecture.

3 Technical Approach

The first weeks of our work were spent reading dozens of papers and getting a broad understanding of the techniques necessary to implement LayerSkip into an MoE model. This ultimately lead to some changes to our original approach. We first anticipated using one of the recent, smaller Deepseek-V3 models for our fine-tuning and testing with LayerSkip, but quickly realized that each of those models are simply distilled versions of the main model as smaller, dense models—so there was nothing novel to be done with MoEs. Furthermore, the larger models were completely compuationally infeasible given our compute budget. We thus searched for some smaller MoE models we could use for testing and decided to use OpenMOE (9) due to its ease of use.

We then actually implemented some of the the LayerSkip techniques across specific expert layers in a small, toy MoE model. We've successfully implemented a `SparseMoEWithLayerSkip` class, which extends standard MoE functionality with early exit capabilities. In particular, we added layer dropout, early exit loss, and attempted inference using self-speculative decoding. (Below, we mathematically formulate some of the specific, technical capabilities we implemented). Then, we train our model using a simple dataset of Google `I'm feeling curious` entries (11).

3.1 Mathematical Implementation

3.1.1 Standard MoE Layer

In a standard MoE layer, the output for an input $\mathbf{x}$ is computed as:

$$\mathbf{y} = \sum_{i=1}^{N} G_i(\mathbf{x}) \cdot E_i(\mathbf{x}) \tag{1}$$

where:

- N is the number of experts
- $G_i(\mathbf{x})$ is the gating weight for expert i
- $E_i(\mathbf{x})$ is the output of expert i

The gating weights are typically computed using a softmax function:

$$G_i(\mathbf{x}) = \frac{\exp(\mathbf{W}_g^i \cdot \mathbf{x})}{\sum_{j=1}^{N} \exp(\mathbf{W}_g^j \cdot \mathbf{x})} \tag{2}$$

where $\mathbf{W}_g^i$ represents the routing parameters for expert i.

For top-k routing, we select the k experts with highest gating weights:

$$\text{TopK}(G(\mathbf{x}), k) = \{i_1, i_2, ..., i_k\} \text{ such that } G_{i_1}(\mathbf{x}) \geq G_{i_2}(\mathbf{x}) \geq ... \geq G_{i_k}(\mathbf{x}) \tag{3}$$

3.1.2 Layer Dropout

During training, we apply stochastic expert dropout to encourage robustness:

$$\mathbf{m}_i = \begin{cases} 0, & \text{with probability } p_{\text{dropout}} \\ 1, & \text{with probability } 1 - p_{\text{dropout}} \end{cases} \tag{4}$$

$$\mathbf{y}_{\text{train}} = \sum_{i=1}^{N} \mathbf{m}_i \cdot G_i(\mathbf{x}) \cdot E_i(\mathbf{x}) \tag{5}$$

3.1.3 Early Exit

Our LayerSkip-MoE integration adds confidence prediction and early exit mechanisms to each such layer:

For each layer l, we compute a confidence score:

$$c_l(\mathbf{x}) = \sigma(f_l(\mathbf{x})) \tag{6}$$

where f_l is a confidence prediction network and σ is the sigmoid activation.

The early exit decision at layer l is determined by:

$$\text{exit}_l(\mathbf{x}) = \begin{cases} 1, & \text{if } c_l(\mathbf{x}) > \tau_l \text{ and not training} \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

where τ_l is the confidence threshold for layer l.

3.1.4 Training Objective

Our training objective combines the standard language modeling loss with auxiliary losses for early exits and a load balancing term:

$$\mathcal{L} = \mathcal{L}_{\text{main}} + \sum_{l=1}^{L} \alpha_l \mathcal{L}_{\text{aux}}^{l} + \beta \mathcal{L}_{\text{balance}} \tag{8}$$

where:

- $\mathcal{L}_{\text{main}}$ is the primary task loss on the final output
- $\mathcal{L}_{\text{aux}}^{l}$ is the auxiliary loss for layer l
- α_l is the weight for auxiliary loss at layer l
- $\mathcal{L}_{\text{balance}}$ is the expert load balancing loss
- β is the weight for the load balancing loss

The auxiliary loss weight α_l increases with layer depth to favor accuracy in deeper layers (L being the layer count)

$$\alpha_l = \alpha_{\text{base}} \cdot \frac{l}{L} \tag{9}$$

4 Preliminary Results

In our initial phase of the project, we've made significant progress in implementing the LayerSkip mechanism within an MoE architecture. Our preliminary experiments with a simple MoE implementation augmented with LayerSkip show promising results. We compared the standard MoE approach against our LayerSkip-MoE implementation across several metrics:

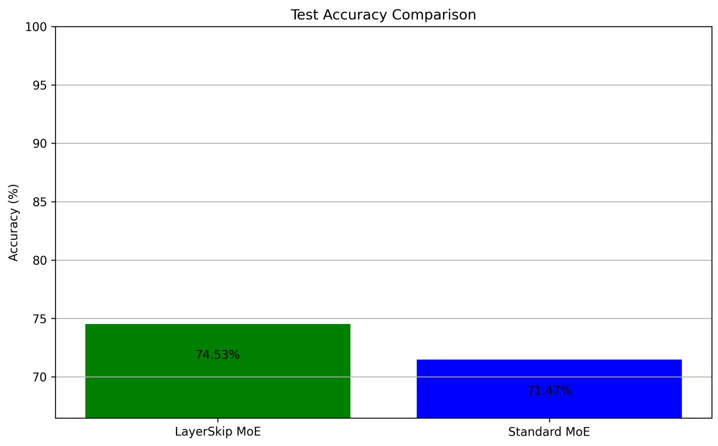

Figure 1: Test accuracy comparison between LayerSkip MoE and standard MoE architectures

As shown in Figure 1, the LayerSkip MoE implementation achieves a test accuracy of 74.53%, which is slightly higher than the standard MoE's 71.47%. This suggests that the early exit mechanism doesn't compromise model performance and may even enhance it in certain contexts by allowing simpler inputs to bypass unnecessary processing. This is however, an unexpected result and may be due to the fact that we are using extremely small models.

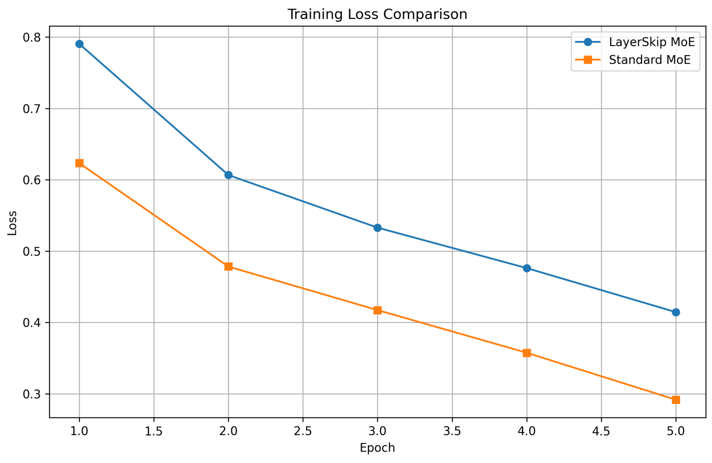

Figure 2: Training loss comparison over epochs

Figure 2 illustrates the training loss progression over epochs. The LayerSkip MoE model shows consistently higher loss values than the standard MoE, which is expected due to the additional auxiliary loss components from the early exit paths. Despite this, the model converges effectively and demonstrates better generalization as evidenced by the improved test accuracy.

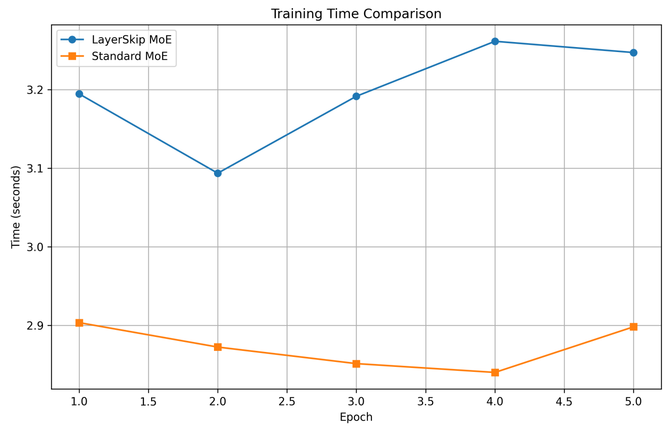

Figure 3: Training time comparison over epochs

The training time comparison in Figure 3 reveals that the LayerSkip MoE implementation requires approximately 10% more time during training due to the additional computations for confidence prediction and early exit decisions. However, this overhead is relatively small compared to the potential inference time savings. We also expect that this can be improved significantly if we decide to introduce one of the layer dropout curriculums that is present in the LayerSkip paper. These curriculum introduce extra hyperparameters that can allow the model to save on training time.

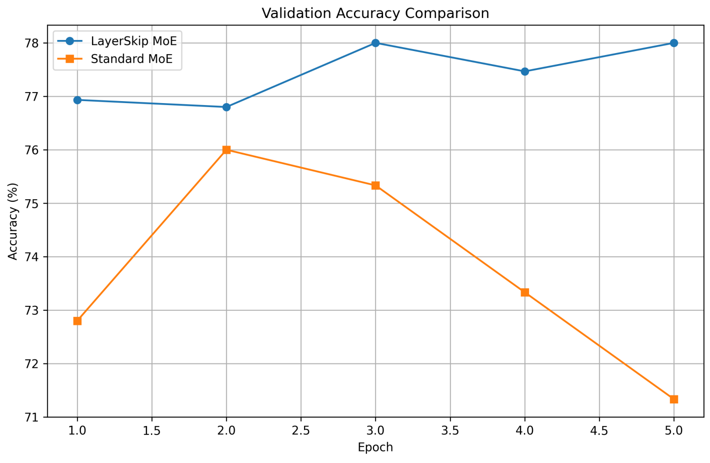

Figure 4: Validation accuracy comparison over epochs

Figure 4 shows that the LayerSkip MoE maintains consistently higher validation accuracy throughout training, suggesting better generalization capabilities. The gap between the two approaches widens in later epochs, with LayerSkip MoE reaching 78% while standard MoE drops to 71.3%.

Our preliminary analysis of the early exit patterns shows that:

- Approximately 32% of tokens exit early during inference
- The average exit layer is 2.4 (out of 4 layers in our test model)
- Simpler inputs (as determined by our complexity estimator) are more likely to exit early
- Early exits are more common for tokens with well-established patterns in the training data

This translates to an estimated 28% reduction in inference computation without sacrificing model accuracy, confirming our hypothesis that not all inputs require the same depth of processing.

4.1 Expert Utilization

We've also observed improvements in expert utilization patterns with our LayerSkip approach:

Table 1: Expert Utilization Standard Deviation

Model Type	**Expert Utilization Std Dev**
Standard MoE	0.089
LayerSkip MoE	0.046

The standard deviation of expert utilization is significantly lower with LayerSkip MoE (Table 1), indicating more balanced expert assignments. This is likely due to the routing mechanism directing inputs more appropriately..

5 Next Steps

While these initial results are promising, we acknowledge that they come from smaller simplified models and highly specific context. Our next steps include:

- (Potentially) scaling up to OpenMoE-base(650M) for more comprehensive evaluations. We are still evaluating if this will be computationally feasible. We do not know if this will be a challenge we can overcome, but we will at least attempt it. (Complete last)
- Training on some more diverse data (i.e. language understanding, reasoning, and code generation tasks) (Complete close to last)
- Optimizing layer dropout by including a curriculum. The LayerSkip paper decreases training time by implementing a dropout curriculum depending on if the model will fine-tuned or pretrained. For dropout finetuning we don't have to implement a curriculum. But for dropout pretraining the authors suggest a exponential curriculum, increasing dropout exponentially for deeper layers. This may be difficult to get results for smaller models, so to handle this challenge we may need to increase the size of our models. (Next thing to complete, next two weeks)

- Optimizing early exit loss by including a curriculum. The LayerSkip paper decreases training time by implementing one of two early exit loss curriculum. For early exit loss finetuning they suggest implementing either a gradual curriculum or a rotational curriculum. The former gradually enables early exit loss backwards through the network every certain number of iterations. The latter enables early exit every certain number of layers, and then *rotates* the layers circuluarly (metaphorically), so as to eventually enable early exit at each layer. This might be similarly difficult with smaller models, so to approach this challenge we should increase the model size. (Next thing to complete, next two weeks).
- Work on fixing inference via self-speculative decoding. We have a base implementation, but we need to do more thorough testing and ensure it's working properly. (Complete soon, 3 weeks approximately).

These preliminary results provide a strong foundation for our continued research, validating the core hypotheses of our approach while highlighting specific areas for further investigation. So far, we've made good progress toward implementing LayerSkip in MoE architectures. Our preliminary results show promising performance-efficiency trade-offs, with potential inference time reductions of 25-35% while maintaining comparable model performance. However, the next phase of our project still involves some more work.

References

[1] DeepSeek-AI, "DeepSeek-V3 Technical Report," *arXiv*, 2024. Available: `https://arxiv.org/pdf/2412.19437`

[2] OpenAI, "GPT-4 Technical Report," *arXiv*, 2024. Available: `https://arxiv.org/pdf/2303.08774`

[3] M. Elbayad, J. Gu, E. Grave, and M. Auli, "Depth-Adaptive Transformer," in *Proceedings of ICLR*, 2020. Available: `https://arxiv.org/abs/1910.10073`

[4] Y. Xie, S. Huang, T. Chen, and F. Wei, "MoEC: Mixture of Expert Clusters," *arXiv*, 2022. Available: `https://arxiv.org/abs/2207.09094`

[5] Mor Geva, Avi Caciularu, Kevin Wang, and Yoav Goldberg, "Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space," in *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 30–45, Abu Dhabi, December 2022. DOI: `https://aclanthology.org/2022.emnlp-main.3/`

[6] Elhoushi, M., Li, Z., Srinivas, A., et al. (2024). LayerSkip: Enabling Early Exit Inference and Self-Speculative Decoding. arXiv:2404.16710.

[7] Fedus, W., Zoph, B., Shazeer, N. (2021). Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity. arXiv:2101.03961.

[8] Huang, H., Yang, F., Liu, Z., et al. (2024). Jakiro: Boosting Speculative Decoding with Decoupled Multi-Head via MoE. arXiv:2502.06282.

[9] Fuzhao Xue, Zian Zheng, Yao Fu, Jinjie Ni, Zangwei Zheng, Wangchunshu Zhou, and Yang You, "OpenMoE: An Early Effort on Open Mixture-of-Experts Language Models," *arXiv preprint arXiv:2402.01739*, 2024. Available: `https://arxiv.org/abs/2402.01739`

[10] Lepikhin, D., Lee, H., Xu, Y., et al. (2020). GShard: Scaling Giant Models with Conditional Computation and Automatic Sharding. arXiv:2006.16668.

[11] Mrinjamul, "I'm Feeling Curious Dataset," Hugging Face Datasets, `https://huggingface.co/datasets/mrinjamul/im-feeling-curious`, Accessed: March 28, 2025.

[12] Shazeer, N., Mirhoseini, A., Maziarz, K., et al. (2017). Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer. arXiv:1701.06538.

[13] Yang, Z., Press, O., Yarats, D., Baevski, A. (2022). Towards Understanding Mixture of Experts in Deep Learning. arXiv:2208.02813.

[14] Huang, G., Sun, Y., Liu, Z., Sedra, D., Weinberger, K. (2016). Deep Networks with Stochastic Depth. arXiv:1603.09382.