From Dense to Dynamic: Token-Difficulty Driven MoEfication of Pre-Trained LLMs

Anonymous ACL submission

Abstract

Training large language models (LLMs) for different inference constraints is computationally expensive, limiting control over efficiency-accuracy trade-offs. Moreover, once trained, these models typically process tokens uniformly, regardless of their complexity, leading to static and inflexible behavior. In this paper, we introduce a post-training optimization framework, DynaMoE, that adapts a pre-trained dense LLM to a token-difficulty-driven Mixture-of-Experts model with minimal fine-tuning cost. This adaptation makes the model dynamic, with sensitivity control to customize the balance between efficiency and accuracy. DynaMoE features a token-difficulty-aware router that predicts the difficulty of tokens and directs them to the appropriate sub-networks or experts, enabling larger experts to handle more complex tokens and smaller experts to process simpler ones. Our experiments demonstrate that DynaMoE can generate a range of adaptive model variants of the existing trained LLM with a single fine-tuning step, utilizing only $10B$ tokens, a minimal cost compared to the base model's training. Each variant offers distinct trade-offs between accuracy and performance. Compared to the baseline post-training optimization framework, Flextron, our method achieves similar aggregated accuracy across downstream tasks, despite using only $\frac{1}{9}$th of their fine-tuning cost.

1 Introduction

Large language models (LLMs) have significantly advanced the field of natural language processing, showcasing strong capabilities in addressing complex tasks (Brown et al., 2020; Touvron et al., 2023a; Wei et al., 2022). However, their large size presents challenges, particularly in terms of high memory and computational demands, which can limit their deployment in resource-constrained settings. To address this, LLMs must be optimized for specific memory and computational constraints (Touvron et al., 2023b). However, designing multi-billion-parameter models for every use case is not cost-effective, as it demands substantial training time, data, and resources.

Some prior works have focused on adapting large LLMs for resource-constrained use cases by distilling knowledge from larger models into smaller ones (Hsieh et al., 2023) or pruning model parameters to reduce computational demands (Sun et al., 2024). While these methods effectively enable the use of large LLMs in low-resource scenarios, they often lead to performance degradation and require careful balancing between efficiency and accuracy. Alternatively, other approaches have investigated many-in-one LLM designs, MatFormer (Devvrit et al., 2023) and SortedNet (Valipour et al., 2024), to employ multiple sub-networks within a single model to accommodate different computational budgets. These architectures use nested structures integrated into the standard LLM framework. However, they require non-standard methodologies and significantly longer, more resource-intensive training processes, which can offset the intended efficiency benefits.

Mixture-of-Experts (MoE) models (Shazeer et al., 2017; Du et al., 2021; Fedus et al., 2022; Zoph et al., 2022; He, 2024) have emerged as a promising alternative to dense models, offering improved efficiency by sparsely activating select sub-modules or experts. This selective activation enables MoEs to achieve high performance while using fewer computational resources during inference. However, training MoEs from scratch remains resource-intensive and each expert becomes static, often requiring fixed compute budget irrespective of the input complexity.

Flextron (Cai et al., 2024) explored a post-training methodology by integrating the MoE concept into a nested elastic structure within the MLP layers, creating heterogeneous experts of different sizes, selected by a router conditioned on the

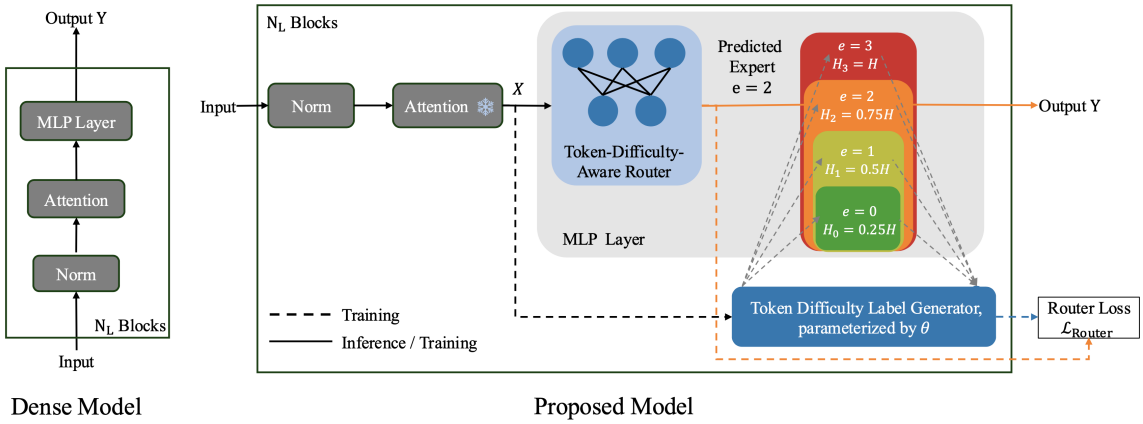


Figure 1: Overview of our proposed post-training optimization framework, DynaMoE. The left part represents the base pre-trained LLM, while the right part shows the adapted DynaMoE model.

input data. However, the lack of supervision in the router training leads to sub-optimal input complexity adaptation. Furthermore, the router lacks a parameter to customize its sensitivity to token complexity, limiting its flexibility and performance in handling diverse use-cases. Salehi et al. (2023) proposed an input-adaptive approach that predicts the difficulty of input data and dynamically adjusts the network's width accordingly. In the absence of ground-truth difficulty labels, they relied on heuristic methods for label generation, which may limit precision and consistency in difficulty estimation.

To address their shortcomings, we introduce DynaMoE, a post-training optimization framework designed to transform a dense LLM into a token-difficulty-driven MoE model. DynaMoE leverages the insight that not all tokens require the full capacity of a model's weights. For example, in the sentence "Geoffrey did his PhD at the university of Edinburgh", simpler tokens like "at the university of" are predictable using prior context, while more complex tokens like "Edinburgh" demand broader contextual understanding. To maximize efficiency, DynaMoE selectively activates nested sub-components of the MLP, referred as experts, based on the predicted difficulty of each token. To this end, we make the following contributions:

- The framework includes a novel token-difficulty-aware router, trained to predict token hardness and assign it to the appropriate expert dynamically.
- Due to the lack of ground truth notion of hardness, we introduce a method to derive token difficulty labels which serve as supervision signals for training the router. This approach allows a token to have varying difficulty labels across different layers.
- A simplified post-training optimization framework that efficiently adapts a pre-trained dense LLM into a token-difficulty-driven MoE model, featuring a sensitivity parameter to customize the efficiency vs accuracy trade-off.

2 Method

In this section, we describe our proposed post-training optimization framework, DynaMoE, which transforms a dense LLM into an MoE model for adaptive inference based on token difficulty. The process involves three key steps: (1) defining heterogeneous experts by splitting the MLP layers of the dense LLM; (2) generating token labels during training to represent token difficulty; and (3) training a router to predict token difficulty while fine-tuning the model. We detail these steps in the below sub-sections.

2.1 Defining Heterogeneous Experts

In this work, we focus on defining experts into the MLP layers of the LLM (Devvrit et al., 2023), as these layers account for the majority of the compute and operate on a token-by-token basis. The overview of DynaMoE is depicted in Fig. 1. The left part of the figure denotes the base pre-trained model which consists of the normalization layers, attention layers and the MLP layers in each transformer block. The right part shows the adapted DynaMoE model, where the original single MLP layer is transformed into multiple nested FFN blocks or experts. Such expert formation intro-

duces no additional parameters to the base model, aside from the router. This design draws inspiration from adaptive width reduction in transformer (Salehi et al., 2023) and recent works like Matformer (Devvrit et al., 2023) and Flextron (Cai et al., 2024). The attention layers remain frozen, and the MLP layers adapted to nested experts are fine-tuned in DynaMoE.

Let D and H denote the embedding and the hidden dimensions of the MLP layer respectively. The input to the MLP layer is $\boldsymbol{X} \in \mathbb{R}^{B\times D}$ and the output is $\boldsymbol{Y} \in \mathbb{R}^{B\times D}$, where B is the batch dimension. The MLP layer with two fully connected layers is represented by weight matrices $\boldsymbol{W}^{(IN)} \in \mathbb{R}^{H\times D}$ and $\boldsymbol{W}^{(OUT)} \in \mathbb{R}^{D\times H}$. In order to get best results, we first rearrange these fully-connected layers, $\boldsymbol{W}^{(IN)}$ and $\boldsymbol{W}^{(OUT)}$, to have the most important rows/columns in the beginning of the matrix so that they can be included in all of the experts (Samragh et al., 2023). There are a total of E experts indexed using $e \in \{0, 1, \ldots, E-1\}$. Each expert gets a portion H_e of the weight matrices $\boldsymbol{W}^{(IN)}$ and $\boldsymbol{W}^{(OUT)}$, sliced over the hidden dimension H. The value H_e is obtained as a fraction of H as,

$$H_e = \left\lfloor \left(\frac{e+1}{E}\right) \cdot H \right\rfloor, \tag{1}$$

consequently, $H_0 < H_1 < \cdots < H_{E-1}$ and $H_{E-1} = H$. Note that the expert with index $E-1$ utilizes the full MLP layer. The restriction of the matrices $\boldsymbol{W}^{(IN)}$ and $\boldsymbol{W}^{(OUT)}$ to the expert width H_e is obtained using the slicing operator that selects the first H_e rows and columns respectively as

$$\boldsymbol{W}_e^{(IN)} = \boldsymbol{W}^{(IN)}[0 : H_e, :], \tag{2}$$

$$\boldsymbol{W}_e^{(OUT)} = \boldsymbol{W}^{(OUT)}[:, 0 : H_e]. \tag{3}$$

With σ as the activation function, the output $\boldsymbol{Y}_e$ of the MLP layer corresponding to the expert e can thus be obtained as,

$$\boldsymbol{Y}_e = \sigma\left(\boldsymbol{X} \cdot \left(\boldsymbol{W}_e^{(IN)}\right)^T\right) \cdot \left(\boldsymbol{W}_e^{(OUT)}\right)^T. \tag{4}$$

2.2 Generating Token Difficulty Label

We aim to train a token-difficulty-aware router to dynamically assign tokens to an appropriate expert. But there is no ground-truth label denoting token difficulty to train such a router. To this end, we propose a method to estimate the token difficulty and generate a derived-ground-truth difficulty label during training. This is shown as "Token Difficulty Label Generator" in Fig. 1.

First, we pass the input to all experts and generate the output $\boldsymbol{Y}_e$ for each $e \in [E]$. Then, for each token $b \in [B]$ and each expert $e \in [E]$, we compute a similarity score $S_{b,e}$ that measures how similar is the output of the expert e compared to the output of the full MLP layer $e = E-1$ for that token. We calculate this similarity as,

$$S_{b,e} = \frac{\langle \boldsymbol{Y}_e[b, :], \boldsymbol{Y}_{E-1}[b, :] \rangle}{\langle \boldsymbol{Y}_{E-1}[b, :], \boldsymbol{Y}_{E-1}[b, :] \rangle}. \tag{5}$$

Here, $\langle \cdot, \cdot \rangle$ denotes the dot-product between two vectors. We use dot-product in this calculation as it accounts for both the magnitude and the direction of the tensors being compared.

Finally, we generate a derived ground-truth hardness label l_b, representing the target expert index for token b. Given a threshold θ, we assign l_b as the smallest expert index e satisfying $S_{b,e} > \theta$, that is, $l_b = \min\{e \in [E] \mid S_{b,e} > \theta\}$. We say that a token is easier if it has a smaller label l_b, that is the similarity score for a smaller expert is higher than threshold θ. In such cases, processing the token with the smaller expert incurs less compute without much compromise in the accuracy. During the forward pass of fine-tuning of DynaMoE, we generate the token difficulty labels. These labels are then used in the backward pass to compute the router loss, which trains the router.

2.3 Training a Token-Difficulty-Aware Router

The output of a router is in $\mathbb{R}^{B\times E}$, denoting logits over the E experts. Each router is parameterized by two linear layers, projecting the token embedding from dimension D to U and subsequently to E. In our experiments, we use $U = 256$, resulting in total parameters added by the routers across all layers to $33.6M$, which is only 0.51% of the base model size.

We train the router using the derived labels from Section 2.2. The objective is to learn the expert prediction using the derived-ground-truth labels to mimic token assignment based on their complexity and need. Hence, we impose the cross-entropy loss on the router to guide to this behavior and call it as the router loss. The overall objective function of DynaMoE is given as,

$$\mathcal{L} = \lambda_{LLM} \cdot \mathcal{L}_{LLM} + \lambda_{Router} \cdot \mathcal{L}_{Router}. \tag{6}$$

Here, $\mathcal{L}_{LLM}$ is the main LLM Cross-entropy loss and $\mathcal{L}_{Router}$ is the router loss. λ_{LLM} and λ_{Router}

	Cost (#Tokens)	Params	ARC-e	LAMBADA	PIQA	WinoGrande	Avg4	SciQ	HellaSwag	ARC-c	Avg7
Base Mistral 7B	-	7B	80.2	75.1	80.8	75.5	**77.8**	96.4	61.4	50.5	74.2
DynaMoE $\theta = 0.9$	**10B**	6B	75.0	71.0	78.3	71.8	**74.0**	95.2	57.9	41.5	70.1
DynaMoE $\theta = 0.8$	**10B**	5.1B	69.9	68.0	78.0	66.0	**70.5**	94.2	54.5	35.2	66.5
DynaMoE $\theta = 0.7$	**10B**	4.6B	66.0	65.9	75.4	63.4	67.7	93.6	52.1	31.7	64.0
Base Llama2-7B †	-	6.5B	75.1	71.5	77.5	69.1	73.3				
Flextron †	93.57B	4.1B	68.6	65.1	76.1	63.7	68.3				

Table 1: Evaluation of DynaMoE models with different sensitivity factor θ on downstream tasks, using 0-shot non-normalized accuracy metric. Our base model is Mistral 7B (Jiang et al., 2023). (†): results from Flextron (Cai et al., 2024) used as our baseline. *Params* denotes the average number of total activated parameters, aggregated over the downstream tasks. *Avg4* averages over *ARC-e, LAMBDA, PIQA, WinoGrande*, while *Avg7* averages over all tasks.

are hyper-parameters, denoting the weights of the respective losses.

3 Experimental Set up

3.1 Model and Dataset

Model: DynaMoE provides a simplified post-training approach to convert any dense LLM to an MoE model with a tunable sensitivity factor θ, the similarity threshold, to achieve desired latency reduction and the tolerance for drop in accuracy. DynaMoE integrates seamlessly with any transformer model, regardless of the architecture. To showcase the effectiveness of the method, we use Mistral 7B model (Jiang et al., 2023), a widely-used open-source pre-trained language model, as the base model.

Dataset: For DynaMoE fine-tuning, we use a small subset (10B tokens) of the Falcon RefinedWeb dataset (Penedo et al., 2023). Falcon RefinedWeb is an open-source dataset which contains high-quality web data. This minimal fine-tuning overhead enables a cost-effective conversion of any pre-trained LLM into an MoE variant for faster inference.

3.2 Training Details

We first reorder the pre-trained weight matrices (Samragh et al., 2023) in the MLP blocks before fine-tuning DynaMoE so that the most important weights can be included in all the experts. We run the dense LLM on a subset of the data and collect the absolute activations from the MLP layer, $Y \in R^{B,H}$. This subset can be a very small portion of the training data, 0.004% tokens of the Falcon RefinedWeb in our case. Subsequently, we aggregate the activations along the batch dimension and across all the subset samples to obtain an importance score for each neuron in the hidden dimension. Using this importance score, we sort the MLP matrices.

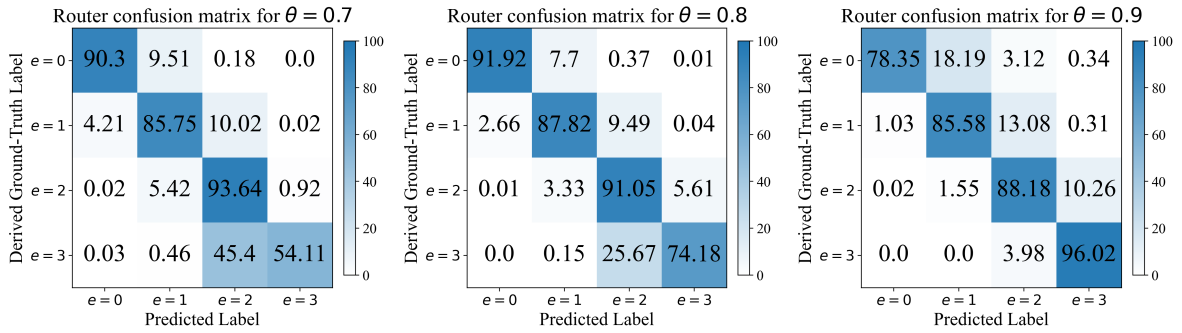
Next, we fine-tune the DynaMoE model using only $10B$ tokens with AdamW optimizer (Loshchilov and Hutter, 2017) and a fixed learning rate of 10^{-5}. We keep the attention layers frozen. We set λ_{LLM} to 0.2 and λ_{Router} to 1 in Equation (6), the objective function for fine-tuning. We experiment with different values of threshold $\theta \in \{0.7, 0.8, 0.9\}$ to build a family of DynaMoE models with varying sensitivity parameter. A low sensitivity parameter, that is a smaller value of θ, makes the system less reactive, favoring smaller experts for most tokens and only escalating to bigger experts for significantly complex tokens. And a high sensitivity parameter makes the system more reactive, escalating to bigger experts even for moderately complex tokens. We use 4 experts ($E = 4$) with sizes $0.25H$, $0.5H$, $0.75H$, and H respectively. We denote the size of expert with index e as H_e.

4 Results

4.1 Evaluation

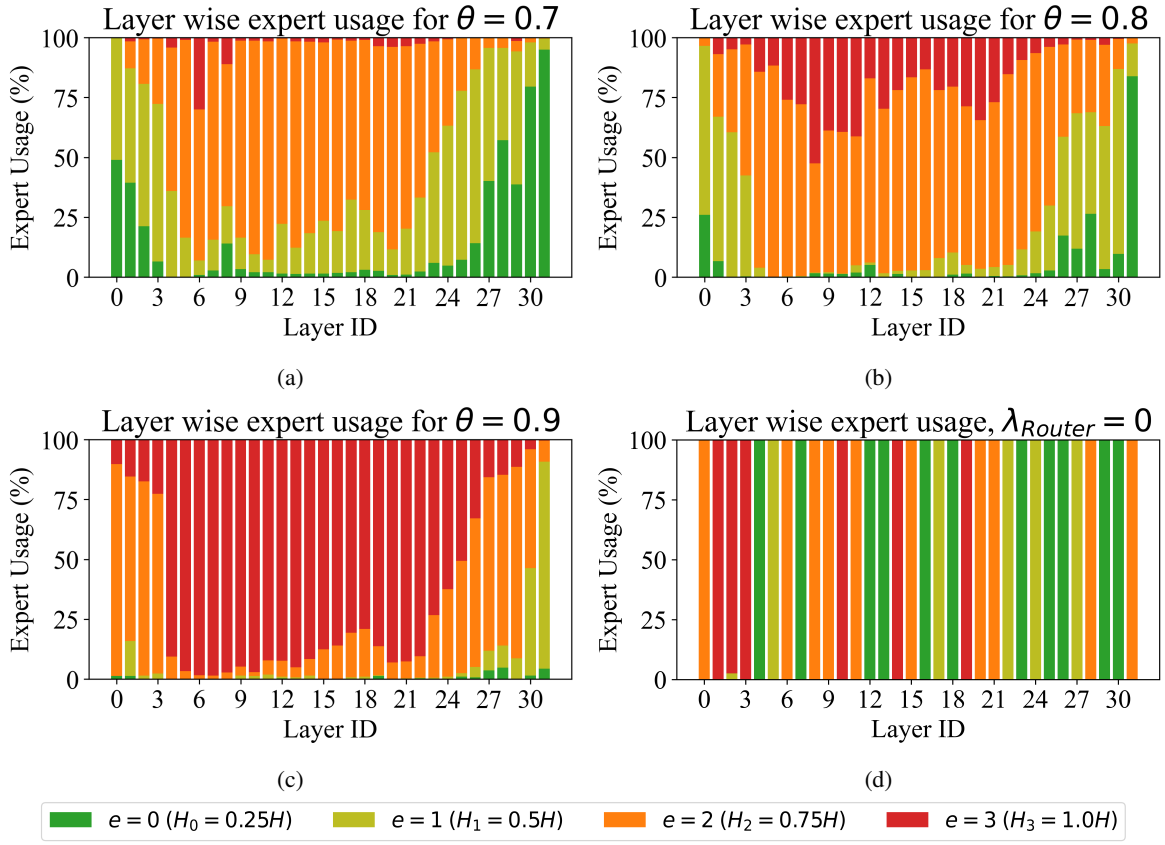
We evaluate the DynaMoE models on 7 downstream tasks using LM Evaluation Harness (Gao et al., 2024) and report the 0-shot non-normalized accuracy metric in Table 1. The selected evaluation tasks include ARC (Easy and Challenge) (Clark et al., 2018), HellaSwag (Zellers et al., 2019), PIQA (Bisk et al., 2019), SciQ (Welbl et al., 2017), WinoGrande (Sakaguchi et al., 2019), and LAMBADA (Paperno et al., 2016).

DynaMoE is compared to two baselines, the



(a) (b) (c)

Figure 2: Confusion matrix for the router's classification task in DynaMoE. The strong diagonal pattern in the matrices reflects high classification accuracy. The misclassified tokens often occurred in neighboring expert classes.



(a) (b) (c) (d)

Figure 3: Layer-wise expert usage pattern of DynaMoE models with varying θ, aggregated across 7 downstream tasks. Each layer uses experts of varying sizes depending on the inputs. The sensitivity parameter θ regulates how readily tokens are routed to larger experts based on their difficulty: a lower θ favors smaller experts, while a higher θ prioritizes larger experts. 3d shows the case when DynaMoE is trained without the router loss λ_{Router}. In this case, the model tends to assign a specific expert per layer instead of dynamically selecting experts based on token difficulty.

base Mistral 7B model and the Flextron model (Cai et al., 2024) using Avg4 and Avg7 in Table 1. Compared to Mistral 7B, DynaMoE with $\theta = 0.8$ improves efficiency by activating only $5.1B$ of $7B$ parameters on average, with an 7.3 point accuracy drop after fine-tuning on only $10B$ tokens on the downstream tasks. The number of activated parameters adapts dynamically to token difficulty. For reference, Flextron fine-tunes on $93.57B$ tokens, activating $4.1B$ of $6.5B$ parameters, with a 5 point accuracy drop from its base model, Llama2-7B (Touvron et al., 2023a). We emphasize that with only $\frac{1}{9}$th of the Flextron's fine-tuning cost, our results for DynaMoE with $\theta = 0.7$ are com-

parable to Flextron. Accuracy improves with increase in fine-tuning cost, but to keep the adaption lightweight, we opt for a smaller cost.

4.2 Analysis of Token-Difficulty-Aware Router

We assess the performance of the Token-Difficulty-Aware router by gathering its predictions from all layers across the 7 downstream tasks outlined in Section 4.1. These predictions are then compared to the ground truth labels derived in Section 2.2. Using both sets of labels, we compute the router's overall classification accuracy.

We present the confusion matrices for the router's classification tasks across all DynaMoE models in Fig. 2. Notably, the matrices exhibit a strong diagonal pattern, indicating high classification accuracy. Furthermore, when the router misclassifies tokens, the errors predominantly occurred in neighboring expert classes, underscoring the router's effectiveness in distinguishing token difficulty levels.

4.3 Experts usage analysis

We visualize the expert usage patterns across all layers in Fig. 3. For each model, the Y-axis represents the percentage of tokens routed to a specific expert, while the X-axis indicates the layer index. Notably, a token's perceived difficulty may vary across layers, hence it can be routed to different experts as the token progresses through the model. The visualization shows that all experts are utilized in varying proportions across layers, reflecting an aggregated behavior over the 7 tasks. However, during inference, the model adapts to the data, with simpler queries predominantly engaging lower-compute experts to maximize efficiency.

The parameter θ affects expert usage in DynaMoE models by controlling how quickly tokens are routed to larger experts based on difficulty. At lower θ values (e.g., $\theta = 0.7$), smaller experts ($e = 2$) dominate across layers, optimizing for efficiency. In contrast, at higher θ values (e.g., $\theta = 0.9$), larger experts ($e = 3$) are utilized more frequently, prioritizing accuracy over efficiency. This shift demonstrates θ's role in balancing computational resource allocation and prediction accuracy.

Ablating the Router Loss: To examine the router loss's role in expert allocation, we train a DynaMoE model without it, relying solely on the LLM loss to train the router. Tokens are routed using the router's predicted expert indices without explicit difficulty supervision. The resulting expert usage pattern, shown in Fig. 3d, reveals that the model converges to using specific experts per layer instead of dynamically allocating experts based on token difficulty. In contrast, when router loss is applied, expert usage adapts dynamically to token difficulty across layers.

5 Conclusion

We present DynaMoE, a post-training optimization framework that converts a standard pre-trained dense LLM into a token-difficulty-driven MoE model. DynaMoE incorporates a lightweight router to predict the token difficulty and routes them to an appropriate expert. To train this router, we propose a novel method to derive the token difficulty labels, which act as supervision signals. DynaMoE generates adaptive model variants with sensitivity control, allowing customization of the trade-off between efficiency and accuracy.

Limitations

While the proposed post-training optimization framework, DynaMoE, and the token-difficulty-aware router provide a general-purpose approach for token-difficulty adaptive processing, this work does not explore their application to pre-trained heterogeneous mixture of expert (HMoE) models (Wang et al., 2024). Such models, with their built-in experts of varying capabilities, could benefit from the integration of DynaMoE, which provides a direct mechanism for routing tokens to appropriate experts based on token difficulty. Incorporating DynaMoE with HMoE models offers a promising direction for future exploration. Additionally, while DynaMoE makes the base model input-adaptive, offering inference efficiency, this work evaluates efficiency only in terms of the number of active parameters and does not include the efficiency measured in a real deployment scenario.

References

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2019. Piqa: Reasoning about physical commonsense in natural language. In *AAAI Conference on Artificial Intelligence*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss,

Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Ma teusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *ArXiv*, abs/2005.14165.

Ruisi Cai, Saurav Muralidharan, Greg Heinrich, Hongxu Yin, Zhangyang Wang, Jan Kautz, and Pavlo Molchanov. 2024. Flextron: Many-in-one flexible large language model. *ArXiv*, abs/2406.10260.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *ArXiv*, abs/1803.05457.

Devvrit, Sneha Kudugunta, Aditya Kusupati, Tim Dettmers, Kaifeng Chen, Inderjit S. Dhillon, Yulia Tsvetkov, Hannaneh Hajishirzi, Sham M. Kakade, Ali Farhadi, and Prateek Jain. 2023. Matformer: Nested transformer for elastic inference. *ArXiv*, abs/2310.07707.

Nan Du, Yanping Huang, Andrew M. Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, Barret Zoph, Liam Fedus, Maarten Bosma, Zongwei Zhou, Tao Wang, Yu Emma Wang, Kellie Webster, Marie Pellat, Kevin Robinson, Kathleen S. Meier-Hellstern, Toju Duke, Lucas Dixon, Kun Zhang, Quoc V. Le, Yonghui Wu, Z. Chen, and Claire Cui. 2021. Glam: Efficient scaling of language models with mixture-of-experts. *ArXiv*, abs/2112.06905.

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: scaling to trillion parameter models with simple and efficient sparsity. *J. Mach. Learn. Res.*, 23(1).

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2024. A framework for few-shot language model evaluation.

Xu Owen He. 2024. Mixture of a million experts. *arXiv preprint arXiv:2407.04153*.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. 2023. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8003–8017, Toronto, Canada. Association for Computational Linguistics.

Albert Qiaochu Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, L'elio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *ArXiv*, abs/2310.06825.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and R. Fernández. 2016. The lambada dataset: Word prediction requiring a broad discourse context. *ArXiv*, abs/1606.06031.

Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. 2023. The RefinedWeb dataset for Falcon LLM: outperforming curated corpora with web data, and web data only. *arXiv preprint arXiv:2306.01116*.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. An adversarial winograd schema challenge at scale.

Mohammadreza Salehi, Sachin Mehta, Aditya Kusupati, Ali Farhadi, and Hannaneh Hajishirzi. 2023. Sharcs: Efficient transformers through routing with dynamic width sub-networks. In *Conference on Empirical Methods in Natural Language Processing*.

Mohammad Samragh, Mehrdad Farajtabar, Sachin Mehta, Raviteja Vemulapalli, Fartash Faghri, Devang Naik, Oncel Tuzel, and Mohammad Rastegari. 2023. Weight subcloning: direct initialization of transformers using larger pretrained ones. *ArXiv*, abs/2312.09299.

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

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. 2024. A simple and effective pruning approach for large language models. In *The Twelfth International Conference on Learning Representations*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. Llama: Open and efficient foundation language models. *ArXiv*, abs/2302.13971.

Hugo Touvron, Louis Martin, Kevin R. Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Daniel M. Bikel, Lukas Blecher, Cristian Cantón Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony S. Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel M. Kloumann, A. V. Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, R. Subramanian, Xia Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zhengxu Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023b. Llama 2: Open foundation and fine-tuned chat models. *ArXiv*, abs/2307.09288.

Mojtaba Valipour, Mehdi Rezagholizadeh, Hossein Rajabzadeh, Parsa Kavehzadeh, Marzieh Tahaei, Boxing Chen, and Ali Ghodsi. 2024. Sortednet: A scalable and generalized framework for training modular deep neural networks. *Preprint*, arXiv:2309.00255.

An Wang, Xingwu Sun, Ruobing Xie, Shuaipeng Li, Jiaqi Zhu, Zhen Yang, Pinxue Zhao, J. N. Han, Zhanhui Kang, Di Wang, Naoaki Okazaki, Cheng zhong Xu, and Tencent Hunyuan. 2024. Hmoe: Heterogeneous mixture of experts for language modeling. *ArXiv*, abs/2408.10681.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed H. Chi, F. Xia, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *ArXiv*, abs/2201.11903.

Johannes Welbl, Nelson F. Liu, and Matt Gardner. 2017. Crowdsourcing multiple choice science questions. *ArXiv*, abs/1707.06209.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Annual Meeting of the Association for Computational Linguistics*.

Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. 2022. ST-MoE: designing stable and transferable sparse expert models. *arXiv preprint arXiv:2202.08906*.