
UMoE: Unifying Attention and FFN with Shared Experts

Yuanhang Yang¹, Chaozheng Wang², Jing Li³

¹Institute of Science Tokyo, Tokyo, Japan

²The Chinese University of Hong Kong, Hong Kong, China

³Hong Kong Polytechnic University, Hong Kong, China

yang.y.ea2c@m.isct.ac.jp

czwang23@cse.cuhk.edu.hk

jing-amelia.li@polyu.edu.hk

Abstract

Sparse Mixture of Experts (MoE) architectures have emerged as a promising approach for scaling Transformer models. While initial works primarily incorporated MoE into feed-forward network (FFN) layers, recent studies have explored extending the MoE paradigm to attention layers to enhance model performance. However, existing attention-based MoE layers require specialized implementations and demonstrate suboptimal performance compared to their FFN-based counterparts. In this paper, we aim to unify MoE designs in attention and FFN layers by introducing a novel reformulation of the attention mechanism, that reveals an underlying FFN-like structure within attention modules. Our proposed architecture, UMoE, achieves superior performance through attention-based MoE layers while enabling efficient parameter sharing between FFN and attention components.

1 Introduction

Scaling plays a crucial role in advancing the capabilities of large language models [1, 2, 3]. However, this scaling advantage comes with substantial computational costs, making continued scaling increasingly impractical. Sparse Mixture-of-Experts (MoE) architectures have emerged as a promising solution by selectively activating only a subset of model parameters—termed experts—for each input [4, 5, 6, 7]. This approach effectively decouples model size from computational cost, enabling efficient scaling with minimal overhead.

Recent work has demonstrated the effectiveness of MoE in Transformer architectures [8, 9, 10, 5, 11], particularly when applied to feed-forward neural network (FFN) layers. Building on this success, several studies have explored extending MoE to attention layers [12, 13, 14], indicating potential for performance gains through attention scaling. Despite the potential, we find that existing MoE attention layers demonstrate suboptimal performance compared to FFN-MoE approaches, when provided with similar computational and parametric budgets. This performance gap challenges the practical utility of attention-MoE architectures, as parameters allocated to scaling attention layers might be more effectively utilized for scaling FFNs instead.

We identify two distinctions between attention-MoE and FFN-MoE implementations that likely account for the observed performance differential: (1) the different expert design between attention and FFN layers, and (2) attention-MoE’s necessity to compromise the expressiveness of vanilla attention mechanisms to accommodate sparse computation [12]. Motivated by these observations, we investigate a compelling question: can we reformulate attention to reveal an underlying structure compatible with the same expert design as FFN layers, without compromising the expressive power

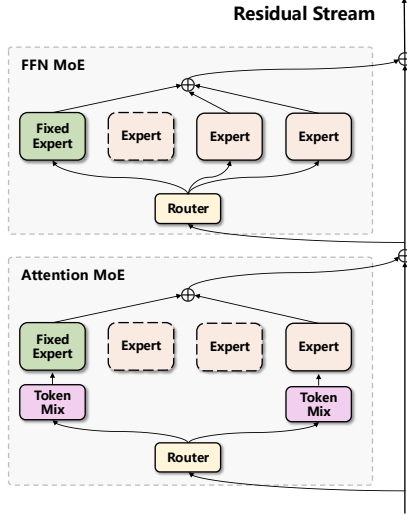


Figure 1: Illustration of a UMoE layer, which incorporates MoE into both FFN and attention modules with shared experts. The primary distinction between attention-MoE and FFN-MoE lies in an additional token mixing operation.

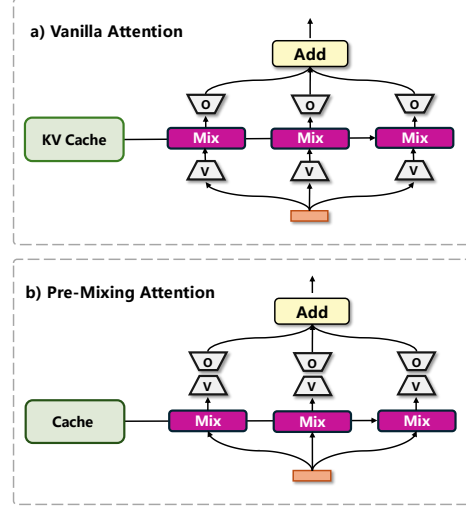


Figure 2: Two formulations of the multi-head attention mechanism. (a) Vanilla attention interleaves mixing operations with value and output projections. (b) Pre-mixing attention performs token mixing prior to projections.

of the attention mechanism? This is a challenging question due to the inherent complexity of attention mechanisms, including multiple projections and softmax calculations, which fundamentally differ from the straightforward two-matrix multiplication pattern of FFNs.

To bridge this structural gap, we reformulate the attention mechanism to reveal its underlying FFN-like structure. Our reformulation decomposes attention into two sequential operations: token mixing and token-wise expert processing. The token-wise expert processing, consisting of two consecutive matrix multiplications, can be implemented as an FFN with a small intermediate size. This implementation naturally aligns with recent advances in fine-grained FFN expert design [6, 15, 3], enabling unified expert architectures and parameter sharing across both attention and FFN layers.

Based on this insight, we introduce UMoE, a unified MoE architecture that abstracts Transformer layers into three fundamental components: *experts*, *token mixing operations*, and *routers*, as shown in Fig. 1. The experts, implemented as standard two-layer FFNs, serve as the primary components for token processing and knowledge storage. The token mixing operations facilitate contextual information exchange through weighted summation of tokens. Routers are employed to dynamically dispatch tokens to the most relevant experts to enable sparse computation. In UMoE, the distinction between the FFN and attention layers lies solely in the expert inputs: FFN layers process tokens independently, while attention layers process tokens simultaneously through weighted summation. This unified design not only simplifies the architecture but also enables parameter-efficient scaling through expert sharing between attention and FFN components.

To evaluate the effectiveness of UMoE, we conduct extensive experiments across various model sizes and tasks, including pre-training and zero-shot evaluations. With the reformulated attention mechanism, the attention-based MoE layers of UMoE match or exceed the performance of previous FFN-based MoE layers. Moreover, by sharing parameters across attention and FFN modules, UMoE achieves superior performance in fully MoE architectures while maintaining the same parameter count. We also present a detailed routing analysis of UMoE, revealing expert specialization patterns across modules, with higher-ranked experts demonstrating interpretable attention patterns. Our code is available at <https://github.com/ysngki/UMoE>.

2 Related Work

Sparse Mixture-of-Experts (MoE). Sparse Mixture-of-Experts (MoE) models have gained increasing attention for their ability to scale model capacity while maintaining computational effi-

ciency [4, 9, 10, 5, 11]. The core component of these models is the sparsely activated MoE sub-layer, which selectively activates different parameter subsets for different inputs. In recent Transformer-based implementations, MoE architectures primarily replace feed-forward network (FFN) layers with MoE sub-layers. Each MoE layer consists of a collection of experts, denoted as $\{E_i\}_{i=1}^N$, where each expert E_i is implemented as an FFN. Tokens are routed to a subset of experts through a routing mechanism, with the top- k router [4] being the most prevalent approach. Despite advances in routing mechanisms [16, 17, 18, 19, 15], the top- k router remains widely adopted due to its simplicity and robust performance [9]. For a given token $x \in \mathbb{R}^d$, where d is the hidden dimension, and a trainable weight matrix $W_r \in \mathbb{R}^{N \times d}$, the top- k router computes the probability distribution over experts as:

$$p = \text{softmax}(W_r x). \quad (1)$$

The set of top- k experts $\mathcal{T}$ is then selected based on p , where $|\mathcal{T}| = k$. Each expert processes the token independently and the final output of the MoE layer is computed as the weighted combination of these k experts’ outputs:

$$y = \sum_{i \in \mathcal{T}} p_i E_i(x), \quad (2)$$

where each expert is implemented as an FFN with two matrices and a non-linear activation function.

MoE for Attention. Several recent approaches have explored extending the MoE paradigm to attention layers in Transformers [12, 13], with a primary focus on expert design. Because attention layers lack the consecutive matrix multiplication pattern found in FFNs, these approaches necessitate expert designs that differ from FFN-MoE models. The Mixture-of-Attention (MoA) [12] propose to conceptualize individual attention heads as experts, scaling attention layers by increasing the number of attention heads. However, introducing sparsity into attention layers presents a significant challenge: query vectors computed by a specific expert (or head) require corresponding key and value vectors from the same expert, necessitating identical expert activation across all tokens. To address this constraint, MoA implements distinct query and output projections per head while maintaining shared key and value projections across attention heads.

SwitchHead [13] presents an alternative approach to implementing the MoE paradigm in attention layers. Rather than treating entire attention heads as experts, SwitchHead designates individual projection matrices within heads as experts. A straightforward implementation maintains four separate MoE sub-layers per head for query, key, value, and output projections. While scaling all projections yields performance improvements, empirical results show that value and output projections benefit most significantly from scaling.

In contrast to these approaches, UMoE unifies attention-MoE and FFN-MoE through a novel reformulation of the multi-head attention mechanism, enabling the shared expert design and parameters across both attention and FFN layers.

Other Related Work. Several studies have explored connections between MoE and attention from different perspectives. MoH [20] proposes using MoE for pruning attention heads in LLMs by continuing pre-training with a routing function. During inference, certain output projections (W_o), viewed as experts, are selectively skipped based on routing decisions. Taking a different approach, MH-MoE [21] incorporates concepts from multi-head attention to enhance FFN-based MoE models. Instead of routing original input tokens to experts, MH-MoE decomposes each token into multiple low-dimensional sub-tokens, which are then processed in parallel by diverse sets of experts.

3 Method

The attention mechanism is the core of Transformers [22], processing token hidden states to capture contextual relationships. However, its structure differs from FFN layers, which complicates the unification of MoE designs across both modules. In this section, we present two alternative formulations of attention, pre-mixing and post-mixing, that reveal an inherent FFN-like structure within attention layers. Based on these formulations, we introduce a novel MoE architecture, UMoE.

```

def UMoELayer(x, X):
    # x: [1, d], X: [n, d]

    ### Attention MoE
    indices, probs = TopKRouter(x) # Assign token x to Experts

    residual_x = x.copy()
    K = X @ W_k
    q_shared = x @ W_q
    for i, p in zip(indices, probs):
        q = q_shared + x @ W_a[i] @ W_b[i]
        # K and V (the hidden states X) are shared across experts.
        y = Attention(Q=q, K=K, V=X)
        residual_x += p * Experts[i](y)
    x = residual_x

    ### FFN MoE
    indices, probs = TopKRouter(x) # Assign token x to Experts
    residual_x = x.copy()
    for i, p in zip(indices, probs):
        residual_x += p * Experts[i](x)
    return residual_x

```

Figure 3: Implementation details of a UMoE layer. The input consists of a sequence X containing n token hidden states and x representing the final hidden state. For simplicity, this implementation focuses on computing the output for the last token.

3.1 Formulations of Attention

Preliminaries. Consider a sequence of token hidden states $\mathbf{X} \in \mathbb{R}^{n \times d}$, where n is the sequence length and d is the hidden dimension. In multi-head attention, each token attends to all other tokens in the sequence through query, key, and value projections. For a single token x (e.g., the last token in the sequence for simplicity), its attention output is computed as:

$$q = x\mathbf{W}_q, \mathbf{K} = \mathbf{X}\mathbf{W}_k, \mathbf{V} = \mathbf{X}\mathbf{W}_v, \quad (3)$$

$$a = \text{softmax}\left(\frac{q\mathbf{K}^\top}{\sqrt{d_k}}\right), \quad o = a\mathbf{V}, \quad (4)$$

where $\mathbf{W}_q, \mathbf{W}_k \in \mathbb{R}^{d \times d_k}$ and $\mathbf{W}_v \in \mathbb{R}^{d \times d_v}$ are learnable matrices, respectively, and $a \in \mathbb{R}^n$ is the attention weight. To enhance representation capacity, this process is repeated h times in parallel, and the outputs are combined:

$$y = [o_1; o_2; \dots; o_h]\mathbf{W}_o, \quad (5)$$

where $\mathbf{W}_o \in \mathbb{R}^{hd_v \times d}$ projects the concatenated outputs back to the original dimension d .

Pre-Mixing Formulation. While multi-head attention is typically expressed using concatenation, it can be equivalently expressed as a sum of per-head outputs, which helps reveal its connection to FFN layers. By decomposing $\mathbf{W}_o$ into small matrices $\mathbf{W}_o^i \in \mathbb{R}^{d_v \times d}$ along the feature dimension, we can express the output as:

$$y = \sum_{i=1}^h o_i \mathbf{W}_o^i = \sum_{i=1}^h (a_i \mathbf{X} \mathbf{W}_v^i) \mathbf{W}_o^i \quad (6)$$

$$= \sum_{i=1}^h (a_i \mathbf{X}) (\mathbf{W}_v^i \mathbf{W}_o^i). \quad (7)$$

This reformulation provides two distinct interpretations, as shown in Fig. 2:

- Eq. 6: The conventional view where value vectors are first aggregated then projected back into the hidden space with an output projection.

- Eq. 7: A new interpretation where token hidden states are first aggregated into contextualized representations, i.e., weighted averages of all tokens, before being processed by the value ($\mathbf{W}_v^i$) and output ($\mathbf{W}_o^i$) projections. We term this formulation as *pre-mixing* attention.

While both interpretations yield same outputs, the pre-mixing formulation enables the grouping of $\mathbf{W}_o$ and $\mathbf{W}_v$. This grouping reveals that pre-mixing attention exhibits a two-layer structure analogous to FFN modules, which can be implemented as a linear FFN with no activation function.

Post-Mixing Formulation. Alternatively, we can rearrange the computation as:

$$\mathbf{y} = \sum_{i=1}^h \mathbf{a}_i (\mathbf{X} \mathbf{W}_v^i \mathbf{W}_o^i). \quad (8)$$

In this formulation, token hidden states are transformed by two successive projections independently for each token, before being aggregated using the attention weights.

3.2 UMoE

By grouping $\mathbf{W}_v$ and $\mathbf{W}_o$, both pre-mixing and post-mixing attention can be naturally interpreted as a MoE architecture, aligning with established FFN-MoE practices. Using pre-mixing attention as an example, let the expert $E(\mathbf{x}) := \mathbf{x} \mathbf{W}_v \mathbf{W}_o$. The multi-head attention can then be reformulated as:

$$\mathbf{y} = \sum_{i=1}^h E_i(\mathbf{a}_i \mathbf{X}). \quad (9)$$

By increasing the number of experts and introducing a routing mechanism, such as a top-k router, we derive a MoE architecture, denoted as UMoE-Att. The output of a UMoE-Att layer is:

$$\mathbf{y} = \sum_{i \in \mathcal{T}} p_i E_i(\mathbf{a}_i \mathbf{X}), \quad \text{where } \mathcal{T} \text{ is the set of activated experts.} \quad (10)$$

Referring to Eq. 2, we observe that the primary distinction between FFN-MoE layers and UMoE-Att layers lies in their expert inputs: FFN experts operate on individual token hidden states $\mathbf{x}$, while attention experts process weighted combinations of all token hidden states. This reveals a relationship: FFN-MoE layers can be interpreted as a specialized case of pre-mixing attention layers where the attention matrix is constrained to an identity matrix, limiting each token to self-attention only.

Fully MoE Architecture. Both the experts in UMoE-Att and the FFN layers of Transformer consist of two consecutive matrices. While attention layer experts utilize a relatively small intermediate size (d_v), FFN layers typically employ larger dimensions. Recent advances in FFN-MoE models suggest the efficacy of using FFN layers with reduced intermediate sizes as experts [6, 15, 3]. This insight enables the direct adoption of experts in attention layers for FFN layers, resulting in a fully MoE architecture, denoted as UMoE. Fig. 1 illustrates the architecture of a UMoE layer., where the MoE paradigm is applied to both FFN and attention layers using a shared expert set. Notably, to facilitate parameter sharing, experts are implemented as two-layer FFNs with an intermediate size of d_v and incorporate a non-linear activation function between matrix multiplications.

Pre-mixing Implementation. The token mixing operation in pre-mixing attention is a weighted summation over token hidden states, which can be implemented as vanilla attention, accepting Q , K , V matrices as input and producing an output matrix. Each token generates distinct query vectors for different experts, while values (hidden states) and their associated keys are shared across experts. To generate expert-dependent queries for input tokens, each expert requires an additional query projection matrix, leading to a notable increase in parameters. To mitigate the parameter count disparity with existing MoE models, where experts typically comprise two matrices, we employ low-rank matrices [23] for query projection within UMoE experts. For a given token $\mathbf{x}$, the query for expert i is computed as:

$$\mathbf{q}_i = \mathbf{x} \mathbf{W}_q + \mathbf{x} \mathbf{W}_a^i \mathbf{W}_b^i, \quad (11)$$

where the first term is shared across all experts, while the second term is expert-specific with unique parameters, $\mathbf{W}_a^i \in \mathbb{R}^{d \times r}$ and $\mathbf{W}_b^i \in \mathbb{R}^{r \times d_k}$, for each expert. Fig. 3 presents the pseudo-code of a UMoE layer.

Putting It All Together. As illustrated in Fig. 1, UMoE integrates three key components: (1) experts implemented as fine-grained FFNs with dual low-rank query projection matrices, (2) pre-mixing attention mechanism utilizing shared keys and values across experts, and (3) the top-k router for expert selection. It is noteworthy that while MoA [12] also shares keys and values across experts, the values of MoA are the results after applying a value linear transformation to the input token hidden states. In contrast, the values of UMoE directly refer to the input token hidden states.

3.3 Discussion

Vanilla Attention vs Pre-mixing Attention. Vanilla attention and pre-mixing attention differ in terms of KV cache requirements and computational complexity. During inference, vanilla attention requires caching multiple keys and values per token, whereas pre-mixing attention requires only one key and hidden state per token. While grouped-query attention (GQA) is commonly adopted to reduce the KV cache size in attention layers, it cannot be combined with pre-mixing attention, since the latter already maintains only a single key-value pair per token. Instead, multi-head latent attention (MLA) [8], a promising alternative to GQA, can be applied by introducing a down-projection to the hidden states before the token-mixing operation. Regarding the computation, while vanilla attention performs weighted summation over low-dimensional value vectors, pre-mixing attention operates on input token hidden states, introducing a modest increase in computational complexity. This modest increase, however, becomes increasingly negligible as models scale to larger dimensions, effectively amortizing the additional computational overhead. A detailed comparative analysis is presented in Table 8 (A.1). Additionally, the abstract formulation of UMoE opens avenues for future research to explore more computationally efficient token mixing alternatives, such as linear attention mechanisms [24, 25].

Pre-mixing Attention vs Post-mixing Attention. As illustrated in Fig. 4, post-mixing attention processes individual tokens through experts prior to mixing. The architectural distinction between pre-mixing and post-mixing variants represents different perspectives on token-parameter interactions in attention layers. Recent interpretability studies have drawn parallels between the two-matrix multiplication pattern of FFNs and associative memory modules, where value neurons, i.e. columns of the second matrix in FFNs, are retrieved by inputs [26, 27, 28]. Within this framework, pre-mixing attention leverages token mixing to generate contextualized inputs for precise retrieval. In contrast, post-mixing attention can be conceptualized as an ensemble of independent retrievals executed by preceding tokens. Our preliminary experiments (A.2) demonstrates a significant performance advantage of pre-mixing attention over its post-mixing counterpart. This observation suggests that generating contextualized inputs for token-parameter interactions more effectively aligns with the principles of attention mechanisms.

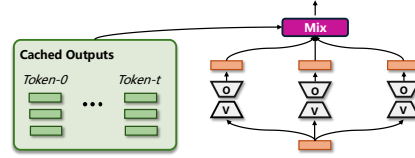


Figure 4: Post-Mixing Attention.

4 Experiments

4.1 Setup

Datasets. We conduct language modeling pretraining on two datasets: FineWeb-Edu 100B [29] and Wikitext-103 [30]. FineWeb-Edu has shown superior data efficiency when evaluated on knowledge-intensive benchmarks. Wikitext-103, consisting of approximately 100M tokens, is a smaller corpus that has been widely adopted in previous studies [16, 13]. We apply the LLaMA tokenizer [31] with a 32K vocabulary size to both datasets. The zero-shot performance of models trained on FineWeb-Edu is evaluated using the `lm-evaluation-harness` framework [32].

Baselines. We compare UMoE against three categories of baselines: dense models, FFN-MoE models with fine-grained experts [8, 3], and attention-MoE models (specifically MoA [12] and SwitchHead [13]). Attention-MoE models are configured with identical expert parameters¹. We

¹SwitchHead represents an exception, as it requires the number of experts to be divisible by the number of attention heads.

Table 1: Comparison of Dense and Sparse Mixture-of-Experts (MoE) Models for Language Modeling. In MoE models, $A \times B$ denotes B experts per layer with size A in ‘#Total’ columns, while B in ‘#Active’ columns indicates the number of experts activated per token. Gray entries in ‘#Total’ columns indicate shared parameters between attention and FFN modules in UMoE models. UMoE-Att refers to UMoE variants with MoE only applied to attention modules.

Model	Params	Attention		FFN		PPL (↓)		MACs
		#Total	#Active	#Total	#Active	Fineweb	Wikitext	
Base Models								
Dense	134 M	768	768	3072	3072	25.79	30.41	525 G
Fine-grained FFN-MoE	535 M	768	768	192×128	192×16	21.19	27.94	530 G
MoA	525 M	192×116	192×4	3072	3072	22.28	27.57	486 G
SwitchHead	533 M	192×116	192×4	3072	3072	22.91	29.47	542 G
UMoE-Att	547 M	192×116	192×4	3072	3072	20.81	27.45	611 G
UMoE	540 M	192×128	192×4	192×128	192×16	20.44	26.67	616 G
Large Models								
Dense	1.1 B	2048	2048	5632	5632	17.53	25.46	4.59 T
Fine-grained FFN-MoE	3.8 B	2048	2048	512×64	512×11	16.09	25.47	4.61 T
MoA	3.6 B	512×57	512×4	5632	5632	16.72	25.14	3.99 T
SwitchHead	3.7 B	512×60	512×4	5632	5632	16.48	27.24	4.62 T
UMoE-Att	3.8 B	512×57	512×4	5632	5632	16.03	25.53	4.73 T
UMoE	3.6 B	512×64	512×4	512×64	512×11	15.95	25.44	4.75 T

implement all MoE models with a fixed expert per layer, following recommendations for optimal model performance [8, 33]. UMoE adopts the pre-mixing attention mechanism, which consistently outperforms post-mixing variants.

Experimental Setup. UMoE is implemented as a decoder-only Transformer with rotary position embedding [34], following deepseek-MoE [6]. The load balancing loss proposed by Switch Transformer [5] is adopted to encourage a balanced load across experts. The experts in our experiments are implemented as two-layer MLPs to ensure fair comparison with the baselines.²

We evaluate two model configurations: The base models comprise 12 layers with a hidden size of 768, while the large models consist of 24 layers with a hidden size of 2048. These configurations yield dense models with 134M and 1.1B parameters, respectively. MoE variants replace all attention or FFN layers with MoE layers. Due to computational constraints, unless otherwise specified, models are pretrained on 50B tokens from FineWeb-Edu with batch size of 1024. For Wikitext-103, following Csordás et al. [13], models are trained for 100k steps, though models typically overfit within 20k steps. Detailed hyperparameters are provided in A.4.

4.2 Comparison with Baselines

Results. From Table 1, we observe that UMoE shows consistent superiority across different model sizes and datasets. In the base model regime, UMoE achieves the best performance. Notably, the attention-only variant of UMoE exhibits substantial improvements over previous attention-based approaches. Even without parameter sharing, UMoE-Att establishes itself as a compelling alternative to traditional FFN-based MoE models. The parameter sharing mechanism between attention and FFN modules further enhances the effectiveness without increasing the total parameter count. Despite equalizing the number of activated experts across all baselines, we observe subtle computational discrepancies due to different attention layer implementations, as measured by MACs (multiplication accumulation operation). We additionally perform a MAC-matched comparison by increasing the number of activated experts of baseline models. Table 2 shows that even under comparable computational constraints, UMoE method achieves the lowest perplexity.

²UMoE is designed as a flexible framework agnostic to the expert implementation. Using more advanced gated MLP variants, such as SwiGLU, may lead to improved performance, as the gating mechanism can enhance the expressiveness of both attention and feed-forward layers.

Table 2: MAC-matched comparison for base models by increasing the number of activated experts of baseline models.

Model	MACs	Active Params	PPL ($\downarrow$)	
			Fineweb	Wikitext
FFN-MoE	617 G	768 + 192 $\times$ 22	20.80	27.39
MoA	621 G	192 $\times$ 8 + 3072	22.00	27.63
SwitchHead	649 G	192 $\times$ 12 + 3072	21.57	28.13
UMoE-Att	611 G	192 $\times$ 4 + 3072	20.81	27.45
UMoE	616 G	192 $\times$ 4 + 192 $\times$ 16	20.44	26.67

Table 3: Parameter sharing strategies. $\checkmark$ indicates shared components between modules while $\times$ indicates separate components.

Component	UMoE	-	-	-
Fixed Experts	$\times$	$\checkmark$	$\times$	$\checkmark$
Router	$\times$	$\checkmark$	$\checkmark$	$\times$
# Params	540 M	536 M	540 M	537 M
PPL	22.82	23.11	23.05	23.02

Table 4: Zero-shot accuracy on downstream tasks. The best score is marked in **bold**.

Model	Params	HellaSwag	PIQA	ARC-E	ARC-C	RACE	Lambada	MMLU	Wino	Avg.
<i>Base Models</i>										
Dense	134 M	33.58	62.35	46.09	24.74	27.75	19.97	24.8	49.8	36.14
MoA	525 M	37.82	65.58	51.34	26.19	28.83	22.33	25.1	50.7	38.49
SwitchHead	533 M	37.19	66.12	50.55	26.59	28.14	21.73	25.2	50.9	38.30
FFN-MoE	535 M	39.69	66.43	52.95	26.71	29.76	23.46	25.3	52.1	39.55
UMoE (Att)	547 M	40.72	67.36	51.77	27.82	29.76	23.66	25.9	52.5	39.94
UMoE	540 M	41.28	66.65	51.86	29.01	28.71	23.77	26.6	52.6	40.06
<i>Large Models</i>										
Dense	1.1 B	48.45	69.26	58.85	32.17	33.11	31.75	27.4	53.8	44.35
MoA	3.6 B	50.61	70.28	61.47	33.22	32.38	33.15	28.6	54.7	45.55
SwitchHead	3.7 B	51.90	70.83	62.34	33.69	33.27	33.66	28.8	55.7	46.27
FFN-MoE	3.8 B	52.74	71.52	64.23	35.67	33.30	34.00	29.2	56.3	47.12
UMoE (Att)	3.8 B	53.20	71.44	63.30	34.39	32.82	34.78	29.3	57.4	47.08
UMoE	3.6 B	53.17	72.47	64.23	35.75	32.44	35.32	30.4	56.9	47.58

In larger-scale models, UMoE maintains its competitive advantage. While MoA shows marginally better performance on Wikitext-103, this result may not fully reflect model capabilities given the relatively small size of Wikitext-103 (100M tokens) compared to the model scale. Following established practice [16], we further report validation perplexity. Fig. 5 shows that UMoE demonstrates faster convergence and lower validation perplexity compared to baselines, indicating enhanced modeling capabilities. This superior performance translates to downstream tasks, with Table 4 showing UMoE consistently achieving the highest average zero-shot accuracy across diverse tasks.

Efficiency. Following Zhang et al. [12], Jin et al. [20], we employ MACs³ as an efficiency metric, as it remains independent of hardware implementations. As shown in Table 1, the pre-mixing attention introduces a modest computational overhead, resulting in approximately 1.17 $\times$ slowdown for base models. However, this slowdown becomes increasingly negligible as models scale up; in large models, UMoE introduces only 1.03 $\times$ slowdown compared to the dense baseline. This favorable scaling behavior arises from the different growth rates in computational complexity: expert processing scales quadratically with hidden dimension, while token aggregation in attention layers scales linearly.

4.3 Ablations

We conducted ablation experiments using base models trained on FineWeb-Edu with 20B tokens.

Parameter Sharing Analysis. We investigated various sharing strategies across FFN and attention layers for fixed experts [6, 33] and routers. As shown in Table 3, all configurations achieved comparable perplexity. Our default configuration, which employs separate fixed experts and routers across FFN and attention layers, yielded the optimal perplexity.

Expert Allocation. As suggested in Section 3.2, FFN-MoE layers can be interpreted as a specialized case of pre-mixing attention layers with an identity matrix as attention matrix. This interpretation raises a question: *does UMoE perform better when allocating more experts to attention layers rather than FFN layers?* According to Table 5, we observe a trend when gradually shifting expert

³MACs is measured using the DeepSpeed Flops Profiler.

Table 5: Impact of expert allocation between Attention and FFN layers (total experts = 20).

Model	# Expert		PPL
	Attention	FFN	
UMoE	4	16	22.82
	8	12	22.63
	12	8	22.44
	16	4	22.50
	20	0	21.75

Table 6: Effect of Activation Functions in Expert Modules. ✓ indicates experts with activation functions while ✗ indicates experts without activation functions.

Model	Act. Function	PPL
UMoE	✓	22.82
	✗	24.43
UMoE (Att)	✓	23.37
	✗	23.99

Table 7: Top tokens for selected experts in the last attention and FFN layer of UMoE.

Expert ID	Top Tokens in Attention Layer	Top Tokens in FFN Layer
3	_Each , This , Every , Each , _This	This , _This , Every , _Each , _Another
10	_Film , _video , _lab , _film , _Video	Tag , _Font , _ISBN , twitter , _DNS
46	The , _The , _the , the , _Our	_a , _his , _my , Your , _Your
64	". , % , . , : , ."	_relatively , _extremely , _a , _very , _Very

allocation from FFN to attention modules while maintaining a total activated expert size of 20. The model achieves its best perplexity when all experts are allocated to attention layers. This finding provides empirical evidence supporting our theoretical interpretation that FFN layers function as a specialized form of attention, with the attention mechanism exhibiting greater expressiveness. However, increasing the number of attention experts introduces substantial computational overhead due to token mixing operations. Future research could explore efficient attention alternatives within the attention MoE framework.

Activation Function. Table 6 presents our investigation into the impact of activation functions in UMoE. The results demonstrate that incorporating activation functions between matrix multiplications within experts consistently improves model performance, reinforcing the crucial role of non-linearity in deep learning architectures. Notably, while the removal of activation functions reduces the experts to pure linear transformations in both FFN and attention modules, UMoE remains trainable. We attribute this robustness to the preserved non-linearity from token mixing operations and layer normalization. Nevertheless, the consistent performance degradation underscores the importance of activation functions in model expressiveness, particularly in the context of shared expert architectures.

4.4 Expert Specialization

Table 7 presents the routing patterns in the final layer of UMoE, where experts are shared between attention and FFN modules while maintaining distinct routers. Notably, certain token categories consistently route to the same experts across both modules, as evidenced by experts 3 and 46. Expert 3 consistently processes determiners, while expert 46 specializes in demonstrative pronouns.

The analysis also reveals divergent specialization patterns that highlight the complexity of shared expert architectures. A notable example is expert 64, which exhibits distinct specializations: processing consecutive punctuation marks in the attention layer while handling degree adverbs in the FFN layer. This phenomenon suggests that shared experts can develop multiple specializations, potentially leading to more efficient parameter utilization. However,

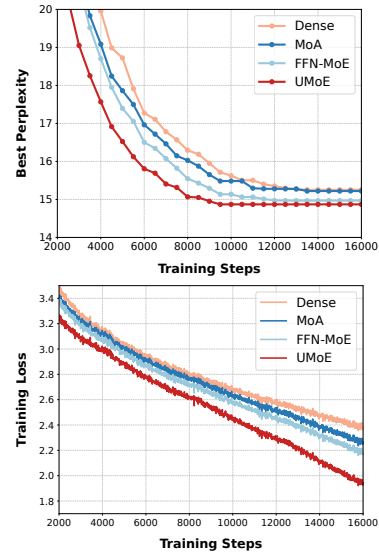


Figure 5: Best valid PPL (top) and training loss (bottom) on Wikitext.

it also raises important questions about potential knowledge conflicts within individual experts, indicating promising directions for future research in routing mechanism design for shared expert architectures.

We also provide an analysis on the attention maps of UMoE in A.5, which confirms that higher-ranked experts show more focused attention distributions on relevant tokens compared to lower-ranked ones.

5 Conclusion

The paper proposes UMoE, a novel architecture that unifies MoE designs for attention and FFN layers. The key insight is a reformulation of the attention mechanism that allows the value and output projections to be grouped into FFN-like experts. This unification enables parameter sharing across attention and FFN layers, resulting in a fully MoE architecture that improves performance without introducing additional parameters. The paper presents extensive experiments demonstrating UMoE’s superiority over existing MoE architectures in terms of perplexity on language modeling datasets and accuracy on zero-shot tasks.

As for future work, we are looking at replacing the token mixing mechanism with more efficient alternatives to enable scaling up the number of activated experts in attention layers. In addition, we are also interested in investigating architectures that unify attention and FFN into a single layer, given our finding that FFN layers function as a specialized case of attention with reduced expressiveness. Exploring parameter sharing across different Transformer layers [35, 36] is also a promising direction.

References

- [1] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *CoRR*, abs/2001.08361, 2020. URL <https://arxiv.org/abs/2001.08361>.
- [2] Elias Frantar, Carlos Riquelme Ruiz, Neil Houlsby, Dan Alistarh, and Utku Evci. Scaling laws for sparsely-connected foundation models. In *The Twelfth International Conference on Learning Representations*, 2024.
- [3] Jan Ludziejewski, Jakub Krajewski, Kamil Adamczewski, Maciej Pióro, Michał Krutul, Szymon Antoniak, Kamil Ciebia, Krystian Król, Tomasz Odrzygóźdź, Piotr Sankowski, Marek Cygan, and Sebastian Jaszczur. Scaling laws for fine-grained mixture of experts. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pages 33270–33288. PMLR, 21–27 Jul 2024.
- [4] Noam Shazeer, *Azalia Mirhoseini, *Krzysztof Maziarczyk, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017.
- [5] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: scaling to trillion parameter models with simple and efficient sparsity. *J. Mach. Learn. Res.*, 23(1), January 2022. ISSN 1532-4435.
- [6] Damai Dai, Chengqi Deng, Chenggang Zhao, R.x. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y. Wu, Zhenda Xie, Y.k. Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and Wenfeng Liang. DeepSeekMoE: Towards ultimate expert specialization in mixture-of-experts language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1280–1297, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.70. URL <https://aclanthology.org/2024.acl-long.70/>.
- [7] 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. In *Proceedings of the 41st International Conference on Machine Learning, ICML’24*. JMLR.org, 2024.
- [8] DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo,

- Jiaqi Ni, Jiashi Li, Jiawei Wang, Jin Chen, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu, Ruoyu Zhang, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Shuting Pan, T. Wang, Tao Yun, Tian Pei, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wanbiao Zhao, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaokang Zhang, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xinnan Song, Xinxia Shan, Xinyi Zhou, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. X. Zhu, Yang Zhang, Yanhong Xu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Yu, Yi Zheng, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Ying Tang, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yu Wu, Yuan Ou, Yuchen Zhu, Yudian Wang, Yue Gong, Yuheng Zou, Yujia He, Yukun Zha, Yunfan Xiong, Yunxian Ma, Yuting Yan, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhipeng Xu, Zhiyu Wu, Zhongyu Zhang, Zhuoshu Li, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Ziyi Gao, and Zizheng Pan. Deepseek-v3 technical report, 2024.
- [9] Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, Yuling Gu, Shane Arora, Akshita Bhagia, Dustin Schwenk, David Wadden, Alexander Wettig, Binyuan Hui, Tim Dettmers, Douwe Kiela, Ali Farhadi, Noah A. Smith, Pang Wei Koh, Amanpreet Singh, and Hannaneh Hajishirzi. Olmoe: Open mixture-of-experts language models. *CoRR*, abs/2409.02060, 2024.
- [10] Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, L  lio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Th  ophile Gervet, Thibaut Lavril, Thomas Wang, Timoth  e Lacroix, and William El Sayed. Mixtral of experts. *CoRR*, abs/2401.04088, 2024. URL <https://doi.org/10.48550/arXiv.2401.04088>.
- [11] Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. St-moe: Designing stable and transferable sparse expert models. *arXiv preprint arXiv:2202.08906*, 2022.
- [12] Xiaofeng Zhang, Yikang Shen, Zeyu Huang, Jie Zhou, Wenge Rong, and Zhang Xiong. Mixture of attention heads: Selecting attention heads per token. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 4150–4162. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.EMNLP-MAIN.278. URL <https://doi.org/10.18653/v1/2022.emnlp-main.278>.
- [13] R  bert Csord  s, Piotr Piekos, Kazuki Irie, and J  rgen Schmidhuber. Switchhead: Accelerating transformers with mixture-of-experts attention. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL <https://openreview.net/forum?id=80SS169GAz>.
- [14] Xiaoye Qu, Daize Dong, Xuyang Hu, Tong Zhu, Weigao Sun, and Yu Cheng. Llama-moe v2: Exploring sparsity of llama from perspective of mixture-of-experts with post-training. *CoRR*, abs/2411.15708, 2024.
- [15] Yuanhang Yang, Shiyi Qi, Wenchao Gu, Chaozheng Wang, Cuiyun Gao, and Zenglin Xu. XMoE: Sparse models with fine-grained and adaptive expert selection. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 11664–11674. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-acl.694.

- [16] Stephen Roller, Sainbayar Sukhbaatar, Arthur Szlam, and Jason E Weston. Hash layers for large sparse models. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL <https://openreview.net/forum?id=1MgDDWb1ULW>.
- [17] Mike Lewis, Shruti Bhosale, Tim Dettmers, Naman Goyal, and Luke Zettlemoyer. BASE layers: Simplifying training of large, sparse models. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 6265–6274. PMLR, 2021. URL <http://proceedings.mlr.press/v139/lewis21a.html>.
- [18] Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Y Zhao, Andrew M. Dai, Zhifeng Chen, Quoc V Le, and James Laudon. Mixture-of-experts with expert choice routing. In *Advances in Neural Information Processing Systems*, 2022. URL <https://openreview.net/forum?id=jdJo1HIVinI>.
- [19] Quzhe Huang, Zhenwei An, Nan Zhuang, Mingxu Tao, Chen Zhang, Yang Jin, Kun Xu, Kun Xu, Liwei Chen, Songfang Huang, and Yansong Feng. Harder task needs more experts: Dynamic routing in MoE models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12883–12895, Bangkok, Thailand, August 2024. Association for Computational Linguistics. URL <https://aclanthology.org/2024.acl-long.696/>.
- [20] Peng Jin, Bo Zhu, Li Yuan, and Shuicheng Yan. Moh: Multi-head attention as mixture-of-head attention. *CoRR*, abs/2410.11842, 2024. doi: 10.48550/ARXIV.2410.11842. URL <https://doi.org/10.48550/arXiv.2410.11842>.
- [21] Xun Wu, Shaohan Huang, Wenhui Wang, Shuming Ma, Li Dong, and Furu Wei. Multi-head mixture-of-experts. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL <https://openreview.net/forum?id=dyZ8GJztX>.
- [22] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2017.
- [23] Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL <https://openreview.net/forum?id=nZeVKeeFYf9>.
- [24] Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL <https://openreview.net/forum?id=y8Rm4VNRPH>.
- [25] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *CoRR*, abs/2312.00752, 2023. URL <https://doi.org/10.48550/arXiv.2312.00752>.
- [26] Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2021. URL <https://aclanthology.org/2021.emnlp-main.446/>.
- [27] Zeyu Liu, Tim Dettmers, Xi Lin, Veselin Stoyanov, and Xian Li. Towards a unified view of sparse feed-forward network in pretraining large language model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 15038–15061, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.930. URL <https://aclanthology.org/2023.emnlp-main.930/>.
- [28] Zeping Yu and Sophia Ananiadou. Neuron-level knowledge attribution in large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 3267–3280. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-main.191. URL <https://aclanthology.org/2024.emnlp-main.191/>.
- [29] Guilherme Penedo, Hynek Kydlíček, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale. *CoRR*, abs/2406.17557, 2024. URL <https://doi.org/10.48550/arXiv.2406.17557>.

- [30] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.
- [31] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem 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, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288, 2023. URL <https://doi.org/10.48550/arXiv.2307.09288>.
- [32] 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. A framework for few-shot language model evaluation, 07 2024. URL <https://zenodo.org/records/12608602>.
- [33] Samyam Rajbhandari, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, and Yuxiong He. Deepspeed-moe: Advancing mixture-of-experts inference and training to power next-generation AI scale. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 18332–18346. PMLR, 2022.
- [34] Jianlin Su, Murtadha H. M. Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568: 127063, 2024. URL <https://doi.org/10.1016/j.neucom.2023.127063>.
- [35] Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. In *International Conference on Learning Representations, ICLR, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL <https://openreview.net/forum?id=HyzdRiR9Y7>.
- [36] Róbert Csordás, Kazuki Irie, Jürgen Schmidhuber, Christopher Potts, and Christopher D. Manning. MoEut: Mixture-of-experts universal transformers. In *Advances in Neural Information Processing Systems, NeurIPS*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/321387ba926b8e58d3591c0aeb52ffc2-Abstract-Conference.html.

Table 8: Computational complexity analysis of vanilla attention and pre-mixing attention mechanisms. Entries in gray denote identical complexity terms between the two mechanisms. Here, N denotes sequence length, d represents hidden dimension, h indicates the number of attention heads (activated experts), d_k and d_v are key and value dimensions respectively, and r is the rank of query projection matrices in the pre-mixing attention.

Operation	Vanilla	Pre-mixing
Output Projection	$\mathcal{O}(N \cdot d_v \cdot d \cdot h)$	$\mathcal{O}(N \cdot d_v \cdot d \cdot h)$
Value Projection	$\mathcal{O}(N \cdot d_v \cdot d \cdot h)$	$\mathcal{O}(N \cdot d_v \cdot d \cdot h)$
Key Projection	$\mathcal{O}(N \cdot d_k \cdot d \cdot h)$	$\mathcal{O}(N \cdot d_k \cdot d)$
Query Projection	$\mathcal{O}(N \cdot d_k \cdot d \cdot h)$	$\mathcal{O}(N \cdot d_k \cdot d + N \cdot (d_k + d) \cdot r \cdot h)$
QK Multiplication	$\mathcal{O}(N^2 \cdot d_k \cdot h)$	$\mathcal{O}(N^2 \cdot d_k \cdot h)$
Weighted Sum	$\mathcal{O}(N^2 \cdot d_v \cdot h)$	$\mathcal{O}(N^2 \cdot d \cdot h)$

A Technical Appendices and Supplementary Material

A.1 Complexity

Table 8 presents a detailed computational complexity comparison between vanilla attention and pre-mixing attention mechanisms. The key distinctions lies in two operations: key projection and weighted sum computation. In key projection, pre-mixing attention achieves lower complexity. However, this efficiency is partially offset in the weighted sum operation, where the complexity increases due to the full dimensional mixing. It is worth mentioning that the computational complexity of weighted sum grows linearly with the hidden dimension, while the FFN computation grows quadratically. The computational overhead of weighted sum becomes less significant as model size increases.

```
def PostMixingMoE(X):
    # X: [n, d]

    ### Attention MoE
    # Independent Expert Processing
    all_token_outputs = []
    all_token_keys = []

    for this_token in X:
        indices, probs = TopKRouter(this_token)

        for i, p in zip(indices, probs):
            y = p * Experts[i](this_token)
            all_token_outputs.append(y)

            k = W_k[i](this_token)
            all_token_keys.append(k)

    # Mixing
    Q = (X @ W_q).unsqueeze(1) # [n, 1, d_q]
    K = all_token_keys.reshape(n, k, d_q)
    V = all_token_outputs.reshape(n, k, d_v)

    attn_output = Attention(Q, K, V) # [n, 1, d_v]

    ### FFN MoE
    ...
```

Figure 6: Implementation of a UMoe layer based on post-mixing attention. X is a sequence of n token hidden states. In the attention layer, tokens are processed by their top- k experts independently. The output embeddings of all tokens are aggregated according to the attention weights.

A.2 Post-mixing vs Pre-mixing

Fig. 6 presents the pseudo-code of a UMoE layer based on post-mixing attention. We conducted preliminary experiments on Wikitext-103 and FineWeb-Edu datasets. As illustrated in Fig. 7, MoE models incorporating pre-mixing attention demonstrate substantially superior performance compared to their post-mixing counterparts.

As discussed in Section 3.1, these two reformulations of the attention mechanism are mathematically equivalent, owing to the absence of non-linear transformations within the matrix chain multiplication in attention heads. However, when grouping the value and output projections and implementing them as an FFN with a non-linear activation function, these formulations yield distinct outputs in the MoE layers. Fig. 8 illustrates MoE models implemented based on these two reformulations. Both approaches can be interpreted as extensions of conventional FFN-based MoE models. Specifically, the pre-mixing approach enhances FFN-based MoE models by contextualizing the inputs, while post-mixing attention enables MoE layers to incorporate other tokens’ outputs in generating the final output. Future research could explore the synergistic combination of post- and pre-mixing approaches to fully leverage contextual information.

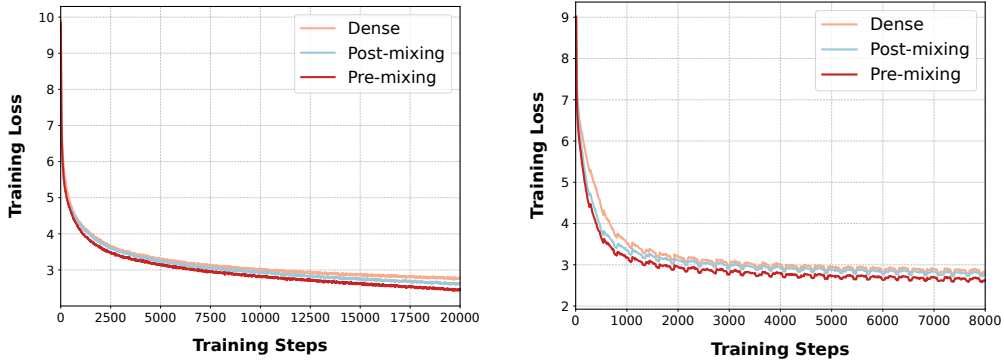


Figure 7: Loss curves of UMoE with attention MoE layers implemented on post-mixing and pre-mixing attention, respectively. Models are trained on Wikitext-103 (left) and FineWeb-Edu (right).

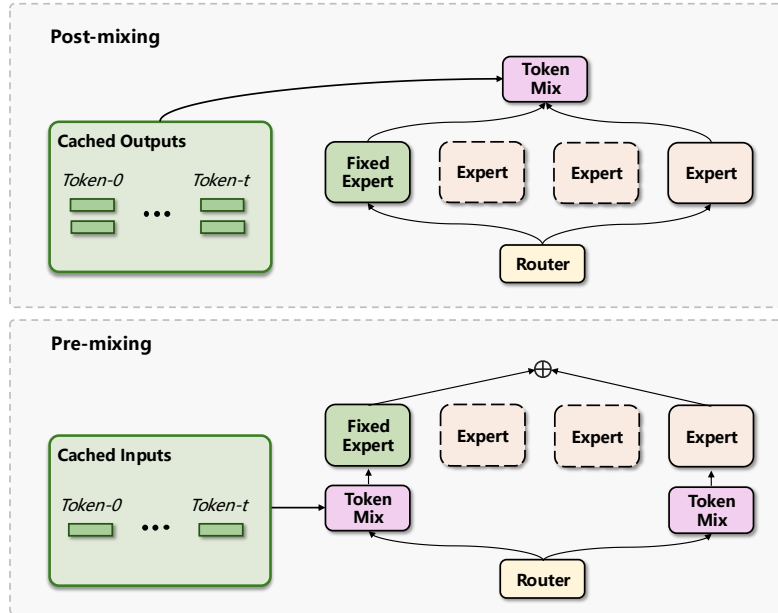


Figure 8: Two implementations of UMoE based on pre-mixing and post-mixing attention, respectively.

Table 9: Comparison of training throughput and inference latency across models. Inference latency denotes the pre-filling time (1024 tokens) measured on a single H100 GPU. All results are averaged over multiple runs.

Model Type	Model	Model Size	Pre-filling Latency (s)	Training Throughput (tokens/s)
Base Model	GPT	134M	0.1376	126,508
	FFN-MoE	535M	0.1974	74,415
	MoA	525M	0.1843	90,394
	SwitchHead	533M	0.2253	69,374
	UMoE-Att	547M	0.2006	76,538
	UMoE	540M	0.2197	71,461
Large Model	GPT	1.1B	0.2950	11,634
	FFN-MoE	3.8B	0.4080	9,799
	MoA	3.6B	0.3624	12,321
	SwitchHead	3.7B	0.4328	8,882
	UMoE-Att	3.8B	0.4120	11,377
	UMoE	3.6B	0.4392	9,194

A.3 Wall-clock Efficiency Analysis

Table 9 presents a quantitative comparison of end-to-end training throughput and inference latency across various model architectures. All experiments were conducted on a single NVIDIA H100 GPU. Inference latency corresponds to the pre-filling time for sequences of 1024 tokens, while training throughput is measured in tokens per second.

Across both scales, all Mixture-of-Experts (MoE) variants exhibit comparable inference latency but remain slower than the dense baseline, despite their similar theoretical MAC counts. This discrepancy primarily arises from additional computational overheads introduced by expert routing and sparse expert execution. Among MoE models, SwitchHead and UMoE show slightly higher latency due to their architectural designs—SwitchHead applies two MoE layers per attention head, whereas UMoE employs two MoE layers within each Transformer block.

These results underscore a persistent challenge in MoE-based architectures: while their theoretical efficiency is well established, current hardware and software infrastructures (e.g., routing kernels, expert parallelization) are not yet fully optimized to realize these potential speedups. Consequently, our main paper reports MAC-based efficiency as the primary metric. We anticipate that advances in GPU kernel design and distributed parallelization will further mitigate the observed wall-clock inefficiencies in future implementations.

A.4 Hyperparameters

Table 10 and 11 give the parameters used for based and large models, respectively. It takes roughly a week for pretraining base models on FineWeb-Edu datasets. MoA and SwitchHead utilize identical parameters as UMoE-Att, excluding the low-rank query projections. Table 12 details the hyperparameters used during training. For the Wikitext-103 dataset, we adopt the same hyperparameter configuration as Csordás et al. [13].

A.5 Attention Analysis

We analyze the attention patterns in UMoE by visualizing expert-specific attention maps. While UMoE (Large) contains 64 experts per layer across 24 layers, we focus on the top 8 experts (ranked by router scores) to maintain tractability. Each expert utilizes its own query projection matrix, allowing us to compute attention maps regardless of activation status.

To investigate attention behavior, we examine two inputs:

- *“Context: William Shakespeare wrote the famous play Romeo and Juliet in the late 16th century. Question: Who wrote Romeo and Juliet? The Answer is”*

Table 10: Hyperparameters of Base Models. MoA and SwitchHead use the same hyperparameters as UMoE-Att.

Hyperparameter	Dense	FFN-MoE	UMoE-Att	UMoE
Context Length	1024	1024	1024	768
Number of Layers	12	12	12	12
Hidden Size	768	768	768	768
Attention Heads	4	4	4	4
FFN Size	3072	192×16	3072	192×16
Query (Key) Dimension	512	512	512	512
Value Dimension	192	192	–	–
Query Lora Rank	–	–	16	16
Number of MoE layers	–	12	12	12
Expert Size	–	192	192	192
Experts per MoE Layer	–	128	116	128
FFN Experts per Token	–	16	–	16
Attention Experts per Token	–	–	4	4

Table 11: Hyperparameters of Large Models. MoA and SwitchHead use the same hyperparameters as UMoE-Att.

Hyperparameter	Dense	FFN-MoE	UMoE-Att	UMoE
Context Length	1024	1024	1024	1024
Number of Layers	24	24	24	24
Hidden Size	2048	2048	2048	2048
Attention Heads	4	4	4	4
FFN Size	5632	512×11	5632	512×11
Query (Key) Dimension	512	512	512	512
Value Dimension	512	512	–	–
Query Lora Rank	–	–	36	36
Number of MoE layers	–	24	24	24
Expert Size	–	512	512	512
Experts per MoE Layer	–	64	57	64
FFN Experts per Token	–	11	–	11
Attention Experts per Token	–	–	4	4

- “Context: Tokyo is the capital city of Japan and has a population of over 37 million people in its metropolitan area. Question: What is the capital of Japan? The Answer is”

UMoE successfully predicted the correct answers for both inputs, even after removing the context. For the final token in each input, we collected attention maps from the top 8 experts across all layers.

Fig. 9 presents attention maps for the final token prediction, revealing distinct patterns between higher and lower-ranked experts. Higher-ranked experts demonstrate more focused attention distributions that align with task requirements. For instance, in the Shakespeare question, Expert_0 and Expert_32 show pronounced attention weights on task-critical tokens "who wrote". Similarly, for the Tokyo question, the top expert exhibits sophisticated attention patterns by focusing on key contextual elements like "Japan" and "capital". These observations suggest that the routing mechanism effectively identifies experts capable of extracting task-relevant information through specialized attention patterns.

To create a comprehensive visualization, we aggregated attention maps of all layers by summing them, as shown in Fig. 10. The values on the left (e.g., E0: 3.83) represent the sum of router scores

Table 12: Training Hyperparameters on FineWeb-Edu and Wikitext-103.

Hyperparameter	FineWeb-Edu	Wikitext
Global Batch Size	1024	64
Learning Rate	4e-4	2.5e-4
Training Steps	50000	100000
LR Scheduler	cosine	cosine
Warmup Ratio	0.05	0.05
GPU	H100	H100

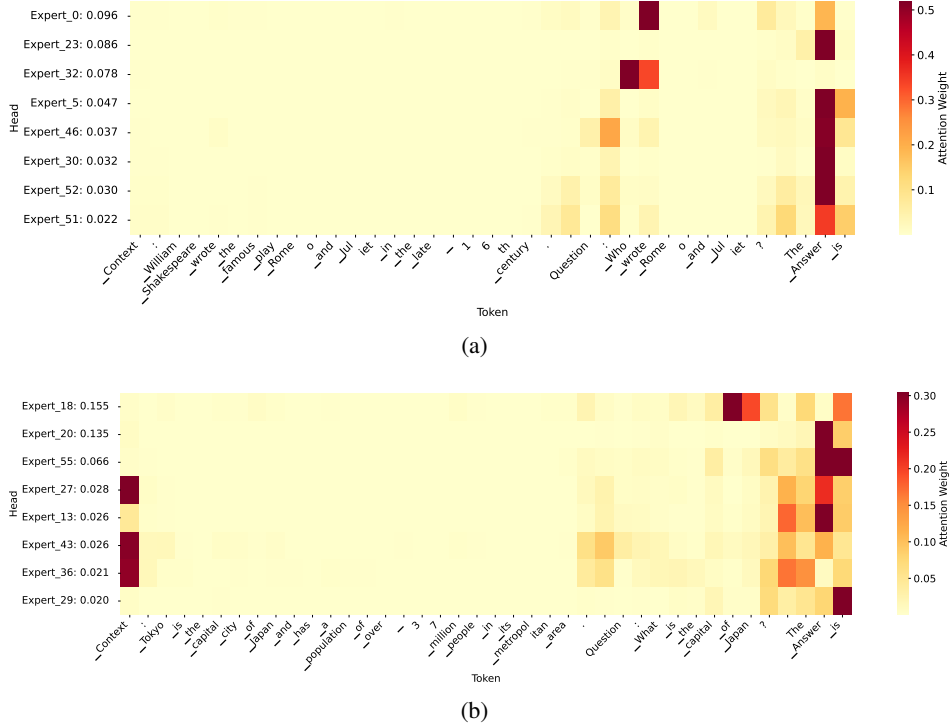


Figure 9: Representative attention maps. The heatmaps show the attention weights of the last token produced by top 8 experts, ranked by their router scores. (a) Attention patterns for the Shakespeare question, where higher-ranked experts (e.g., Expert_0, Expert_32) demonstrate focused attention on question-relevant tokens. (b) Attention patterns for the Tokyo question, showing similar task-specific attention concentration among top experts.

received by experts at each rank position across all layers. The accumulated patterns reveal that higher-ranked experts (particularly E0 and E1) maintain more targeted attention distributions focused on question-relevant tokens, while lower-ranked experts tend to focus heavily on the initial token,

Notably, we observe minimal attention paid to answer tokens present in the context. This phenomenon aligns with the conceptualization of experts (two consecutive matrices) as key-value memory modules, where input serves as a query. In other words, the output of attention layers is the composition of values, i.e., columns of the second matrix, in the activated experts, rather than token hidden states. This suggests that the token mixing should focus on building appropriate query for accurate compositions. Therefore, the last token should pay attention to the tokens relevant to the answer token, rather than the answer itself.

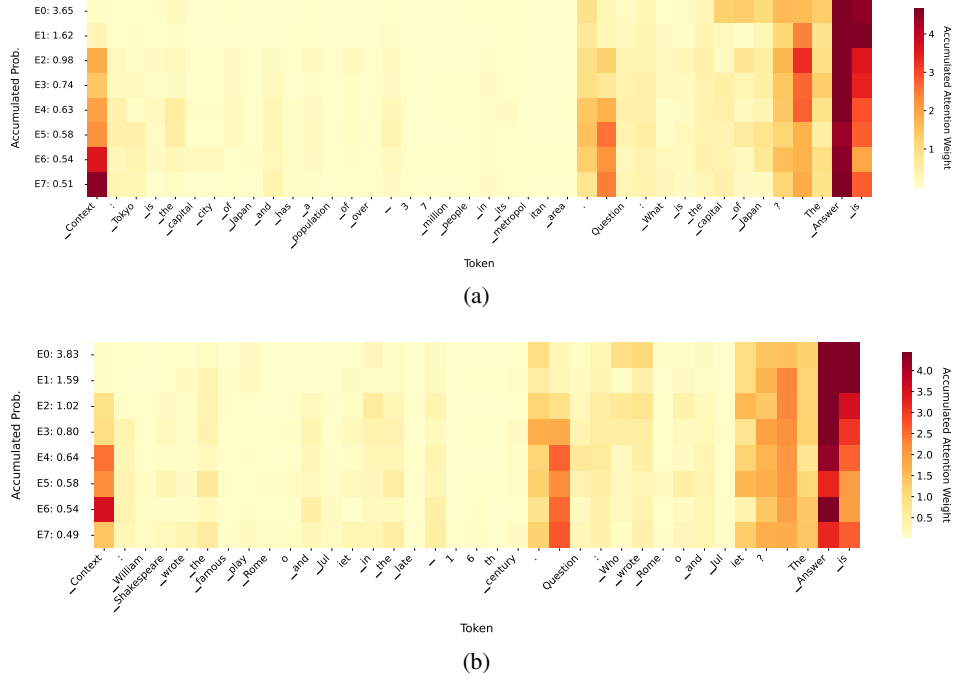


Figure 10: Layer-wise accumulated attention weights across the model. The values on the left (e.g., E0: 3.83) represent the sum of router scores for experts. Higher-ranked experts (E0, E1) consistently show more focused attention distributions compared to lower-ranked experts.

A.6 Limitations

One limitation of UMoE is, as discussed in Section 4.2, the modest additional computational cost introduced by reformulating the attention mechanism in relatively small models. Additionally, the datasets used in this paper do not cover mathematics or code, given the scope of our work and limited computational resources. Considering recent research on MoE models’ reasoning limitations, exploring how reasoning abilities scale with UMoE may provide valuable insights for future work.

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [\[Yes\]](#)

Justification: In the abstract and introduction, we claim that our proposed attention-MoE layers outperform previous attention-MoE methods and match or exceed the performance of FFN-MoE models.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [\[Yes\]](#)

Justification: The paper provides a "Limitations" section (A.6).

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: The paper does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: This paper proposes a novel model architecture. The details of experiments are provided in Section 4 and A.4. Our code is also available for reproduction.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The code is released. The datasets used by this paper are all publicly available.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: The details of experiments are provided in Section 4 and A.4.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: Considering we are performing language modeling pretraining, it would be too computationally expensive for this given our limited computational budget.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: Section A.4 provides information on the time of training and Table 12 provides the information of GPU.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [Yes]

Justification: The research conducted in the paper conform with the NeurIPS Code of Ethics.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification: There is no societal impact of the work performed since the focus of this work is to propose a general architecture.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: There is no model or data released.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We cite the datasets and the code package used for model evaluation. The FineWeb-Edu dataset is under ODC-By 1.0 license and the Wikitext-103 is released under CC BY-SA 3.0.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [\[Yes\]](#)

Justification: We released our code for reproduction with a document on how to run it.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [\[NA\]](#)

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [\[NA\]](#)

Justification: The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: LLMs are used only for writing.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>) for what should or should not be described.