Expert Attention: MoE-Based Head Decoupling and Pruning for Pretrained Encoders

Anonymous ACL submission

Abstract

Encoder-only models benefit from bidirectional attention, enabling high parallelism and strong throughput, making them suitable for large-scale supervised tasks. However, their inference efficiency remains a bottleneck in real-world deployment. We propose Expert Attention, a Mixture-of-Experts (MoE)–based method that decouples each attention head as an independent expert. A gating mechanism dynamically selects which heads to activate, guided by a two-stage training strategy of load balancing followed by specialization. After training, a Top-1 selection strategy prunes unused heads, significantly improving throughput. Unlike prior pruning methods, our approach is purely architectural—requiring no complex scoring functions—making it simple and practical. Experiments show that Expert Attention achieves substantial speedups with minimal performance loss, outperforming existing attention head pruning techniques.

1 Introduction

In recent years, large language models have made significant advances across a wide range of natural language processing (NLP) tasks. Decoder-only architectures such as Qwen, DeepSeek, and LLaMA (Yang et al., 2024; DeepSeek-AI et al., 2024; Touvron et al., 2023) have scaled to hundreds of billions of parameters, achieving remarkable results in generative benchmarks. However, the autoregressive nature of these models limits their inference efficiency. As model size increases, throughput tends to decline, making large decoder-only models less suitable for real-time or high-volume applications.

In contrast, encoder-only models such as BERT (Devlin et al., 2019) and XLM-R (Conneau et al., 2020), offer a structural advantage. Their bidirectional attention mechanism enables full parallelism during inference, making them ideal for large-scale classification and tagging tasks. In scenarios where generation is not required, encoder-only models deliver substantially better throughput compared to autoregressive architectures.

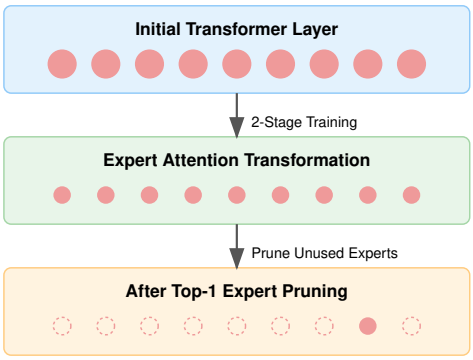

Figure 1: Expert Attention workflow: transforming a standard Transformer layer into expert-based modules, then pruning inactive experts to improve inference efficiency.

Despite these advantages, pretrained encoders still face performance bottlenecks in ultra-scale deployment scenarios - such as multilingual sentiment tagging over massive social media corpora - where both inference speed and model size become critical constraints. This challenge has spurred extensive research into model pruning and compression. Existing pruning techniques fall into two broad categories:

- **Coarse-grained pruning** techniques, such as layer dropping and attention head pruning (Michel et al., 2019a; Voita et al., 2019a), simplify the model structure but may lead to noticeable performance degradation.

- **Fine-grained pruning** methods, including structured sparsity and weight magnitude pruning (Zafrir et al., 2021; Sanh et al., 2020), typically require specialized inference frameworks and involve complex trade-offs between compression rate and real-world latency.

Alternatively, Mixture-of-Experts (MoE) models (Shazeer et al., 2017; Lepikhin et al., 2021) have recently gained attention for enabling capacity scaling without proportional increases in computation. However, most existing MoE-based methods focus on *expansion*—adding FFN-style adapters or routing layers—to improve performance, rather than enabling model compression. Moreover, MoE has rarely been applied at the granularity of attention heads, leaving a structural gap unaddressed.

Previous approaches to attention head pruning primarily rely on predefined importance metrics, auxiliary losses, or supervised signals to determine which heads should be pruned (Michel et al., 2019a; Voita et al., 2019a; Li et al., 2021a). While these methods effectively reduce redundancy within attention layers, they often overlook the feed-forward networks (FFNs), which account for the majority of parameters and computation in Transformer architectures. To address these limitations, we introduce a novel framework that conceptualizes attention heads as independent experts, allowing dynamic selection and pruning, as depicted in Figure 1.

- We propose **Expert Attention**, a novel MoE-based restructuring of Transformer layers, where each attention head is decoupled into a standalone expert module, paired with its own feed-forward block.
- We design a **progressive, layer-wise MoE transformation and training schedule**, combining load balancing and expert specialization, to gradually integrate the MoE structure into pretrained encoders.
- We introduce a **Top-1 expert pruning strategy** based on actual usage frequency, enabling aggressive and efficient model compression without requiring complex scoring functions or auxiliary supervision.
- Our method is **purely architectural** - easy to implement, hardware-agnostic, and free of heavy dependencies - while achieving significant gains in inference throughput and parameter efficiency, outperforming existing attention head pruning techniques.

Overall, our approach demonstrates that MoE is not only a tool for scaling up models but also a powerful mechanism for structured compression when applied at the right granularity.

2 Related Works

Attention Head Pruning Transformer-based models have demonstrated that not all attention heads contribute equally to performance. Michel et al. (2019a) revealed that many attention heads could be pruned with little to no degradation in performance. Voita et al. (2019a) introduced auxiliary losses to identify and prune underperforming heads, while HeadMask (Michel et al., 2019a) utilized learnable head masking. Li et al. (2021a) proposed a dynamic, end-to-end learnable pruning mechanism that integrates pruning decisions into the training loss.

Transformer Compression Numerous efforts have been made to compress Transformer models for efficient deployment. Coarse-grained strategies include LayerDrop (Fan et al., 2019a) and DynaBERT (Hou et al., 2020), which drop layers or adapt width and depth during runtime. In contrast, fine-grained approaches such as Movement Pruning (Sanh et al., 2020) and quantization-based methods like Q8BERT (Zafrir et al., 2019) aim at reducing parameter counts while preserving precision. However, these methods often incur non-negligible implementation complexity and may require custom inference frameworks.

Mixture-of-Experts (MoE) Architectures MoE architectures provide a scalable way to increase model capacity with conditional computation. GShard (Lepikhin et al., 2020) and Switch Transformer (Fedus et al., 2022) activate only a subset of experts for each input, significantly reducing computation. TaskMoE (Zhao et al., 2022) further enhances MoE for multi-task settings. Despite their success, these designs are mainly focused on feed-forward (FFN) layers and are rarely applied to the attention mechanism itself.

MoE for Compression and Specialization Recent efforts have begun to leverage MoE not only for expansion but also for compression. Sparse MoE approaches (Roller et al., 2021) and hashing techniques (Riquelme et al., 2021) reduce computation by sparsifying expert selection. Yet, these methods primarily address FFN modules and ignore redundancy in attention heads, leaving a gap in structured compression strategies at the attention level.

Our Contribution Our method bridges the gap between MoE scalability and Transformer pruning.

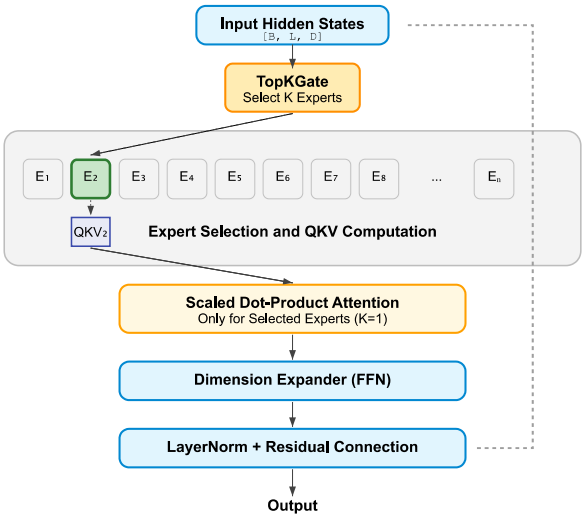

Figure 2: Overview of the Expert Attention mechanism. The input hidden states are routed by a Top-k gate to select the most relevant expert heads. Only the selected experts compute QKV projections and perform self-attention. The resulting outputs are passed through a shared Expander FFN, which restores the hidden dimension and replaces the original feed-forward network. Finally, layer normalization and residual connection are applied to produce the final output.

By treating each attention head as an independent expert and introducing a shared, lightweight Expander FFN, we enable dynamic expert selection and post-training pruning without requiring auxiliary losses or supervision. This architectural-level restructuring complements previous work and introduces a novel compression perspective in Transformer models.

3 Method

3.1 Expert Attention

We propose **Expert Attention**, a modular reformation of the Transformer encoder layer that replaces the standard multi-head attention and shared feed-forward structure with a sparse, expert-driven computation framework, as illustrated in Figure 2.

In a standard Transformer layer, multiple attention heads are computed in parallel and merged, followed by a shared feed-forward network (FFN). In contrast, we **decouple each attention head** into an independent computation path. Each head is paired with a dedicated *expander FFN*, forming a standalone expert module. These experts are dynamically selected at runtime using a gating mechanism, allowing only the most relevant ones to be activated for a given input.

This design introduces three key components:

- **Independent attention experts**, each with its own self-attention computation.
- **Expander FFNs**, which restore output dimensionality and replace the original FFN.
- **Top-k gating**, which selects a sparse subset of expert modules per input.

The output of the selected expert paths is aggregated, normalized, and combined with residual connections. This structure supports sparse inference, facilitates specialization, and enables expert-level pruning with minimal performance drop. We detail each of these components in the following subsections.

3.1.1 Attention as Expert Module

In Expert Attention, we reinterpret each attention head as an independent expert module. Unlike standard multi-head attention, where heads are computed jointly and fused, our design detaches each head into a self-contained unit with its own query, key, and value projections:

$$Q_i = XW_i^Q, \quad K_i = XW_i^K, \quad V_i = XW_i^V$$

Each expert computes scaled dot-product attention individually:

$$\text{Head}_i(X) = \text{softmax}\left(\frac{Q_i K_i^\top}{\sqrt{d_k}}\right) V_i$$

These heads are not concatenated or merged. Instead, they act as standalone expert branches that will be selectively routed during inference.

By treating each head independently, we allow each to specialize without interference from others. This modularization also lays the foundation for subsequent pruning, as unused heads can be easily removed.

3.1.2 Expander FFN

In standard Transformer layers, the feed-forward network (FFN) is a two-layer fully connected block applied after the multi-head attention output, projecting from the hidden dimension d to an intermediate dimension and back to d. This design introduces a significant number of parameters and dominates the overall model size.

In Expert Attention, we replace the original FFN with a simplified and shared **Expander FFN**, which serves two purposes:

1. **Dimensional adaptation:** Since each attention expert produces an output of shape (B, L, d_{head}), where $d_{\text{head}} = d/h$, we need to project this lower-dimensional output back to the full hidden dimension d. This projection ensures compatibility with the residual connection and layer normalization applied afterward.

2. **Functional substitution:** The Expander FFN also replaces the original FFN, providing the necessary non-linear transformation and capacity to support downstream representation learning.

3. **Parameter efficiency:** Unlike the original FFN, which maps $d \to d_{\text{ff}} \to d$, our Expander FFN only maps $d_{\text{head}} \to d_{\text{ff}} \to d$. This greatly reduces the number of parameters, especially when $d_{\text{head}} \ll d$.

Formally, the Expander FFN is defined as:

$$\text{ExpFFN}(x) = W^2 \cdot \text{GELU}(W^1 x)$$

where $W^1 \in R^{d_{\text{ff}} \times d_{\text{head}}}$ and $W^2 \in R^{d \times d_{\text{ff}}}$. All experts share the same Expander FFN parameters, which promotes efficiency while maintaining model capacity through sparse expert selection.

This lightweight design enables the model to remain modular, prune-friendly, and highly efficient in inference, as shown in Figure 2.

3.1.3 Top-1 Gating

To enable efficient expert selection and support pruning, we adopt a simplified routing strategy inspired by Switch Transformer (Fedus et al., 2022). Instead of using softmax-based weighting over multiple experts, we select only the single most relevant expert (i.e., $k = 1$) for each input.

The gating mechanism takes the representation of the [CLS] token, $x_{\text{CLS}} \in R^d$, and projects it to a score vector over N experts:

$$g = \text{Linear}(x_{\text{CLS}}) \in R^N$$

We then select the index of the top-scoring expert:

$$i^* = \arg\max_j g_j$$

No softmax is applied, and no weighting is performed. The selected expert i^* alone is responsible for processing the input. This hard routing design not only reduces runtime computation but also allows unused experts to be completely removed after training.

3.1.4 Integration and Output

As illustrated in the bottom portion of Figure 2, once the top-1 expert i^* is selected, its attention head processes the input and passes the result through the shared Expander FFN:

$$E = \text{ExpFFN}(\text{Head}_{i^*}(X))$$

We then apply residual connection and layer normalization, using the original input X:

$$Y = \text{LayerNorm}(E + X)$$

This design ensures that only a single expert is executed per layer and per input. Because experts are fully decoupled, those that are rarely or never selected can be pruned entirely—enabling efficient post-training compression without any structural reconfiguration.

3.2 Further training

Training an Expert Attention model poses unique challenges due to the sparse and modular nature of its computation. In particular, directly applying MoE transformation to all layers at once often leads to instability or training collapse, especially when fine-tuning on downstream tasks. To address this, we adopt a progressive training strategy with two critical components: **layer-wise MoE conversion** and a **two-stage optimization schedule**.

Layer-wise MoE Conversion. Instead of transforming all encoder layers into Expert Attention modules simultaneously, we convert one layer at a time in a staged manner. At each training epoch, one additional Transformer layer is replaced by its MoE counterpart. Once all target layers have been converted, we continue training the full model for a small number of extra epochs (typically 2) to stabilize learning and allow for full adaptation.

For example, when training a 12-layer encoder with 6 MoE-converted layers over 8 epochs, we proceed as follows:

- Epochs 1–6: progressively convert 1 new layer per epoch.

- Epochs 7–8: full MoE model trained as-is for final optimization.

This approach prevents sudden optimization shocks, allowing each newly introduced expert layer to adapt gradually to the model's dynamics.

Two-Stage Learning: Load Balancing → Specialization. We further split the training process into two functional stages to align with the final goal of pruning underutilized experts:

- **Stage 1 – Balanced Exploration.** During the layer-wise conversion phase (before all target layers are MoE-enabled), we apply a load balancing loss to encourage even expert utilization. This prevents early expert collapse and ensures that each expert gets adequate learning signals. The loss encourages the routing gate to distribute inputs more uniformly across available experts.

- **Stage 2 – Expert Specialization.** After all MoE layers are in place, we disable the load balancing objective and allow the gating mechanism to focus solely on task performance. In this phase, experts naturally specialize, and frequently selected experts are reinforced, while unselected ones begin to fade. This directly supports the goal of pruning, where underused experts can later be removed without harming performance. Importantly, the Top-1 gating strategy is preserved throughout to ensure sparsity and modularity.

This two-phase training process bridges the gap between robust early learning and efficient late-stage specialization. It ensures both high-quality optimization and structural sparsity, laying the groundwork for expert pruning in subsequent steps.

3.3 Expert Pruning

After training, many experts in the MoE-enabled layers are observed to be rarely or never activated. To reduce model size and accelerate inference, we propose a usage-driven pruning strategy that removes underutilized experts in a structured and interpretable manner.

Usage-based Expert Selection. Each MoE layer routes inputs to one of several expert heads based on a Top-1 gating decision. After training, we analyze validation-time expert usage across all layers by recording how frequently each expert is selected. For each layer, we rank experts by their activation frequency and identify those that contribute meaningfully to inference.

Pruning Policy. For every MoE layer, we retain only the top-m most frequently used experts and discard the rest. The choice of m can be tuned to control the trade-off between model compression and performance retention. Since our architecture uses hard routing (Top-1) and each expert is structurally independent, the unused branches can be safely removed without any additional fine-tuning.

Architecture Simplification. Following pruning, we statically replace each dynamic MoE layer with a simplified deterministic attention layer that only includes the selected experts. This change eliminates the runtime cost of gating and makes inference more predictable and hardware-efficient.

Effectiveness. This approach yields substantial reductions in parameter count and inference latency. Unlike unstructured pruning, our method operates at the architectural level—making the resulting model easy to deploy and analyze. Experimental results show that pruning has minimal impact on downstream task performance while significantly improving throughput and efficiency.

4 Experiments

4.1 Expert Attention vs Traditional Methods

In this section, we compare our method, **Expert Attention**, with several traditional attention head pruning techniques. All experiments are conducted on the BERT-base-uncased model as the baseline, and pruning is performed on the MNLI task. The pruning and fine-tuning results for various baseline methods are sourced from the study in (Li et al., 2021a). Our goal is to compare the performance of traditional attention head pruning methods with the newly proposed Mixture-of-Experts (MoE) approach, highlighting the difference in their efficiency and performance under identical conditions.

The baselines considered for this experiment include:

- **Michel et al.**: This method prunes attention heads based on predefined importance scores. The importance score is computed by using gradient-based analysis to remove the least important heads. (Michel et al., 2019b)

- **Pipelined DSP**: A dynamic sparse pruning method that reduces the number of attention heads during the post-training phase while attempting to preserve the model's performance. It prunes heads based on an iterative strategy, adjusting the pruning thresholds.(Li et al., 2021b)

- **Voita et al.**: This method uses auxiliary losses during training to identify and prune underperforming heads. It utilizes a binary gating mechanism to control which heads remain active during training. (Voita et al., 2019b)
- **STE (Straight-Through Estimator)**: A pruning method that applies threshold-based pruning using a straight-through estimator to approximate gradients in the binary decision process of pruning heads. (Fan et al., 2019b)
- **Joint DSP**: This approach dynamically prunes heads based on a joint optimization strategy, where the pruning decisions are integrated into the model's training loss, making pruning an end-to-end learnable process. (Li et al., 2021a)

As shown in Table 1, when comparing the Expert Attention method with the baseline methods using the same Unpruned Heads values, we observe the following:

While Joint DSP achieves some of the best performance metrics in terms of accuracy, **Expert Attention significantly outperforms the traditional pruning methods in terms of model size reduction**. Our method benefits from an architecture reconfiguration, where pruning is performed at the layer level, along with an attention head decoupling and MoE-based model pruning approach. This allows Expert Attention to achieve a much larger reduction in model size compared to Joint DSP and other baselines, while retaining superior accuracy. For example, when Expert Attention retains only 12 unpruned heads, it achieves an 88.5% reduction in model size, which is much greater than the reduction seen in Joint DSP and other baseline methods.

Moreover, when comparing methods that result in the same model size reduction (e.g., Model Size Reduction = 16%, corresponding to Joint DSP's unpruned heads = 72 and our method's unpruned heads = 120), **Expert Attention clearly outperforms the baselines in terms of accuracy**. This demonstrates the effectiveness of our MoE-based approach in reducing model size while maintaining high performance, even with aggressive pruning strategies.

In Table 2, we detail the effect of different MoE-layer configurations on throughput and model performance. This analysis focuses on the increase in throughput as the number of MoE layers is increased. From the results, we observe a marked improvement in throughput as more MoE layers are added, along with a significant reduction in model parameters.

In extreme pruning scenarios, for example, when 11 out of 12 layers are pruned (i.e., layer 11/12), our model achieves a remarkable 88.50% reduction in model parameters and a 332.84% increase in throughput, while still retaining 81.11% of its accuracy. In comparison, Joint DSP (1), which prunes aggressively, achieves only a 26% reduction in model size and 72.78% of its accuracy, with a throughput of approximately 1995 inferences/sec (33% increase).

This indicates that our method not only reduces more parameters (88.50% vs. 26%) but also significantly outperforms Joint DSP (1) in terms of throughput (332.84% increase vs. 33% increase) while retaining a higher level of accuracy (81.11% vs. 72.78%).

This demonstrates the power of our MoE-based approach, which provides a better balance between computational efficiency and model performance. Our method handles very aggressive pruning with far more effective parameter reduction and throughput improvements, while still maintaining a high level of performance compared to traditional methods like Joint DSP.

4.2 Experiments on XLM-R-Base Model

In this experiment, we apply our proposed method and framework to the XLM-R-Base model, another widely-used pretrained encoder-only model with 12 attention heads and 12 layers. This aims to demonstrate that our approach is not restricted to a single model but is broadly effective across different encoder-only architectures. Building upon the results shown in Section 4.1 with BERT, this experiment validates that our Expert Attention technique consistently preserves performance and improves efficiency in diverse multilingual settings. We focus particularly on assessing how pruning affects model accuracy, parameter efficiency, and inference throughput, offering a comprehensive analysis of the trade-offs introduced by our method across different encoder-only pretrained models.

For the dataset, we use the Unified Multilingual Sentiment Analysis Benchmark (UMSAB(Team, 2024)), which is a comprehensive benchmark for evaluating sentiment analysis models across multiple languages.UMSAB integrates sentiment-annotated tweets from eight languages, including Arabic, English, French, German, Hindi, Italian,

Unpruned Heads	Michel et al.	Pipelined DSP	Voita et al.	STE	Joint DSP	Expert-attention (ours)	Model Size Reduction (base vs ours)
120	84.6	84.41	84.18	84.59	84.97	85.01	4% / **16.6%**
96	84.24	83.27	82.95	83.93	84.41	84.09	8% / **32.6%**
72	82.47	82.95	83.24	82.81	83.48	82.41	13% / **48.6%**
48	79.26	79.1	76.08	82.31	83.22	79.49	17% / **64.5%**
36	70.82	76.29	31.68	82.20	82.51	74.68	19% / **80.5%**
12	40.59	56.29	76.91	73.79	79.74	68.53	24% / **88.5%**

Table 1: Comparison of attention head pruning methods with Expert Attention on the BERT-base-uncased model, evaluated on the MNLI task. The table compares pruning techniques at different unpruned head levels, showing their impact on accuracy (Acc) and model size reduction.

Model (Unpruned Heads)	Throughputs/seds	Model Size Reduction (%)	Acc	Accuracy Performance Retained (%)
bert-base-uncased(144)	1500	0	84.90	/
Joint DSP (1)	$\approx$ 1995 ($\uparrow$33%)	-26%	61.79	72.78%
layer 2/12 (120)	1787.13 ($\uparrow$16.56%)	16.60%	**85.01**	**100.12%**
layer 4/12 (96)	2150.64 ($\uparrow$40.26%)	-32.60%	84.09	99.05%
layer 6/12 (72)	2603.85 ($\uparrow$40.26%)	-48.60%	82.41	97.14%
layer 8/12 (48)	3211.63 ($\uparrow$109.47%)	-64.50%	79.49	93.63%
layer 10/12 (36)	4933.59 ($\uparrow$221.78%)	-80.50%	74.68	87.97%
layer 11/12 (12)	**6636.58 ($\uparrow$332.84%)**	**-88.50%**	68.53	81.11%

Table 2: Performance comparison of pruning methods in terms of throughput, accuracy, and model size reduction. The first two rows represent baseline methods: BERT-base-uncased (144) (no pruning) and Joint DSP (1) (extreme pruning with only one unpruned head). The remaining rows show the results for MoE-based layer pruning, where the number of unpruned heads per layer is progressively reduced (e.g., layer 2/12 means 2 unpruned heads out of 12). The table highlights the effects of pruning on throughput (inferences/sec, measured with batch size 128), model size reduction, and accuracy retention, with Expert Attention achieving higher throughput and more substantial model size reductions while maintaining a competitive level of accuracy.

Portuguese, and Spanish. Each dataset within UMSAB is designed for three-way sentiment classification (positive, negative, and neutral) and is derived from existing sentiment analysis datasets such as SemEval, Deft, SB-10K, SAIL, Sentipolc, SentiBR, and InterTASS. This diverse multilingual testbed is particularly suitable for assessing the performance of sentiment classification models in cross-lingual settings.

Following the training framework described earlier, we apply the Expert Attention method to fine-tune the model at the layer level. Specifically, we progressively transform 2, 4, 6, 8, 10, and 11 layers into MoE layers, and conduct experiments at each stage to evaluate the impact of our approach.

Analysis of Experimental Results Figures 3 (a) and (b) summarize the key performance and efficiency metrics observed during the progressive layer-wise MoE transformation and pruning of the XLM-R-Base model.

From Figure 3 (a), we observe that the **F1 score remains relatively stable as the number of modified layers increases from 2 to 10, both before and after pruning**. The model effectively retains most of its classification performance despite the incremental introduction of the Expert Attention modules. However, once 11 layers are modified, there is a noticeable drop in the F1 score, indicating that aggressive pruning at this stage leads to a decline in task performance.

In terms of throughput, the benefits of our method become clear: **the throughput steadily increases with the number of MoE layers, rising from around 900 samples/sec to over 3500 samples/sec** after pruning at 11 modified layers. This represents a substantial improvement in inference speed compared to the baseline throughput of approximately 700 samples/sec, highlighting the efficiency gains from pruning redundant attention heads.

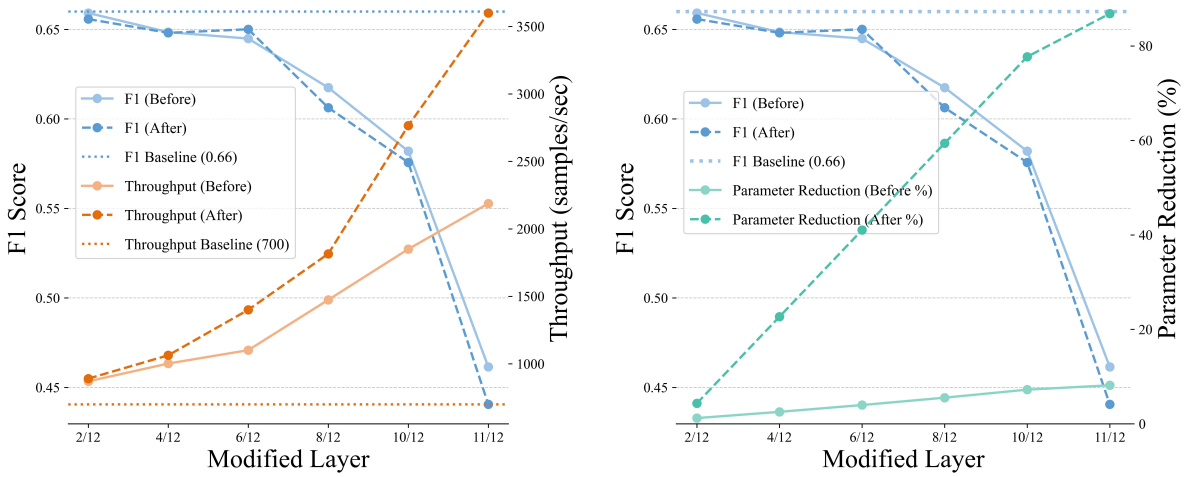

Figure 3: (a) F1 score vs throughput before and after pruning (left); (b) F1 score vs parameter reduction before and after pruning (right).

Figure 3 (b) further illustrates the relationship between parameter reduction and model performance. The parameter count reduction before pruning is modest when few layers are modified, but it grows significantly with more layers transformed. After pruning, the parameter reduction is much more pronounced, reaching up to nearly 90% when 11 layers are modified. Notably, this substantial compression incurs only a moderate performance drop until the last pruning step, which aligns with the trend observed in Figure 3 (a).

Overall, these results demonstrate that our Expert Attention approach can effectively compress the XLM-R-Base model, enhancing inference efficiency while preserving strong multilingual sentiment classification performance across most pruning levels. However, when comparing with the previous experiments on the single-language BERT model, we observe a notable difference in performance retention. Specifically, under extreme pruning (modified layer = 11), the BERT-base model maintains approximately 81.11% of its original performance on the MNLI task, whereas the XLM-R-Base model retains only about 69.9% on the multilingual sentiment classification benchmark. This larger performance degradation in XLM-R-Base likely stems from the added complexity of handling multiple languages simultaneously, which demands richer and more diverse representations. Consequently, aggressive pruning in a multilingual setting poses greater challenges for preserving model accuracy, emphasizing the need for careful pruning strategies tailored to multilingual scenarios.

For a detailed comparison of F1 scores across individual languages, please refer to Appendix Table 3, which provides a comprehensive breakdown of the per-language performance difference.

5 Conclusion

We propose *Expert Attention*, a Mixture-of-Experts (MoE) architecture that decouples each attention head into an independent expert module while sharing a common Expander feed-forward block. Combined with a dynamic Top-1 gating mechanism and progressive layer-wise training, this design enables effective expert specialization and load balancing. Unlike prior methods, our simple yet effective pruning strategy removes underutilized experts based solely on usage frequency, without relying on complex masking or scoring functions, thereby significantly reducing model size and improving inference throughput with minimal overhead.

Experiments on BERT-base and XLM-R-Base validate the effectiveness and generality of our approach. Expert Attention achieves up to 90% parameter reduction while maintaining strong task performance. The trade-off between accuracy and efficiency is more pronounced for the multilingual XLM-R-Base under extreme pruning, reflecting the challenge of preserving multilingual representations.

Overall, our method provides a practical, architecture-level solution for structured compression of Transformer encoders, balancing efficiency and performance, and facilitating deployment in real-world applications where speed and model size are critical.

6 Limitations

Our Expert Attention method has so far been designed and evaluated primarily on encoder-only Transformer models. Extending it to decoder-only or encoder-decoder architectures introduces additional challenges due to the presence of components such as cross-attention and autoregressive mechanisms. These factors complicate direct adaptation and require more sophisticated approaches.

We recognize these challenges and plan to address them in future work by exploring finer-grained gating mechanisms and adaptive pruning strategies tailored to these architectures and specific downstream tasks. This will help broaden the applicability and impact of our method.

References

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 8440–8451. Association for Computational Linguistics.

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, and 81 others. 2024. Deepseek-v3 technical report. *CoRR*, abs/2412.19437.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Angela Fan, Edouard Grave, and Armand Joulin. 2019a. Reducing transformer depth on demand with structured dropout. *arXiv preprint arXiv:1909.11556*.

Angela Fan, Edouard Grave, and Armand Joulin. 2019b. Reducing transformer depth on demand with structured dropout. *arXiv preprint arXiv:1909.11556*.

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39.

Lu Hou, Lifeng Ma, Qun Zhou, Yanyan Song, Jing Liu, Xiaodong Li, and Xuanjing Huang. 2020. Dynabert: Dynamic bert with adaptive width and depth. In *Advances in Neural Information Processing Systems*, volume 33, pages 9782–9793.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2021. Gshard: Scaling giant models with conditional computation and automatic sharding. In *ICLR*.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Melvin Johnson, Wolfgang Macherey, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2020. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*.

Jiaoda Li, Ryan Cotterell, and Mrinmaya Sachan. 2021a. Differentiable subset pruning of transformer heads. *Trans. Assoc. Comput. Linguistics*, 9:1442–1459.

Junnan Li, Ryan Cotterell, and Mrinmaya Sachan. 2021b. Differentiable subset pruning of transformer heads. *Transactions of the Association for Computational Linguistics*, 9:1027–1041.

Paul Michel, Omer Levy, and Graham Neubig. 2019a. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems*, volume 32.

Paul Michel, Omer Levy, and Graham Neubig. 2019b. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems*, pages 14014–14024.

Carlos Riquelme, Akhilesh Srinivas, Shakir Mohamed, Jonathan Heek, Andrew Brock, Aurko Roy, Barret Zoph, and Noam Shazeer. 2021. Hash layers for large sparse models. In *Advances in Neural Information Processing Systems*, volume 34, pages 13976–13990.

Stephen Roller, Joseph G Sartran, Kurt Shuster, Eric Michael Smith, Y-Lan Boureau, Jason Weston, and Emily Dinan. 2021. Sparse mixture of experts are noisy learners. In *Proceedings of the MLSys 2023 Conference*.

Victor Sanh, Zhiqing Xu, Thomas Wolf, Teven Le Scao, Sylvain Gugger, Mariama Drame, Siva Reddy Saini, and Alexander Rush. 2020. Movement pruning: Adaptive sparsity by fine-tuning. In *Advances in Neural Information Processing Systems*, volume 33, pages 20378–20389.

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

UMSAB Team. 2024. Unified multilingual sentiment analysis benchmark (umsab). https://github.com/UMSAB/UMSAB. Accessed: 2025-05-19.

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, and 49 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019a. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. *arXiv preprint arXiv:1905.09418*.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019b. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5797–5808.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, and 40 others. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Ofir Zafrir, Guy Boudoukh, Peter Izsak, and Moshe Wasserblat. 2019. Q8bert: Quantized 8bit bert. *arXiv preprint arXiv:1910.06188*.

Ofir Zafrir, Guy Boudoukh, Peter Izsak, and Moshe Wasserblat. 2021. Prunebert: Compressing bert by progressive module pruning. *arXiv preprint arXiv:2106.16113*.

Wayne Xin Zhao, Jingyuan Zhang, Jing Liu, Kelong Wang, Yanyan Lan, Junjie Liu, Sheng Ma, Maosong Sun, and Zhiyuan Liu. 2022. Taskmoe: Learning task-specific experts for multi-task learning. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1750–1763.

A Training Details

All experiments were conducted on an NVIDIA GeForce RTX 4090 GPU with 24GB memory, running Ubuntu 20.04, CUDA 11.7, and PyTorch 2.3.

Key training settings are as follows:

- Batch size: 128
- Optimizer: AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.999$
- Learning rate: initial 1×10^{-4}
- Learning rate warm-up: linear warm-up during the first epoch, increasing from 1×10^{-5} to 1×10^{-4}

Language	2/12 layer		6/12 layer		8/12 layer		10/12 layer		11/12 layer	
	Original	Pruned	Original	Pruned	Original	Pruned	Original	Pruned	Original	Pruned
ar	0.6283	0.6271	0.604	0.6177	0.5988	0.578	0.573	0.5473	0.4731	0.4336
de	0.7302	0.7224	0.7062	0.7011	0.6901	0.6644	0.6426	0.656	0.4691	0.4256
en	0.6863	0.6463	0.6605	0.6568	0.6642	0.6271	0.642	0.6422	0.4961	0.4337
es	0.6449	0.6431	0.6194	0.6335	0.5847	0.5546	0.5683	0.5579	0.457	0.4154
fr	0.6839	0.6945	0.6895	0.6945	0.6606	0.6212	0.5777	0.5657	0.4452	0.4186
in	0.47	0.4797	0.5035	0.4991	0.4638	0.4531	0.4503	0.4645	0.3872	0.3735
it	0.6442	0.6596	0.6673	0.6927	0.6159	0.5801	0.5806	0.5747	0.4656	0.4054
pt	0.7002	0.7123	0.7089	0.7051	0.6617	0.6423	0.6218	0.5976	0.4985	0.4293
average	0.6485	0.6481	0.6449	0.6501	0.6175	0.5901	0.582	0.5757	0.4615	0.4168

Table 3: Language Performance Across Different Layers