Improving Recursive Transformers with Mixture of LoRAs

Anonymous ACL submission

Abstract

Parameter sharing in recursive transformers reduces model size but collapses layer-wise expressivity. We propose Mixture of LoRAs (MoL), a lightweight mechanism that uses low-rank experts to simulate a Mixture-of-Experts layer. By replacing feed-forward networks (FFNs) in the shared transformer with these conditional modules, we effectively restore the expressivity lost due to parameter sharing. We pretrain a modernised recursive architecture, ModernALBERT, integrating rotary embeddings, GeGLU, FlashAttention, and a distillation-based initialisation. Across GLUE, SQuAD-v2, and BEIR, ModernALBERT (50M–120M) achieves state-of-the-art performance among compact models and surpasses larger fully parameterised baselines. We also propose an expert-merging procedure that compresses MoL into a single adapter at inference while preserving accuracy, enabling efficient deployment. Our results show that conditional computation effectively restores the expressivity lost under aggressive parameter sharing in recursive transformers. [1]

1 Introduction

Large language models (LLMs) power modern NLP and deliver strong reasoning, understanding, and generation performance. Their large parameter counts and high memory use, however, make training, fine-tuning, and downstream adaptation computationally expensive (Vaswani et al., 2017; Brown et al., 2020; Sanh et al., 2020).

Parameter sharing reduces the memory footprint of transformer models by applying the same weights across layers instead of learning distinct parameters. ALBERT demonstrates this approach by using extensive cross-layer sharing to build compact yet strong language models (Lan et al., 2019). Analyses show, however, that matching the performance of fully parameterised transformers can force the shared layer to grow wider, increasing its hidden dimension and computational cost (Lan et al., 2019). This trade-off undermines efficiency and motivates approaches that preserve expressivity while minimising both parameters and computation.

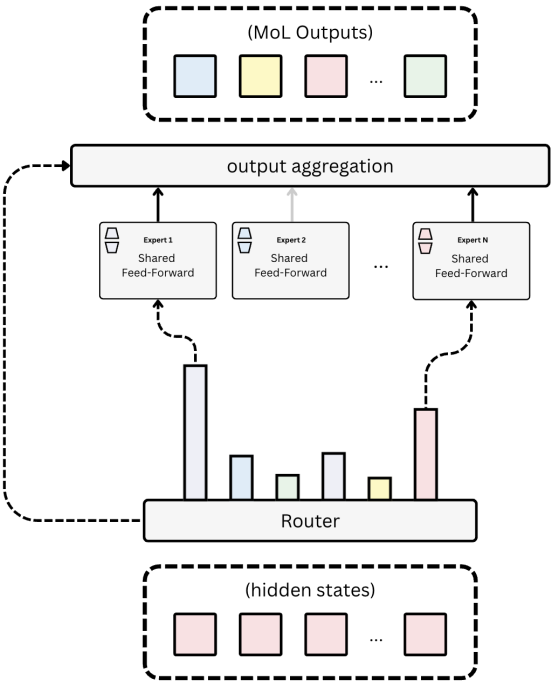

Figure 1: Structure of the Mixture of LoRAs (MoL) layer. Each MoL layer combines a shared feed-forward network (FFN) with several LoRA experts, allowing conditional computation without significantly increasing parameters.

Recent advances in large-scale models show the effectiveness of Mixture-of-Experts (MoE) architectures in enhancing transformer expressivity. By activating a subset of specialised expert modules for each input, MoE models significantly expand a transformer's representational capacity and deliver strong performance across diverse tasks (Shazeer et al., 2017; Fedus et al., 2021; Lepikhin et al.,

[1] Our models are available at LINK ANONYMISED

2020). Despite this success, MoE architectures introduce substantial parameter growth, making direct application to shared or lightweight architectures incompatible with the goal of parameter efficiency.

Existing approaches attempt to mitigate the expressivity loss from parameter sharing through static parameter-efficient techniques, such as MiniALBERT (Nouriborji et al., 2023) and Relaxed Recursive Transformers (Bae et al., 2024a). While these methods attach depth-specific parameters to shared blocks to partially restore layer-specific behaviour, they lack token-dependent routing and leave the shared feed-forward network (FFN) unmodified. Alternatively, dynamic parameter-efficient fine-tuning (PEFT) approaches that utilise mixtures of modules (Wang et al., 2022; Pfeiffer et al., 2020; Li et al., 2024) introduce token-level flexibility. Yet, these mechanisms are designed almost exclusively for downstream adaptation, rather than being leveraged to restore expressivity in a recursive architecture during pretraining. Building on these concepts, this work explores whether integrating a Mixture of LoRAs directly into the internal states of the FFN can more effectively restore the expressivity lost due to parameter sharing.

To investigate this, we introduce ModernALBERT, a recursive transformer that restores expressivity through Mixture of LoRAs (MoL), a mechanism for conditional computation within the shared architecture. We replace select FFNs in the shared transformer with MoL modules, embedding a small pool of LoRA experts directly into the network. By routing tokens to a sparse subset of these experts, MoL allows the shared backbone to exhibit distinct behaviours for different inputs. This approach reestablishes the layer-wise functional diversity that parameter sharing typically eliminates, all while preserving strict parameter efficiency (Hu et al., 2021).

We integrate several modern architectural components that improve training stability and runtime efficiency: gated GELU-style feed-forward networks (e.g., GeGLU) (Shazeer, 2020), rotary position embeddings (RoPE; Su et al., 2021), and I/O-aware attention implementations such as FlashAttention (Dao et al., 2022). Empirical results demonstrate that ModernALBERT achieves state-of-the-art performance among compact transformer models, outperforming prior baselines on GLUE (Wang et al., 2019), SQuAD (Rajpurkar et al., 2018), and BEIR (Thakur et al., 2021) benchmarks. Our key contributions include:

- We introduce Mixture of LoRAs (MoL), a conditional-computation layer that integrates low-rank experts into a shared FFN, restoring expressivity in recursive transformers.
- We build ModernALBERT, a family of compact recursive models that integrate RoPE, GeGLU, and FlashAttention, and we release all code and training checkpoints.
- We provide model variants ranging from 50M to 120M parameters and evaluate them extensively across GLUE (Wang et al., 2019), SQuAD (Rajpurkar et al., 2018), and BEIR (Thakur et al., 2021).
- We show that MoL consistently improves performance over mixture-of-adapters and static parameter-efficient modules in controlled ablations, and we introduce a merging strategy that compresses experts into a single adapter for efficient deployment.

2 Related Works

Parameter-efficient language models were significantly advanced by ALBERT, which introduced factorised embedding parameterisation and aggressive cross-layer parameter sharing to dramatically reduce the memory footprint compared to models like BERT (Lan et al., 2019). However, this strict static parameter tying, while efficient, introduces a trade-off by limiting overall expressivity and capacity. Early recursive architectures, such as the Universal Transformer (Dehghani et al., 2018), also explored layer tying, but noted that matching the performance of fully parameterised models often required wider layers, increasing computational cost. This highlights a central challenge: static parameter sharing saves memory, but can create a performance bottleneck.

A parallel and orthogonal approach to scaling models efficiently is MoE (Shazeer et al., 2017; Lepikhin et al., 2020; Fedus et al., 2021). MoE models, such as the Switch Transformer, increase model capacity without a proportional rise in computation by routing each token to a sparse subset of experts. Conditional computation has also been adapted for parameter-efficient fine-tuning (PEFT), where experts are often small plug-in modules such as LoRA adapters (Hu et al., 2021; Houlsby

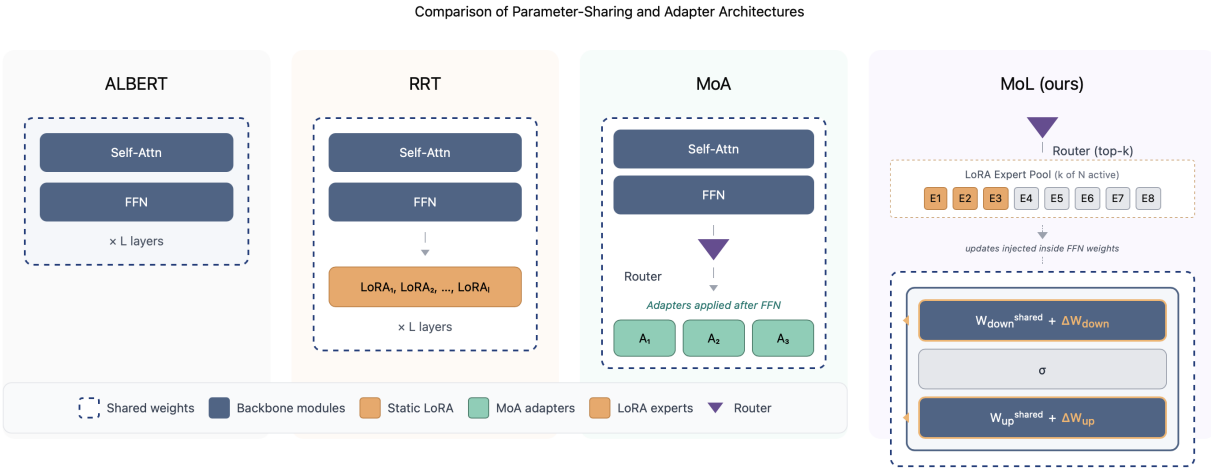

Figure 2: Comparison of parameter-sharing and adapter architectures. ALBERT shares a single self-attention and FFN block across all layers without adaptation. RRT augments the shared block with depth-specific static LoRA adapters. MoA keeps the shared backbone fixed and uses a router to select adapters applied after the FFN output. In contrast, MoL (ours) routes among a pool of LoRA experts whose low-rank updates are injected directly into the shared FFN weights, enabling token-conditional modulation while preserving parameter sharing in the backbone.

et al., 2019). Mixture-of-Adapters approaches (e.g., MoLE, MixLoRA, AdaMix, Adapter Fusion) have proven effective for multi-task fine-tuning (Wu et al., 2024; Li et al., 2024; Mehrish et al., 2023; Pfeiffer et al., 2020; Feng et al., 2024). However, these frameworks are predominantly designed to be applied on top of existing models, rather than being integral to the base architecture during pretraining.

Beyond modularity, modern architectural improvements have shown measurable gains in both efficiency and performance. Enhancements such as rotary positional embeddings, gated feed-forward networks, and I/O-optimised attention implementations reduce training overhead while improving model quality. ModernBERT demonstrates that these design choices can yield substantial performance gains over standard transformer baselines without increasing computational cost (Warner et al., 2024).

Recent work has explored the combination of parameter sharing with lightweight, layer-specific modules. Relaxed Recursive Transformers introduce static LoRA modules into a recursive shared block, pairing this architecture with early exiting strategies and sophisticated initialisation techniques for both the low-rank adapters and the underlying recursive network (Bae et al., 2024b). These additions demonstrate that depth-specific parameters can partially recover the performance lost due to strict parameter sharing while maintaining inference efficiency. Similarly, MiniALBERT employs bottleneck adapters to enhance the expressivity of shared backbones, showing that compact transformers can achieve strong performance when combined with targeted, low-cost modules (Nouriborji et al., 2023). While these approaches effectively augment parameter-shared models through static means, integrating dynamic conditional computation into a compact backbone remains an open direction.

ModernALBERT addresses this limitation by introducing an architecture that combines the parameter efficiency of ALBERT-style recursive sharing with the adaptive expressivity of token-conditional expert modulation.

3 ModernALBERT

ModernALBERT is a compact transformer architecture that combines recursive parameter sharing with lightweight, parameter-efficient modules to enhance representational capacity. We outline how recursive weight sharing reduces model size while preserving effective depth, then introduce the **Mixture of LoRAs (MoL)**, which injects flexibility into shared layers through MoE-style low-rank experts. We also describe several auxiliary techniques that further stabilise training and improve performance across model sizes: knowledge distillation, improved initialisation, and architectural refinements.

3.1 Parameter-Sharing through Recursions

Fully-parameterised transformers, such as BERT, have demonstrated strong performance across a wide range of benchmarks and tasks. A stan-

dard transformer layer consists of a multi-head self-attention (MHA) module followed by a feed-forward network (FFN), with residual connections and layer normalisation (LN) applied around each sub-layer (Nguyen and Salazar, 2019). Mathematically, the i-th layer can be expressed as:

$$h_i^{\text{att}} = h_{i-1} + \text{MHA}(\text{LN}(h_{i-1}); \theta_i) \quad (1)$$

$$h_i = h_i^{\text{att}} + \text{FFN}(\text{LN}(h_i^{\text{att}}); \phi_i) \quad (2)$$

For brevity, let the output of the full layer be:

$$h_i = \Psi(h_{i-1}; \theta_i, \phi_i), \quad (3)$$

where θ_i and ϕ_i denote the parameters of the attention and feed-forward modules for the i-th layer, respectively. Note that in the fully-parameterised formulation, each layer has its own distinct parameters, making deep models costly and hard to deploy. Let the hidden dimension be d and the FFN expansion rate be 4; then for a transformer of depth N, the parameter count is approximately:

$$|\Theta| \approx N\Big(4d^2 + 2 \times (4d^2)\Big) = 12Nd^2 \quad (4)$$

To address the high parameter cost of fully-parameterised transformers, parameters can be shared across multiple layers, allowing models to maintain depth while reducing the number of unique parameters. In a recursive transformer with group size G and total depth $K \cdot G$, the layers are divided into K groups, where all layers within a group share parameters. Denoting the shared attention and feed-forward parameters for group g as $\theta^{(g)}$ and $\phi^{(g)}$, the i-th layer (with i and g starting from 1) can be compactly written as:

$$h_i = \Psi(h_{i-1}; \theta^{(\lceil i/G \rceil)}, \phi^{(\lceil i/G \rceil)}) \quad (5)$$

In the extreme case of full parameter sharing (as in ALBERT), all layers share the same parameters, i.e., $\theta^{(g)} = \theta$ and $\phi^{(g)} = \phi$ for all layers. More generally, in a recursive transformer with total depth $K \cdot G$ and K groups of size G, the total number of unique parameters is approximately

$$|\Theta| \approx K \cdot (4d^2 + 8d^2) = 12Kd^2 \quad (6)$$

Compared to a fully-parameterised transformer with $N = K \cdot G$ layers, the recursive transformer reduces the number of unique parameters by a factor of K/N:

$$\frac{|\Theta|}{|\Theta_{\text{full}}|} \approx \frac{12Kd^2}{12Nd^2} = \frac{K}{N} = \frac{1}{G} \quad (7)$$

3.2 Improved Flexibility through Mixture of LoRAs

We address the loss of representational capacity from weight sharing by integrating Mixture-of-Experts (MoE) layers, a mechanism that has proven highly effective in recent large language models. A standard MoE layer consists of a router and E feed-forward modules called experts (Fedus et al., 2021). For a given input token representation h, the router assigns a probability $p_i(h)$ to each expert i, and the output of the MoE layer is a weighted sum of the expert outputs:

$$\text{MoE}(h) = \sum_{i=1}^{E} p_i(h) \cdot \text{FFN}_i(h) \quad (8)$$

where $p_i(h)$ is the routing weight for expert i and FFN_i is the feed-forward network corresponding to expert i. This allows different tokens to leverage different subsets of experts, effectively enabling conditional computation and increasing the model's flexibility.

Although MoE layers are highly effective, they come at the cost of significantly increasing the total number of parameters, which goes against our goal of building compact, efficient models. To retain the benefits of MoE while keeping the parameter count low, we adopt LoRA modules (Hu et al., 2021) to mimic the behavior of a mixture of experts. In this setup, a shared feed-forward network is augmented with multiple lightweight LoRA adapters, which act as distinct experts and enable conditional computation.

A standard LoRA module replaces a weight matrix $W \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$ with a low-rank update:

$$W' = W + \frac{\alpha}{r} AB \quad (9)$$

where $A \in \mathbb{R}^{d_{\text{in}} \times r}$ and $B \in \mathbb{R}^{r \times d_{\text{out}}}$ are trainable matrices of rank $r \ll d_{\text{in}}, d_{\text{out}}$, drastically reducing the number of additional parameters.

Building on this idea, we define a shared feed-forward network $\text{FFN}_{\text{shared}}$ with parameters $\theta =$

Variant	Layers	Groups	MoL Groups	Hidden Dim	Intermediate Dim / Expert Dim
Tiny	14	7	6, 7	768	1152 / 2624
Medium	12	3	3	1024	2624 / 4096
Base	16	4	3,4	1024	2624 / 4096
Large	24	6	3,4,5,6	1024	2624 / 4096

Table 1: Comparison of ModernALBERT model variants. MoL layers replace the shared FFNs at the indicated positions, each with 8 experts and top-2 routing.

$\{W_{\text{down}}, W_{\text{up}}\}$, and augment it with E LoRA modules $\psi_i = \{A_{i,1}, B_{i,1}, A_{i,2}, B_{i,2}\}$ for each expert i. Given an input h, the FFN modified by the i-th LoRA expert is:

$$\text{FFN}(h; \theta, \psi_i) = \left(W_{\text{up}} + \tfrac{\alpha}{r} A_{i,2} B_{i,2}\right) \sigma\Big(\left(W_{\text{down}} + \tfrac{\alpha}{r} A_{i,1} B_{i,1}\right) h\Big) \tag{10}$$

where r is the LoRA rank, α is a scaling factor, and σ is the activation function. Finally, the output of the Mixture of LoRAs (MoL) layer is computed by selecting experts via routing weights $p_i(h)$:

$$\text{MoL}(h) = \sum_{i=1}^{E} p_i(h) \cdot \text{FFN}(h; \theta, \psi_i) \tag{11}$$

We adopt a sparse top-2 routing strategy, selecting only the two experts with the highest routing weights for each token to further improve efficiency. Let $S_2(h)$ denote the indices of the top-2 experts for input h, then the sparse MoL output is:

$$\text{MoL}_{\text{top-2}}(h) = \sum_{i \in S_2(h)} p_i(h) \cdot \text{FFN}(h; \theta, \psi_i) \tag{12}$$

where $p_i(h)$ is re-normalised over the selected top-2 experts to sum to 1.

3.3 Effects of Knowledge Distillation and Initialisations

We found that initialising the parameters of ModernALBERT from a fully-parameterised model, in our case ModernBERT, significantly improves performance. We follow a step-wise initialisation strategy from the teacher model, where selected layers from the teacher are mapped to initialise the shared layer groups in ModernALBERT, providing a strong starting point for subsequent training. In addition to this, we apply knowledge distillation (Hinton et al., 2015; Sanh et al., 2020) from ModernBERT, using its predictions as soft targets to further guide the training of ModernALBERT. This combination of informed initialisation and distillation is critical for data efficiency, allowing the model to converge rapidly despite reduced training time and a pre-training budget of only 30B tokens (compared to the 1.7T tokens used for ModernBERT).

3.4 Training Settings

We adopt a multi-stage pre-training curriculum to maximise data efficiency within our limited token budget. The training process is divided into two distinct phases. We first warm up the model on the RedPajamas-1T dataset (Computer, 2023) for 20,000 to 30,000 steps. Following this initialisation phase, we transition to the RefinedWeb dataset (Penedo et al., 2023) for the remainder of the training, continuing for an additional 70,000 to 80,000 steps.

Throughout both stages, we maintain a fixed global batch size of 384 and a maximum sequence length of 1024 tokens. Optimisation is performed using the AdamW optimiser (Kingma and Ba, 2017). We employ a linear warmup schedule, peaking at a learning rate of 5×10^{-4} or 5×10^{-5}, followed by a linear decay. This regimen allows ModernALBERT to effectively adapt the distilled knowledge from the teacher model while refining the shared parameters and LoRA modules on high-quality web data.

3.5 ModernALBERT Variants

In addition to parameter-sharing and MoL, ModernALBERT incorporates several architectural improvements to enhance training stability, efficiency, and expressivity. We adopt pre-layer normalisation (Pre-Norm) (Nguyen and Salazar, 2019) and GeGLU activations (Shazeer, 2020) in the feed-forward networks to improve gradient flow and representation capacity. To further improve efficiency, we use rotary embeddings (Su et al., 2021) in the at-

tention mechanism and FlashAttention (Dao et al., 2022) for fast and memory-efficient attention computation.

The MoL layers are placed at the end of each group by replacing the shared FFN with a Mixture of LoRAs layer. Each MoL consists of 8 experts with top-2 routing, except the tiny model, which has 4 experts and a top-1 routing, enabling conditional computation and enhanced flexibility without significant parameter overhead.

We design four variants of ModernALBERT (Table 1) to target different model scales. This design balances model depth, parameter efficiency, and representational flexibility across different scales, enabling ModernALBERT to achieve high performance while maintaining compactness and efficiency.

4 Experimental Results

We evaluate ModernALBERT on natural language understanding, question answering, and information retrieval benchmarks. Our goal is to show that adding the Mixture-of-LoRAs (MoL) layer to a recursive, parameter-shared architecture provides substantial efficiency gains without sacrificing representational capacity. As shown in Table 2, ModernALBERT performs strongly across model scales.

4.1 GLUE Benchmark Results

We present GLUE benchmark results (Wang et al., 2019) in Table 2, where ModernALBERT performs strongly across model scales. The largest variant, ModernALBERT-large (120M parameters), achieves a GLUE average of **88.72**, surpassing the fully parameterised ModernBERT-base (149M parameters, 88.45) and outperforming previous compact models such as NomicBERT and MosaicBERT.

In the compact regime, ModernALBERT-tiny (50M) and ModernALBERT-medium (55M) consistently outperform lightweight baselines such as DistilBERT, TinyBERT, and MiniLM. Notably, ModernALBERT-tiny achieves the strongest overall performance among compact models with a GLUE average of **84.95**, surpassing MiniLM (33M, 84.49) despite having a different architectural design and stronger parameter sharing constraints.

The architectural benefits appear most clearly on tasks that require precise semantic matching and inference from limited data, where ModernALBERT-large achieves state-of-the-art results among base-class models on **RTE (88.44)**, **STS-B (92.1)**, and **MRPC (92.7)**.

For smaller datasets like CoLA and RTE, we further stabilise training by pre-training the router on MNLI and freezing its parameters. This allows the model to exploit a more robust, generalised routing policy, though we find the degree of improvement varies depending on the specific downstream objective. These optimisations preserve general capacity, keeping ModernALBERT competitive on high-resource tasks.

4.2 Question Answering Performance

We evaluate extractive question answering capabilities on SQuAD-v2 (Rajpurkar et al., 2018), as reported in Table 3. ModernALBERT-base achieves an F1 score of 92.8, outperforming the larger ModernBERT-base (92.6) and ALBERT-xxlarge (92.5). These results indicate that the Mixture-of-LoRAs approach transfers well to token-level classification tasks. ModernALBERT also reaches an Exact Match score of 86.1, showing that the model can localise answer spans accurately despite heavy parameter sharing.

4.3 Information Retrieval Results

Performance on the BEIR information retrieval benchmark (Thakur et al., 2021) is detailed in Table 4. ModernALBERT shows strong domain generalisation, achieving the best average performance among compact models on tasks such as TREC-COVID, FiQA, and ArguAna. Specifically, on ArguAna (argument retrieval), ModernALBERT scores 48.82, outperforming ModernBERT (35.7). This indicates that our routing mechanism allows the model to adapt effectively to specialised domains and shifting distributions, a capability often lacking in static recursive transformers.

4.4 Ablation Studies

To isolate the contributions of our architectural choices, we conducted ablation studies comparing the MoL design against alternative conditional computation strategies and initialisation methods (detailed comprehensively in Appendix B).

Our comparative analysis reveals that integrating LoRA experts directly into the weight space of the shared FFN (MoL) consistently outperforms the Mixture-of-Adapters (MoA) approach, where adapters are placed sequentially after the FFN. This suggests that modifying the internal representations of the shared layer facilitates a deeper and more

Model	Params	Seq.	Single Sentence		Paraphrase and Similarity			Natural Language Inference			Avg
			CoLA	SST-2	MRPC	STS-B	QQP	MNLI	QNLI	RTE	
Base Models											
BERT (Devlin et al., 2019)	110M	512	59.0	93.1	89.5	89.4	91.4	85.4	91.6	78.2	84.84
RoBERTa (Liu et al., 2019)	125M	512	63.6	94.8	90.2	91.2	91.9	87.6	92.8	78.7	86.35
DeBERTav3 (He et al., 2023a)	183M	512	**69.2**	95.6	89.5	91.6	**92.4**	**90.0**	**94.0**	83.8	88.26
MosaicBERT-128 (Portes et al., 2024)	137M	128	58.2	93.5	89.0	90.3	92.0	85.6	91.4	83.0	85.38
NomicBERT-2048 (Nussbaum et al., 2025)	137M	2048	50.0	93.0	88.0	90.0	92.0	86.0	92.0	82.0	84.13
GTE-en-MLM (Li et al., 2023)	137M	8192	57.0	93.4	92.1	90.2	88.8	86.7	91.9	84.8	85.99
ModernBERT (Warner et al., 2024)	149M	8192	65.1	**96.0**	92.2	91.8	92.1	89.1	93.9	87.4	88.45
Compact Baselines											
DistilBERT (Sanh et al., 2019)	66M	512	51.0	91.0	88.2	89.4	84.9	82.4	85.3	75.0	80.90
TinyBERT (Jiao et al., 2020)	66M	512	46.3	91.7	87.5	90.0	90.6	84.1	90.5	78.0	82.34
MiniLM (33M) (Wang et al., 2020)	33M	512	56.5	92.6	88.7	91.0	91.3	84.8	91.5	79.5	84.49
MiniLM (66M) (Wang et al., 2020)	66M	512	49.2	92.0	88.4	-	91.0	84.0	91.0	71.5	-
ModernALBERT (Ours)											
ModernALBERT-tiny	50M	1024	58.4	93.0	90.2	90.4	90.5	84.6	91.3	81.2	84.95
ModernALBERT-medium	55M	1024	62.3	94.0	91.0	91.2	91.0	86.6	92.0	81.6	86.21
ModernALBERT-base	75M	1024	64.6	95.2	91.7	91.8	91.5	88.2	93.0	85.6	87.70
ModernALBERT-large	120M	1024	66.4	95.5	**92.7**	**92.1**	92.0	88.9	93.7	**88.44**	**88.72**

Table 2: GLUE benchmark results for base, compact baselines, and ModernALBERT models. Scores are reported for each task category. **Avg** is the unweighted average across all tasks.

Model	F1 (%)	Exact (%)
BERT-base	88.6	80.6
RoBERTa-base	91.7	84.7
ALBERT-xxlarge	92.5	84.5
ModernBERT-base	92.6	85.2
Ours		
ModernALBERT-tiny	90.0	82.9
ModernALBERT-medium	90.4	82.9
ModernALBERT-base	92.8	**86.1**
ModernALBERT-large	**92.9**	85.9

Table 3: Results on the SQuAD-v2 Benchmark.

effective integration of conditional computation. Furthermore, scaling the expert count from a single expert to 8 experts with top-2 routing yields clear performance gains, confirming that the improvements stem from the dynamic routing capacity of the mixture model rather than simply adding trainable parameters.

Finally, we compare our ModernALBERT model with the Relaxed Recursive Transformer (RRT) baseline using the same initialisation strategy and training both models with the masked language modelling (MLM) objective. As shown in Appendix Table 8, ModernALBERT achieves a higher GLUE average (**81.94 vs. 80.95**), indicating that integrating conditional computation directly into the shared FFN weights yields more effective training than using static LoRA modules. The training configuration was kept uniform across all ablation experiments, with all models trained for a consistent 30k steps on the Wikipedia corpus, using a batch size of 128 and a maximum sequence length of 1024.

5 Discussion

The increase in capacity in the MoL architecture arises primarily from expert specialisation. As the router learns to activate different LoRA modules for different inputs, each module becomes responsible for modelling particular aspects of the data distribution, such as syntactic cues, semantic relations, or domain-specific patterns. This division of responsibility yields richer internal representations and helps mitigate the expressivity limitations inherent to fully shared architectures. Although the additional experts provide clear performance gains, dynamic routing naturally increases inference cost because several experts may be activated per token (Shazeer et al., 2017; Lepikhin et al., 2020; Fedus et al., 2021).

To address this inference overhead, we introduce an expert merging procedure that compresses the dynamic MoL layer into a single static adapter for deployment. A detailed analysis of this optimization, including our dynamic Exponential Moving Average (EMA) merging strategy and comprehensive efficiency benchmarks, is provided in Appendix A. Crucially, this merging process demonstrates that the learned expressivity of the router can be effectively compressed into a static repre-

Task	BERT	RoBERTa	DeBERTaV3	NomicBERT	GTE-en	ModernBERT	ModernALBERT
NFCorpus	24.3	20.4	8.0	25.7	**26.3**	23.7	24.30
SciFact	51.3	45.6	22.6	52.0	54.1	**57.0**	56.90
TREC-Covid	49.5	52.2	48.4	63.0	49.7	72.1	**72.85**
FiQA	22.8	26.1	11.5	23.5	30.1	28.8	**30.43**
ArguAna	31.6	35.2	26.1	35.5	35.7	35.7	**48.82**
Avg. (Subset)	38.9	37.7	20.2	41.0	41.4	41.6	**46.66**

Table 4: Performance on selected BEIR benchmark datasets.

sentation, reducing inference latency to match that of a standard dense model while preserving most of its original performance.

6 Limitations

While the proposed model improves parameter efficiency and adaptive capacity through the integration of MoL layers, several limitations remain. First, despite the architectural optimisations and merging strategies explored in this work, the computational complexity of mixture-of-experts routing remains relatively high compared to fully shared or purely dense alternatives. Although our design mitigates dispatching overhead through lightweight routing and optional merging, further advances in efficient expert selection and load balancing are necessary to fully realise the scalability benefits of conditional computation.

Second, the model employs global attention across all tokens without incorporating locality-aware mechanisms. Recent architectures such as ModernBERT have demonstrated that local or block-sparse attention can substantially improve both efficiency and long-sequence modelling performance. Consequently, our approach may underperform on tasks requiring extended context retention or fine-grained reasoning over long inputs. Future work could address these issues by integrating local attention patterns or hybrid attention schemes within the recursive parameter-sharing framework.

7 Conclusion and Future Work

In this work, we explored several strategies to enhance the representational power of recursive transformers. Building upon the principle of parameter sharing, we introduced the Mixture-of-LoRAs (MoL) layer, a novel mechanism that integrates conditional computation into the shared transformer architecture. The proposed layer demonstrated substantial improvements in performance while maintaining parameter efficiency, highlighting its effectiveness as a scalable and adaptable extension to recursive parameter-shared models.

For future research, extending this framework to multimodal settings offers a promising direction for studying conditional representations across diverse input types. Moreover, applying parameter sharing in conjunction with the proposed Mixture-of-LoRAs to large autoregressive language models could provide a path toward reducing their parameter count and memory footprint, while preserving or even enhancing their representational capacity. Such integration may enable the development of more efficient yet powerful large-scale generative models.

References

Sanghwan Bae, Adam Fisch, Hrayr Harutyunyan, Zongyi Ji, Sungdong Kim, and Tal Schuster. 2024a. Relaxed recursive transformers: Effective parameter sharing with layer-wise lora. *arXiv preprint arXiv:2410.20672*.

Seongjin Bae and 1 others. 2024b. Effective parameter sharing with layer-wise lora. Introduces Relaxed Recursive Transformers / layer-wise LoRA (openreview/arXiv).

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. *Advances in Neural Information Processing Systems (NeurIPS) / arXiv preprint*.

Together Computer. 2023. Redpajama: An open dataset for training large language models. In *Thirty-eighth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. 2018. Universal transformers. ArXiv preprint.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*.

William Fedus, Barret Zoph, and Noam Shazeer. 2021. Switch transformers: Scaling to trillion-parameter models with simple and efficient sparsity. ArXiv / conference.

W. Feng and 1 others. 2024. Mixture-of-loras: An efficient multitask tuning for large language models. *arXiv preprint*.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2023a. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. *Preprint*, arXiv:2111.09543.

Shwai He, Run-Ze Fan, Liang Ding, Li Shen, Tianyi Zhou, and Dacheng Tao. 2023b. Merging experts into one: Improving computational efficiency of mixture of experts. *Preprint*, arXiv:2310.09832.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *Preprint*, arXiv:1503.02531.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. *Preprint*, arXiv:1902.00751.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. ArXiv preprint.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. Tinybert: Distilling bert for natural language understanding. *Findings of EMNLP*.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization. *Preprint*, arXiv:1412.6980.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. ArXiv preprint.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2020. Gshard: Scaling giant models with conditional computation and automatic sharding. ArXiv preprint / ICLR.

D. Li, Y. Ma, N. Wang, and 1 others. 2024. Mixlora: Enhancing large language models fine-tuning with lora-based mixture of experts. *arXiv preprint*.

Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. 2023. Towards general text embeddings with multi-stage contrastive learning. *Preprint*, arXiv:2308.03281.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *Preprint*, arXiv:1907.11692.

A. Mehrish and 1 others. 2023. Adaptermix: Exploring the efficacy of mixture-of-adapters. *arXiv preprint*.

Toan Q Nguyen and Julian Salazar. 2019. Transformers without tears: Improving the normalization of self-attention. *arXiv preprint arXiv:1910.05895*.

Mohammadmahdi Nouriborji and 1 others. 2023. Minialbert: Model distillation via parameter-efficient recursive student. In *EACL / ACL (workshop / proceedings)*.

Zach Nussbaum, John X. Morris, Brandon Duderstadt, and Andriy Mulyar. 2025. Nomic embed: Training a reproducible long context text embedder. *Preprint*, arXiv:2402.01613.

Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. 2023. The refinedweb dataset for falcon llm: Outperforming curated corpora with web data, and web data only. *Preprint*, arXiv:2306.01116.

Jonas Pfeiffer, Andreas Rücklé, Klaus Schütze, and 1 others. 2020. Non-destructive task composition for transfer learning (adapterfusion). *arXiv preprint / ACL/EACL workshop*.

Jacob Portes, Alex Trott, Sam Havens, Daniel King, Abhinav Venigalla, Moin Nadeem, Nikhil Sardana, Daya Khudia, and Jonathan Frankle. 2024. Mosaicbert: A bidirectional encoder optimized for fast pretraining. *Preprint*, arXiv:2312.17482.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. *Preprint*, arXiv:1806.03822.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *Preprint*, arXiv:1910.01108.

Noam Shazeer. 2020. Glu variants improve transformer. ArXiv preprint (describes GeGLU, SwiGLU, etc.).

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. ICLR workshop / arXiv preprint.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. 2021. Roformer: Enhanced transformer with rotary position embedding. ArXiv preprint.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models. *Preprint*, arXiv:2104.08663.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. Glue: A multi-task benchmark and analysis platform for natural language understanding. *Preprint*, arXiv:1804.07461.

Wenhui Wang, Fang Wei, Li Dong, Hangbo Bao, Nan Yang, Ming Zhou, and 1 others. 2020. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *NeurIPS*.

Yaqing Wang, Subhabrata Agarwal, Subhransu Mukherjee, Xiaodong Liu, Jing Gao, Ahmed Hassan Awadallah, and Jianfeng Gao. 2022. Adamix: Mixture-of-adaptations for parameter-efficient model tuning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5744–5760.

Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, and 1 others. 2024. Modernbert: A modern bidirectional encoder for fast, memory-efficient, and long-context fine-tuning and inference. *arXiv preprint*.

Xun Wu, Shaohan Huang, Furu Wei, and 1 others. 2024. Mixture of lora experts. *arXiv preprint / OpenReview*.

A Optimising MoL for Inference

The proposed MoL layer augments a parameter-shared recursive transformer with lightweight LoRA experts coordinated by a routing network. While this design captures the flexibility of Mixture-of-Experts architectures, the conditional routing mechanism introduces latency overhead during inference by evaluating multiple experts per token. To address this, we propose compressing the MoL layer into a single static adapter for deployment.

A.1 Expert Merging Strategies

To eliminate conditional branching, we synthesise a single dense adapter Δ_{merged} from the pool of E experts $\{\Delta_1, \ldots, \Delta_E\}$. This aggregation is defined by a weighting vector $\mathbf{w} \in \mathbb{R}^E$:

$$\Delta_{\text{merged}} = \sum_{j=1}^{E} w_j \Delta_j. \tag{13}$$

During fine-tuning, the dynamic routing mechanism is disabled, and the model updates Δ_{merged} directly (He et al., 2023b). This procedure effectively converts the sparse MoL layer into a standard, static LoRA module, reducing the inference cost to that of a dense model. As shown in Table 5, this approach yields substantial efficiency gains; ModernALBERT-tiny achieves 9.46 ms latency and 106,527 tokens/s throughput while using only 0.196 GB of GPU memory.

We investigate two strategies for managing this aggregation during fine-tuning.

A.1.1 Uniform Initialization

In the first strategy, we initialise the merged adapter by averaging the pre-trained experts uniformly ($w_j = 1/E$). At each step, the merged adapter Δ_{merged} is formed by averaging the experts, and the model parameters are updated via standard back-propagation. This method assumes a neutral starting point, allowing the aggregate representation to adapt to the downstream task without reliance on token-specific routing.

A.1.2 Dynamic EMA Merging

The second strategy updates the merging weights $\mathbf{w}$ dynamically throughout fine-tuning based on the router's activation patterns. We initialise $\mathbf{w}$ uniformly ($w_j = 1/E$), allowing the EMA updates to gradually refine the distribution from a neutral starting point. Let $p_{i,t} \in \mathbb{R}^E$ be the router probability vector for token t in sample i. We first compute a per-sample routing vector r_i and a batch-level average r_b:

$$r_i = \frac{1}{T_i} \sum_{t=1}^{T_i} p_{i,t}, \qquad r_b = \frac{1}{B} \sum_{i=1}^{B} r_i. \tag{14}$$

We then update the global merging weights $\mathbf{w}$ using an exponential moving average (EMA) of these batch statistics:

$$\mathbf{w}_{\text{new}} = \alpha \mathbf{w}_{\text{old}} + (1 - \alpha) r_b, \tag{15}$$

where α is a decay factor. Regarding the router itself, we adopt a task-dependent approach: for data-rich tasks like MNLI, the router is trained jointly to refine expert selection; for smaller datasets (STS-B, RTE, SST-2), the router is frozen to leverage pre-trained specialisation while only the EMA weights adapt.

As shown in Table 6, the EMA-based strategy consistently outperforms uniform initialization (Vanilla). By dynamically adjusting the expert contributions based on routing statistics, EMA preserves nearly all of the unmerged model's accuracy, with particularly strong gains on RTE (86.28 vs 84.83) and SST-2. These results confirm that incorporating the router's learned preferences into the merging process, even when the router itself is frozen, provides a superior adaptation mechanism compared to a static uniform initialization.

B Detailed Ablation Studies

To supplement the findings discussed in Section 4, Table 7 provides the comprehensive results of our ablation studies comparing the Mixture-of-Adapters (MoA) and Mixture-of-LoRAs (MoL) configurations.

In the MoA setup, a set of trainable adapters and a routing mechanism are appended sequentially *after* the shared feed-forward network. Conversely, the proposed MoL design embeds LoRA experts directly into the weight space of the shared FFN. The empirical results demonstrate a distinct advantage for the MoL integration. For instance, when holding the parameter count roughly equivalent ($\sim$ 75M), the MoL architecture with 8 experts and top-2 routing achieves a GLUE average of 77.24, outperforming the equivalent MoA configuration (76.87). This validates our hypothesis that modifying the internal representations of the shared

Model	Latency (ms) ↓	Throughput (token/s) ↑	Memory (GB) ↓
ModernBERT Baseline			
ModernBERT-base	14.44	69519	0.570
ModernBERT-large	21.30	48117	1.5
ModernALBERT (No Expert Merging)			
ModernALBERT-tiny	13.40	72810	0.196
ModernALBERT-medium	12.34	80631	0.207
ModernALBERT-base	19.64	53453	0.291
ModernALBERT-large	31.45	32270	0.459
ModernALBERT (Merged Experts)			
ModernALBERT-tiny	**9.46**	**106527**	**0.196**
ModernALBERT-medium	9.72	106208	0.207
ModernALBERT-base	12.71	80731	0.291
ModernALBERT-large	18.87	54483	0.459

Table 5: Inference efficiency comparison of ModernBERT and ModernALBERT models. The proposed MoL expert merging converts conditional routing into a static adapter, significantly reducing inference latency and increasing throughput.

Method	MNLI	STS-B	RTE	SST-2
No Merging	88.2	91.8	85.6	95.2
Vanilla	87.9	91.5	84.83	94.8
EMA	**88.0**	**91.7**	**86.28**	**95.1**

Table 6: Effect of Expert Merging on Different Datasets.

FFN directly via LoRA yields a more effective integration of conditional computation than merely appending adaptive layers to the block output.

Furthermore, Table 7 highlights the positive scaling behaviour of the MoL design. Increasing the number of experts from 1 (GLUE average: 76.08) to 8 (GLUE average: 77.24) confirms that the performance gains are driven by the conditional capacity and specialisation of the mixture model, rather than the simple addition of trainable parameters.

Model	Params	Num Experts	Top-K	Single Sentence		Paraphrase		NLI		Avg
				CoLA	SST-2	MRPC	STS-B	MNLI	RTE	
ModernBERT	245M	-	-	34.05	88.76	88.72	88.22	81.53	75.45	76.12
ModernALBERT (Base)	56M	-	-	37.39	88.76	86.76	87.40	81.14	72.92	75.73
Mixture of Adapters (MoA) - Adapters placed after FFN										
MoA	64M	1	1	38.49	88.64	87.25	87.68	80.77	71.84	75.45
MoA	69M	4	1	38.49	**90.71**	87.50	87.32	80.61	74.36	76.17
MoA	75M	8	2	41.80	89.10	87.74	87.56	81.05	74.00	76.87
Mixture of LoRAs (MoL) - Proposed Method										
MoL	73M	1	1	40.06	88.99	87.74	87.38	81.03	73.28	76.08
MoL	74M	4	1	40.40	89.00	87.99	87.33	80.91	73.40	76.50
MoL	75M	8	2	**42.97**	90.36	88.23	88.01	81.15	74.72	**77.24**

Table 7: Ablation Study: Mixture of Adapters (MoA) vs. Mixture of LoRAs (MoL). MoL consistently outperforms MoA at similar parameter counts.

Model	Params	Single Sentence		Paraphrase/Similarity			Natural Language Inference			Avg
		CoLA	SST-2	MRPC	STS-B	QQP	MNLI	RTE	QNLI	
ModernALBERT	75M	**45.69**	89.90	**88.72**	**89.60**	**90.86**	**83.74**	**76.17**	**90.82**	**81.94**
Relaxed Recursive Transformers	73M	40.76	**90.02**	87.99	88.99	90.86	83.25	75.09	90.66	80.95

Table 8: Ablation on Initialisation: We use the same initialisation as Relaxed Recursive Transformers (RRT) and compare our Mixture-of-LoRAs (MoL) strategy with their layer-wise LoRAs on GLUE tasks.