Mixture-of-Recursions: Learning Dynamic Recursive Depths for Adaptive Token-Level Thinking

Sangmin Bae [*1] **Yujin Kim** [*1] **Reza Bayat** [*2] **Sungnyun Kim** [1] **Jiyoun Ha** [3] **Tal Schuster** [4] **Adam Fisch** [4]
Hrayr Harutyunyan [5] **Ziwei Ji** [4] **Aaron Courville** [†2 6] **Se-Young Yun** [†1]

Abstract

Scaling language models unlocks impressive capabilities, but the accompanying computational and memory demands make both training and deployment expensive. Existing efficiency efforts typically target either parameter sharing or adaptive computation, leaving open the question of how to attain both simultaneously. We introduce *Mixture-of-Recursions* (MoR), a unified framework that combines the two axes of efficiency inside a single Recursive Transformer. MoR reuses a shared stack of layers across recursion steps to achieve parameter efficiency, while lightweight routers enable adaptive token-level thinking by dynamically assign recursion depth to tokens, thereby focusing quadratic attention computation only where it is most useful. Further enhancing its efficiency, MoR incorporates a recursion-wise key-value caching mechanism that eliminates redundant memory access across recursion steps by selectively storing only the key-value caches for designated tokens. Across pretraining runs at model scales ranging from 135M to 1.7B parameters, MoR forms a new Pareto frontier: at equal training FLOPs and smaller model sizes, it significantly lowers validation perplexity and improves few-shot accuracy, while delivering higher throughput compared with vanilla and existing recursive baselines. These gains demonstrate that MoR is an effective path towards large-model quality without incurring large-model cost.

[*]Equal contribution [†]Corresponding authors [1]Kim Jaechul Graduate School of Artificial Intelligence, KAIST [2]Mila-Quebec AI Institute [3]Google Cloud [4]Google DeepMind [5]Google Research (Google co-authors performed only an advisory role in this paper.) [6]Department of Computer Science and Operations Research, Université de Montréal. Correspondence to: Sangmin Bae <bsmn0223@kaist.ac.kr>, Yujin Kim <yujin399@kaist.ac.kr>, Reza Bayat <reza.bayat@mila.quebec>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

1. Introduction

Scaling Transformer networks to hundreds of billions of parameters has unlocked impressive few-shot generalization and reasoning abilities (Brown et al., 2020; Chowdhery et al., 2023; Grattafiori et al., 2024; OpenAI, 2023; Reid et al., 2024; DeepSeek-AI et al., 2024). However, the accompanying memory footprint and computational requirements make both training and deployment outside hyperscale data centers challenging (Patterson et al., 2021; Momeni et al., 2024). This has motivated researchers to seek alternative "efficient" designs (Tay et al., 2022; Wan et al., 2023). Among the different axes of efficiency, *parameter efficiency* (Dehghani et al., 2018; Bae et al., 2024; Shazeer et al., 2017; Fedus et al., 2022)—reducing or sharing model weights—and *adaptive computation* (Raposo et al., 2024; Schuster et al., 2022; Fedus et al., 2022; Leviathan et al., 2023)—spending more compute only when it is needed—are promising, actively studied research direction.

One proven route to parameter efficiency is *layer tying*, in which a shared set of weights is reused across multiple layers (Dehghani et al., 2018; Lan et al., 2019; Gholami & Omar, 2023; Bae et al., 2024; Takase & Kiyono, 2021). For adaptive computation, a common approach is *early-exiting*, which dynamically allocates compute by exiting earlier in the network when predicting simpler tokens (Elhoushi et al., 2024; Schuster et al., 2022; Elbayad et al., 2020; Bae et al., 2023). In contrast, an architecture that effectively unifies both parameter efficiency and adaptive computation is still missing. Recursive Transformers (Bae et al., 2024; Fan et al., 2024; Giannou et al., 2023; Yang et al., 2023; Saunshi et al., 2025; Geiping et al., 2025), models that repeatedly apply the same set of shared layers multiple times, offer a strong foundation due to weight sharing. However, prior attempts at dynamic recursion have often been constrained by practical hurdles, such as requiring additional specialized training or deployment inefficiency. This has led most approaches to default to a fixed-depth recursion, incapable of delivering adaptive token-level compute allocation.

In this work, we introduce *Mixture-of-Recursions* (MoR), a unified framework that fully leverages the potential of Recursive Transformers (see Figure 1). MoR trains lightweight

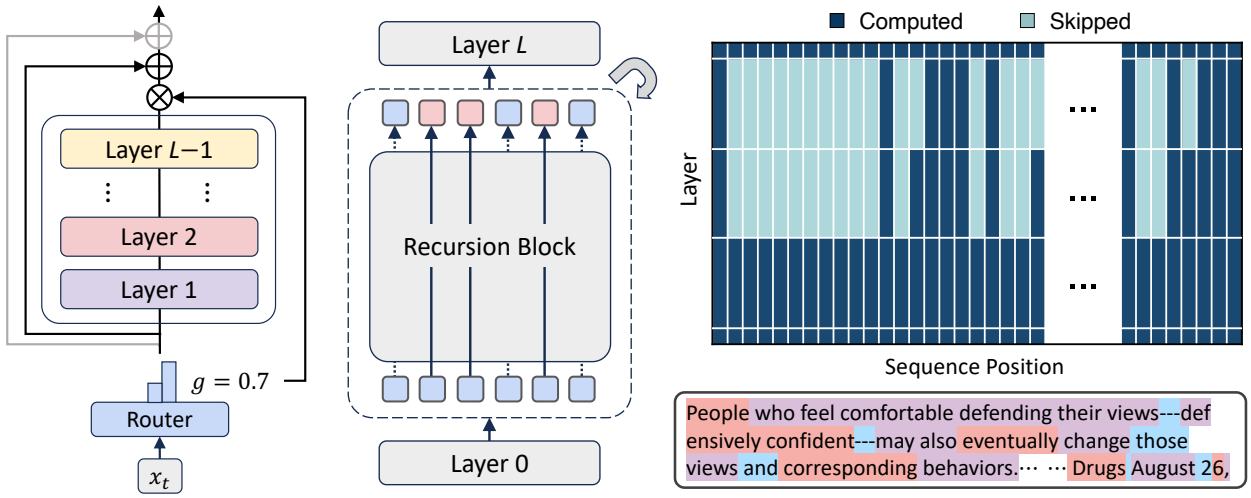


Figure 1: Overview of Mixture-of-Recursions (MoR). (*Left*) Each recursion step consists of a fixed stack of layers and a router that determines whether each token should pass through or exit. This recursion block corresponds to the gray box in the middle. (*Middle*) The full model structure, where the shared recursion step is applied up to N_r times for each token depending on the router decision. (*Right*) An example routing pattern showing token-wise recursion depth, where darker cells indicate active computation through the recursion block. Below shows the number of recursion steps of each text token, shown in colors: 1, 2, and 3.

routers end-to-end to assign token-specific recursion depths: it decides how many times a shared parameter block is applied to each token according to its required depth of "thinking", thereby directing computation to where it is most needed. This dynamic, token-level recursion inherently facilitates recursion-wise key–value (KV) caching, selectively storing and retrieving key–value pairs corresponding to each token's assigned recursion depth. This targeted caching strategy reduces memory traffic, improving throughput without post-hoc modifications. Therefore, MoR simultaneously (i) ties weights to cut parameters, (ii) routes tokens to cut redundant FLOPs, and (iii) caches key-values recursion-wise to cut memory IO—within a single architecture.

Conceptually, MoR provides a *pre-training* framework for latent space reasoning—performing non-verbal thinking by iteratively applying a single parameter block (Hao et al., 2024; Geiping et al., 2025; Goyal et al., 2023). However, unlike approaches that deliberate on augmented continuous prompts (Liu et al., 2024b; Goyal et al., 2023; Hao et al., 2024; Shen et al., 2025), MoR enables this latent thinking directly during the decoding of each token (Zelikman et al., 2024). Furthermore, routing mechanism facilitates adaptive reasoning along the model's vertical axis[1], moving beyond the fixed thinking depth common in prior work (Geiping et al., 2025; Tack et al., 2025). MoR enables models to efficiently adjust their thinking depth on a per-token basis, unifying parameter efficiency with adaptive computation.

[1]While thinking occurs along the depth axis, it is analogous to continuous thoughts along the horizontal sequence axis.

Contributions. In summary, our key contributions in this paper are as follows.

- **Unified framework for efficient language modeling:** We present *Mixture-of-Recursions* (MoR), the first architecture to unify efficiency paradigms—parameter sharing (§2.1), token-level adaptive thinking depth (§2.2.1), and memory-efficient KV caching (§2.2.2)—within a single framework.
- **Dynamic recursion routing:** We introduce a router trained from scratch to assign dynamic per-token recursion depths. This aligns training with inference-time behavior and eliminates the need for costly, performance-degrading post-hoc routing stages used in conventional methods.
- **Extensive empirical validation:** Across models from 135M to 1.7B parameters[2] under equal compute budgets, MoR establishes a new Pareto frontier by improving validation loss and few-shot accuracy relative to vanilla and recursive baselines (§3.1, §3.2).
- **Efficient architecture:** MoR dramatically reduces training FLOPs by selectively engaging only essential sequences in attention operations. Simultaneously, reduction in KV cache sizes leads to enhanced inference throughput itself, further boosted by continuous depth-wise batching (§3.3).

[2]These are base model sizes, while MoR models have fewer unique parameters due to parameter sharing.

2. Method

2.1. Preliminary

Recursive Transformers. The standard Transformer (Vaswani et al., 2017) constructs token representations through a stack of L unique layers, each with a self-attention and a feed-forward network. At time step t, the hidden state h evolves as: $\mathbf{h}_t^{\ell+1} = f\big(\mathbf{h}_t^{\ell}; \Phi_\ell\big)$, where $\ell = 0, \ldots, L-1$ and Φ_ℓ represents the parameters of the ℓ-th layer. Recursive Transformers (Bae et al., 2024; Fan et al., 2024; Giannou et al., 2023; Yang et al., 2023; Saunshi et al., 2025) aim to reduce parameter count by reusing layers across depth. Instead of having L distinct sets of weights, they partition the model into N_r recursion *blocks*, where each block uses a shared pool of parameters Φ'. This design allows for more computation (by increasing the effective network depth) without increasing parameter size.

Parameter-sharing strategies. We examine four parameter-sharing strategies: *Cycle*, *Sequence*, and their variants *Middle-Cycle* and *Middle-Sequence*. Table 3 summarizes details of these strategies. In Cycle sharing, recursion blocks are reused cyclically. For example, consider an original non-recursive model with L=9 layers and its recursive counterpart using N_r=3 recursions. Under the "Cycle" strategy, the layers are shared and unrolled as $[(0, 1, 2), (0, 1, 2), (0, 1, 2)]$. In "Sequence" sharing, each recursion block reuses the same layer consecutively before moving to the next, resulting in $[(0, 0, 0), (1, 1, 1), (2, 2, 2)]$ for the same configuration. Furthermore, the "Middle" variants preserve full-capacity parameters at the first and last layers (Φ_0 and Φ_{L-1}), while sharing weights among the intermediate layers.

Limitations in prior works. Although model parameters are tied, the distinct KV caches are typically used for each depth. This design fails to reduce the cache sizes, meaning the high retrieval latency still remains a severe inference bottleneck. Moreover, most existing recursive models simply apply a fixed recursion depth to all tokens, ignoring the varying complexity. While post-hoc methods like early-exiting methods can introduce some adaptivity, they often require separate training phases that can degrade performance (Schuster et al., 2022; Elhoushi et al., 2024; Bae et al., 2024). Ideally, the recursion depth should be learned dynamically during pretraining, allowing the model to adapt its computational path to each token's difficulty in a data-driven manner. However, such dynamic paths introduce a new challenge: exited tokens will have missing KV pairs at subsequent recursion depths. Addressing this would require a parallel decoding mechanism (Bae et al., 2023; Elhoushi et al., 2024; Kim et al., 2023) to efficiently compute the actual KV pairs, but this requires separate, complex engineering and complicates the system.

2.2. Mixture-of-Recursions

We propose *Mixture-of-Recursions* (MoR)—a framework that dynamically adjusts recursion step for each token during *pretraining* and *inference*. The core of MoR lies in two components: a routing mechanism that assigns token-specific recursion steps to adaptively concentrate computation on more challenging tokens, and a KV caching strategy that defines how KV pairs are stored and selectively utilized for attention at each recursive step.

2.2.1. Routing Strategies: Expert-choice vs. Token-choice

Expert-choice routing. (Figure 2a) Inspired by top-k gating in MoD models (Raposo et al., 2024), in expert-choice routing, each recursion depth becomes an expert and selects their preferred top-k tokens (e.g., for $N_r = 3$, we have three experts: Expert 1 applies the first recursion step, Expert 2 applies the second recursion, and so on). At each recursion step r, the corresponding router uses the hidden state $\mathcal{H}_t^r$ (input to the r-th recursion block) and its routing parameters θ_r to compute a scalar score $g_t^r = \mathcal{G}(\theta_r^\top \mathcal{H}_t^r)$ for token t. Here, $\mathcal{G}$ represents an activation function like `sigmoid` or `tanh`. Then, the top-k tokens are selected to pass through the recursion block:

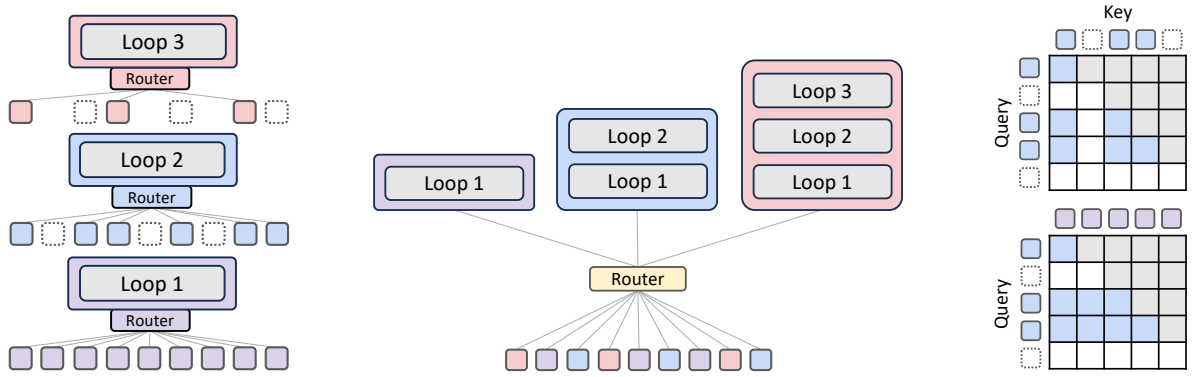
$$\mathcal{H}_t^{r+1} = \begin{cases} g_t^r f(\mathcal{H}_t^r,\, \Phi') + \mathcal{H}_t^r, & \text{if } g_t^r > P_\beta(G^r) \\ \mathcal{H}_t^r, & \text{otherwise} \end{cases} \tag{1}$$

where $P_\beta(G^r)$ is the β-percentile threshold over all scores at recursion step r.

To ensure coherent progression through steps, we adopt *hierarchical filtering*: only tokens selected at recursion step r can be re-evaluated at r+1. This simulates early-exit behavior while learning from scratch. As deeper layers tend to encode abstract and sparse information (Li et al., 2022; Yang et al., 2024; Nawrot et al., 2024), this mechanism prioritizes computation for only the most demanding tokens.

Token-choice routing. (Figure 2b) Unlike expert-choice, where token selection is made at each recursion step, token-choice commits each token to a full sequence of recursion blocks from the start. Formally, given the hidden state $\mathcal{H}_t^1$ (in Middle-Cycle strategy, $\mathcal{H}_t^1 = h_t^1$), the router computes a non-linear function (`softmax` or `sigmoid`) over experts: $g_t = \mathcal{G}(\theta_r^\top \mathcal{H}_t^1)$, where g_t^j denotes the routing score for expert $j \in \{1, \ldots, N_r\}$. The token is assigned to expert $i = \arg\max_j g_t^j$ (top-1 gating), which corresponds to sequentially applying the recursion i times. The hidden state is then updated recursively as:

$$\mathcal{H}_t^{r+1} = \begin{cases} g_t^r f(\mathcal{H}_t^r,\, \Phi') + \mathcal{H}_t^1, & \text{if } r = i \\ g_t^r f(\mathcal{H}_t^r,\, \Phi'), & \text{otherwise} \end{cases} \tag{2}$$



(a) Expert-choice routing (b) Token-choice routing (c) Caching mechanism

Figure 2: Architectural components of Mixture-of-Recursions (MoR). (a) *Expert-choice routing:* At each recursion step, a router selects top-k tokens to continue, progressively narrowing the set of active tokens with depth. (b) *Token-choice routing:* Each token is assigned a fixed recursion step at the outset via a single routing decision, defining its complete compute path through the model. (c) *KV caching strategies:* Each square in the matrix represents whether a token (row) attends to another token's cached key (column). In "recursion-wise KV caching" (*Top*), only the keys of currently selected (non-dropped) tokens at each recursion step are cached (blue), and attention is restricted only to these entries. In "recursive KV sharing" (*Bottom*), all keys of previous tokens are cached at the first recursion step (purple) and shared across subsequent recursion steps.

To compare routing strategies under equal compute, we align the token allocation budgets of expert-choice with that of token-choice. Specifically, we calibrate token capacity (i.e., top-k) of expert-choice to match the expected token distribution of token-choice routing with perfect load balancing. In perfectly balanced token-choice, each token is assigned to recursion depth $i \in \{1, \ldots, N_r\}$ with equal probability $1/N_r$. Thus, recursion step j processes $(N_r - j + 1)/N_r$ of the tokens. For example, when $N_r = 3$, recursion steps 1, 2, and 3 handle $\{3/3, 2/3, 1/3\}$ of tokens, respectively.

Strengths and limitations. (Table 1–*Left*) Although expert-choice routing guarantees perfect load balancing with static top-k selection, it suffers from information leakage (Zhou et al., 2022; Wang et al., 2024; Raposo et al., 2024). This violation of causality during training forces to exploit an auxiliary router or a regularization loss (Zhou et al., 2022; Raposo et al., 2024), aiming to precisely detect top-k tokens at inference without access to future token information. Meanwhile, token-choice is free from such leakage, but requires a balancing loss or loss-free algorithms (Wang et al., 2024; Fedus et al., 2022; Zoph et al., 2022) due to its inherent load balancing challenges. We explore each of these components for MoR in detail (§L.1).

2.2.2. KV Caching Strategies: Recursion-wise Caching vs. Recursive Sharing

Dynamic-depth models often struggle with KV cache consistency during autoregressive decoding. When a token exits early, its corresponding keys and values in deeper layers will be missing, which can be crucial information for subsequent tokens. Some methods attempt to reuse stale entries (Schuster et al., 2022) or run parallel decoding (Bae et al., 2023), but these solutions still introduce overhead and complexity. To this end, we design and explore two KV cache strategies tailored to MoR models: *recursion-wise caching* and *recursive sharing*.

Recursion-wise KV caching. (Figure 2c–*Top*) Inspired by Raposo et al. (2024), we cache KV pairs selectively: only tokens routed to a given recursion step store their key–value entries at that level. Thereby, the KV cache size at each recursion depth is determined exactly by the capacity factor in expert-choice, or according to actual balancing ratios in token-choice. Attention is then restricted to those locally cached tokens. This design promotes block-local computation, which improves memory efficiency and reduces IO demands.

Recursive KV sharing. (Figure 2c–*Bottom*) A key design choice for our MoR model is that all tokens traverse at least the first recursion block[3]. We leverage this by caching KV pairs exclusively at this initial step and reusing them across all subsequent recursions. Therefore, the query length might get shorter at each recursion depth based on the selection capacity, but the key and value lengths will consistently maintain the full sequence. This ensures that all tokens can access to past context without recomputation, despite any distribution mismatch.

Strengths and limitations. (Table 1–*Right*) Recursion-wise caching cuts KV memory and IO to approximately $(N_r + 1)/2N_r$ times across the entire model (when assum-

[3]Though it is not a strict requirement of the MoR framework.

Table 1: Comparison of routing strategies and key-value caching strategies. (*Left*) Summary of two routing strategies: expert-choice and token-choice, highlighting their pros, cons, and mitigating solutions from previous works (Raposo et al., 2024; Wang et al., 2024; Zoph et al., 2022). (*Right*) Relative cost efficiency of caching strategies against a vanilla Transformer (normalized to 1). Here, N_r denotes the number of recursions, and k ($k < N_{\text{ctx}}$) denotes the number of selected tokens per layer. KV cache memory and IO are measured across the entire model, whereas attention FLOPs are reported per layer.

	Expert-choice	**Token-choice**
Pros	Static compute budget	No leakage
Cons	Causality violation	Load imbalance
∟ Sol	Aux Rout, Aux Loss	Bal Loss, Loss-free

	Recursion-wise Caching	**Recursive Sharing**
KV Memory	$(N_r+1)/2N_r$	$1/N_r$
KV Cache IO	$(N_r+1)/2N_r$	1
Attn FLOPs	k^2/N_{ctx}^2	k/N_{ctx}

Table 2: Comparison of MoR, Recursive, and Vanilla Transformers under both fixed FLOPs (16.5e9) and token (20B) settings. All models are trained on FineWeb-Edu (Penedo et al., 2024) and evaluated by validation negative log-likelihood (NLL) and few-shot accuracy. For the isoFLOP rows, the number of training tokens (N_{tok}) varies by model efficiency. For the fixed-token rows, we report the effective FLOPs consumed. For the model sizes, we report non-embedding parameter counts.

	MoR		**Recursion**		**Pretrain**			**NLL↓**	**Few-shot Accuracy↑**						
Models	Type	KV	Share	N_R	Param	FLOPs	N_{tok}	FineWeb	LD	HS	PQ	WG	ARC	MMLU	Avg
Vanilla	-	-	-	-	315M	16.5	20B	2.7824	32.0	37.8	65.6	50.5	39.6	28.0	42.3
Recursive	-	-	M-Cyc	2	167M	16.5	20B	2.8079	31.0	37.1	66.7	52.3	**40.8**	27.5	42.6
	-	-	M-Cyc	3	118M	16.5	20B	2.8466	29.8	35.9	65.0	52.3	39.0	27.2	41.5
	-	-	M-Cyc	4	98M	16.5	19B	2.8781	28.2	35.4	65.5	52.5	38.0	26.8	41.0
MoR (ours)	Expert	✗	M-Cyc	2	167M	16.5	27B	**2.7511**	**34.4**	**39.3**	65.7	51.2	39.6	**28.1**	**43.1**
	Expert	✗	M-Cyc	3	118M	16.5	30B	2.7925	33.1	37.9	**66.9**	52.1	38.3	27.4	42.6
	Expert	✗	M-Cyc	4	98M	16.5	30B	2.8204	30.1	37.3	65.0	51.1	38.9	27.4	41.6
	Expert	✗	M-Cyc	2	167M	12.3	20B	2.7749	33.2	38.3	65.2	**52.6**	40.1	**28.1**	42.9
	Expert	✗	M-Cyc	3	118M	11.0	20B	2.8246	31.9	37.0	65.7	50.5	38.3	27.4	41.8
	Expert	✗	M-Cyc	4	98M	11.0	20B	2.8519	30.2	36.5	64.3	52.3	38.6	27.2	41.5
	Token	✗	M-Cyc	3	118M	16.5	30B	2.9163	27.6	34.1	63.8	50.6	37.4	26.8	40.0
	Expert	✓	M-Cyc	3	118M	16.5	31B	2.7983	31.7	37.2	65.1	51.0	39.0	27.1	41.9

ing capacity factors follow a sequence like N_r/N_r, $\cdots$, $1/N_r$ over N_r recursion steps). It also reduces per-layer attention FLOPs to a factor of $(k/N_{\text{ctx}})^2$ of those in vanilla models, resulting in substantially improved efficiency for both training and inference phases. Meanwhile, recursive sharing can yield maximal memory savings by globally reusing context. Specifically, significant speedups can be achieved by skipping KV projection and prefill operations at shared depths (Sun et al., 2024). However, attention FLOPs only decrease by a factor of k/N_{ctx}, and high volume of KV IO still leads to a decoding bottleneck.

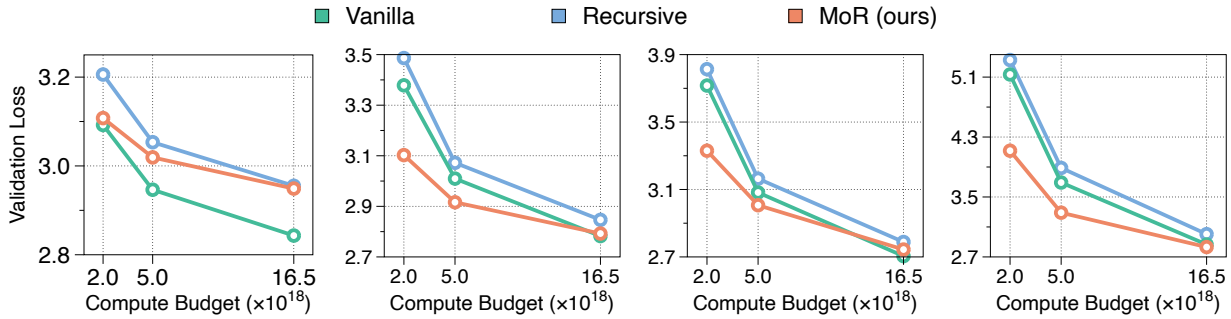
3. Experiments

We pretrain our models from scratch using a LLaMA-based Transformer architecture[4] (Dubey et al., 2024), implemented with the SmolLM open-source configurations (Allal et al., 2024), on a deduplicated subset of the FineWeb-Edu dataset (Penedo et al., 2024). We evaluate few-shot accuracy on seven benchmarks. Details are described in Appendix D.

3.1. Main Results

MoR outperforms baselines with fewer parameters under equal train compute. Under an equal training budget of 16.5e9 FLOPs, we compared our Mixture-of-Recursions (MoR) model against both Vanilla and Recursive Transformers. As shown in Table 2, the MoR model using expert-choice routing and two recursions $N_r = 2$ not only achieves a lower validation loss but also surpasses the vanilla baseline in average few-shot accuracy (43.1% vs. 42.3%). Remarkably, this superior performance is achieved despite using nearly 50% fewer parameters. This is attributed to MoR's higher computational efficiency, which allows it to process more training tokens within the same FLOPs budget. Furthermore, as N_r increases to 3 or 4, MoR maintains its competitive accuracy, consistently outperforming the recursive baselines while remaining within a tight margin of the full-capacity vanilla model.

[4]Experiments on Llama are conducted without direction or involvement from Google advisors.



(a) 135M-based model (b) 360M-based model (c) 730M-based model (d) 1.7B-based model

Figure 3: Validation loss as a function of compute budget across four model sizes: 135M, 360M, 730M, and 1.7B parameters. MoR consistently outperforms recursive baselines and matches or exceeds the Vanilla Transformer at larger scales, despite using significantly fewer parameters (approximately one-third due to layer tying with $N_R = 3$). These results highlight MoR's favorable scaling behavior.

MoR outperforms baselines with less compute at equal data. To isolate architectural differences, we analyze performance under a fixed number of training tokens (20B). Specifically, our MoR model with $N_r = 2$ outperforms both vanilla and recursive baselines—achieving lower validation loss and higher average accuracy—despite using 25% less fewer training FLOPs. This theoretical efficiency translates into significant practical gains: compared to the vanilla baseline, our model reduces training time by 19% and cuts peak memory usage by 25%. These improvements stem from our hierarchical filtering and recursion-wise attention mechanism, which shortens sequence lengths to achieve a superior compute-accuracy trade-off, even during pretraining.

MoR performance varies with routing and caching strategies. We also evaluate a few design variants within MoR framework, specifically with $N_r = 3$ that is lightweight and still comparable with Vanilla. In this case, using token-choice routing yields lower performance (40.0%) compared to expert-choice routing (42.6%), indicating that routing granularity plays a pivotal role in model performance. Additionally, applying KV cache sharing slightly reduces performance compared to independent caching, while providing improved memory efficiency. This trade-off remains favorable for practical deployment when memory usage is a key concern.

3.2. IsoFLOP Analysis

A core criterion for evaluating a new model architectural design is whether performance continues to improve as model and compute scales grow (Kaplan et al., 2020). Therefore, we evaluate MoR against both Vanilla and Recursive Transformers across a wide range of model sizes and computational budgets to show that it maintains competitive or superior predictive performance as the scale increases.

Experimental Setup. We experiment with four scales—135M, 360M, 730M, and 1.7B parameters—fixing the number of recursions to three for both Recursive and MoR configurations, resulting in roughly one-third the number of unique parameters. Each model is pretrained under three different FLOPs budgets: 2e9, 5e9, and 16.5e9.

MoR is a scalable and parameter-efficient architecture. As shown in Figure 3, MoR consistently outperforms recursive baselines across all model sizes and compute budgets. While it underperforms the vanilla model at the smallest model size (135M)—likely due to a recursive capacity bottleneck—this gap closes rapidly at scale. For >360M parameters, MoR not only matches but often exceeds the Vanilla Transformer, particularly under low and mid-range budgets. Overall, these results highlight that MoR is a scalable and efficient alternative to standard Transformers. It achieves strong validation performance with significantly lower parameter counts, making it a strong candidate for both pretraining and large-scale deployment. Further details are presented in Appendix E.

3.3. Throughput Evaluation

We highlight the substantial improvement in inference throughput of Mixture-of-Recursions over Vanilla Transformers in continuous depth-wise batching scenarios (Hooper et al., 2023; Pope et al., 2022). This practical setting maintains high and consistent GPU utilization by immediately replacing completed sequences with incoming tokens during decoding. MoR seamlessly integrates with this setup, dynamically substituting tokens that exit early at shallow recursion depths with new tokens. In practice, all MoR variants outperform the vanilla baseline, trading a slight increase in log-likelihood for significant throughput gains. Detailed settings and results are summarized in Appendix F.

4. Conclusion

Mixture-of-Recursions (MoR) presents a unified Transformer architecture that simultaneously leverages parameter sharing, adaptive recursion depth, and efficient KV caching without compromising model quality. By dynamically assigning recursion depth to tokens via lightweight routers and selectively caching key-value states for selected tokens, MoR reduces both quadratic attention computation and redundant memory access costs. Extensive empirical evaluations show that MoR lowers validation perplexity and improves average few-shot accuracy compared to both vanilla and previous recursive baselines, even with higher inference throughput. These results demonstrate that MoR offers an effective path towards achieving large-model capabilities with significantly reduced computational and memory overhead.

Acknowledgements This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No. RS-2024-00457882, AI Research Hub Project).

References

Ainslie, J., Lee-Thorp, J., de Jong, M., Zemlyanskiy, Y., Lebrón, F., and Sanghai, S. GQA: training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pp. 4895–4901, 2023a. doi: 10.18653/V1/2023.EMNLP-MAIN.298.

Ainslie, J., Lee-Thorp, J., De Jong, M., Zemlyanskiy, Y., Lebrón, F., and Sanghai, S. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245*, 2023b.

Allal, L. B., Lozhkov, A., Bakouch, E., von Werra, L., and Wolf, T. Smollm - blazingly fast and remarkably powerful, 2024.

Bae, S., Ko, J., Song, H., and Yun, S.-Y. Fast and robust early-exiting framework for autoregressive language models with synchronized parallel decoding. In Bouamor, H., Pino, J., and Bali, K. (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 5910–5924, Singapore, dec 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.362. URL https://aclanthology.org/2023.emnlp-main.362.

Bae, S., Fisch, A., Harutyunyan, H., Ji, Z., Kim, S., and Schuster, T. Relaxed recursive transformers: Effective parameter sharing with layer-wise lora. *arXiv preprint arXiv:2410.20672*, 2024.

Bengio, E., Bacon, P.-L., Pineau, J., and Precup, D. Conditional computation in neural networks for faster models. *arXiv preprint arXiv:1511.06297*, 2015.

Brandon, W., Mishra, M., Nrusimha, A., Panda, R., and Ragan-Kelley, J. Reducing transformer key-value cache size with cross-layer attention. *Neural Information Processing Systems*, 2024. doi: 10.48550/arXiv.2405.12981.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

Chen, C., Borgeaud, S., Irving, G., Lespiau, J.-B., Sifre, L., and Jumper, J. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv: 2302.01318*, 2023.

Chen, Y., Shang, J., Zhang, Z., Xie, Y., Sheng, J., Liu, T., Wang, S., Sun, Y., Wu, H., and Wang, H. Inner thinking transformer: Leveraging dynamic depth scaling to foster adaptive internal thinking. *arXiv preprint arXiv:2502.13842*, 2025.

Cheng, J. and Van Durme, B. Compressed chain of thought: Efficient reasoning through dense representations. *arXiv preprint arXiv:2412.13171*, 2024.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

Dai, D., Dong, L., Ma, S., Zheng, B., Sui, Z., Chang, B., and Wei, F. Stablemoe: Stable routing strategy for mixture of experts. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7085–7095, 2022.

Dai, M., Yang, C., and Si, Q. S-grpo: Early exit via reinforcement learning in reasoning models. *arXiv preprint arXiv:2505.07686*, 2025.

DeepSeek-AI, Liu, A., Feng, B., Xue, B., Wang, B., Wu, B., Lu, C., Zhao, C., Deng, C., Zhang, C., Ruan, C., Dai, D., Guo, D., Yang, D., Chen, D., Ji, D., Li, E., Lin, F., Dai, F., Luo, F., Hao, G., Chen, G., Li, G., Zhang, H., Bao, H., Xu, H., Wang, H., Zhang, H., Ding, H., Xin, H., Gao, H., Li, H., Qu, H., Cai, J. L., Liang, J., Guo, J., Ni, J., Li, J., Wang, J., Chen, J., Chen, J., Yuan, J., Qiu, J., Li, J., Song, J., Dong, K., Hu, K., Gao, K., Guan, K., Huang, K., Yu, K., Wang, L., Zhang, L., Xu, L., Xia, L., Zhao, L., Wang, L., Zhang, L., Li, M., Wang, M., Zhang, M., Zhang, M., Tang, M., Li, M., Tian, N., Huang, P., Wang,

P., Zhang, P., Wang, Q., Zhu, Q., Chen, Q., Du, Q., Chen, R. J., Jin, R. L., Ge, R., Zhang, R., Pan, R., Wang, R., Xu, R., Zhang, R., Chen, R., Li, S. S., Lu, S., Zhou, S., Chen, S., Wu, S., Ye, S., Ye, S., Ma, S., Wang, S., Zhou, S., Yu, S., Zhou, S., Pan, S., Wang, T., Yun, T., Pei, T., Sun, T., Xiao, W. L., Zeng, W., Zhao, W., An, W., Liu, W., Liang, W., Gao, W., Yu, W., Zhang, W., Li, X. Q., Jin, X., Wang, X., Bi, X., Liu, X., Wang, X., Shen, X., Chen, X., Zhang, X., Chen, X., Nie, X., Sun, X., Wang, X., Cheng, X., Liu, X., Xie, X., Liu, X., Yu, X., Song, X., Shan, X., Zhou, X., Yang, X., Li, X., Su, X., Lin, X., Li, Y. K., Wang, Y. Q., Wei, Y. X., Zhu, Y. X., Zhang, Y., Xu, Y., Xu, Y., Huang, Y., Li, Y., Zhao, Y., Sun, Y., Li, Y., Wang, Y., Yu, Y., Zheng, Y., Zhang, Y., Shi, Y., Xiong, Y., He, Y., Tang, Y., Piao, Y., Wang, Y., Tan, Y., Ma, Y., Liu, Y., Guo, Y., Wu, Y., Ou, Y., Zhu, Y., Wang, Y., Gong, Y., Zou, Y., He, Y., Zha, Y., Xiong, Y., Ma, Y., Yan, Y., Luo, Y., You, Y., Liu, Y., Zhou, Y., Wu, Z. F., Ren, Z. Z., Ren, Z., Sha, Z., Fu, Z., Xu, Z., Huang, Z., Zhang, Z., Xie, Z., Zhang, Z., Hao, Z., Gou, Z., Ma, Z., Yan, Z., Shao, Z., Xu, Z., Wu, Z., Zhang, Z., Li, Z., Gu, Z., Zhu, Z., Liu, Z., Li, Z., Xie, Z., Song, Z., Gao, Z., and Pan, Z. Deepseek-v3 technical report. *arXiv preprint arXiv: 2412.19437*, 2024.

Dehghani, M., Gouws, S., Vinyals, O., Uszkoreit, J., and Kaiser, L. Universal transformers. *International Conference on Learning Representations*, 2018.

Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., Goyal, A., Hartshorn, A., Yang, A., Mitra, A., Sravankumar, A., Korenev, A., Hinsvark, A., Rao, A., Zhang, A., Rodriguez, A., Gregerson, A., Spataru, A., Rozière, B., Biron, B., Tang, B., Chern, B., Caucheteux, C., Nayak, C., Bi, C., Marra, C., McConnell, C., Keller, C., Touret, C., Wu, C., Wong, C., Ferrer, C. C., Nikolaidis, C., Allonsius, D., Song, D., Pintz, D., Livshits, D., Esiobu, D., Choudhary, D., Mahajan, D., Garcia-Olano, D., Perino, D., Hupkes, D., Lakomkin, E., AlBadawy, E., Lobanova, E., Dinan, E., Smith, E. M., Radenovic, F., Zhang, F., Synnaeve, G., Lee, G., Anderson, G. L., Nail, G., Mialon, G., Pang, G., Cucurell, G., Nguyen, H., Korevaar, H., Xu, H., Touvron, H., Zarov, I., Ibarra, I. A., Kloumann, I. M., Misra, I., Evtimov, I., Copet, J., Lee, J., Geffert, J., Vranes, J., Park, J., Mahadeokar, J., Shah, J., van der Linde, J., Billock, J., Hong, J., Lee, J., Fu, J., Chi, J., Huang, J., Liu, J., Wang, J., Yu, J., Bitton, J., Spisak, J., Park, J., Rocca, J., Johnstun, J., Saxe, J., Jia, J., Alwala, K. V., Upasani, K., Plawiak, K., Li, K., Heafield, K., Stone, K., and et al. The llama 3 herd of models. *CoRR*, abs/2407.21783, 2024. doi: 10.48550/ARXIV.2407.21783.

Elbayad, M., Gu, J., Grave, E., and Auli, M. Depth-adaptive transformer. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJg7KhVKPH.

Elhoushi, M., Shrivastava, A., Liskovich, D., Hosmer, B., Wasti, B., Lai, L., Mahmoud, A., Acun, B., Agarwal, S., Roman, A., et al. Layerskip: Enabling early exit inference and self-speculative decoding. *arXiv preprint arXiv:2404.16710*, 2024.

Fan, Y., Du, Y., Ramchandran, K., and Lee, K. Looped transformers for length generalization. *arXiv preprint arXiv:2409.15647*, 2024.

Fedus, W., Zoph, B., and Shazeer, N. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *The Journal of Machine Learning Research*, 23(1):5232–5270, 2022.

Gadhikar, A., Majumdar, S. K., Popp, N., Saranrittichai, P., Rapp, M., and Schott, L. Attention is all you need for mixture-of-depths routing. *arXiv preprint arXiv:2412.20875*, 2024.

Gatmiry, K., Saunshi, N., Reddi, S. J., Jegelka, S., and Kumar, S. Can looped transformers learn to implement multi-step gradient descent for in-context learning? *arXiv preprint arXiv:2410.08292*, 2024.

Ge, S., Zhang, Y., Liu, L., Zhang, M., Han, J., and Gao, J. Model tells you what to discard: Adaptive kv cache compression for llms. *International Conference on Learning Representations*, 2023. doi: 10.48550/arXiv.2310.01801.

Geiping, J., McLeish, S., Jain, N., Kirchenbauer, J., Singh, S., Bartoldson, B. R., Kailkhura, B., Bhatele, A., and Goldstein, T. Scaling up test-time compute with latent reasoning: A recurrent depth approach. *arXiv preprint arXiv:2502.05171*, 2025.

Gholami, S. and Omar, M. Do generative large language models need billions of parameters? *arXiv preprint arXiv: 2309.06589*, 2023.

Giannou, A., Rajput, S., Sohn, J.-y., Lee, K., Lee, J. D., and Papailiopoulos, D. Looped transformers as programmable computers. In *International Conference on Machine Learning*, pp. 11398–11442. PMLR, 2023.

Goyal, S., Ji, Z., Rawat, A. S., Menon, A. K., Kumar, S., and Nagarajan, V. Think before you speak: Training language models with pause tokens. *arXiv preprint arXiv:2310.02226*, 2023.

Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., Yang, A., Fan, A., Goyal, A., Hartshorn, A., Yang, A., Mitra, A., Sravankumar, A., Korenev, A., Hinsvark, A., Rao, A., Zhang, A., Rodriguez, A.,

Gregerson, A., Spataru, A., Roziere, B., Biron, B., Tang, B., Chern, B., Caucheteux, C., Nayak, C., Bi, C., Marra, C., McConnell, C., Keller, C., Touret, C., Wu, C., Wong, C., Ferrer, C. C., Nikolaidis, C., Allonsius, D., Song, D., Pintz, D., Livshits, D., Wyatt, D., Esiobu, D., Choudhary, D., Mahajan, D., Garcia-Olano, D., Perino, D., Hupkes, D., Lakomkin, E., AlBadawy, E., Lobanova, E., Dinan, E., Smith, E. M., Radenovic, F., Guzmán, F., Zhang, F., Synnaeve, G., Lee, G., Anderson, G. L., Thattai, G., Nail, G., Mialon, G., Pang, G., Cucurell, G., Nguyen, H., Korevaar, H., Xu, H., Touvron, H., Zarov, I., Ibarra, I. A., Kloumann, I., Misra, I., Evtimov, I., Zhang, J., Copet, J., Lee, J., Geffert, J., Vranes, J., Park, J., Mahadeokar, J., Shah, J., van der Linde, J., Billock, J., Hong, J., Lee, J., Fu, J., Chi, J., Huang, J., Liu, J., Wang, J., Yu, J., Bitton, J., Spisak, J., Park, J., Rocca, J., Johnstun, J., Saxe, J., Jia, J., Alwala, K. V., Prasad, K., Upasani, K., Plawiak, K., Li, K., Heafield, K., Stone, K., El-Arini, K., Iyer, K., Malik, K., Chiu, K., Bhalla, K., Lakhotia, K., Rantala-Yeary, L., van der Maaten, L., Chen, L., Tan, L., Jenkins, L., Martin, L., Madaan, L., Malo, L., Blecher, L., Landzaat, L., de Oliveira, L., Muzzi, M., Pasupuleti, M., Singh, M., Paluri, M., Kardas, M., Tsimpoukelli, M., Oldham, M., Rita, M., Pavlova, M., Kambadur, M., Lewis, M., Si, M., Singh, M. K., Hassan, M., Goyal, N., Torabi, N., Bashlykov, N., Bogoychev, N., Chatterji, N., Zhang, N., Duchenne, O., Çelebi, O., Alrassy, P., Zhang, P., Li, P., Vasic, P., Weng, P., Bhargava, P., Dubal, P., Krishnan, P., Koura, P. S., Xu, P., He, Q., Dong, Q., Srinivasan, R., Ganapathy, R., Calderer, R., Cabral, R. S., Stojnic, R., Raileanu, R., Maheswari, R., Girdhar, R., Patel, R., Sauvestre, R., Polidoro, R., Sumbaly, R., Taylor, R., Silva, R., Hou, R., Wang, R., Hosseini, S., Chennabasappa, S., Singh, S., Bell, S., Kim, S. S., Edunov, S., Nie, S., Narang, S., Raparthy, S., Shen, S., Wan, S., Bhosale, S., Zhang, S., Vandenhende, S., Batra, S., Whitman, S., Sootla, S., Collot, S., Gururangan, S., Borodinsky, S., Herman, T., Fowler, T., Sheasha, T., Georgiou, T., Scialom, T., Speckbacher, T., Mihaylov, T., Xiao, T., Karn, U., Goswami, V., Gupta, V., Ramanathan, V., Kerkez, V., Gonguet, V., Do, V., Vogeti, V., Albiero, V., Petrovic, V., Chu, W., Xiong, W., Fu, W., Meers, W., Martinet, X., Wang, X., Wang, X., Tan, X. E., Xia, X., Xie, X., Jia, X., Wang, X., Goldschlag, Y., Gaur, Y., Babaei, Y., Wen, Y., Song, Y., Zhang, Y., Li, Y., Mao, Y., Coudert, Z. D., Yan, Z., Chen, Z., Papakipos, Z., Singh, A., Srivastava, A., Jain, A., Kelsey, A., Shajnfeld, A., Gangidi, A., Victoria, A., Goldstand, A., Menon, A., Sharma, A., Boesenberg, A., Baevski, A., Feinstein, A., Kallet, A., Sangani, A., Teo, A., Yunus, A., Lupu, A., Alvarado, A., Caples, A., Gu, A., Ho, A., Poulton, A., Ryan, A., Ramchandani, A., Dong, A., Franco, A., Goyal, A., Saraf, A., Chowdhury, A., Gabriel, A., Bharambe, A., Eisenman, A., Yazdan, A., James, B., Maurer, B., Leonhardi, B., Huang, B., Loyd, B., Paola, B. D., Paranjape, B., Liu, B., Wu, B., Ni, B., Hancock, B., Wasti, B., Spence, B., Stojkovic, B., Gamido, B., Montalvo, B., Parker, C., Burton, C., Mejia, C., Liu, C., Wang, C., Kim, C., Zhou, C., Hu, C., Chu, C.-H., Cai, C., Tindal, C., Feichtenhofer, C., Gao, C., Civin, D., Beaty, D., Kreymer, D., Li, D., Adkins, D., Xu, D., Testuggine, D., David, D., Parikh, D., Liskovich, D., Foss, D., Wang, D., Le, D., Holland, D., Dowling, E., Jamil, E., Montgomery, E., Presani, E., Hahn, E., Wood, E., Le, E.-T., Brinkman, E., Arcaute, E., Dunbar, E., Smothers, E., Sun, F., Kreuk, F., Tian, F., Kokkinos, F., Ozgenel, F., Caggioni, F., Kanayet, F., Seide, F., Florez, G. M., Schwarz, G., Badeer, G., Swee, G., Halpern, G., Herman, G., Sizov, G., Guangyi, Zhang, Lakshminarayanan, G., Inan, H., Shojanazeri, H., Zou, H., Wang, H., Zha, H., Habeeb, H., Rudolph, H., Suk, H., Aspegren, H., Goldman, H., Zhan, H., Damlaj, I., Molybog, I., Tufanov, I., Leontiadis, I., Veliche, I.-E., Gat, I., Weissman, J., Geboski, J., Kohli, J., Lam, J., Asher, J., Gaya, J.-B., Marcus, J., Tang, J., Chan, J., Zhen, J., Reizenstein, J., Teboul, J., Zhong, J., Jin, J., Yang, J., Cummings, J., Carvill, J., Shepard, J., McPhie, J., Torres, J., Ginsburg, J., Wang, J., Wu, K., U, K. H., Saxena, K., Khandelwal, K., Zand, K., Matosich, K., Veeraraghavan, K., Michelena, K., Li, K., Jagadeesh, K., Huang, K., Chawla, K., Huang, K., Chen, L., Garg, L., A, L., Silva, L., Bell, L., Zhang, L., Guo, L., Yu, L., Moshkovich, L., Wehrstedt, L., Khabsa, M., Avalani, M., Bhatt, M., Mankus, M., Hasson, M., Lennie, M., Reso, M., Groshev, M., Naumov, M., Lathi, M., Keneally, M., Liu, M., Seltzer, M. L., Valko, M., Restrepo, M., Patel, M., Vyatskov, M., Samvelyan, M., Clark, M., Macey, M., Wang, M., Hermoso, M. J., Metanat, M., Rastegari, M., Bansal, M., Santhanam, N., Parks, N., White, N., Bawa, N., Singhal, N., Egebo, N., Usunier, N., Mehta, N., Laptev, N. P., Dong, N., Cheng, N., Chernoguz, O., Hart, O., Salpekar, O., Kalinli, O., Kent, P., Parekh, P., Saab, P., Balaji, P., Rittner, P., Bontrager, P., Roux, P., Dollar, P., Zvyagina, P., Ratanchandani, P., Yuvraj, P., Liang, Q., Alao, R., Rodriguez, R., Ayub, R., Murthy, R., Nayani, R., Mitra, R., Parthasarathy, R., Li, R., Hogan, R., Battey, R., Wang, R., Howes, R., Rinott, R., Mehta, S., Siby, S., Bondu, S. J., Datta, S., Chugh, S., Hunt, S., Dhillon, S., Sidorov, S., Pan, S., Mahajan, S., Verma, S., Yamamoto, S., Ramaswamy, S., Lindsay, S., Lindsay, S., Feng, S., Lin, S., Zha, S. C., Patil, S., Shankar, S., Zhang, S., Zhang, S., Wang, S., Agarwal, S., Sajuyigbe, S., Chintala, S., Max, S., Chen, S., Kehoe, S., Satterfield, S., Govindaprasad, S., Gupta, S., Deng, S., Cho, S., Virk, S., Subramanian, S., Choudhury, S., Goldman, S., Remez, T., Glaser, T., Best, T., Koehler, T., Robinson, T., Li, T., Zhang, T., Matthews, T., Chou, T., Shaked, T., Vontimitta, V., Ajayi, V., Montanez, V., Mohan, V., Kumar, V. S., Mangla, V., Ionescu, V., Poenaru, V., Mihailescu, V. T., Ivanov, V., Li, W., Wang, W., Jiang, W., Bouaziz, W.,

Constable, W., Tang, X., Wu, X., Wang, X., Wu, X., Gao, X., Kleinman, Y., Chen, Y., Hu, Y., Jia, Y., Qi, Y., Li, Y., Zhang, Y., Zhang, Y., Adi, Y., Nam, Y., Yu, Wang, Zhao, Y., Hao, Y., Qian, Y., Li, Y., He, Y., Rait, Z., DeVito, Z., Rosnbrick, Z., Wen, Z., Yang, Z., Zhao, Z., and Ma, Z. The llama 3 herd of models. *arXiv preprint arXiv: 2407.21783*, 2024.

Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Han, S., Pool, J., Tran, J., and Dally, W. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015.

Hao, S., Sukhbaatar, S., Su, D., Li, X., Hu, Z., Weston, J., and Tian, Y. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*, 2024.

Hooper, C., Kim, S., Mohammadzadeh, H., Genc, H., Keutzer, K., Gholami, A., and Shao, S. Speed: Speculative pipelined execution for efficient decoding. *arXiv preprint arXiv:2310.12072*, 2023.

Hooper, C., Kim, S., Mohammadzadeh, H., Mahoney, M. W., Shao, Y. S., Keutzer, K., and Gholami, A. Kvquant: Towards 10 million context length llm inference with kv cache quantization. *arXiv preprint arXiv: 2401.18079*, 2024.

Hou, L., Huang, Z., Shang, L., Jiang, X., Chen, X., and Liu, Q. Dynabert: Dynamic bert with adaptive width and depth. *Advances in Neural Information Processing Systems*, 33:9782–9793, 2020.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W., et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

Huang, G., Sun, Y., Liu, Z., Sedra, D., and Weinberger, K. Q. Deep networks with stochastic depth. In Leibe, B., Matas, J., Sebe, N., and Welling, M. (eds.), *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part IV*, volume 9908 of *Lecture Notes in Computer Science*, pp. 646–661. Springer, 2016. doi: 10.1007/978-3-319-46493-0_39. URL https://doi.org/10.1007/978-3-319-46493-0_39.

Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., and Kalenichenko, D. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2704–2713, 2018.

Jaegle, A., Gimeno, F., Brock, A., Vinyals, O., Zisserman, A., and Carreira, J. Perceiver: General perception with iterative attention. In *International conference on machine learning*, pp. 4651–4664. PMLR, 2021.

Jiang, A. Q., Sablayrolles, A., Roux, A., Mensch, A., Savary, B., Bamford, C., Chaplot, D. S., Casas, D. d. l., Hanna, E. B., Bressand, F., et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

Jiang, G., Quan, G., Ding, Z., Luo, Z., Wang, D., and Hu, Z. Flashthink: An early exit method for efficient reasoning. *arXiv preprint arXiv:2505.13949*, 2025.

Kang, H., Zhang, Q., Kundu, S., Jeong, G., Liu, Z., Krishna, T., and Zhao, T. Gear: An efficient kv cache compression recipe for near-lossless generative inference of llm. *arXiv preprint arXiv: 2403.05527*, 2024.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Kim, S., Mangalam, K., Moon, S., Malik, J., Mahoney, M. W., Gholami, A., and Keutzer, K. Speculative decoding with big little decoder. *Advances in Neural Information Processing Systems*, 36:39236–39256, 2023.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. Albert: A lite bert for self-supervised learning of language representations. *International Conference on Learning Representations*, 2019.

Lepikhin, D., Lee, H., Xu, Y., Chen, D., Firat, O., Huang, Y., Krikun, M., Shazeer, N., and Chen, Z. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*, 2020.

Leviathan, Y., Kalman, M., and Matias, Y. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Li, Z., You, C., Bhojanapalli, S., Li, D., Rawat, A. S., Reddi, S. J., Ye, K., Chern, F., Yu, F., Guo, R., et al. The lazy neuron phenomenon: On emergence of activation sparsity in transformers. *arXiv preprint arXiv:2210.06313*, 2022.

Liu, A., Liu, J., Pan, Z., He, Y., Haffari, R., and Zhuang, B. Minicache: Kv cache compression in depth dimension for large language models. *Advances in Neural Information Processing Systems*, 37:139997–140031, 2024a.

Liu, L., Pfeiffer, J., Wu, J., Xie, J., and Szlam, A. Deliberation in latent space via differentiable cache augmentation. *arXiv preprint arXiv:2412.17747*, 2024b.

Liu, Z., Desai, A., Liao, F., Wang, W., Xie, V., Xu, Z., Kyrillidis, A., and Shrivastava, A. Scissorhands: Exploiting the persistence of importance hypothesis for llm kv cache compression at test time. *Advances in Neural Information Processing Systems*, 36:52342–52364, 2023a.

Liu, Z., Wang, J., Dao, T., Zhou, T., Yuan, B., Song, Z., Shrivastava, A., Zhang, C., Tian, Y., Re, C., et al. Deja vu: Contextual sparsity for efficient llms at inference time. In *International Conference on Machine Learning*, pp. 22137–22176. PMLR, 2023b.

Luo, Y., Luo, G., Ji, J., Zhou, Y., Sun, X., Shen, Z., and Ji, R. γ-mod: Exploring mixture-of-depth adaptation for multimodal large language models. *arXiv preprint arXiv:2410.13859*, 2024.

Mofakhami, M., Bayat, R., Mitliagkas, I., Monteiro, J., and Zantedeschi, V. Performance control in early exiting to deploy large models at the same cost of smaller ones. *arXiv preprint arXiv:2412.19325*, 2024.

Momeni, A., Rahmani, B., Scellier, B., Wright, L. G., McMahon, P. L., Wanjura, C. C., Li, Y., Skalli, A., Berloff, N. G., Onodera, T., Oguz, I., Morichetti, F., del Hougne, P., Gallo, M. L., Sebastian, A., Mirhoseini, A., Zhang, C., Marković, D., Brunner, D., Moser, C., Gigan, S., Marquardt, F., Ozcan, A., Grollier, J., Liu, A. J., Psaltis, D., Alù, A., and Fleury, R. Training of physical neural networks. *arXiv preprint arXiv: 2406.03372*, 2024.

Nawrot, P., Łańcucki, A., Chochowski, M., Tarjan, D., and Ponti, E. M. Dynamic memory compression: Retrofitting llms for accelerated inference. *arXiv preprint arXiv:2403.09636*, 2024.

OpenAI. Gpt-4 technical report. *PREPRINT*, 2023.

Panda, P., Sengupta, A., and Roy, K. Conditional deep learning for energy-efficient and enhanced pattern recognition. In *2016 design, automation & test in europe conference & exhibition (DATE)*, pp. 475–480. IEEE, 2016.

Patterson, D., Gonzalez, J., Le, Q., Liang, C., Munguia, L.-M., Rothchild, D., So, D., Texier, M., and Dean, J. Carbon emissions and large neural network training. *arXiv preprint arXiv: 2104.10350*, 2021.

Penedo, G., Kydlíček, H., allal, L. B., Lozhkov, A., Mitchell, M., Raffel, C., Werra, L. V., and Wolf, T. The fineweb datasets: Decanting the web for the finest text data at scale. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=n6SCkn2QaG.

Pfau, J., Merrill, W., and Bowman, S. R. Let's think dot by dot: Hidden computation in transformer language models. *arXiv preprint arXiv:2404.15758*, 2024.

Pope, R., Douglas, S., Chowdhery, A., Devlin, J., Bradbury, J., Levskaya, A., Heek, J., Xiao, K., Agrawal, S., and Dean, J. Efficiently scaling transformer inference. *arXiv preprint arXiv: 2211.05102*, 2022.

Raposo, D., Ritter, S., Richards, B., Lillicrap, T., Humphreys, P. C., and Santoro, A. Mixture-of-depths: Dynamically allocating compute in transformer-based language models. *arXiv preprint arXiv:2404.02258*, 2024.

Reid, M., Savinov, N., Teplyashin, D., Lepikhin, D., Lillicrap, T., baptiste Alayrac, J., Soricut, R., Lazaridou, A., Firat, O., Schrittwieser, J., Antonoglou, I., Anil, R., Borgeaud, S., Dai, A., Millican, K., Dyer, E., Glaese, M., Sottiaux, T., Lee, B., Viola, F., Reynolds, M., Xu, Y., Molloy, J., Chen, J., Isard, M., Barham, P., Hennigan, T., McIlroy, R., Johnson, M., Schalkwyk, J., Collins, E., Rutherford, E., Moreira, E., Ayoub, K., Goel, M., Meyer, C., Thornton, G., Yang, Z., Michalewski, H., Abbas, Z., Schucher, N., Anand, A., Ives, R., Keeling, J., Lenc, K., Haykal, S., Shakeri, S., Shyam, P., Chowdhery, A., Ring, R., Spencer, S., Sezener, E., Vilnis, L., Chang, O., Morioka, N., Tucker, G., Zheng, C., Woodman, O., Attaluri, N., Kocisky, T., Eltyshev, E., Chen, X., Chung, T., Selo, V., Brahma, S., Georgiev, P., Slone, A., Zhu, Z., Lottes, J., Qiao, S., Caine, B., Riedel, S., Tomala, A., Chadwick, M., Love, J., Choy, P., Mittal, S., Houlsby, N., Tang, Y., Lamm, M., Bai, L., Zhang, Q., He, L., Cheng, Y., Humphreys, P., Li, Y., Brin, S., Cassirer, A., Miao, Y., Zilka, L., Tobin, T., Xu, K., Proleev, L., Sohn, D., Magni, A., Hendricks, L. A., Gao, I., Ontañón, S., Bunyan, O., Byrd, N., Sharma, A., Zhang, B., Pinto, M., Sinha, R., Mehta, H., Jia, D., Caelles, S., Webson, A., Morris, A., Roelofs, B., Ding, Y., Strudel, R., Xiong, X., Ritter, M., Dehghani, M., Chaabouni, R., Karmarkar, A., Lai, G., Mentzer, F., Xu, B., Li, Y., Zhang, Y., Paine, T. L., Goldin, A., Neyshabur, B., Baumli, K., Levskaya, A., Laskin, M., Jia, W., Rae, J. W., Xiao, K., He, A., Giordano, S., Yagati, L., Lespiau, J.-B., Natsev, P., Ganapathy, S., Liu, F., Martins, D., Chen, N., Xu, Y., Barnes, M., May, R., Vezer, A., Oh, J., Franko, K., Bridgers, S., Zhao, R., Wu, B., Mustafa, B., Sechrist, S., Parisotto, E., Pillai, T. S., Larkin, C., Gu, C., Sorokin, C., Krikun, M., Guseynov, A., Landon, J., Datta, R., Pritzel, A., Thacker, P., Yang, F., Hui, K., Hauth, A., Yeh, C.-K., Barker, D., Mao-Jones, J., Austin, S., Sheahan, H., Schuh, P., Svensson, J., Jain, R., Ramasesh, V., Briukhov, A., Chung, D.-W., von Glehn, T., Butterfield, C., Jhakra, P., Wiethoff, M., Frye, J., Grimstad, J., Changpinyo, B., Lan, C. L., Bortsova, A.,

Wu, Y., Voigtlaender, P., Sainath, T., Smith, C., Hawkins, W., Cao, K., Besley, J., Srinivasan, S., Omernick, M., Gaffney, C., Surita, G., Burnell, R., Damoc, B., Ahn, J., Brock, A., Pajarskas, M., Petrushkina, A., Noury, S., Blanco, L., Swersky, K., Ahuja, A., Avrahami, T., Misra, V., de Liedekerke, R., Iinuma, M., Polozov, A., York, S., van den Driessche, G., Michel, P., Chiu, J., Blevins, R., Gleicher, Z., Recasens, A., Rrustemi, A., Gribovskaya, E., Roy, A., Gworek, W., Arnold, S., Lee, L., Lee-Thorp, J., Maggioni, M., Piqueras, E., Badola, K., Vikram, S., Gonzalez, L., Baddepudi, A., Senter, E., Devlin, J., Qin, J., Azzam, M., Trebacz, M., Polacek, M., Krishnakumar, K., yiin Chang, S., Tung, M., Penchev, I., Joshi, R., Olszewska, K., Muir, C., Wirth, M., Hartman, A. J., Newlan, J., Kashem, S., Bolina, V., Dabir, E., van Amersfoort, J., Ahmed, Z., Cobon-Kerr, J., Kamath, A., Hrafnkelsson, A. M., Hou, L., Mackinnon, I., Frechette, A., Noland, E., Si, X., Taropa, E., Li, D., Crone, P., Gulati, A., Cevey, S., Adler, J., Ma, A., Silver, D., Tokumine, S., Powell, R., Lee, S., Chang, M., Hassan, S., Mincu, D., Yang, A., Levine, N., Brennan, J., Wang, M., Hodkinson, S., Zhao, J., Lipschultz, J., Pope, A., Chang, M. B., Li, C., Shafey, L. E., Paganini, M., Douglas, S., Bohnet, B., Pardo, F., Odoom, S., Rosca, M., dos Santos, C. N., Soparkar, K., Guez, A., Hudson, T., Hansen, S., Asawaroengchai, C., Addanki, R., Yu, T., Stokowiec, W., Khan, M., Gilmer, J., Lee, J., Bostock, C. G., Rong, K., Caton, J., Pejman, P., Pavetic, F., Brown, G., Sharma, V., Lučić, M., Samuel, R., Djolonga, J., Mandhane, A., Sjösund, L. L., Buchatskaya, E., White, E., Clay, N., Jiang, J., Lim, H., Hemsley, R., Labanowski, J., Cao, N. D., Steiner, D., Hashemi, S. H., Austin, J., Gergely, A., Blyth, T., Stanton, J., Shivakumar, K., Siddhant, A., Andreassen, A., Araya, C., Sethi, N., Shivanna, R., Hand, S., Bapna, A., Khodaei, A., Miech, A., Tanzer, G., Swing, A., Thakoor, S., Pan, Z., Nado, Z., Winkler, S., Yu, D., Saleh, M., Maggiore, L., Barr, I., Giang, M., Kagohara, T., Danihelka, I., Marathe, A., Feinberg, V., Elhawaty, M., Ghelani, N., Horgan, D., Miller, H., Walker, L., Tanburn, R., Tariq, M., Shrivastava, D., Xia, F., Chiu, C.-C., Ashwood, Z., Baatarsukh, K., Samangooei, S., Alcober, F., Stjerngren, A., Komarek, P., Tsihlas, K., Boral, A., Comanescu, R., Chen, J., Liu, R., Bloxwich, D., Chen, C., Sun, Y., Feng, F., Mauger, M., Dotiwalla, X., Hellendoorn, V., Sharman, M., Zheng, I., Haridasan, K., Barth-Maron, G., Swanson, C., Rogozińska, D., Andreev, A., Rubenstein, P. K., Sang, R., Hurt, D., Elsayed, G., Wang, R., Lacey, D., Ilić, A., Zhao, Y., Aroyo, L., Iwuanyanwu, C., Nikolaev, V., Lakshminarayanan, B., Jazayeri, S., Kaufman, R. L., Varadarajan, M., Tekur, C., Fritz, D., Khalman, M., Reitter, D., Dasgupta, K., Sarcar, S., Ornduff, T., Snaider, J., Huot, F., Jia, J., Kemp, R., Trdin, N., Vijayakumar, A., Kim, L., Angermueller, C., Lao, L., Liu, T., Zhang, H., Engel, D., Greene, S., White, A., Austin, J., Taylor, L., Ashraf, S., Liu, D., Georgaki, M., Cai, I., Kulizhskaya, Y., Goenka, S., Saeta, B., Vodrahalli, K., Frank, C., de Cesare, D., Robenek, B., Richardson, H., Alnahlawi, M., Yew, C., Ponnapalli, P., Tagliasacchi, M., Korchemniy, A., Kim, Y., Li, D., Rosgen, B., Ashwood, Z., Levin, K., Wiesner, J., Banzal, P., Srinivasan, P., Yu, H., Çağlar Ünlü, Reid, D., Tung, Z., Finchelstein, D., Kumar, R., Elisseeff, A., Huang, J., Zhang, M., Zhu, R., Aguilar, R., Giménez, M., Xia, J., Dousse, O., Gierke, W., Yeganeh, S. H., Yates, D., Jalan, K., Li, L., Latorre-Chimoto, E., Nguyen, D. D., Durden, K., Kallakuri, P., Liu, Y., Johnson, M., Tsai, T., Talbert, A., Liu, J., Neitz, A., Elkind, C., Selvi, M., Jasarevic, M., Soares, L. B., Cui, A., Wang, P., Wang, A. W., Ye, X., Kallarackal, K., Loher, L., Lam, H., Broder, J., Holtmann-Rice, D., Martin, N., Ramadhana, B., Toyama, D., Shukla, M., Basu, S., Mohan, A., Fernando, N., Fiedel, N., Paterson, K., Li, H., Garg, A., Park, J., Choi, D., Wu, D., Singh, S., Zhang, Z., Globerson, A., Yu, L., Carpenter, J., de Chaumont Quitry, F., Radebaugh, C., Lin, C.-C., Tudor, A., Shroff, P., Garmon, D., Du, D., Vats, N., Lu, H., Iqbal, S., Yakubovich, A., Tripuraneni, N., Manyika, J., Qureshi, H., Hua, N., Ngani, C., Raad, M. A., Forbes, H., Bulanova, A., Stanway, J., Sundararajan, M., Ungureanu, V., Bishop, C., Li, Y., Venkatraman, B., Li, B., Thornton, C., Scellato, S., Gupta, N., Wang, Y., Tenney, I., Wu, X., Shenoy, A., Carvajal, G., Wright, D. G., Bariach, B., Xiao, Z., Hawkins, P., Dalmia, S., Farabet, C., Valenzuela, P., Yuan, Q., Welty, C., Agarwal, A., Chen, M., Kim, W., Hulse, B., Dukkipati, N., Paszke, A., Bolt, A., Davoodi, E., Choo, K., Beattie, J., Prendki, J., Vashisht, H., Santamaria-Fernandez, R., Cobo, L. C., Wilkiewicz, J., Madras, D., Elqursh, A., Uy, G., Ramirez, K., Harvey, M., Liechty, T., Zen, H., Seibert, J., Hu, C. H., Elhawaty, M., Khorlin, A., Le, M., Aharoni, A., Li, M., Wang, L., Kumar, S., Lince, A., Casagrande, N., Hoover, J., Badawy, D. E., Soergel, D., Vnukov, D., Miecnikowski, M., Simsa, J., Koop, A., Kumar, P., Sellam, T., Vlasic, D., Daruki, S., Shabat, N., Zhang, J., Su, G., Zhang, J., Liu, J., Sun, Y., Palmer, E., Ghaffarkhah, A., Xiong, X., Cotruta, V., Fink, M., Dixon, L., Sreevatsa, A., Goedeckemeyer, A., Dimitriev, A., Jafari, M., Crocker, R., FitzGerald, N., Kumar, A., Ghemawat, S., Philips, I., Liu, F., Liang, Y., Sterneck, R., Repina, A., Wu, M., Knight, L., Georgiev, M., Lee, H., Askham, H., Chakladar, A., Louis, A., Crous, C., Cate, H., Petrova, D., Quinn, M., Owusu-Afriyie, D., Singhal, A., Wei, N., Kim, S., Vincent, D., Nasr, M., Choquette-Choo, C. A., Tojo, R., Lu, S., de Las Casas, D., Cheng, Y., Bolukbasi, T., Lee, K., Fatehi, S., Ananthanarayanan, R., Patel, M., Kaed, C., Li, J., Sygnowski, J., Belle, S. R., Chen, Z., Konzelmann, J., Põder, S., Garg, R., Koverkathu, V., Brown, A., Dyer, C., Liu, R., Nova, A., Xu, J., Petrov, S., Hassabis, D., Kavukcuoglu, K., Dean, J., and Vinyals, O. Gemini 1.5: Unlocking multimodal understanding

across millions of tokens of context. *arXiv preprint arXiv: 2403.05530*, 2024.

Saunshi, N., Dikkala, N., Li, Z., Kumar, S., and Reddi, S. J. Reasoning with latent thoughts: On the power of looped transformers. *arXiv preprint arXiv:2502.17416*, 2025.

Schuster, T., Fisch, A., Gupta, J., Dehghani, M., Bahri, D., Tran, V. Q., Tay, Y., and Metzler, D. Confident adaptive language modeling. *arXiv preprint arXiv: 2207.07061*, 2022.

Shazeer, N. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*, 2019.

Shazeer, N. M., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q. V., Hinton, G. E., and Dean, J. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *International Conference on Learning Representations*, 2017.

Shen, Z., Yan, H., Zhang, L., Hu, Z., Du, Y., and He, Y. Codi: Compressing chain-of-thought into continuous space via self-distillation. *arXiv preprint arXiv:2502.21074*, 2025.

Sun, Y., Dong, L., Zhu, Y., Huang, S., Wang, W., Ma, S., Zhang, Q., Wang, J., and Wei, F. You only cache once: Decoder-decoder architectures for language models. *Advances in Neural Information Processing Systems*, 37: 7339–7361, 2024.

Tack, J., Lanchantin, J., Yu, J., Cohen, A., Kulikov, I., Lan, J., Hao, S., Tian, Y., Weston, J., and Li, X. Llm pretraining with continuous concepts. *arXiv preprint arXiv:2502.08524*, 2025.

Takase, S. and Kiyono, S. Lessons on parameter sharing across layers in transformers. *arXiv preprint arXiv:2104.06022*, 2021.

Tay, Y., Dehghani, M., Bahri, D., and Metzler, D. Efficient transformers: A survey. *ACM Computing Surveys*, 55(6): 1–28, 2022.

Teerapittayanon, S., McDanel, B., and Kung, H.-T. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd international conference on pattern recognition (ICPR)*, pp. 2464–2469. IEEE, 2016.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wan, Z., Wang, X., Liu, C., Alam, S., Zheng, Y., Liu, J., Qu, Z., Yan, S., Zhu, Y., Zhang, Q., Chowdhury, M., and Zhang, M. Efficient large language models: A survey. *arXiv preprint arXiv: 2312.03863*, 2023.

Wang, L., Gao, H., Zhao, C., Sun, X., and Dai, D. Auxiliary-loss-free load balancing strategy for mixture-of-experts. *arXiv preprint arXiv:2408.15664*, 2024.

Xia, Y., He, T., Tan, X., Tian, F., He, D., and Qin, T. Tied transformers: Neural machine translation with shared encoder and decoder. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 5466–5473, 2019.

Xiao, G., Tian, Y., Chen, B., Han, S., and Lewis, M. Efficient streaming language models with attention sinks. *arXiv preprint arXiv: 2309.17453*, 2023.

Yang, C., Si, Q., Duan, Y., Zhu, Z., Zhu, C., Li, Q., Lin, Z., Cao, L., and Wang, W. Dynamic early exit in reasoning models. *arXiv preprint arXiv:2504.15895*, 2025.

Yang, D., Han, X., Gao, Y., Hu, Y., Zhang, S., and Zhao, H. Pyramidinfer: Pyramid kv cache compression for high-throughput llm inference. *arXiv preprint arXiv:2405.12532*, 2024.

Yang, L., Lee, K., Nowak, R., and Papailiopoulos, D. Looped transformers are better at learning learning algorithms. *arXiv preprint arXiv:2311.12424*, 2023.

Zelikman, E., Harik, G., Shao, Y., Jayasiri, V., Haber, N., and Goodman, N. D. Quiet-star: Language models can teach themselves to think before speaking. *arXiv preprint arXiv:2403.09629*, 2024.

Zeng, D., Du, N., Wang, T., Xu, Y., Lei, T., Chen, Z., and Cui, C. Learning to skip for language modeling. *arXiv preprint arXiv:2311.15436*, 2023.

Zhang, J., Meng, D., Qi, J., Huang, Z., Wu, T., and Wang, L. p-mod: Building mixture-of-depths mllms via progressive ratio decay. *arXiv preprint arXiv:2412.04449*, 2024a.

Zhang, T., Yi, J., Xu, Z., and Shrivastava, A. Kv cache is 1 bit per channel: Efficient large language model inference with coupled quantization. *Advances in Neural Information Processing Systems*, 37:3304–3331, 2024b.

Zhang, Z., Sheng, Y., Zhou, T., Chen, T., Zheng, L., Cai, R., Song, Z., Tian, Y., Ré, C., Barrett, C., et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36:34661–34710, 2023.

Zhou, Y., Lei, T., Liu, H., Du, N., Huang, Y., Zhao, V., Dai, A. M., Le, Q. V., Laudon, J., et al. Mixture-of-experts with expert choice routing. *Advances in Neural Information Processing Systems*, 35:7103–7114, 2022.

Zoph, B., Bello, I., Kumar, S., Du, N., Huang, Y., Dean, J., Shazeer, N., and Fedus, W. St-moe: Designing stable and transferable sparse expert models. *arXiv preprint arXiv:2202.08906*, 2022.

A. Limitations and Future Works

Reasoning MoR models. Recent studies have highlighted the redundancy within reasoning chains and address it by applying token-level adaptive computation, like early-exit mechanisms (Yang et al., 2025; Jiang et al., 2025; Dai et al., 2025). Our MoR framework inherently enables latent reasoning by adaptively determining the necessary recursion depth for individual tokens. Therefore, a crucial future work involves exploring how the router can dynamically learn to adjust to the necessity of chain-of-thought (CoT) chains when post-trained on actual reasoning datasets. Developing advanced routing strategies that explicitly align recursion depth with reasoning complexity may enhance reasoning accuracy, computational efficiency, and even interpretability.

Further scaling model family. Our experiments have been limited to models with up to 1.7 billion parameters due to compute constraints. The natural next step is to train MoR models at larger scales (over 3 billion parameters) on substantially larger corpora. This will allow us to test whether the observed isoFLOP scaling advantages persist and to uncover optimization issues that only emerge at extreme scales. To further reduce overall pre-training costs, we could also explore continued pre-training (i.e., uptraining), starting from existing pre-trained vanilla LLM checkpoints. As future work, we plan to investigate MoR performance using various initialization strategies for recursive transformers, as explored in prior work (Bae et al., 2024).

Adaptive capacity control. Expert-choice routing offers the significant advantage of guaranteeing perfect load balancing through pre-determined capacity factors (Raposo et al., 2024; Zhou et al., 2022). However, a limitation arises when we want to allocate different capacities during inference. Specifically, in our MoR models, we observe that when using an auxiliary loss, the router outputs for selected and unselected tokens are almost perfectly separated. This makes it challenging to adjust top-k values after training. Therefore, a more adaptive model design, which can leverage different capacities during both training and inference phases, is needed to address this limitation.

Compatibility with sparse algorithms. Given MoR's token-level adaptive recursion, we can further optimize computation by integrating structured sparsity. This approach allows for the selective activation of subnetworks or parameters (Liu et al., 2023b), dynamically pruning unnecessary computations at both the token and layer levels (Raposo et al., 2024; Elhoushi et al., 2024). This investigation into sparse model designs promises significant efficiency improvements. We believe many sparsity-based techniques, such as pruning (Han et al., 2015) or quantization (Jacob et al., 2018), are highly complementary to MoR. This will provide deeper insights into effective sparse architectures within recursive models, offering promising directions for future research.

Expansion to multimodal and non-text domains. MoR's recursion block is inherently modality-agnostic, allowing its adaptive depth mechanism to extend beyond text processing. This crucial property enables MoR to readily integrate into vision, speech, and unified multimodal transformer architectures. Applying token-adaptive recursion to long-context video or audio streams holds the potential for even greater memory efficiencies and substantial throughput gains, crucial for real-world applications. By dynamically adjusting the processing depth for each token or segment, MoR could unlock these significant benefits.

B. Related Work

Adaptive computation. Many works have shown that *dynamic* compute allocation can markedly reduce the cost of training and inference, from traditional neural networks (Bengio et al., 2015; Huang et al., 2016; Teerapittayanon et al., 2016; Panda et al., 2016) to large language models (Hou et al., 2020; Elbayad et al., 2020; Fedus et al., 2022; Bae et al., 2023; Elhoushi et al., 2024; Raposo et al., 2024). Early exiting methods learn to halt processing for "easy" samples (e.g., tokens or sequences in language modeling) by skipping the remaining layers (Elbayad et al., 2020; Schuster et al., 2022; Dehghani et al., 2018; Mofakhami et al., 2024). Alternatively, early exits can be combined with speculative decoding techniques (Chen et al., 2023; Leviathan et al., 2023) during inference time by leveraging lower layers for fast drafting (Elhoushi et al., 2024). Recently, *Mixture-of-Depths* (MoD) (Raposo et al., 2024) reframed adaptive depth as a routing problem: a lightweight router at each layer selects a *subset* of tokens to receive the full computation, while the rest bypass the layer, yielding finer-grained conditional compute. This new form of adaptive allocation is well suited to Transformer architectures and has already been extended to other modalities (Zhang et al., 2024a; Luo et al., 2024), highlighting a promising paradigm of dynamic compute at token-level granularity. MoR applies the MoD routing idea to *recursive* Transformers: tokens are dynamically sent through repeated calls of a single, weight-tied block instead of through distinct layers. This shift keeps parameter count constant, allows arbitrarily deep (adaptive) compute beyond the model's physical depth.

Recursive Transformers. Parameter sharing provides an orthogonal path to efficiency (Dehghani et al., 2018; Lan et al., 2019; Xia et al., 2019; Takase & Kiyono, 2021; Bae et al., 2024; Jaegle et al., 2021). The *Universal Transformer* first showed that repeatedly applying a single block can match the representational power of deep, non-shared stacks (Dehghani et al., 2018). Looped Transformers shown to be effective, can act as programmable computers (Giannou et al., 2023), learn iterative data-fitting algorithms (Yang et al., 2023), generalize to much longer inputs on algorithmic tasks (Fan et al., 2024), and illuminate few-shot learning by mimicking multi-step optimizers (Gatmiry et al., 2024). Furthermore, Bae et al. (2024) mitigate the accuracy loss often associated with weight tying by adding low-rank adaptation (LoRA) adapters (Hu et al., 2022) in each loop, yielding *Relaxed Recursive Transformers*. Recent work further demonstrates that Recursive Transformers excel at latent reasoning via recurrent depth (Geiping et al., 2025). While most prior studies focus on the efficiency gains from weight tying, the *recursive* architecture itself offers a second level: inspired by early-exiting (Schuster et al., 2022) and compute routing (Raposo et al., 2024), one can vary the number of recursions per input (e.g., per token), allocating compute only where it is most beneficial during both training and inference.

Routing mechanism. LLMs have increasingly employed routers to enable adaptive computation, primarily in sparse Mixture-of-Experts (MoE) frameworks (Shazeer et al., 2017; Lepikhin et al., 2020; Dai et al., 2022; Zoph et al., 2022), i.e., each token is processed by a subset of expert networks chosen by a learned router, dramatically increasing model capacity without a computational overhead. Early MoE architectures (Lepikhin et al., 2020; Fedus et al., 2022; Jiang et al., 2024) adopted a *token-choice* routing strategy, wherein the router selects the top-k experts for each token based on its hidden state. While effective, this approach often leads to load imbalance across experts, necessitating auxiliary balancing losses. To address this, *expert-choice* routing (Zhou et al., 2022; Guo et al., 2025) has been proposed, wherein each expert selects the tokens to serve, ensuring perfect load balancing and improved efficiency. Building on this, a few works employed trainable routers to determine which layers to skip (Zeng et al., 2023; Raposo et al., 2024; Gadhikar et al., 2024). Unlike traditional early-exit methods, these expert-choice routing mechanisms enforce a static compute budget by capping the number of tokens processed per layer (or depth).

Key-value caching. Key–value (KV) caching stores the per-token key and value tensors produced at each layer during autoregressive decoding; reusing them eliminates quadratic-time recomputation and boosts throughput (Shazeer, 2019; Ge et al., 2023; Liu et al., 2024a; Xiao et al., 2023; Pope et al., 2022; Kang et al., 2024; Brandon et al., 2024). Unfortunately, retaining these tensors quickly saturates GPU memory, especially for long contexts and large batches (Chowdhery et al., 2023; Brandon et al., 2024). Prior work tackles this issue by quantizing KV activations to lower precision (Hooper et al., 2024; Zhang et al., 2024b), discarding entries that contribute little to the final output (Zhang et al., 2023; Liu et al., 2023a), and sharing keys and values across attention heads (Shazeer, 2019; Ainslie et al., 2023b). Brandon et al. (2024) push this idea further, allowing adjacent layers to share the same key and value tensors and achieving additional memory savings with negligible quality loss. Our Mixture-of-Recursions offer a complementary avenue: KV caches generated in early recursions can be reused in later ones, potentially reducing memory consumption even further. This provides the advantage of only needing to run the first recursion during prefill phase (Sun et al., 2024), promising significant speedups for prompt settings over 1 million tokens. Two caching strategies in MoR can be optimized based on their distinct benefits to suit various deployment settings.

Latent reasoning. An emerging line of work enables LLMs to perform reasoning internally within hidden states rather than through explicit verbalization (Goyal et al., 2023; Pfau et al., 2024; Zelikman et al., 2024; Cheng & Van Durme, 2024; Tack et al., 2025). Many approaches adopt a *fixed* latent reasoning depth: they insert special tokens or structured prompts (e.g., a learnable "pause" token (Goyal et al., 2023) or filler punctuation (Pfau et al., 2024)) that allow the model to execute a predetermined number of hidden reasoning passes before producing an answer. Others reuse the model's hidden states in a closed loop for a fixed number of iterations by feeding final hidden states back as input to simulate chain-of-thought (Hao et al., 2024; Shen et al., 2025; Saunshi et al., 2025). Another line of research enhances latent reasoning by augmenting hidden states with intermediate semantic signals (Zelikman et al., 2024; Tack et al., 2025). However, these methods lack the flexibility to allocate computation where it is most needed, leading to unnecessary overhead on easy inputs and insufficient reasoning on complex ones. Building upon recent findings that looping enhances model reasoning capabilities (Chen et al., 2025; Geiping et al., 2025; Saunshi et al., 2025), we believe our MoR framework provides a crucial foundation for bridging adaptive compute and latent reasoning.

C. Details of Mixture-of-Recursions Design Choices

In this section, we provide detailed descriptions of the design choices employed in Mixture-of-Recursions, expanding upon the summary provided in the main pages.

Table 3: Parameter-sharing strategies in Recursive Transformers. This table shows *Cycle*, *Middle-Cycle*, *Sequence*, and *Middle-Sequence* schemes with layer reuse, where Middle-* retains unique first and last layers.

	Cycle Strategy		**Middle-Cycle Strategy**	
Layers	Equation	Figure	Equation	Figure
Last	–		$f\left(\mathbf{h}_t^{L-1}; \Phi_{L-1}\right)$	Layer 3
Recursion	$f\left(\mathbf{h}_t^{\ell}; \Phi'_{(\ell \bmod (L/N_r))}\right)$	Layer 1 Layer 0	$f\left(\mathbf{h}_t^{\ell}; \Phi'_{((\ell-1) \bmod ((L-2)/N_r)+1)}\right)$	Layer 2 Layer 1
First	–		$f\left(\mathbf{h}_t^{0}; \Phi_0\right)$	Layer 0
	Sequence Strategy		**Middle-Sequence Strategy**	
Layers	Equation	Figure	Equation	Figure
Last	–		$f\left(\mathbf{h}_t^{L-1}; \Phi_{L-1}\right)$	Layer 3
Recursion	$f\left(\mathbf{h}_t^{\ell}; \Phi'_{(\ell \bmod N_r)}\right)$	Layer 1 Layer 0	$f\left(\mathbf{h}_t^{\ell}; \Phi'_{((\ell-1) \bmod N_r+1)}\right)$	Layer 2 Layer 1
First	–		$f\left(\mathbf{h}_t^{0}; \Phi_0\right)$	Layer 0

C.1. Parameter-sharing Strategy

Table 3 shows formulation and visualization of four parameter-sharing strategies: Cycle, Middle-Cycle, Sequence and Middle-Sequence. These strategies determine how a shared pool of blocks Φ' are reused across a total of L unrolled layers.

In the **Cycle** strategy, a fixed set of parameters is reused cyclically across all recursion steps. This approach efficiently reduces the number of unique parameters, enabling a compact model representation. However, because the same transformations are applied repeatedly regardless of input variation, it may limit the model's capacity to learn diverse or highly specialized features.

On the other hand, the **Sequence** strategy assigns distinct parameters to each recursion block in sequential order, allowing the model to capture more specialized and diverse representations at different recursion depths. Notably, prior studies have observed that adjacent Transformer layers often learn similar features, suggesting that sharing parameters among sequential layers may cause less performance degradation.

Building upon these strategies, the **Middle** sharing variant further refines parameter reuse by preserving unique parameters at the first and last layers while sharing weights only among the intermediate layers. This approach aims to balance the trade-off between parameter efficiency and representational flexibility, maintaining distinct entry and exit transformations while benefiting from reduced parameter redundancy in the middle layers. Consequently, Middle sharing can capture important input and output nuances more effectively than pure Cycle or Sequence sharing, without significantly increasing model size.

C.2. Routing Strategy

In this section, we provide an in-depth explanation of the two routing strategies employed in Mixture-of-Recursions: *Expert-choice* and *Token-choice* routers. Each approach has distinct advantages and inherent limitations, which we first outline before discussing the mitigation techniques we employed.

Expert-choice Router. The expert-choice router offers several advantages, including a fixed compute budget that simplifies resource management. However, it suffers from a key issue: the top-k selection operation, which requires tokens that appear later in the sequence, violates causality in autoregressive inference. This non-causal dependency can cause unexpected behavior during inference, potentially reducing model reliability.

To address these challenges, we explore two approaches: the auxiliary router and the auxiliary loss (Raposo et al., 2024). The **auxiliary router** is an additional lightweight network trained jointly but used only during inference; it predicts whether a token will be among the top-k. This network is trained with a binary cross-entropy loss, where the main router's outputs serve as logits and the top-k selections define the targets. Importantly, its training is isolated from the main objective through

gradient blocking, so it does not affect the primary model training. The **auxiliary loss** applies the binary cross-entropy loss to the main router, enabling it to simultaneously learn to accurately identify top-k tokens during training and to reliably predict whether each token will be included in the top-k during inference.

Token-choice Router. In contrast, the token-choice router assigns recursion depths on a per-token basis without enforcing a fixed compute budget, thus avoiding leakage of information across tokens and preserving autoregressive properties. However, this introduces load imbalance across experts, which results in uneven token distribution across experts (or recursion depths), potentially causing inefficient compute allocation and unbalanced training.

To mitigate load imbalance, we employ two solutions from existing literature. **Balancing Loss** (Lepikhin et al., 2020; Fedus et al., 2022) regularizes for a more uniform distribution of tokens across experts. For a sequence of length T, a balancing loss for MoR is calculated as follows:

$$\begin{aligned} \mathcal{L}_{\text{Balance}} &= \alpha \sum_{i=1}^{N} q_i P_i, \\ q_i &= \frac{N_r}{T} \sum_{t=1}^{T} \mathbf{1}(\text{Token } t \text{ selects Expert } i), \\ P_i &= \frac{1}{T} \sum_{t=1}^{T} g_t^i, \end{aligned} \tag{3}$$

where N_r is the total number of experts (which is also the number of recursion), g_t^i is the routing score of expert i for token t, q_i represents the fraction of tokens routed to expert i, P_i denotes the average routing scores of expert i, and λ is a hyperparameter controlling the strength of the auxiliary loss.

Loss-free (Wang et al., 2024) utilizes router biasing without explicit regularization. Specifically, this method adjusts per-expert bias terms b_i to balance token assignments across experts. During each training batch, routing scores are computed, and the number of tokens assigned to each expert c_i is counted. The load violation error is calculated as $e_i = \bar{c}_i - c_i$ where $\bar{c}_i$ is the average token count for expert i. Biases are then updated via

$$b_i \leftarrow b_i + u \times \text{sign}(e_i), \tag{4}$$

where u is a bias update rate. The routing scores for selecting expert are calculated as

$$g_{i,t} = \begin{cases} g_{i,t}, & g_{i,t} + b_i = \max\left(\{g_{j,t} + b_j \mid 1 \le j \le N\}\right), \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

C.3. KV Caching Strategy

This work investigates two principal strategies for key-value (KV) caching to optimize memory usage during Recursive Transformer computations: *recursion-wise caching* and *recursive KV sharing*.

Recursion-wise caching keeps separate KV caches for each recursion step, ensuring tokens attend only to the KV pairs generated in their current recursion block. This prevents distribution mismatches between recursion steps and helps maintain model accuracy while reducing memory and computational costs proportionally to the number of recursion steps.

Recursive KV sharing, in contrast, reuses KV pairs computed in the first recursion step for all subsequent steps. Although this approach further lowers memory usage, it introduces potential mismatches as later recursion steps receive KV representations originally intended for earlier steps. Such mismatch can negatively impact model performance when token routing is precise.

Therefore, recursion-wise caching is generally preferred in settings with selective token routing to avoid performance degradation, while recursive KV sharing may be considered when routing is less accurate and memory efficiency is prioritized.

D. Experimental Setup

Training Settings We utilized a Llama-based Transformer architecture (Dubey et al., 2024), referring to the configurations of the open-source SmolLM models (Allal et al., 2024). All models were pretrained on a deduplicated subset of the FineWeb-

Edu dataset (Penedo et al., 2024), which comprises 220 billion tokens sourced from educational materials. Pretraining was conducted using four H100 or A100 GPUs. In our main and isoFLOPs analysis experiments, we utilized a Trapezoid learning rate scheduler, which consists of warmup, stable, and cooldown phases. This approach allows us to efficiently continue pretraining for scaling laws from intermediate checkpoints, eliminating the need to train all models independently. In contrast, for all other experiments, we used a simple cosine annealing scheduler.

Evaluation Settings To assess model performance, we evaluated few-shot accuracy on seven benchmarks—LAMBADA (LD), HellaSwag (HS), PIQA (PQ), WinoGrande (WG), ARC (Easy and Challenge), and MMLU—using the Language Model Evaluation Harness. For all few-shot datasets, excluding LAMBADA, WinoGrande, and MMLU, we normalized accuracy by the byte length of the target string. We adhered to the standard number of shots for each dataset, and used the continuation task specifically for MMLU for simplicity. All evaluation performance measurements were conducted using a single H100 or A100 GPU.

Model Architecture Details Table 4 summarizes the architectural specifications of the four Vanilla Transformer models used as the base for our recursive models. Each model variant differs in scale, ranging from 135M to 1.7B total parameters (including both non-embedding and embedding components). For consistency and comparability, all models are trained using a vocabulary size of 49K and a maximum input sequence length of 2K tokens.

Table 4: Key parameters of four model size variants. A model's size is defined by the total number of its non-embedding and embedding parameters. Three small models utilize Grouped-Query Attention (Ainslie et al., 2023a), reducing the number of key-value heads. We refer to the base configurations of the open-sourced SmolLM models (Allal et al., 2024).

	Base Configuration				Attention & Feed-Forward				Input	
Models	N-emb	Emb	N_L	d_{model}	N_{head}	N_{KV}	d_{head}	d_{inter}	Vocab	L_{ctx}
Vanilla 135M	106M	28M	30	576	9	3	64	1536	49K	2K
Vanilla 360M	315M	47M	32	960	15	5	64	2560	49K	2K
Vanilla 730M	654M	75M	26	1536	24	8	64	4096	49K	2K
Vanilla 1.7B	1.61B	101M	24	2048	32	32	64	8192	49K	2K

E. Expanded Results of IsoFLOP Analysis

Experimental Recap. In the main paper (§3.2) we compared Vanilla, Recursive and our Mixture-of-Recursions (MoR) models under matched *training compute*. Four base model capacities were studied—135M, 360M, 730M and 1.7B parameters. For Recursive and MoR we fix the recursion count to $N_r = 3$, so the number of *unique* parameters is roughly one-third of the Vanilla counterpart. Each architecture is trained once for the *largest* compute budget (16.5EB)[5] and the resulting checkpoint is re-used to obtain the 5EB and 2EB variants, as detailed below.

Trapezoid Learning-rate Schedule with Checkpoint Reuse. To avoid retraining every model from scratch for each FLOPs slice we employ the *trapezoid* schedule. The schedule is

$$\eta(t) = \begin{cases} \frac{t}{w}\,\eta_{\max}, & 0 \le t < w \quad \text{(warm-up)}, \\ \eta_{\max}, & w \le t < p \quad \text{(plateau)}, \\ \eta_{\max}\Big(1 - \frac{t-p}{d}\Big), & p \le t < p + d \quad \text{(cool-down)}, \end{cases} \tag{6}$$

where w denotes the warm-up interval, $p - w$ the constant-LR plateau, and d the cool-down segment.

We save checkpoints at the optimizer steps whose cumulative compute—including the short cool-down tail specific to each slice—reaches 5EB and 2EB, respectively. For the warm-up we allocate $5\,\%$ of the total training steps of the smallest budget (2EB), and we set the cool-down steps to $20\,\%$ of the total training steps for each *corresponding* budget.

Results at a Glance. Table 5 reports NLL on FineWeb-Edu and few-shot accuracy. Three clear takeaways:

[5] 1EB $= 10^{18}$ floating-point operations.

1. **More compute leads to better models.** Higher FLOPs always lowers NLL and lifts accuracy.
2. **Recursive lags.** Weight-sharing alone stays behind Vanilla (e.g. 360M,16.5EB: +0.06 NLL, –1.1pp).
3. **MoR wins.** Token-routed MoR catches up and then beats Vanilla from 360M upward, while using only one-third of the parameters; the edge persists at 730M and 1.7B.

Table 5: **IsoFLOP results using a trapezoid learning-rate schedule.** Negative log-likelihood (NLL ↓) on the FineWeb-Edu validation set and few-shot accuracy (% ↑) on seven downstream tasks for four base model sizes (135M, 360M, 730M, 1.7B), each trained once up to 16.5EB and sliced back to 5EB and 2EB via mid-training checkpoints using a trapezoid learning-rate schedule. Every block of three rows compares Vanilla, Recursive (N_r=3) and Mixture-of-Recursions (MoR) variants.

		Pretrain				Recursion		NLL ↓	Few-shot Accuracy ↑						
Models	**Base**	N-Emb	N_L	FLOPs	N_{tok}	Share	Loop	FineWeb	LD	HS	PQ	WG	ARC	MMLU	Avg
Vanilla	135M	106M	30	2.0e+18	6.5B	-	-	3.0922	22.80	30.93	62.35	51.14	36.28	26.29	38.30
Recursive	135M	42M	1+10+1	2.0e+18	6.1B	M-Cyc	3	3.2058	19.79	29.32	60.17	50.59	34.83	25.40	36.68
MoR	135M	42M	1+10+1	2.0e+18	9.2B	M-Cyc	3	3.1077	21.13	31.00	59.79	49.09	34.87	25.63	36.92
Vanilla	135M	106M	30	5.0e+18	16.1B	-	-	2.9464	26.88	33.69	63.98	51.46	37.08	27.07	40.03
Recursive	135M	42M	1+10+1	5.0e+18	15.1B	M-Cyc	3	3.0534	24.51	31.57	62.40	50.83	35.78	25.94	38.51
MoR	135M	42M	1+10+1	5.0e+18	23.1B	M-Cyc	3	3.0192	22.01	32.53	61.75	49.88	35.39	26.19	37.96
Vanilla	135M	106M	30	16.5e+18	53.3B	-	-	2.8432	30.16	36.51	64.80	53.43	40.17	27.82	42.15
Recursive	135M	42M	1+10+1	16.5e+18	50.0B	M-Cyc	3	2.9552	25.98	33.36	63.98	51.78	36.96	26.68	39.79
MoR	135M	42M	1+10+1	16.5e+18	76.2B	M-Cyc	3	2.9490	22.61	33.99	61.92	47.83	35.95	26.36	38.11
Vanilla	360M	315M	32	2.0e+18	2.4B	-	-	3.3785	17.27	27.90	59.36	51.38	32.10	25.49	35.58
Recursive	360M	118M	1+10+1	2.0e+18	2.4B	M-Cyc	3	3.4864	10.34	26.66	58.00	51.54	30.94	24.94	33.74
MoR	360M	118M	1+10+1	2.0e+18	3.6B	M-Cyc	3	3.1026	24.14	30.53	61.86	50.99	34.74	25.50	37.96
Vanilla	360M	315M	32	5.0e+18	6.0B	-	-	3.0097	25.17	32.10	63.22	48.62	36.01	26.69	38.63
Recursive	360M	118M	1+10+1	5.0e+18	6.0B	M-Cyc	3	3.0722	23.29	31.19	62.62	51.30	35.85	25.99	38.37
MoR	360M	118M	1+10+1	5.0e+18	9.0B	M-Cyc	3	2.9161	28.33	34.53	63.22	51.07	36.70	26.98	40.14
Vanilla	360M	315M	32	16.5e+18	19.8B	-	-	2.7824	31.94	37.92	66.10	51.30	39.70	27.95	42.49
Recursive	360M	118M	1+10+1	16.5e+18	19.8B	M-Cyc	3	2.8466	29.75	35.92	64.91	51.46	39.12	27.18	41.39
MoR	360M	118M	1+10+1	16.5e+18	29.7B	M-Cyc	3	2.7924	33.15	37.94	66.97	52.09	38.46	27.49	42.68
Vanilla	730M	654M	26	2.0e+18	1.2B	-	-	3.7164	07.74	26.58	57.62	51.14	29.74	24.46	32.88
Recursive	730M	252M	1+8+1	2.0e+18	1.2B	M-Cyc	3	3.8136	05.53	26.25	55.77	50.59	29.88	24.63	32.11
MoR	730M	252M	1+8+1	2.0e+18	1.8B	M-Cyc	3	3.3300	17.93	28.74	59.30	51.46	33.14	25.37	35.99
Vanilla	730M	654M	26	5.0e+18	3.1B	-	-	3.0821	22.05	31.99	62.68	50.67	35.88	26.12	38.23
Recursive	730M	252M	1+8+1	5.0e+18	3.1B	M-Cyc	3	3.1640	18.51	30.72	62.13	47.83	35.97	25.84	36.83
MoR	730M	252M	1+8+1	5.0e+18	4.5B	M-Cyc	3	3.0067	26.18	32.76	62.46	50.91	36.93	26.37	39.27
Vanilla	730M	654M	26	16.5e+18	10.1B	-	-	2.7048	34.50	40.29	66.81	49.49	40.82	28.66	43.43
Recursive	730M	252M	1+8+1	16.5e+18	10.1B	M-Cyc	3	2.7886	30.76	37.84	65.51	52.41	39.26	27.51	42.21
MoR	730M	252M	1+8+1	16.5e+18	14.9B	M-Cyc	3	2.7438	32.93	39.55	66.32	54.38	40.00	28.09	43.55
Vanilla	1.7B	1.61B	24	2.0e+18	0.6B	-	-	5.1349	00.00	24.96	51.03	51.38	25.75	23.07	29.37
Recursive	1.7B	0.67B	1+8+1	2.0e+18	0.5B	M-Cyc	3	5.3277	00.00	25.27	51.36	48.62	26.52	22.98	29.13
MoR	1.7B	0.67B	1+8+1	2.0e+18	0.8B	M-Cyc	3	4.1175	01.44	25.80	53.97	49.64	27.56	24.08	30.42
Vanilla	1.7B	1.61B	24	5.0e+18	1.5B	-	-	3.6926	08.33	26.84	57.29	51.30	29.72	24.51	33.00
Recursive	1.7B	0.67B	1+8+1	5.0e+18	1.3B	M-Cyc	3	3.8876	03.14	26.57	54.73	49.17	29.01	24.49	31.19
MoR	1.7B	0.67B	1+8+1	5.0e+18	2.0B	M-Cyc	3	3.2905	17.62	28.32	59.03	49.80	32.14	25.28	35.37
Vanilla	1.7B	1.61B	24	16.5e+18	4.8B	-	-	2.8658	26.94	35.61	64.74	50.59	38.55	26.81	40.54
Recursive	1.7B	0.67B	1+8+1	16.5e+18	4.5B	M-Cyc	3	3.0042	23.25	32.09	62.95	50.75	37.64	26.53	38.87
MoR	1.7B	0.67B	1+8+1	16.5e+18	6.5B	M-Cyc	3	2.8316	28.10	36.18	64.64	50.99	38.68	27.25	40.97

F. Details of Throughput Evaluation

We implement a continuous depth-wise batching system to evaluate decoding throughput. Queries are enqueued and scheduled dynamically during decoding using the FineWeb-Edu test set. In particular, for MoR, when some queries exit early, the vacant slots in the batch were immediately filled with new queries waiting in the queue, maintaining a fully utilized batch at all times.

We conduct decoding on 1,000 query samples, each with an average decoding length of 256 tokens. The lengths of these samples are randomly truncated to simulate variable-length inputs.

For the experiments, we use a fixed batch size of 32 to compare Vanilla Transformers and MoR variants. Notably, MoR employs recursion-wise KV caching, which significantly reduces KV cache size. This reduction alone allows a substantial increase in batch size for throughput gains: MoR with recursion depth 2 can accommodate a batch size of 42 with the same KV cache memory as vanilla, while recursion depths 3 and 4 support batch sizes up to 47.

As shown in Figure 4, combining MoR with continuous depth-wise batching further increases effective batch size and throughput. These results demonstrate that our approach provides a highly efficient architecture for scaling throughput without sacrificing model capacity.

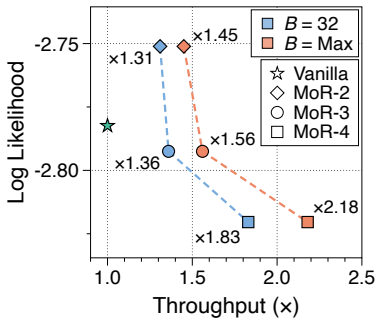


Figure 4: Pareto frontier of inference throughput and log-likehood for MoR and Vanilla Transformer under continuous depth-wise batching scenario.

G. Expanded Results of Parameter Sharing Strategy

We revisit the four weight-tying schemes—*Cycle*, *Sequence*, *Middle-Cycle*, and *Middle-Sequence*—on two base capacities (135 M and 360 M non-embedding parameters) and two recursion depths ($N_r = 2$ and 3). All models were trained from scratch for 10 B tokens under identical optimization hyper-parameters. Validation negative log-likelihood (NLL) on FineWeb-Edu and averaged few-shot accuracy over seven benchmarks are reported in Table 6.

Middle-Cycle is consistently the safest choice.

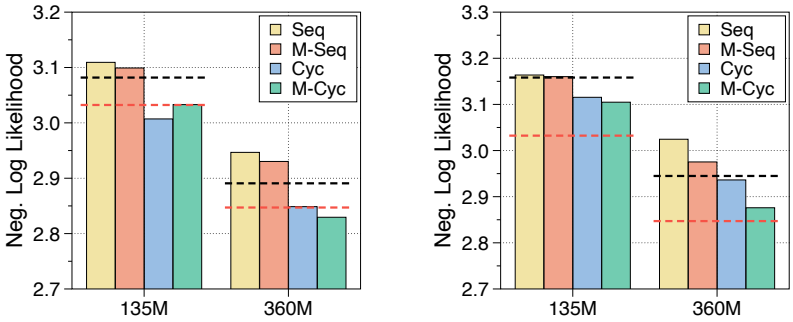
- **360 M models.** Middle-Cycle achieves the *lowest NLL at both depths* (2.8295 vs. 2.8487 for Cycle at $N_r = 2$, and 2.8760 vs. 2.9363 at $N_r = 3$) and enjoys the largest margin in average accuracy (+0.83 pp at $N_r = 2$, +1.67 pp at $N_r = 3$).
- **135 M models.** While Cycle is slightly ahead at the shallow setting (3.0071 vs. 3.0330), Middle-Cycle overtakes when depth rises (3.1048 vs. 3.1154) and shows a steadier accuracy profile.

Sequence variants lag in both quality and stability. Pure Sequence sharing records the worst NLL in all four settings (e.g., 3.1093 at 135 M/$N_r = 2$ and 3.0245 at 360 M/$N_r = 3$), and its accuracy deficit widens with depth (−.85 pp vs. Cycle at 360 M/$N_r = 3$).

Depth amplifies the benefit of Middle-Cycle. Figure 5 plots NLL for each strategy at the two recursion depths (panels Figure 5a and Figure 5b). The gap between Middle-Cycle and the others grows when stepping from two to three recursions, especially for the 360 M model.

Table 6: Comparison of parameter-sharing strategies (Cycle, Sequence, Middle-Cycle, Middle-Sequence) across two model scales (135M and 360M) and two recursion depths ($N_R = 2$ and $N_R = 3$), with all models pretrained from scratch on 10B tokens. We report validation negative log-likelihood (NLL) on FineWeb and few-shot accuracy across seven tasks. Middle-Cycle consistently outperforms other strategies in both NLL and average task accuracy, especially at higher recursion depth. ARC values are averaged over ARC-Easy and ARC-Challenge.

	Pretrain			Recursion		NLL ↓	Few-shot Accuracy ↑						
Base Model	N-Emb	N_L	N_{tok}	Share	Loop	FineWeb	LD	HS	PQ	WG	ARC	MMLU	Avg
Vanilla 135M	106M	30	10B	-	-	3.0323	24.14	31.12	61.15	52.01	34.74	25.95	38.19
Vanilla 135M	53M	15	10B	-	-	3.0818	23.64	30.10	60.94	50.99	35.38	25.93	37.83
Vanilla 135M	35M	10	10B	-	-	3.1582	21.46	29.30	60.01	52.01	34.40	25.53	37.12
Vanilla 135M	53M	15	10B	Cyc	2	3.0071	25.52	31.25	61.10	50.99	36.08	26.11	38.51
Vanilla 135M	53M	15	10B	Seq	2	3.1093	22.39	29.60	61.10	50.12	34.46	25.72	37.23
Vanilla 135M	57M	1+14+1	10B	M-Cyc	2	3.0330	23.40	31.20	61.59	50.59	35.44	25.54	37.96
Vanilla 135M	57M	1+14+1	10B	M-Seq	2	3.0991	21.70	30.06	60.45	49.41	35.20	25.74	37.09
Vanilla 135M	35M	10	10B	Cyc	3	3.1154	21.42	30.14	60.61	49.72	34.15	25.57	36.94
Vanilla 135M	35M	10	10B	Seq	3	3.1637	19.99	29.39	59.25	51.62	33.79	25.32	36.56
Vanilla 135M	39M	1+9+1	10B	M-Cyc	3	3.1048	22.41	30.35	61.04	49.01	34.80	25.91	37.26
Vanilla 135M	39M	1+9+1	10B	M-Seq	3	3.1602	20.69	29.35	61.43	51.30	34.40	25.51	37.11
Vanilla 360M	315M	32	10B	-	-	2.8471	27.27	34.78	64.20	52.80	38.29	26.72	40.68
Vanilla 360M	157M	16	10B	-	-	2.8908	27.01	33.49	64.42	52.09	37.40	26.54	40.16
Vanilla 360M	98M	10	10B	-	-	2.9449	26.41	32.93	63.38	50.36	37.15	26.48	39.45
Vanilla 360M	157M	16	10B	Cyc	2	2.8487	28.47	34.79	63.06	49.96	37.38	26.81	40.08
Vanilla 360M	157M	16	10B	Seq	2	2.9467	26.33	32.49	62.89	52.41	36.37	26.24	39.46
Vanilla 360M	167M	1+15+1	10B	M-Cyc	2	2.8295	28.59	34.98	64.53	50.51	39.68	27.20	40.91
Vanilla 360M	167M	1+15+1	10B	M-Seq	2	2.9303	26.14	32.71	62.79	51.38	36.31	25.73	39.18
Vanilla 360M	98M	10	10B	Cyc	3	2.9363	25.87	32.98	62.89	50.28	36.35	26.54	39.15
Vanilla 360M	98M	10	10B	Seq	3	3.0245	24.55	31.48	63.11	49.25	35.65	25.73	38.30
Vanilla 360M	118M	1+10+1	10B	M-Cyc	3	2.8760	28.51	34.89	64.31	50.51	39.51	27.20	40.82
Vanilla 360M	118M	1+10+1	10B	M-Seq	3	2.9753	24.18	31.89	62.08	49.72	36.47	26.27	38.44



(a) Recursion of two ($N_R = 2$) (b) Recursion of three ($N_R = 3$)

Figure 5: Validation negative log-likelihood (lower is better) on *FineWeb-Edu* for four parameter-sharing strategies—Sequence (*Seq*), Middle-Sequence (*M-Seq*), Cycle (*Cyc*), and Middle-Cycle (M-CYC). Bars are grouped by model capacity (135M vs. 360M parameters); the two panels correspond to recursion depths N_R=2 (left) and N_R=3 (right). Middle-Cycle (teal) consistently attains the lowest NLL, with its margin widening as either model size or depth increases. Horizontal dashed lines mark the *untied* (no-sharing) baselines: the lower red line is the fully untied model, while the upper black line is a parameter-matched truncated model whose footprint equals that of a single recursion.

Behaviour under continued pre-training (up-training). Table 7 extends the study by "up-training" models—continuing from open-sourced SmolLM checkpoints for an additional 5B tokens. Middle-Cycle not only maintains its lead but is also the *only* sharing scheme that closes *both* the NLL and accuracy gap to the full-capacity baseline (e.g., 2.8603→2.8295 NLL and +2.3 pp accuracy at 360 M/$N_r = 2$). The other strategies plateau earlier, hinting at limited capacity-growth headroom.

Table 7: Extended results on parameter sharing strategies with up-training with 5B tokens. Models are trained on FineWeb-Edu and evaluated by train negative log-likelihood (NLL) and few-shot accuracy across seven benchmarks. ARC denotes average of ARC-Easy and ARC-Challenge tasks, MMLU denotes the MMLU-Cont task.

	Pretrain			Recursion			NLL ↓	Few-shot Accuracy ↑						
Base Model	N-Emb	N_L	N_{tok}	Share	Init	Loop	FineWeb	LD	HS	PQ	WG	ARC	MMLU	Avg
Vanilla 360M	315M	32	5B	-	-		2.4825	41.67	50.63	70.35	55.09	46.99	30.82	49.26
Vanilla 360M	157M	16	5B	-	Step	1	2.7168	31.85	37.59	64.74	53.20	41.06	27.34	42.63
Vanilla 360M	157M	16	5B	Cyc	Avg	1	2.8603	22.14	30.36	60.07	48.22	34.99	25.56	36.89
Vanilla 360M	157M	16	5B	Seq	Avg	1	2.7919	25.40	32.35	62.30	50.12	35.88	26.19	38.71
Vanilla 360M	98M	10	5B	-	Step	1	2.8915	26.63	35.03	64.42	52.09	38.75	26.86	40.63
Vanilla 360M	98M	10	5B	Cyc	Avg	1	3.0512	23.19	31.27	62.30	51.22	36.71	26.29	38.50
Vanilla 360M	98M	10	5B	Seq	Avg	1	2.9915	25.67	32.21	62.30	51.38	36.68	26.56	39.14
Vanilla 360M	157M	16	5B	Cyc	Step	2	2.7165	31.30	37.68	64.91	52.17	39.29	27.53	42.15
Vanilla 360M	157M	16	5B	Cyc	Avg	2	2.8263	23.21	30.52	60.55	50.28	36.01	25.50	37.68
Vanilla 360M	157M	16	5B	Cyc	Lower	2	2.8024	27.67	34.71	63.49	49.88	38.12	26.87	40.13
Vanilla 360M	157M	16	5B	Cyc	Upper	2	2.7915	18.26	34.88	63.06	51.85	39.27	26.88	39.03
Vanilla 360M	157M	16	5B	Cyc	Rand	2	2.7575	25.29	34.78	61.64	52.01	38.09	26.62	39.74
Vanilla 360M	157M	16	5B	Seq	Step	2	2.6862	34.14	42.49	67.90	53.35	43.24	28.80	44.99
Vanilla 360M	157M	16	5B	Seq	Avg	2	2.7508	29.01	34.13	63.60	52.09	36.31	26.58	40.29
Vanilla 360M	157M	16	5B	Seq	Lower	2	2.8300	27.50	33.28	63.38	51.38	37.44	26.32	39.88
Vanilla 360M	157M	16	5B	Seq	Upper	2	2.7498	30.49	40.07	65.61	52.25	40.28	28.17	42.81
Vanilla 360M	157M	16	5B	Seq	Rand	2	2.7153	32.00	41.31	66.10	53.35	42.13	28.52	43.90
Vanilla 360M	167M	1+15+1	5B	M-Cyc	Step	2	2.6800	35.47	42.39	67.19	50.99	42.54	28.79	44.56
Vanilla 360M	167M	1+15+1	5B	M-Cyc	Avg	2	2.7314	33.81	40.42	66.87	51.78	41.68	28.17	43.79
Vanilla 360M	167M	1+15+1	5B	M-Cyc	Lower	2	2.7449	30.76	39.50	66.16	50.99	41.12	28.07	42.77
Vanilla 360M	167M	1+15+1	5B	M-Cyc	Upper	2	2.6605	34.41	43.74	67.46	53.20	43.75	28.93	45.25
Vanilla 360M	167M	1+15+1	5B	M-Cyc	Rand	2	2.6730	35.65	43.04	67.74	52.17	42.60	28.62	44.97
Vanilla 360M	167M	1+15+1	5B	M-Seq	Step	2	2.6627	35.09	43.34	67.57	51.22	43.66	28.91	44.97
Vanilla 360M	167M	1+15+1	5B	M-Seq	Avg	2	2.7143	33.92	40.93	66.49	51.70	40.72	28.24	43.66
Vanilla 360M	167M	1+15+1	5B	M-Seq	Lower	2	2.7696	30.74	38.36	65.94	51.78	41.27	27.73	42.64
Vanilla 360M	167M	1+15+1	5B	M-Seq	Upper	2	2.6931	32.66	42.35	67.14	52.80	42.83	28.49	44.38
Vanilla 360M	167M	1+15+1	5B	M-Seq	Rand	2	2.6908	35.07	42.03	66.32	53.75	43.11	28.42	44.78
Vanilla 360M	98M	10	5B	Cyc	Step	3	2.8901	27.46	35.26	63.82	51.54	39.35	27.44	40.81
Vanilla 360M	98M	10	5B	Seq	Step	3	2.8258	30.43	37.37	63.76	52.33	40.55	27.65	42.02
Vanilla 360M	118M	1+10+1	5B	M-Cyc	Step	3	2.7735	31.38	39.31	65.51	50.51	40.70	27.65	42.51
Vanilla 360M	118M	1+10+1	5B	M-Seq	Step	3	2.7678	31.67	39.23	65.89	52.09	40.65	27.90	42.91
Vanilla 360M	118M	1+10+1	5B	M-Cyc	Upper	3	2.8100	28.86	37.61	65.51	52.57	41.44	28.17	42.36

H. Expanded Results of Design Choices for Router

In this section, we report detailed explanation and results for ablations studies about router configurations.

H.1. Details of Router Design and Training Configurations

We investigate various router design choices to improve performance and stability. Specifically, we tune the coefficient values controlling the strength of auxiliary or balancing loss terms (*Coeff*), and adjust the scaling factor applied after the router function (α) to modulate routing weights. Different activation functions (*Func*), such as sigmoid or softmax, are evaluated, with architectural variations (*Arch*) of the router network, including linear layer, 2-layer MLP with GELU activation, Wide-MLP that expands the hidden layer size by a factor of four.

We also incorporate several techniques to stabilize training. To improve training stability, we introduce the *router z-loss* (Zoph et al., 2022), which penalizes large logits produced by the gating network. Large logits can cause numerical instability and hinder effective training of the router. The z-loss is computed as follows:

$$L_z(x) = \frac{1}{B}\sum_{i=1}^{B}\left(\log\sum_{j=1}^{N_r} e^{x_j^{(i)}}\right)^2, \tag{7}$$

where B is the number of tokens in the batch, N_r is the number of experts, and $x \in \mathbb{R}^{B\times N_r}$ denotes the logits input to the router. This regularization encourages the gating network to produce smaller logits, promoting more stable and reliable routing decisions.

Additionally, we apply *capacity warmup*, where the capacity gradually decreases from an initial value to 1.0 over a predefined number of warmup steps. The warmup duration is set equal to the learning rate warmup (i.e., 5% of total training steps). The capacity increases following a smooth cosine schedule based on the current training step. If no warmup is specified, the capacity remains fixed at its initial value throughout training.

H.2. Extended Evaluation of Router Designs

In the expert-choice routing, we evaluate sampling accuracy and dead token ratio to assess the router's selection behavior. The *dead token ratio* measures the proportion of tokens at specific positions within the batch that are consistently unselected during the final recursion step, indicating a positional bias where certain token positions are systematically neglected by the router. The *sampling accuracy* how well the router used during inference predicts whether a token belongs to the top-k tokens identified during training, reflecting the router's ability to consistently select the most relevant tokens. Ideally, high sampling accuracy with a low dead token ratio indicates a router that both identifies important tokens accurately and maintains diversity in token selection.

The results presented in Table 8 indicate that although both the auxiliary router and auxiliary loss methods enhance sampling accuracy, they are also associated with high dead token ratios. In particular, some auxiliary router variants exhibit dead token ratios as high as 66.7%, suggesting that the router always selects tokens from the same positions across inputs, reflecting a positional bias. Notably, employing a linear router architecture in conjunction with auxiliary loss and z-loss effectively reduces the dead token ratio without compromising sampling accuracy.

In token-choice routing, we evaluate the router's ability to balance token assignments across experts using MaxVio (maximum violation) and entropy metrics, which measure load imbalance and distribution uncertainty, respectively. *MaxVio* (Wang et al., 2024) measures the load imbalance across experts:

$$\text{MaxVio} = \frac{\max_i \text{Load}_i - \overline{\text{Load}_i}}{\overline{\text{Load}_i}}, \tag{8}$$

where Load_i denotes the actual number of tokens assigned to the i-th expert, and $\overline{\text{Load}_i}$ represents the expected load per expert assuming perfect balance.

To measure the diversity of token assignments across experts, we compute the *Entropy* of the average selection probabilities for each expert:

$$H = -\sum_{i=1}^{N_r} \overline{p}_i \log \overline{p}_i, \tag{9}$$

where $\overline{p}_i$ is the average probability of selecting the i-th expert over all tokens in the evaluation batch, and N_r is the total number of experts. A higher entropy indicates a more uniform distribution of tokens among experts, reflecting balanced and diverse routing decisions.

Results from Table 9 reveal that applying an explicit balancing loss significantly reduces MaxVio and increases entropy, leading to improved load balance without sacrificing sampling accuracy or overall model performance. Loss-free approaches, while simpler, tend to show higher MaxVio and lower entropy, indicating less balanced token routing. Architectures such

Table 8: Ablation results on using expert-choice router with different routing configurations. Coeff denotes coefficient values for auxiliary loss term, and α denotes scaling term after router function. Dead token ratio denotes the proportion of tokens that remain unselected during the last recursion step, measured within evaluation batch size of 500.

Expert-choice Configurations							Router Metrics		NLL ↓	Few-shot Accuracy ↑						
Sampling	Coeff	Func	α	Arch	Warmup	Z-loss	Dead ↓	Samp-Acc ↑	FineWeb	LD	HS	PQ	WG	ARC	MMLU	Avg
-	-	rand	-	-	✗	✗	0.0	-	2.9335	26.0	33.1	61.6	52.3	35.8	26.2	39.1
Aux Router	-	-	-	MLP	✗	✗	66.7	50.0	NaN	0.0	25.04	49.5	49.6	23.9	23.0	28.5
Aux Router	-	σ	0.1	MLP	✗	✗	0.0	89.2	2.8893	26.1	33.8	62.0	51.5	36.6	26.4	39.4
Aux Router	-	σ	1.0	MLP	✗	✗	66.7	50.0	2.8867	26.4	33.6	63.0	52.4	37.0	24.1	39.8
Aux Router	-	tanh	0.1	MLP	✗	✗	66.7	98.6	2.8720	13.9	31.8	60.7	49.3	35.8	25.8	36.2
Aux Router	-	tanh	1.0	MLP	✗	✗	66.7	97.0	3.0624	18.26	29.7	60.1	50.9	34.6	25.5	36.5
Aux Loss	0.01	-	-	MLP	✗	✗	66.7	50.0	NaN	0.0	25.04	49.5	49.6	23.9	23.0	28.5
Aux Loss	0.01	σ	0.1	MLP	✗	✗	0.0	99.6	2.8967	24.8	33.6	63.3	50.3	36.6	26.6	39.2
Aux Loss	0.01	σ	1.0	MLP	✗	✗	65.9	100.0	2.9189	12.0	31.6	59.4	51.5	33.2	25.3	35.5
Aux Loss	0.01	tanh	0.1	MLP	✗	✗	32.8	99.7	2.9426	23.5	32.4	62.4	49.8	35.6	26.0	38.3
Aux Loss	0.01	tanh	1.0	MLP	✗	✗	0.0	98.8	3.2743	16.4	28.14	58.8	52.2	31.6	24.8	35.3
Aux Loss	0.1	σ	0.1	MLP	✗	✗	0.0	99.8	3.0416	21.5	31.0	61.8	50.3	35.0	26.0	37.6
Aux Loss	0.001	σ	0.1	MLP	✗	✗	0.0	99.1	2.8816	27.6	34.3	63.0	51.6	36.7	26.5	40.0
Aux Loss	0.001	tanh	0.1	MLP	✗	✗	0.0	56.4	2.9933	25.0	32.3	61.5	51.5	36.6	26.0	38.8
Aux Loss	0.001	σ	0.1	Linear	✗	✗	0.1	99.2	2.8667	27.4	34.6	63.2	51.5	37.2	26.5	40.1
Aux Loss	0.001	σ	0.1	W-MLP	✗	✗	0.4	99.2	2.8716	27.8	33.9	62.4	49.9	36.3	26.3	39.4
Aux Loss	0.001	σ	0.1	Linear	✓	✗	4.9	99.1	2.8744	26.0	33.9	62.0	51.2	36.1	26.1	39.2
Aux Loss	0.001	σ	0.1	Linear	✗	✓	0.0	99.3	2.8824	26.9	34.0	63.8	52.3	36.8	26.4	40.0

as MLP and Linear routers perform comparably under balancing loss, with z-loss often contributing to improved routing stability and model accuracy. These findings highlight the importance of explicit balancing mechanisms in maintaining effective token-choice routing behavior.

Table 9: Ablation results on token-choice router under different routing configurations. Coeff denotes coefficient values for balancing loss term, and u values for loss-free method. MaxVio measures the maximum relative deviation of token load assigned to any expert, quantifying the degree of load imbalance.

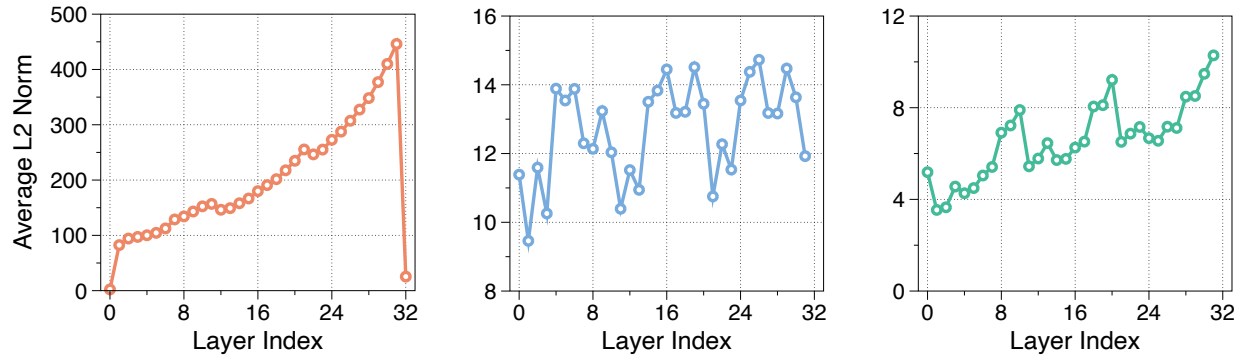
Token-choice Configurations						Router Metrics		NLL ↓	Few-shot Accuracy ↑						
Balancing	Coeff	Func	α	Arch	Z-loss	MaxVio ↓	Entropy ↑	FineWeb	LD	HS	PQ	WG	ARC	MMLU	Avg
-	-	rand	-	-	✓	0.007	1.099	3.0268	24.8	32.0	61.4	52.2	35.5	26.1	38.7
Loss	0.1	soft	1.0	MLP	✓	0.200	1.076	3.0239	24.2	31.9	61.4	51.5	35.7	26.2	38.5
Loss	0.01	soft	1.0	MLP	✓	0.682	0.921	2.9118	28.0	33.3	62.8	49.7	36.4	26.2	39.4
Loss-free	0.01	soft	1.0	MLP	✓	1.788	0.297	2.9078	25.5	32.5	61.3	52.3	36.1	26.0	38.9
Loss-free	0.01	σ	0.1	MLP	✓	0.956	0.646	3.1144	21.8	29.8	60.3	51.6	34.0	25.7	37.2
Loss-free	0.01	σ	1.0	MLP	✓	0.918	0.749	3.0188	23.4	31.3	59.9	50.0	35.2	25.8	37.6
Loss-free	0.001	soft	1.0	MLP	✓	0.852	0.915	2.9081	25.8	33.6	62.8	50.6	37.5	26.7	39.5
Loss-free	0.001	σ	0.1	MLP	✓	1.281	0.551	2.9165	23.9	33.1	61.2	51.6	37.3	26.2	38.9
Loss-free	0.001	σ	1.0	MLP	✓	0.542	0.941	3.0188	24.9	32.0	61.9	51.4	35.5	25.9	38.6
Loss	0.1	soft	1.0	Linear	✓	0.492	0.960	2.9974	23.7	31.3	62.2	50.3	36.7	26.0	38.4
Loss	0.1	soft	1.0	W-MLP	✓	0.384	1.037	3.0293	25.3	31.5	62.2	51.2	36.4	26.3	38.8
Loss	0.1	soft	1.0	Linear	✗	0.266	1.056	2.9358	25.7	32.6	61.9	51.7	36.4	26.5	39.1

I. Expanded Results of KV Cache Sharing

I.1. KV Representation Trends in Recursive Transformers

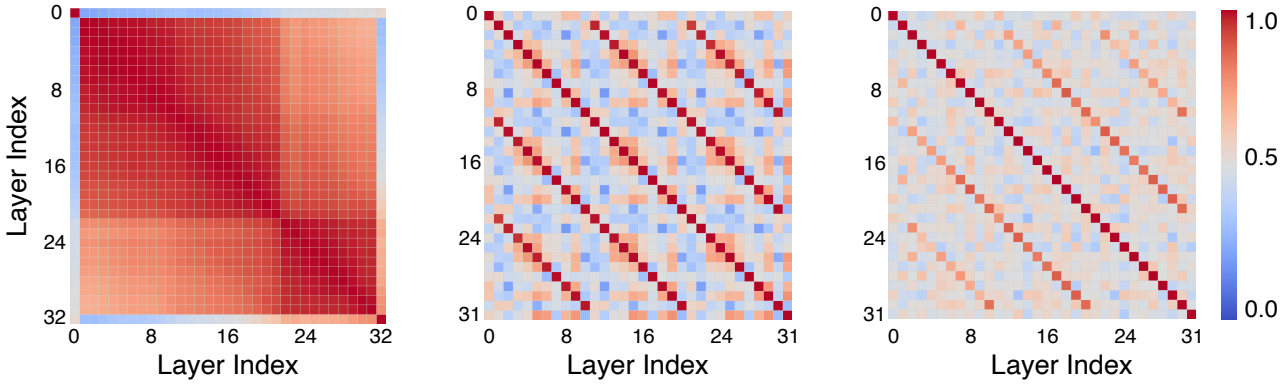
Sharing KV caches across model depths has emerged as a promising approach to improve inference throughput in Vanilla Transformers (Brandon et al., 2024). This technique not only reduces the memory footprint required for KV caches, but also decreases the time needed to transfer them from HBM to SRAM within the GPU hierarchy. Due to the high degree of freedom in Vanilla models—where trainable parameters can be optimized for shared caches—these models exhibit only marginal performance drops when KV caches are shared between adjacent layers.

In contrast, Recursive Transformers have far fewer parameters available for aligning to tied key-value states. Nevertheless, we hypothesize that similar patterns may emerge between shared attention blocks. To investigate this, we pretrained Recursive Transformers using the Middle-Cycle strategy with a recursion depth of 3, decomposed the KV states into magnitude and directional components, and visualized the results in Figure 6 and Figure 7.



(a) Hidden states (b) Key states (c) Value states

Figure 6: Average L2 norm magnitude of (a) hidden states, (b) key states, and (c) value states across layers in a Middle-Cycle Recursive Transformer 360M with 3 recursion steps. Note that the last hidden states correspond to the final hidden states after the last layer normalization.



(a) Hidden states (b) Key states (c) Value states

Figure 7: Cosine similarity matrices showing the layer-wise similarity of (a) hidden states, (b) key states, and (c) value states in Recursive Transformer with Middle-Cycle strategy and recursion depth 3. Results are from a 360M parameter model with 32 layers. The hidden states matrix includes the final hidden states after the last layer normalization.

As shown in Figure 6, the sharing of key and value projection layers across recursion depths leads to clear recursive patterns in the magnitude values. Although the magnitudes of hidden states tend to increase, the projection layers appear to be trained to produce similar signal sizes at corresponding depths within each recursion.

We also measured the cosine similarity between the unit vectors of KV states. In these visualizations, red indicates

higher similarity between the key or value states of each layer, while blue indicates lower similarity. As illustrated in Figure 7, distinct diagonal patterns emerge, suggesting that shared projection layers generate highly similar key and value representations. While sharing value states across recursions appears to be more challenging than sharing key states, these findings suggest that the performance drop from KV cache sharing can remain marginal even in Recursive Transformers.

I.2. Performance Comparison of KV Sharing Strategy

In Table 10, we present the performance results when applying KV cache sharing to Vanilla, Recursive, and MoR models. Especially, we tested various strategies for KV caches, including Cycle or Sequence strategies that share the same concepts as parameter sharing (see subsection C.1 for details). Interestingly, KV cache sharing even improves the performance of vanilla models, where sharing acts as a regularization technique. For recursive models, we aligned the sharing strategy for parameters and KV caches. Despite some variations in the results after applying KV sharing, the Middle-Cycle strategy (the best parameter sharing strategy) showed a slight perplexity drop, albeit not substantial.

When moving to MoR models, they still introduced a small amount of degradation in our best settings (expert-choice router). However, considering the reduced parameter sizes and cache sizes, we believe this minor drop is acceptable. Furthermore, we explored an additional caching strategy (indicated by †) that utilized shared caches for inactive (unselected) positions while updating active positions through actual computation. Although it did not provide additional benefits, it is still worth exploring combinations of KV sharing and actual updates, such as only updating current a single token position, which would not introduce additional memory footprint with minimal FLOPs requirements during decoding.

Table 10: Comparison of KV cache sharing strategies across Vanilla, Recursive, and MoR Transformers. Models are evaluated using negative log-likelihood (NLL) on train set and few-shot accuracy across benchmarks. KV sharing denotes use of recursive KV sharing. † indicates training with KV sharing that updates the KV cache with the outputs generated after each recursion step.

	Pretrain		**Recursion**		**MoR**	**KV Sharing**		**NLL ↓**	**Few-shot Accuracy ↑**						
Models	N-Emb	N_L	Share	Loop	Type	Share	Loop	FineWeb	LD	HS	PQ	WG	ARC	MMLU	Avg
Vanilla	315M	32	-	-	-	-	-	2.8471	27.3	34.8	64.2	52.8	38.3	26.7	40.7
Vanilla	315M	32	-	-	-	Seq	2	2.7848	30.0	36.5	64.6	50.7	39.4	26.9	41.3
Vanilla	315M	32	-	-	-	Cyc	2	2.7650	30.0	36.7	65.2	51.1	39.6	27.5	41.7
Vanilla	295M	30	-	-	-	-	-	2.8069	29.1	35.6	65.1	50.4	38.5	27.3	41.0
Vanilla	295M	30	-	-	-	Seq	3	2.7879	28.3	36.4	64.3	52.7	39.4	27.3	41.4
Vanilla	295M	30	-	-	-	Cyc	3	2.7890	28.9	36.5	64.6	51.4	39.0	27.6	41.3
Recursive	157M	16	Seq	2	-	-	-	2.9467	26.3	32.5	62.9	52.4	36.4	26.2	39.5
Recursive	157M	16	Seq	2	-	Seq	2	2.8904	26.4	33.4	64.0	51.0	37.0	26.9	39.8
Recursive	157M	16	Cyc	2	-	-	-	2.8487	28.5	34.8	63.1	50.0	37.4	28.8	40.1
Recursive	157M	16	Cyc	2	-	Cyc	2	2.8577	26.2	34.5	64.2	51.4	37.3	26.9	40.1
Recursive	167M	1+15+1	M-Cyc	2	-	-	-	2.8295	28.6	35.0	64.5	50.5	39.7	27.2	40.9
Recursive	167M	1+15+1	M-Cyc	2	-	M-Cyc	2	2.8451	27.3	34.7	63.7	50.5	37.8	27.0	40.2
Recursive	98M	10	Seq	3	-	-	-	3.0245	24.6	31.5	63.1	49.3	35.7	25.7	38.3
Recursive	98M	10	Seq	3	-	Seq	3	2.9554	24.2	32.3	62.5	52.7	36.6	26.2	39.1
Recursive	98M	10	Cyc	3	-	-	-	2.9363	25.9	33.0	62.9	50.3	36.4	26.5	39.2
Recursive	98M	10	Cyc	3	-	Cyc	3	2.9155	24.1	32.9	62.4	51.2	37.4	26.7	39.1
Recursive	118M	1+10+1	M-Cyc	3	-	-	-	2.8760	28.5	34.9	64.3	50.5	39.5	27.2	40.8
Recursive	118M	1+10+1	M-Cyc	3	-	M-Cyc	3	2.8854	27.3	33.8	63.3	52.3	37.5	26.8	40.2
MoR	118M	1+10+1	M-Cyc	3	Expert	-	-	2.8667	27.4	34.6	63.2	51.5	37.2	26.5	40.1
MoR	118M	1+10+1	M-Cyc	3	Expert	M-Cyc	3	2.8895	34.0	61.6	50.2	26.0	36.5	27.0	39.2
MoR	118M	1+10+1	M-Cyc	3	Expert	M-Cyc†	3	2.8653	24.8	34.3	62.0	50.1	36.7	26.7	39.1
MoR	118M	1+10+1	M-Cyc	3	Token	-	-	2.9358	25.7	32.6	61.9	51.7	36.4	26.5	39.1
MoR	118M	1+10+1	M-Cyc	3	Token	M-Cyc	3	2.9155	25.7	32.6	61.8	49.4	36.2	26.0	38.6

We also investigated relaxing the constraints on KV sharing in Table 11, similar to the approach taken by Bae et al. (2024) for parameter sharing. Specifically, we re-examined four relaxation techniques for standard Recursive Transformers with very small ranks or prefix lengths. Our results indicate that these techniques do not yield significant performance gains, which makes sense in that they introduce only a small number of additional parameters. Although we hypothesized that incorporating prefix-based approaches (such as adding trainable prefixes to attention) into KV sharing might lead to greater benefits, our experiments did not reveal substantial differences in this regard. Further exploration of more sophisticated techniques for efficiently relaxing KV cache sharing constraints remains an open direction for future research.

Table 11: Comparison of KV Cache sharing and relaxation methods in Recursive Transformers trained on FineWeb-Edu (Penedo et al., 2024) with 10B tokens and 360M parameters. Models are evaluated on train NLL and few-shot accuracy across multiple benchmarks. Relaxation types include encoding trainable parameters on recursion hidden states (Enc), LoRA and DoRA applied to QV matrices and adaptation prompt tuning (Adapt-P). All models use the Middle-Cycle sharing strategy with 3 recursion depths.

	Pretrain		Relaxation			KV Sharing		NLL ↓	Few-shot Accuracy ↑						
Models	N-Emb	N_L	Type	Rank	Len	Share	Loop	FineWeb	LD	HS	PQ	WG	ARC	MMLU	Avg
Recursive	118M	1+10+1	-	-	-	-	-	2.8854	27.3	33.8	63.3	52.3	37.5	26.8	40.2
Recursive	118M	1+10+1	Enc	-	-	-	-	2.8604	27.3	34.6	63.9	53.4	38.6	26.7	40.2
Recursive	124M	1+10+1	LoRA	64	-	-	-	2.8599	27.3	34.6	64.3	50.9	38.0	26.9	39.7
Recursive	124M	1+10+1	DoRA	64	-	-	-	2.8945	26.4	33.6	64.4	50.6	37.4	26.5	39.2
Recursive	126M	1+10+1	Adapt-P	-	256	-	-	2.8626	27.1	34.7	64.0	51.9	37.6	26.8	39.7
Recursive	118M	1+10+1	-	-	-	M-Cyc	3	2.8854	27.3	33.8	63.3	52.3	37.5	26.8	40.2
Recursive	126M	1+10+1	Adapt-P	-	256	M-Cyc	3	2.9030	24.5	33.1	63.0	52.2	26.7	37.6	39.5

J. Compute-optimal Scaling Analysis

Figure 8a shows that Mixture-of-Recursions exhibits a distinct scaling behavior compared to Vanilla and Recursive Transformers under isoFLOPs constraints. The sharper curvature along the model size axis suggests that MoR gains more from increasing parameter count, while its relatively flat trajectory in terms of compute implies that extending training length yields smaller marginal benefits. This indicates that, for MoR, allocating budget toward larger models trained for shorter durations may be more effective, highlighting a different trade-off dynamic from that of baseline architectures.

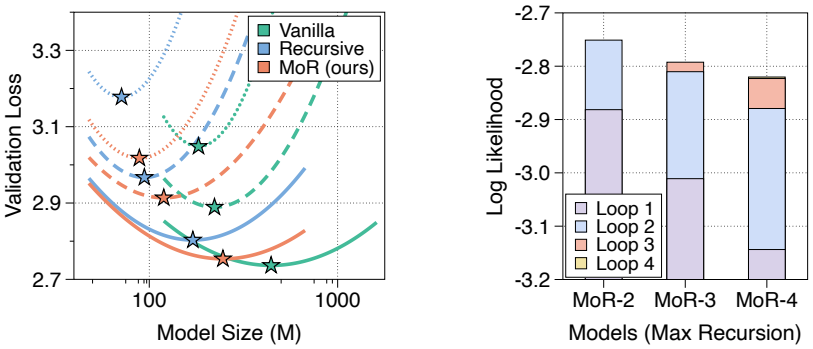
K. Test-time Scaling Analysis

We visualize how log-likelihood evolves across recursion steps in MoR models with $N_r = \{2, 3, 4\}$ in Figure 8b. For each model, we plot the log-likelihoods of tokens that exit at each recursion depth, shown as overlaid bars. As recursion proceeds, the log-likelihood consistently improves, indicating that additional compute through deeper recursion leads to better performance. These results support the view that MoR enables test-time scaling: allocating more recursion steps at inference can improve generation quality.

L. Qualitative Results

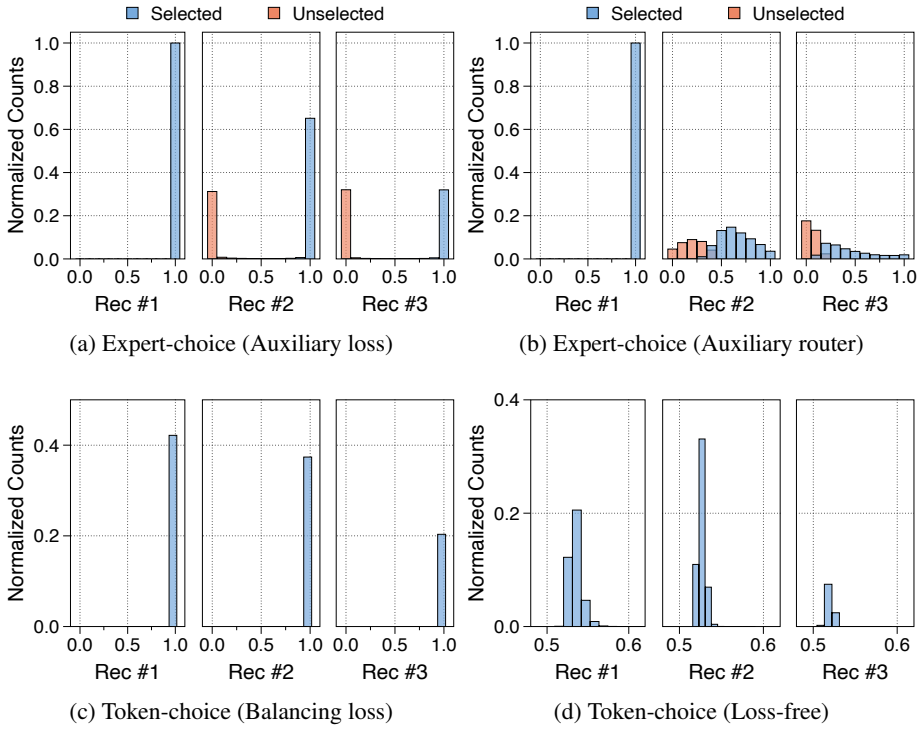
L.1. MoR Router Weights

To gain insight into the optimization of router output distributions, we visualized the results in Figure 9. Our analysis reveals that various routing mechanisms are trained to balance loads of experts according to the desired capacity. Notably, expert-choice routers achieved nearly perfect load balancing with the auxiliary loss, resulting in almost binary values (1 or 0) for selected and unselected tokens, respectively. Other strategies also demonstrated good balancing properties, with reasonable router values that will be multiplied to refine the outputs of corresponding recursion depths. However, most cases failed to converge to optimal balancing (i.e., these are edge cases where they achieved their own optimal load balancing), highlighting the challenges of achieving consistent performance except expert-choice with auxiliary loss.



(a) Compute-optimal scaling (b) Test-time scaling

Figure 8: (a) Compute-optimal scaling analysis. Each star indicates the optimal combination of model size and number of training tokens for each architecture under a fixed compute budget. (c) Test-time scaling analysis showing the cumulative log-likelihood improvement with each additional recursion loop, measured over 500 samples.



(a) Expert-choice (Auxiliary loss) (b) Expert-choice (Auxiliary router)

(c) Token-choice (Balancing loss) (d) Token-choice (Loss-free)

Figure 9: Distribution of router weights for selected and unselected tokens at each recursion step in expert-choice and token-choice MoR (N_r=3). The subplots show results for (a) expert-choice routing with auxiliary loss, (b) auxiliary router, (c) token-choice routing with balancing loss, and (d) loss-free routing.

L.2. Analysis on adaptive computation paths

Table 12 illustrates a qualitative analysis of the recursion depth assigned to each text token across ten diverse samples. This visualization provides a detailed insight into how tokens within each sample exhibit varying levels of recursive processing. Notably, some tokens exit early (blue), while others require deeper processing (purple and red), reflecting the model's ability to focus more compute on challenging parts of the input.

Table 12: Visualization of the recursion depth for each text token, with colors representing the number of recursion steps: 1, 2, and 3. Each row corresponds to a different sample.

Sample	Text
Sample #1	code and will serve as a good introduction to the syntax necessary for creating shell programs. We will continue by ... ... Dedu ction ¿¿ Federal : Char
Sample #2	2 014, 7:57 AM Space X Falcon 9 –R Rocket Suff ers ... ... the Ottoman Empire seems to have succeeded.
Sample #3	children ? Who else (if anyone) would you want to know about your genetic test results? Object ives ... ... We urge you to reject the pipeline and
Sample #4	keep tar sands oil in the ground where it belongs. The representatives were joined by the National Wildlife Federation and ... ... -W . Va ., called the
Sample #5	9 PR Newsw ire . All rights reservedA report released Thursday on the Slide Fire in Oak Creek ... ... he has selected a small hob bit like
Sample #6	Bil bo to accompany the dwarves to fight the enemy. He says, "S ar uman believes it is ... ... to a new area. Light -Trans
Sample #7	mitting Plants After four years of research, most of it in total darkness , a Stanford University plant biologist has ... ... the type of price spikes that could create
Sample #8	an impression of market power abuses or other market failures. In some cases, however, prices may spike to higher ... ... , the sun (and associated physics),
Sample #9	and spaceflight and spacecraft. covers elementary astronomy , Newtonian mechanics , the Sun and related physics and spaceflight . Also ... ... deployment in underlying database and network infrastructure ,
Sample #10	a well established role in scientific computing, and a recent increased presence in desktop computing, it almost certain that contemporary ... ... export them. Functions are memory resident shell