Mixture-of-Recursions

Learning Dynamic Recursive Depths for Adaptive Token-Level Computation

Authors: Sangmin Bae, Yujin Kim, Reza Bayat, et al.
Institutions: KAIST AI, Mila, Google Cloud, Google DeepMind, Google Research

Presented by: Zhaokun Wang
02.02.2026

Presented for: **Is Attention All You Need? The Search for a New Architecture Seminar**

01

The Scaling Challenge

Why do we need more efficient architectures?

The Scaling Challenge

The Transformer Scaling Dilemma

The Promise

Scaling LLMs to **hundreds of billions** of parameters unlocks impressive few-shot generalization and reasoning abilities

The Problem

Accompanying **computational and memory demands** make both training and deployment expensive and challenging outside hyperscale data centers

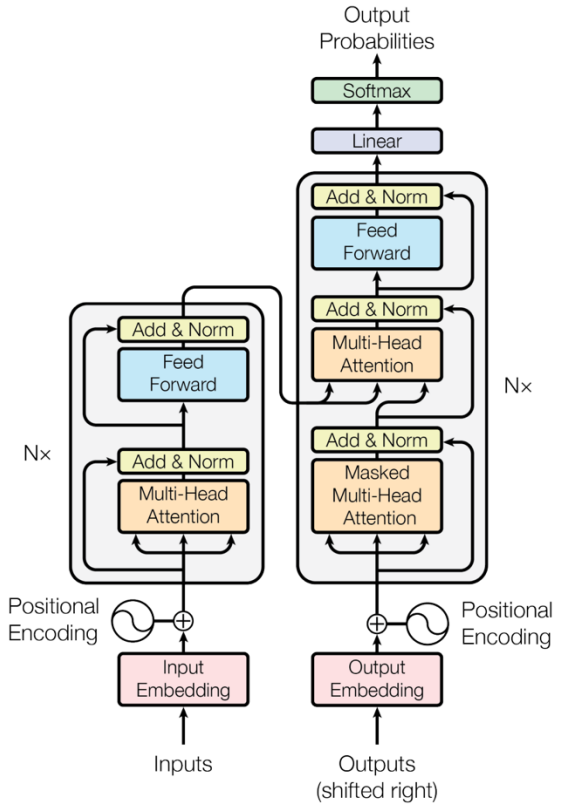


Figure 1: Transformer Architecture

The Scaling Challenge

Two Axes of Efficiency

Parameter Efficiency

Goal: Reduce or share model weights to decrease memory footprint

- **Layer tying:** Reuse shared weights across multiple layers
- **Weight sharing:** Same parameters for different computations

Adaptive Computation

Goal: Spend more compute only when it's actually needed

- **Early-exiting:** Exit earlier for simpler tokens/sequences
- **Dynamic allocation:** Vary compute based on input complexity

02

Existing Approaches

What has been tried before?

Existing Approaches

Recursive Transformers: A Promising Foundation

Core Concept

Recursive Transformers **repeatedly apply the same set of shared layers** multiple times instead of using unique layers at each depth

Key Benefit: Built-in weight sharing provides parameter efficiency

How It Works

1. Partition model into N_r **recursion blocks**
2. Each block uses **shared parameter pool Φ'**
3. Apply recursively for **more computation without more parameters**

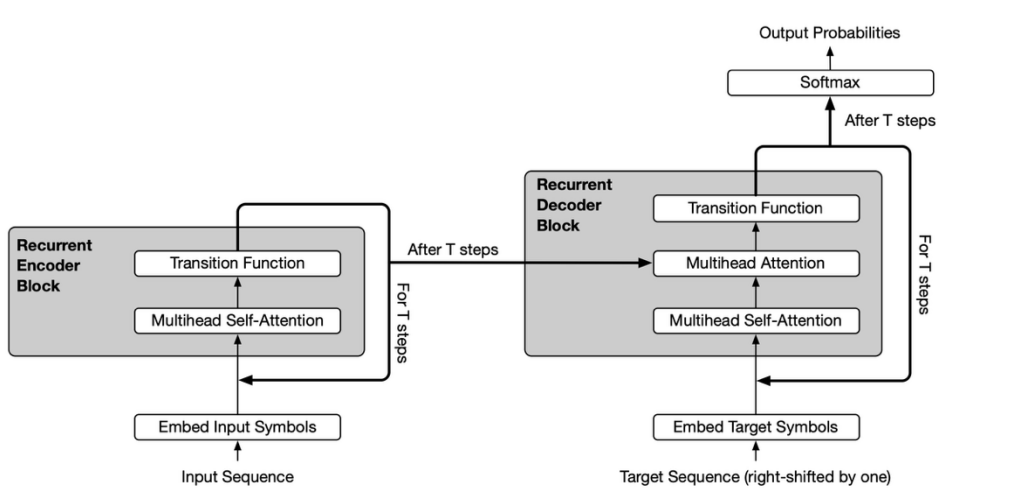


Figure 2: Transformer Architecture & Decoding Costs

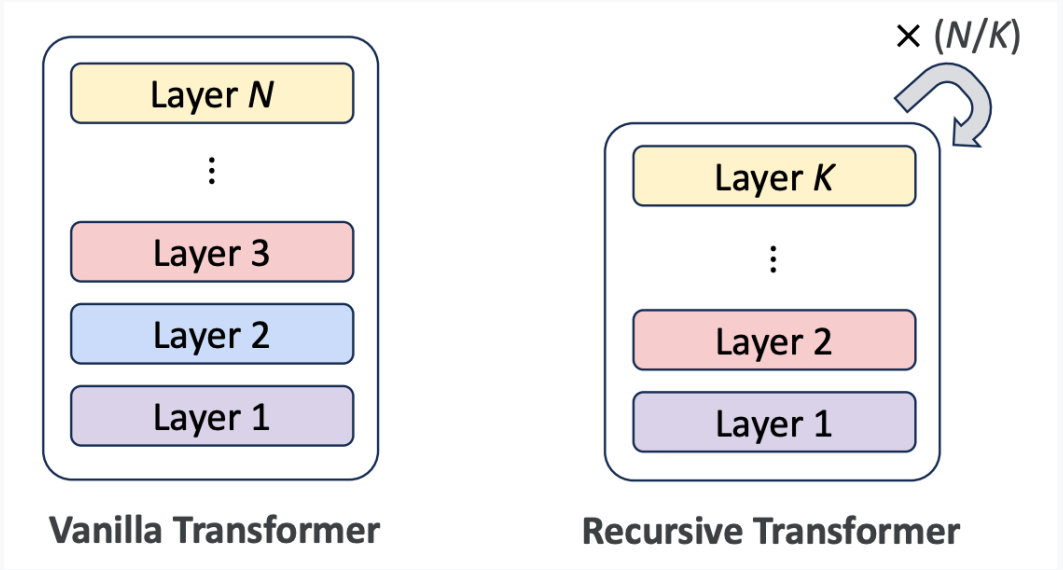


Figure 3: Recursive vs. Vanilla Transformer Architectures

Existing Approaches

Early exiting—stop processing when no more work is needed.

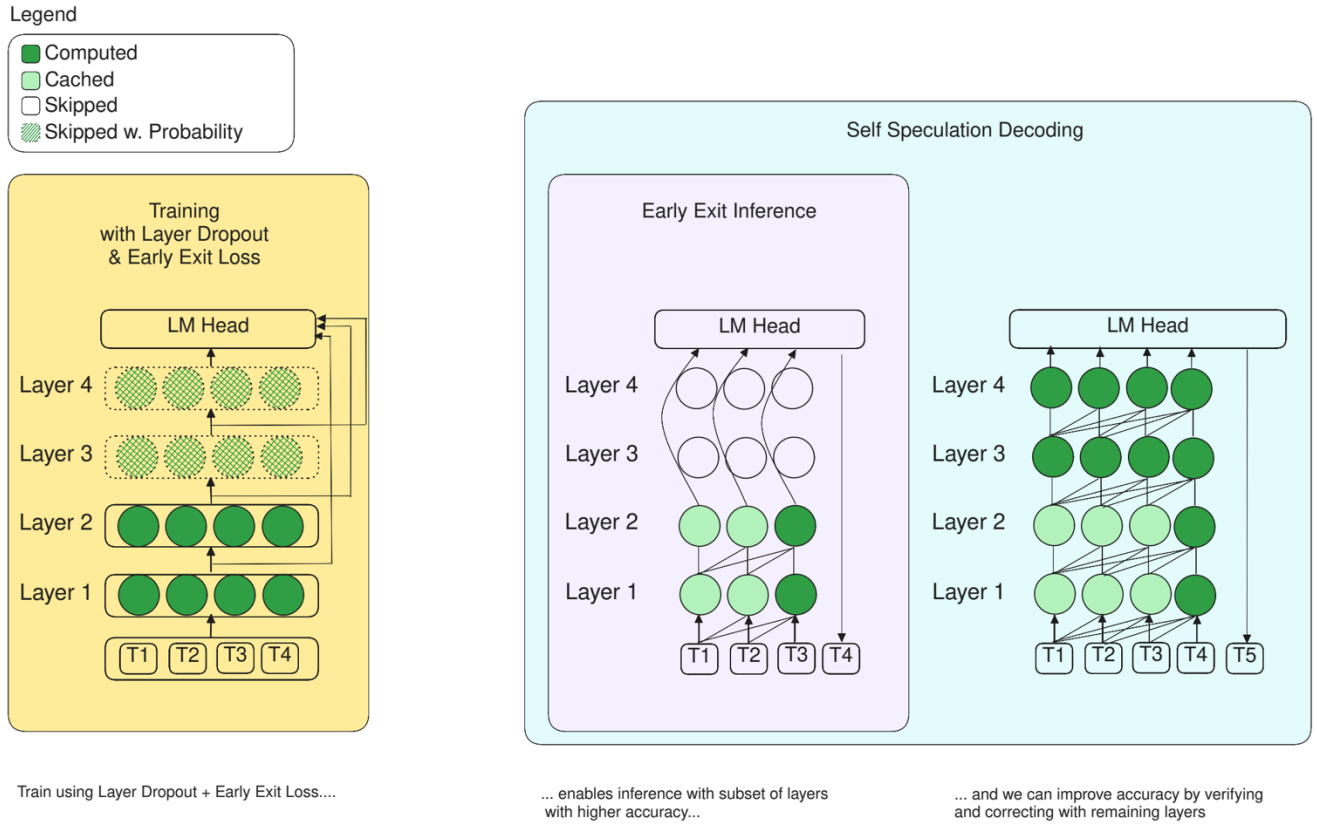


Figure 1 Overview of our end-to-end solution, LayerSkip, showing its 3 components.

Figure 5: LayerSkip Architecture Overview

Existing Approaches

Token-level adaptive computation

- Early exiting: stop processing when confidence is high
- Token routing (Mixture-of-Depths style): only some tokens compute at deeper depth
- Hard parts:
- (1) train–inference causality mismatch
- (2) missing KV entries

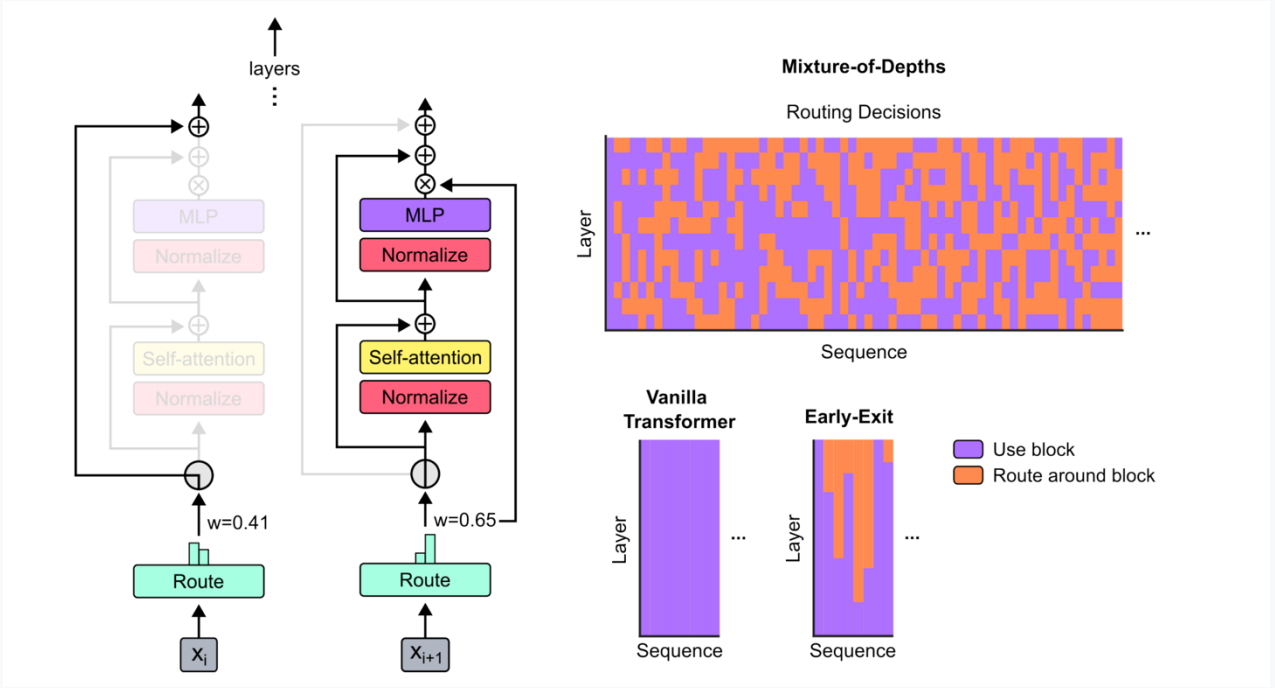


Figure 4: Mixture-of-Depths Routing Mechanism

Existing Approaches

The gap MoR targets

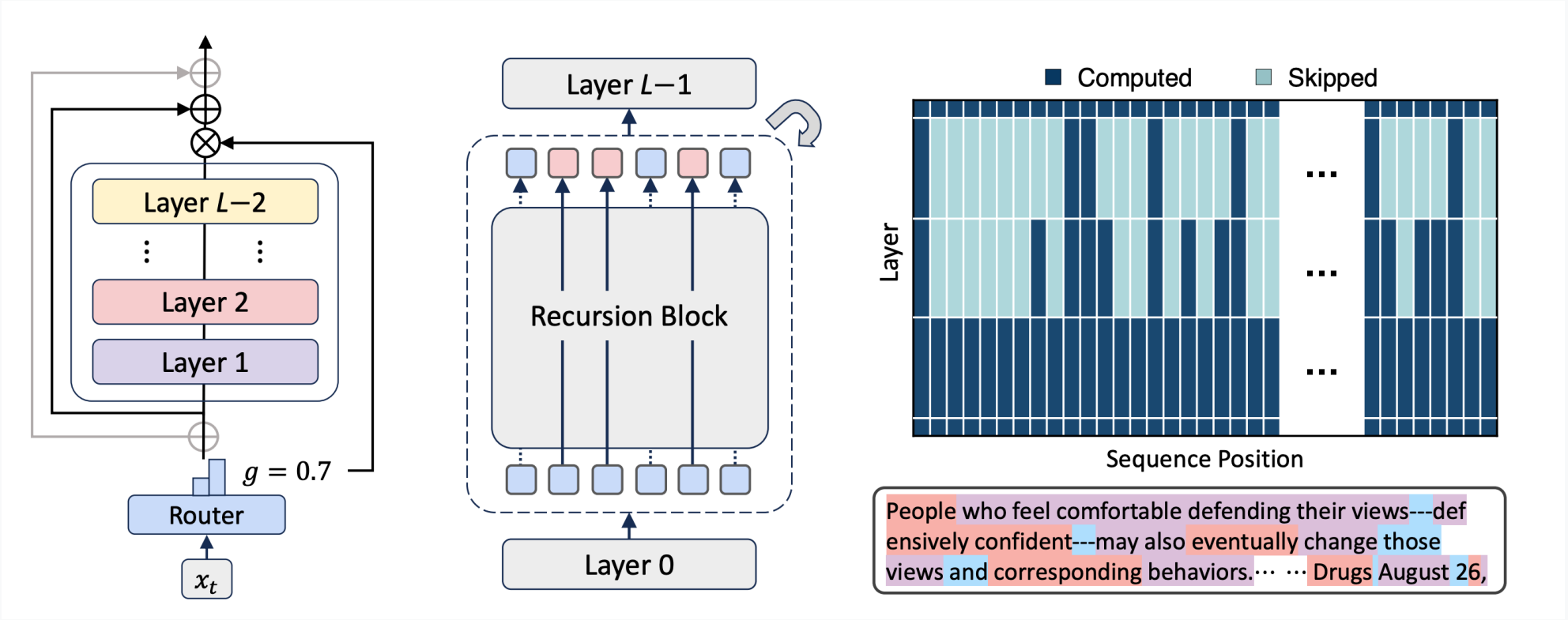
- We want: parameter sharing + token-level adaptive depth
- Prior work often needs post-hoc routing or special inference engineering
- MoR trains routing end-to-end and defines caching to match dynamic depth

03

Mixture-of-Recursions

A unified framework for efficient language modeling

The Core Idea



- Left: Recursion Block (fixed stack + router).
- Middle: dynamic path per token (early exit / continue).
- Right: heatmap of computation (darker = more recursion).

Figure 6: Mixture-of-Recursions (MoR) Core Framework

MoR Architecture Overview

1 Parameter Sharing

Reuse a shared stack of layers across recursion steps

2 Dynamic Routing

Lightweight routers decide per-token recursion depth

3 Efficient KV Caching

Selective caching strategies reduce memory traffic

Parameter Sharing Strategies

Cycle Strategy

Layers are **reused cyclically** across recursion steps

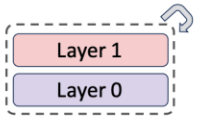
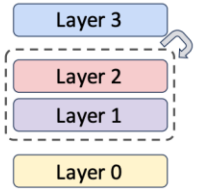
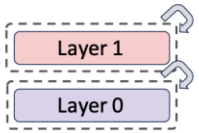
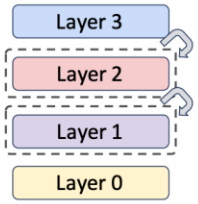
[(0,1,2), (0,1,2), (0,1,2)]

Sequence Strategy

Each recursion **reuses the same layer consecutively**

[(0,0,0), (1,1,1), (2,2,2)]

Table 1: Parameter Sharing Strategies

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

The Core Mechanism: Routing

Goal: assign token-specific recursion depth.

Why?

- “Easy” tokens (e.g., “the”, “a”) need less compute.
- “Hard” tokens need deeper reasoning.

Two strategies:

- Expert-choice routing
- Token-choice routing

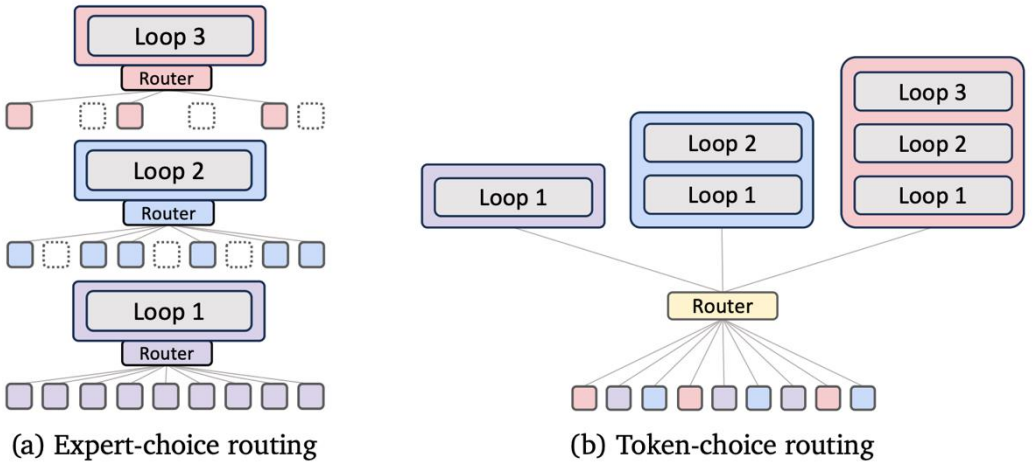


Figure 7: Expert-Choice vs. Token-Choice Routing

Routing Strategy 1: Expert-Choice

- Mechanism:
  Each recursion depth acts as an “expert”. Selects Top-k tokens to process at each step.
- Pros: perfect load balancing (k is fixed).
- Cons: potential information leakage (needs future info for top-k).
- Fix: auxiliary loss to predict top-k without leakage.

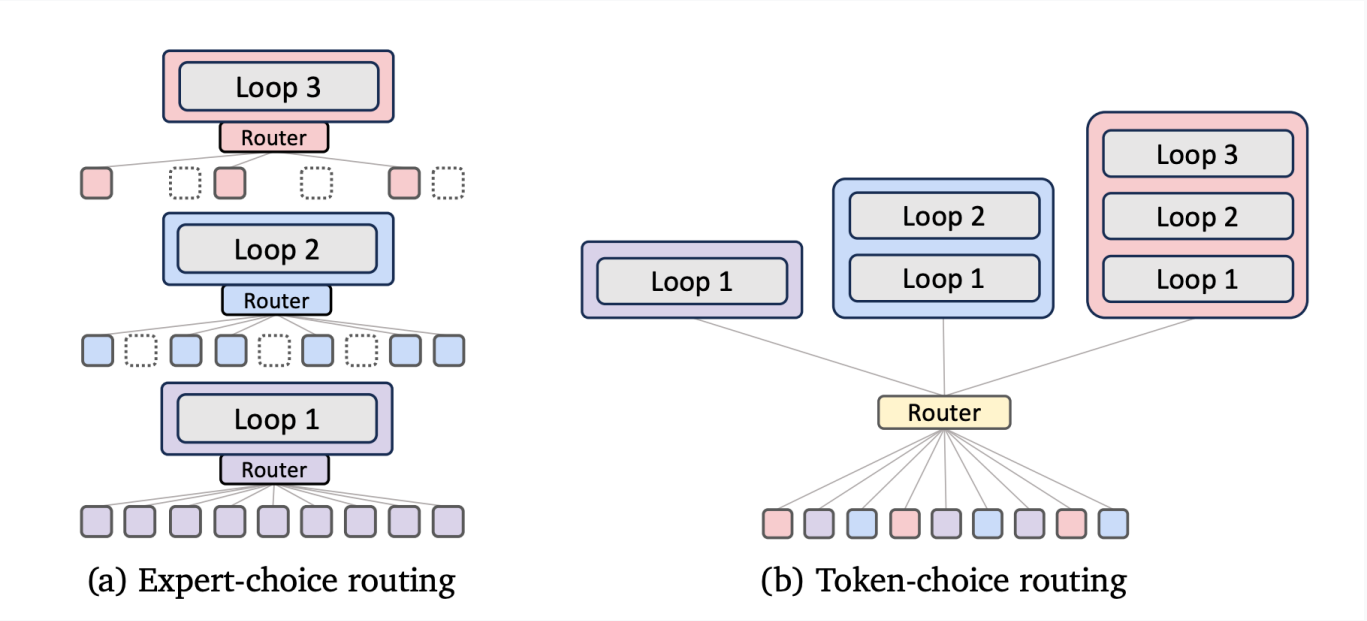


Figure 7: Expert-Choice vs. Token-Choice Routing

Advantages

Perfect load balancing with static top-k selection. **Guaranteed compute budget** simplifies resource management

Challenge

Causality violation: Top-k selection requires future token information, causing information leakage

Routing Strategy 2: Token-Choice

- Mechanism:

  Router decides the entire path at the start.

  Token t is assigned to depth i via softmax / top-1.
- Pros: independent per token, no leakage.
- Cons: load imbalance (some steps empty/overloaded).
- Fix: balancing loss is crucial.

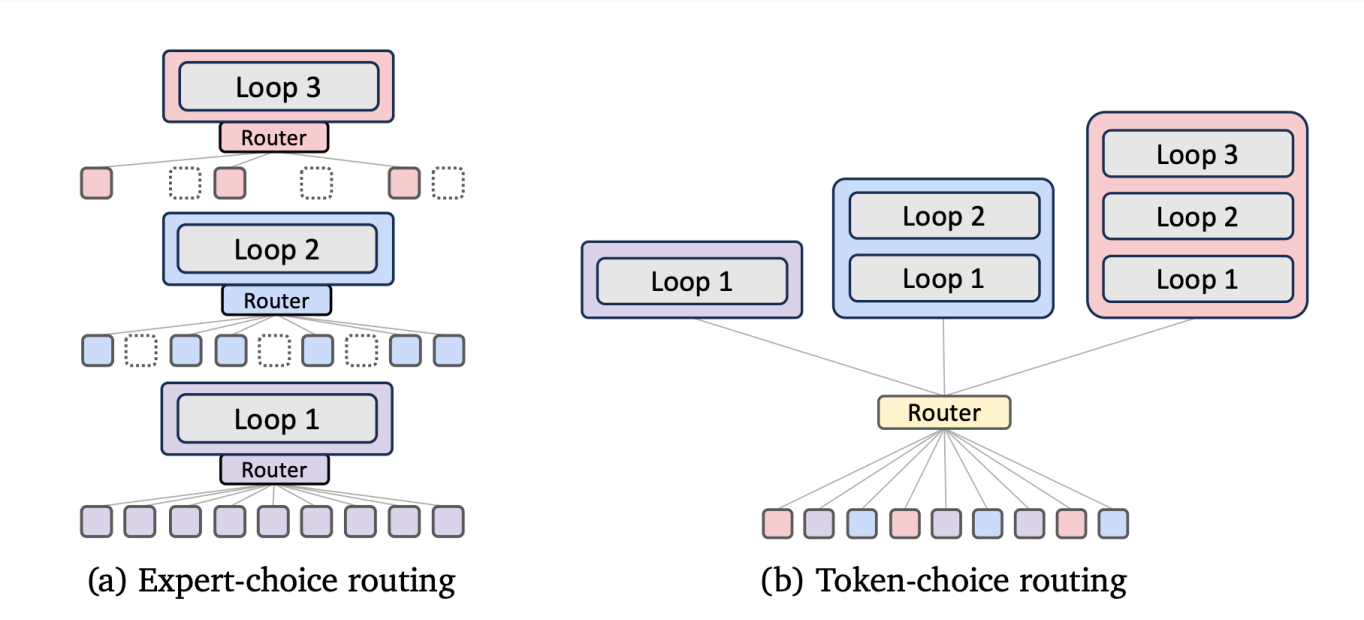


Figure 7: Expert-Choice vs. Token-Choice Routing

Advantages

No information leakage - preserves causality. **Natural autoregressive** behavior without auxiliary components

Challenge

Load imbalance: Requires balancing loss or loss-free algorithms to distribute tokens evenly

Comparing Routing Strategies

Table 2: Comparison of Routing Strategies

Aspect	Expert-Choice	Token-Choice
Compute Budget	Static - fixed top-k	Dynamic - per-token
Load Balancing	Perfect	Requires balancing
Causality	Violation (leakage)	No leakage
Solution	Auxiliary loss or router	Balancing loss / loss-free

When to Use Expert-Choice

- ✓ Need guaranteed compute budget
- ✓ Want perfect load balancing
- ✓ Can handle auxiliary components

When to Use Token-Choice

- ✓ Must preserve strict causality
- ✓ Want simpler architecture
- ✓ Can accept slight performance drop

The KV Cache Bottleneck

- Challenge in dynamic depth:
- If Token A exits at Step 1, and Token B goes to Step 3...
- Token B still needs to attend to Token A.
- Where are Token A's Keys/Values for Step 2 and 3?
- Standard solutions: recompute or parallel decode (slow/complex).
- MoR: specialized caching strategies designed for recursion.

Solution A: Recursion-wise Caching

- Method:
- Cache KV pairs only for tokens active at that step.
- Attention is restricted to locally cached tokens.
- Benefits:
- Significantly reduces KV memory footprint.
- Faster attention ops (shorter effective sequence k < N_ctx).

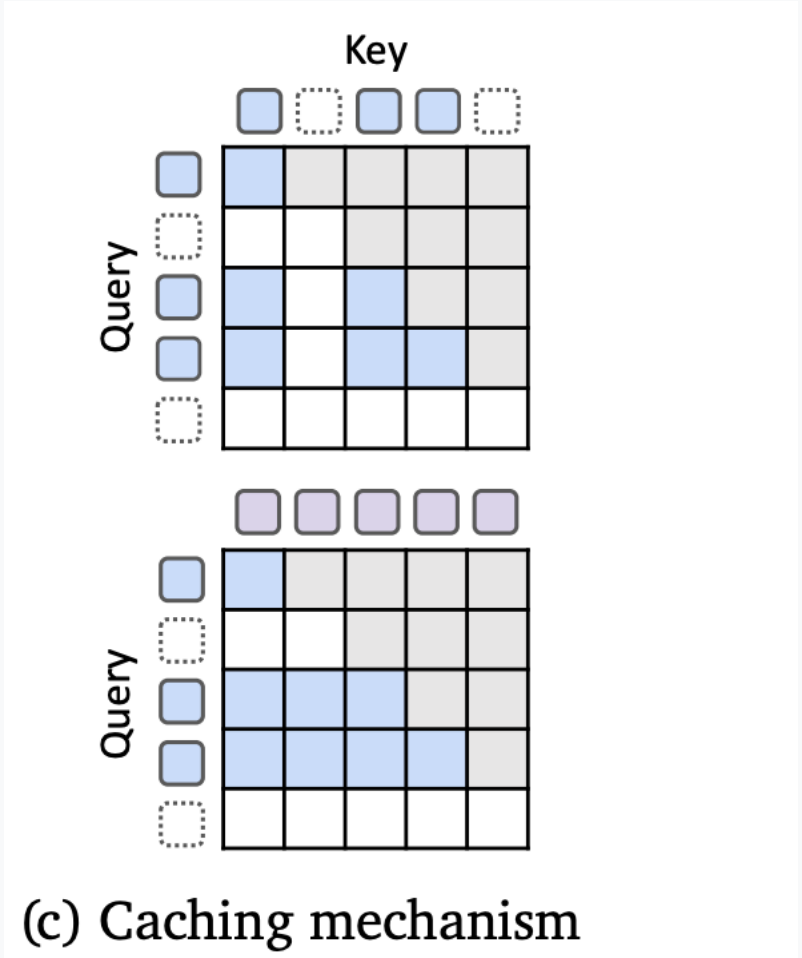


Figure 8: KV Caching Mechanisms

Solution B: Recursive KV Sharing

- Method:
- Compute KV once (at Step 1).
- Reuse KV for all subsequent recursion steps.
- Benefits:
- Maximum memory savings.
- No “missing keys” for deeper recursions.
- Trade-off: potential distribution mismatch (quality drop).

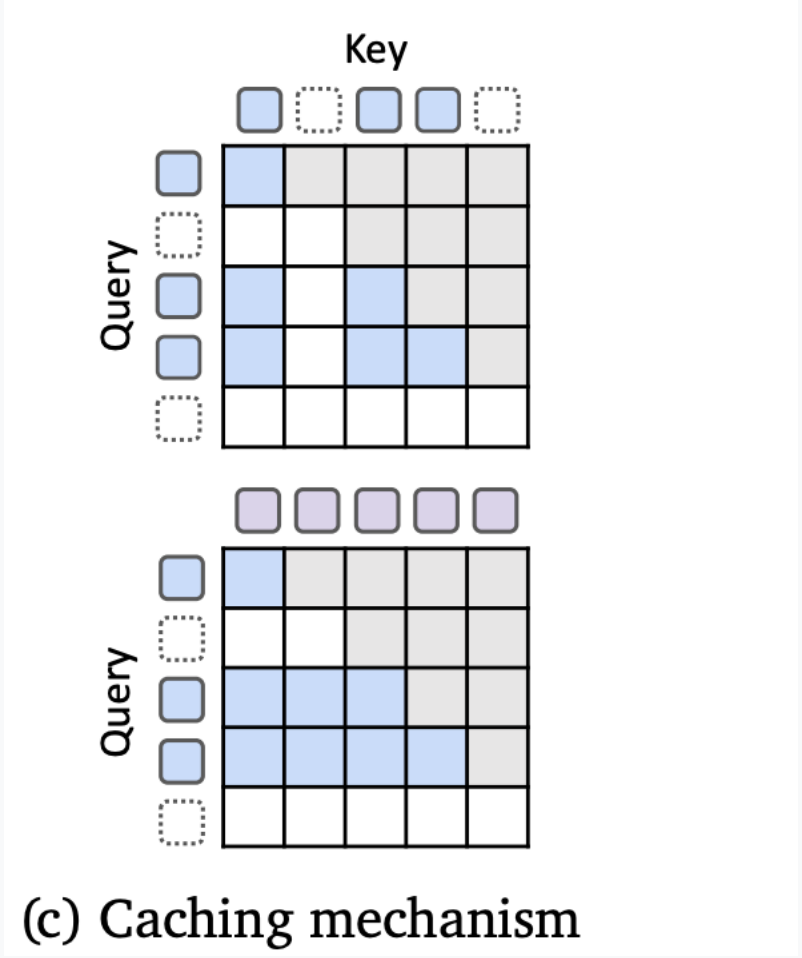
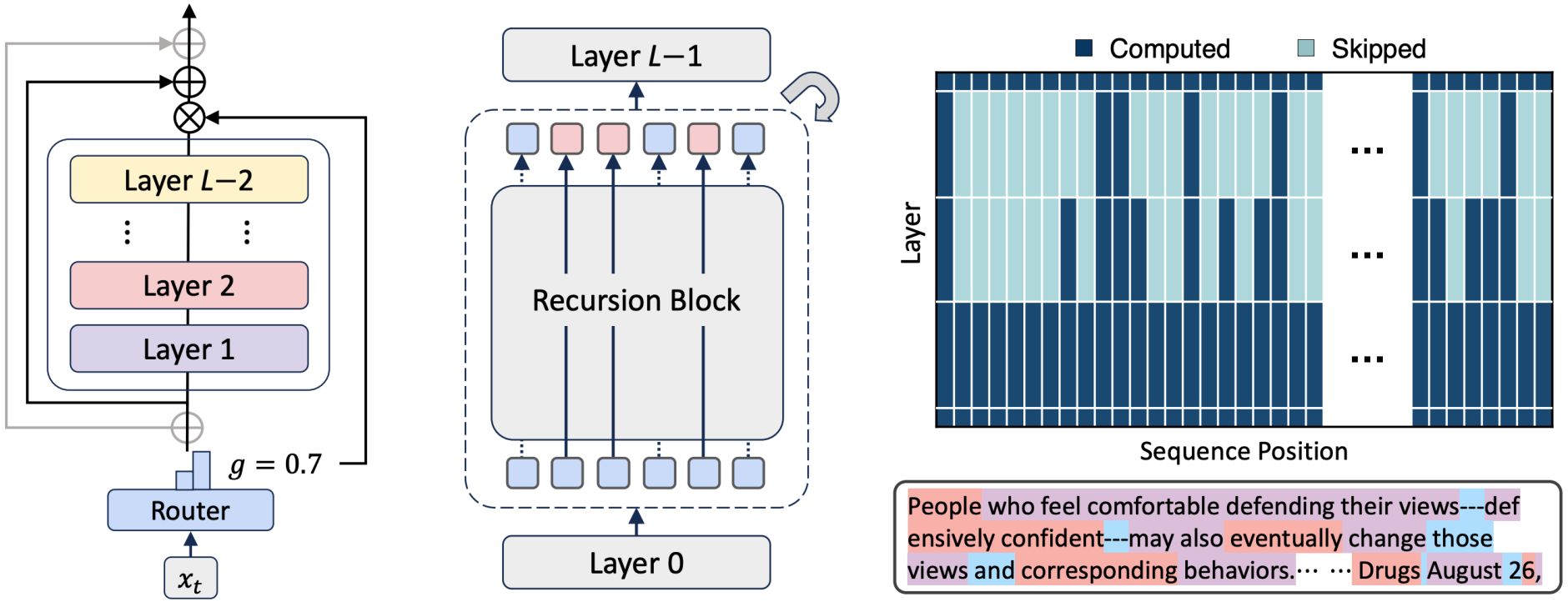


Figure 8: KV Caching Mechanisms

Look Back



- Left: Recursion Block (fixed stack + router).
- Middle: dynamic path per token (early exit / continue).
- Right: heatmap of computation (darker = more recursion).

Figure 6: Mixture-of-Recursions (MoR) Core Framework

04

Experiment & Results

Does MoR deliver on its promises?

Experimental Validation

Experimental Setup

Architecture

Llama-based Transformer architecture following SmolLM open-source model configurations

Dataset

FineWeb-Edu dataset (220B tokens from educational materials) in SmolLM-Corpus

Evaluation

Validation: Negative log-likelihood (NLL)

Few-shot: 6 benchmarks (LD, HS, PQ, WG, ARC, MMLU)

Model Scales

135M parameters	360M parameters
730M parameters	1.7B parameters

Training Settings

Compute Budgets	2e18, 5e18, 16.5e18 FLOPs
Fixed Tokens	20B tokens
Recursion Depths	N_r = 2, 3, 4

Experimental Validation

Results: Performance

Table 3: Main Performance Results

Models	MoR		Recursion		Pretrain			NLL↓	Few-shot Accuracy↑						
	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	Cache	M-Cyc	2	167M	16.5	27B	**2.7511**	**34.4**	**39.3**	65.7	51.2	39.6	**28.1**	**43.1**
	Expert	Cache	M-Cyc	3	118M	16.5	30B	2.7925	33.1	37.9	**66.9**	52.1	38.3	27.4	42.6
	Expert	Cache	M-Cyc	4	98M	16.5	30B	2.8204	30.1	37.3	65.0	51.1	38.9	27.4	41.6
	Expert	Cache	M-Cyc	2	167M	12.3	20B	2.7749	33.2	38.3	65.2	**52.6**	40.1	**28.1**	42.9
	Expert	Cache	M-Cyc	3	118M	11.0	20B	2.8246	31.9	37.0	65.7	50.5	38.3	27.4	41.8
	Expert	Cache	M-Cyc	4	98M	11.0	20B	2.8519	30.2	36.5	64.3	52.3	38.6	27.2	41.5
	Token	Cache	M-Cyc	3	118M	16.5	30B	2.9163	27.6	34.1	63.8	50.6	37.4	26.8	40.0
	Expert	Share	M-Cyc	3	118M	16.5	31B	2.7983	31.7	37.2	65.1	51.0	39.0	27.1	41.9

Higher accuracy—equal compute and token budgets

Experimental Validation

Results: Scaling Analysis

MOR:

- Matches or exceeds vanilla at larger scales (≥360M).
- Consistent improvement across compute budgets.

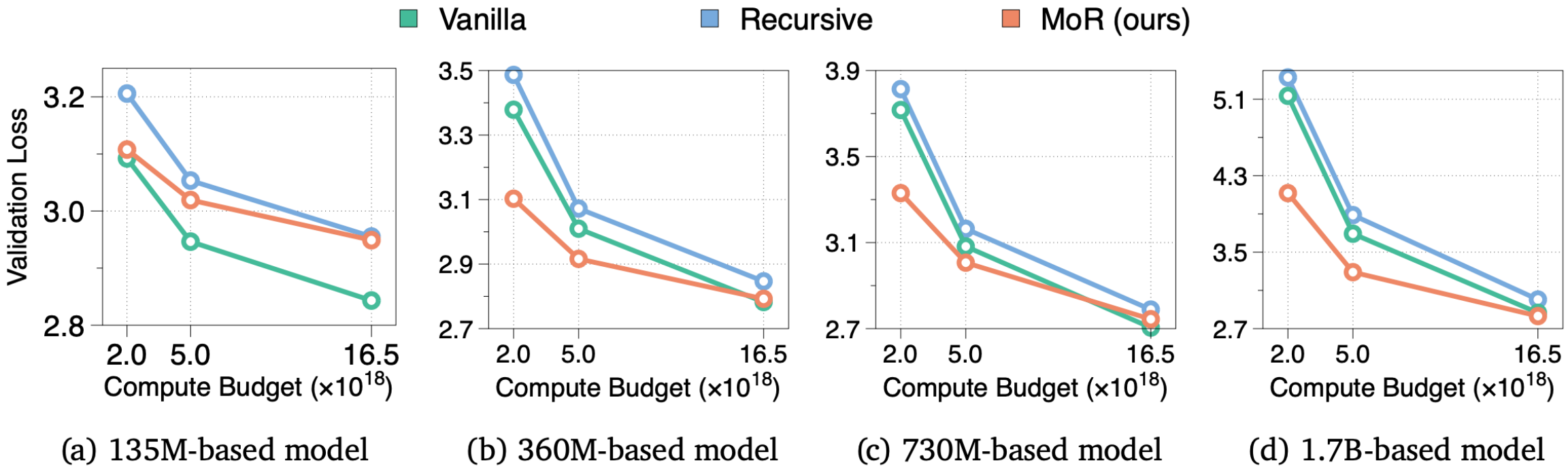


Figure 9: Scaling Analysis: Loss vs. Compute

Scaling—lower loss with the same compute but much smaller models.

Experimental Validation

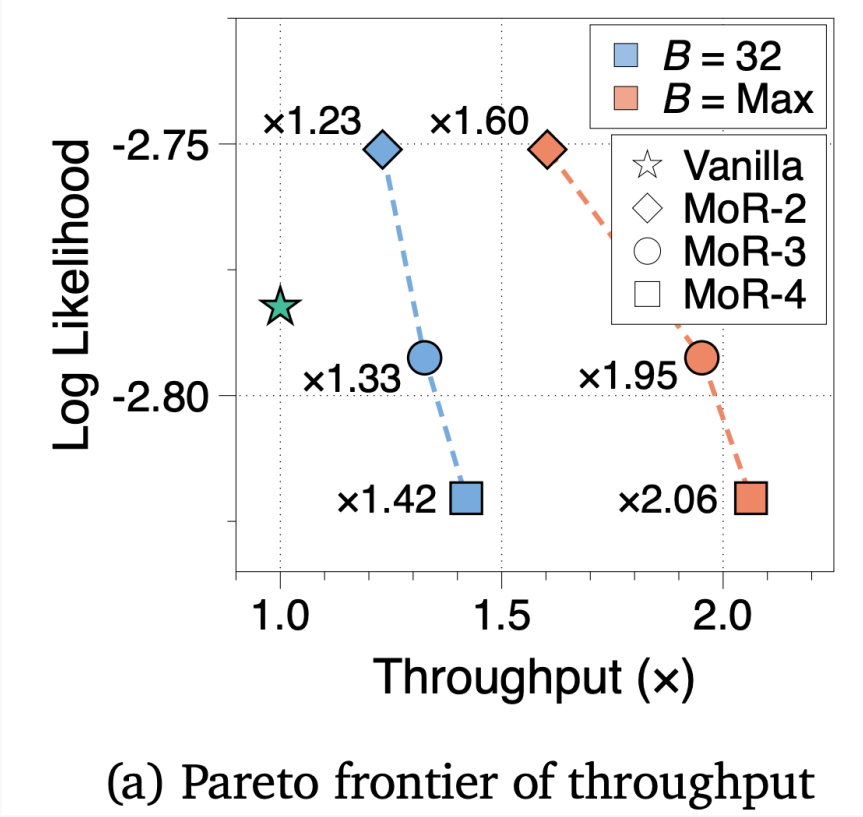
Results: Inference Throughput Gains

Throughput (tokens/sec):

- MoR leverages continuous depth-wise batching.
- Early-exiting tokens are immediately replaced.

Result:

- 1.60×–2.06× speedup vs. vanilla (depending on max recursion).
- Speedup increases with max recursion depth.



(a) Pareto frontier of throughput

Figure 10: Inference Throughput vs. Performance

Experimental Validation

Ablation: Design Choices

Expert-Choice Router

Auxiliary loss + sigmoid + linear works best

Token-Choice Router

Balancing loss + softmax + MLP preferred

Table 4: Ablation Study: Router Design Choices

Expert-choice Router				Performance (↓ / ↓ / ↑)		
Sampling	Func	Arch	z-loss	Dead	NLL	Few-shot
Aux Rout	σ	MLP	✗	0.0	2.8893	39.4
Aux Rout	tanh	MLP	✗	66.7	2.8720	36.2
Aux Loss	σ	MLP	✗	0.0	2.8816	40.0
Aux Loss	tanh	MLP	✗	0.0	2.9933	38.8
Aux Loss	σ	Linear	✗	0.1	**2.8667**	**40.1**
Aux Loss	σ	W-MLP	✗	0.4	2.8716	39.4
Aux Loss	σ	Linear	✓	0.0	2.8824	40.0

Token-choice Router				Performance (↓ / ↓ / ↑)		
Balancing	Func	Arch	z-loss	M-Vio	NLL	Few-shot
Loss (0.1)	soft	MLP	✓	0.200	3.0239	38.5
Loss (0.01)	soft	MLP	✓	0.682	2.9118	39.4
Loss-free	soft	MLP	✓	0.852	2.9081	39.4
Loss-free	σ	MLP	✓	1.281	3.0188	37.6
Loss (0.1)	soft	Linear	✓	0.492	2.9974	38.4
Loss (0.1)	soft	W-MLP	✓	0.384	3.0293	38.8
Loss (0.1)	soft	Linear	✗	0.266	**2.9358**	**39.1**

Experimental Validation

Why Middle-Cycle?

Empirical evidence:

- Consistently lowest validation loss (NLL).
- Better few-shot accuracy vs. pure Cycle/Sequence.

Intuition:

- First layer processes raw embeddings.
- Last layer prepares for vocabulary projection.
- Middle layers perform the heavy “reasoning” loops.

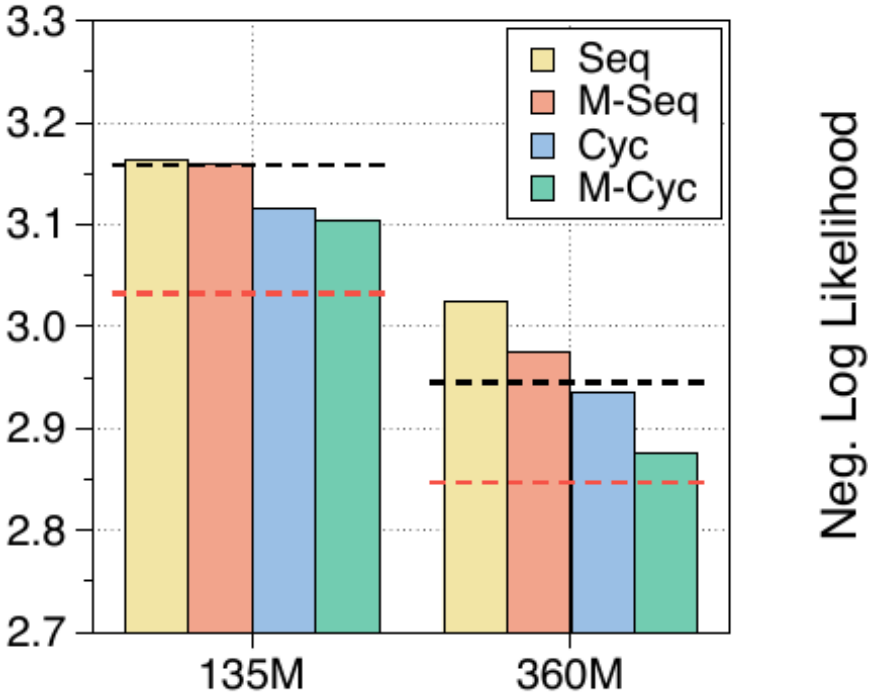


Figure 11 b: Sharing Strategy

Experimental Validation

Ablation: KV Caching

Performance vs. efficiency:

- Recursion-wise caching: best quality (independent contexts).
- KV sharing: slight degradation for expert-choice, but can help token-choice.

Recommendation:

- Use recursion-wise caching for quality.
- Use KV sharing for extreme memory constraints.

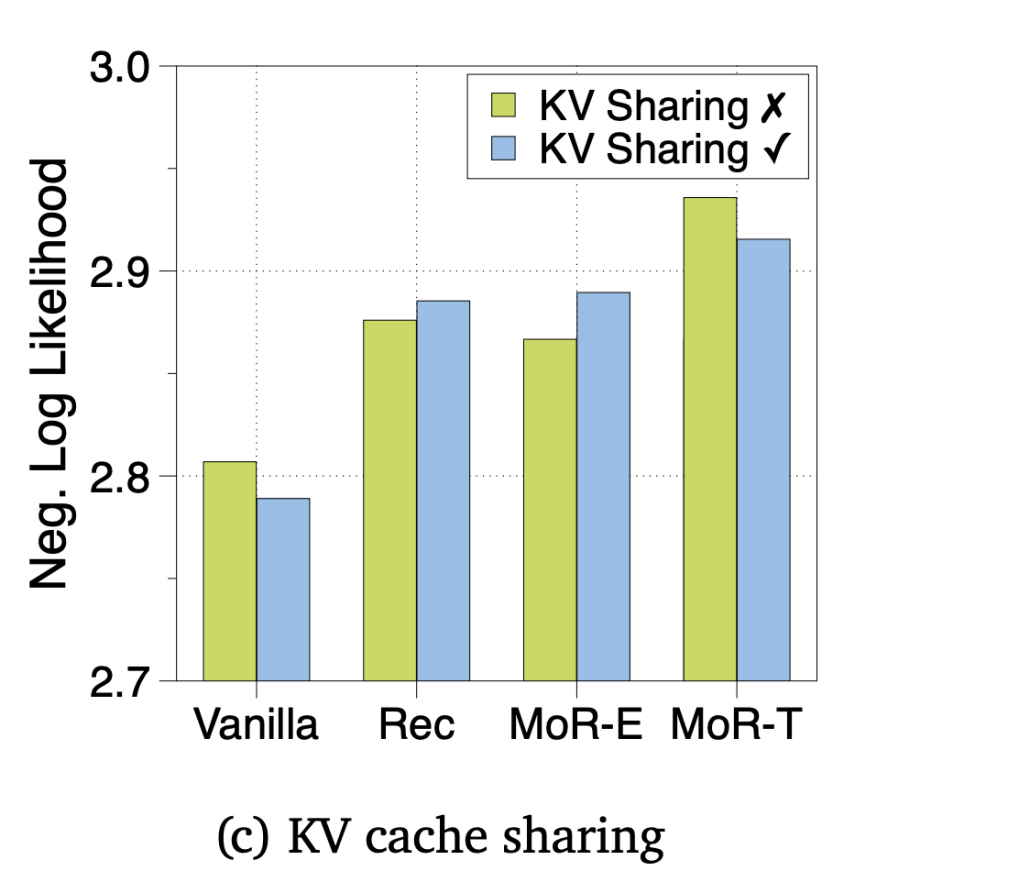


Figure 11 c: KV Caching Efficiency Analysis

Analysis: What does MoR learn?

- Qualitative pattern (Table 17):
- Shallow (1 step): function words, easy suffixes.
- Deep (3 steps): opening complex phrases / content-heavy words.
- Insight: model allocates “thinking time” based on predictability.

steps: 1, 2, and 3

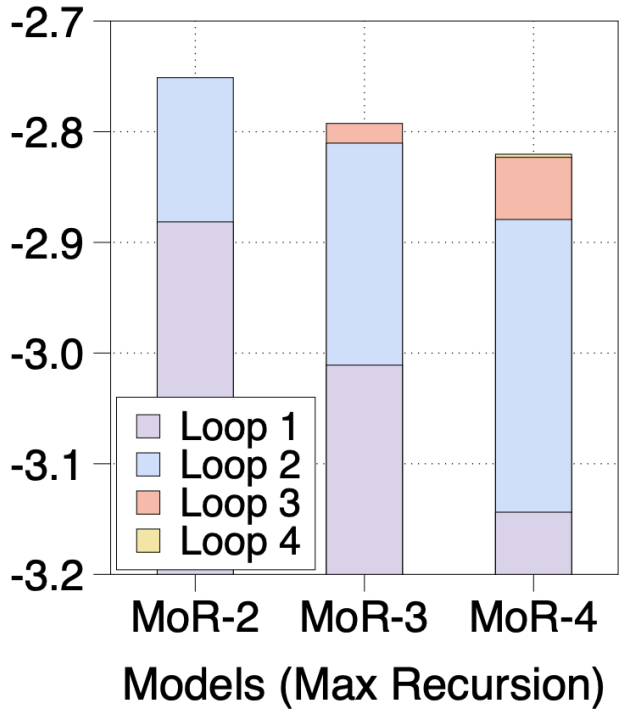
Sample	Text
Sample #1	People who feel comfortable defending their views — def ensively confident — may also eventually change those views and corresponding behaviors . National Election Studies surveys showed that defensive confidence predicted def ection in the 2006 U . S . House elections , above and beyond the impact of various demographic and political variables . Moreover , defensive confidence was also associated with political knowledge and attention to politics and government affairs , but not
Sample #2	2014 , 7 : 57 AM Space X Falcon 9 - R Rocket Suff ers M alf unction , Self - Dest ruct s During Test Flight August 23 , 2014 , 9 : 36 AM Texas Ch osen as Site for SpaceX 's First Commercial Launch pad August 5 , 2014 , 1 : 44 PM South Carolina Prison Find s C ras hed Dr one Car rying Drugs , Ph ones August 1 , 2014 , 2 : 49 PM NASA 's Mars 2020 Rover G ains Seven New Instruments for Exploration August 1 , 2014 , 1 : 30 PM NASA
Sample #3	9 PR New sw ire . All rights reserved A report released Thursday on the Slide Fire in Oak Creek Canyon doesn 't hold back in laying out the danger facing the burned - out area . The U . S . Forest Service issued the report Thursday afternoon stating the biggest concern is the st eeper slopes of Oak Creek Canyon is especially subject to debris flows , rocks l ides , flash flooding and erosion that could become concentrated flow or a landslide . The report said in smaller streams
Sample #4	Bil bo to accompany the dwar ves to fight the enemy . He says , “ S ar uman believes it is only great power that can hold evil in check , but that is not what I have found . I found it is the small everyday deeds of ordinary folk that keep the darkness at bay . Small acts of kindness and love .” That ’ s what Jesus teaches us as well . Warning us that we would live in dark times , He reminded us that because of Him we are “ the light of the world ” (Matt . 5 : 14)
Sample #5	mitting Plants After four years of research , most of it in total darkness , a Stanford University plant biologist has discovered that some plants have a system of fiber optics that can transmit light up and down their tissues in a way similar to the method the telephone company uses to transmit many of its telephone calls . Dr . D ina Mand oli , seeking to find the light - transmit ting properties of plants , had to rely on her sense of touch as she placed tissues of oat , m ung

Figure 12: Qualitative Analysis of Routing Decisions

Experimental Validation

Analysis: What does MoR learn?

- Can we “think longer” at test time?
- Increasing max recursion depth at inference improves log-likelihood.
- Implication: adjustable quality/cost trade-off after training.



(c) Test-time scaling

Figure 13: Test-Time Scaling Effects

Experimental Validation

Comparison Summary

Table 5: Summary of Architectural Comparisons

Feature	Vanilla Transformer	Recursive Transformer	MoR
Weights	Unique (L)	Shared	**Shared**
Compute	Fixed	Fixed	**Adaptive (token-level)**
Memory	High (KV per layer)	High (KV per depth)	**Low (efficient cache / sharing)**
Throughput	Baseline	Similar / modest	**~1.6×–2.1× faster**

05

Conclusion & Discussion

Conclusion

Unified Efficiency Framework

MoR is **first architecture to unify three efficiency paradigms** in single framework: parameter sharing, token-level adaptive thinking depth, and memory-efficient KV caching.

- **Weight tying** → cuts parameters
- **Token routing** → cuts redundant FLOPs
- **Recursion-wise caching** → cuts memory traffic

Conclusion & Discussion

Limitations & Future Work

⚠ Current Limitations

Scale Constraints

Experiments limited to 1.7B parameters. Need validation at larger scales (3B+ parameters).

Adaptive Capacity Control

Challenging to adjust top-k values post-training with auxiliary loss due to perfect separation.

Reasoning Evaluation

Limited exploration on actual reasoning datasets with chain-of-thought.

🚀 Future Research Directions

Larger Scale Training

Train MoR models at 3B+ parameters on larger corpora. Explore continued pre-training from existing checkpoints.

Enhanced Architecture

Depth-specific LoRA, mixture-of-experts principles, expert parallelism for improved quality.

Sparse Integration

Integrate structured sparsity, pruning, quantization for additional efficiency gains.

Multimodal Extension

Extend to vision, speech, unified multimodal architectures. Apply to long-context video/audio.

References

1. **Wei, H., Sun, Y., & Li, Y.** (2025). *DeepSeek-OCR: Contexts Optical Compression*. arXiv preprint **arXiv:2510.18234**.
2. **Raposo, D., et al.** (2024). *Mixture-of-Depths: Dynamically Allocating Compute in Transformer-Based Language Models*. arXiv preprint **arXiv:2404.02258**.
3. **Fedus, W., Zoph, B., & Shazeer, N.** (2022). *Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity*. **Journal of Machine Learning Research**, 23(120), 1–39.
4. **Elhoushi, M., et al.** (2024). *LayerSkip: Enabling Early Exit Inference and Self-Speculative Decoding*. arXiv preprint **arXiv:2404.16710**.
5. **Schuster, T., et al.** (2022). *Confident Adaptive Language Modeling*. **Advances in Neural Information Processing Systems (NeurIPS)**, 35, 17456–17472.

Thank you!