Relaxed Recursive Transformers

Effective Parameter Sharing with Layer-wise LoRA

Paper by Sangmin Bae, Adam Fisch, Hrayr Harutyunyan,
Ziwei Ji, Seungyeon Kim, and Tal Schuster

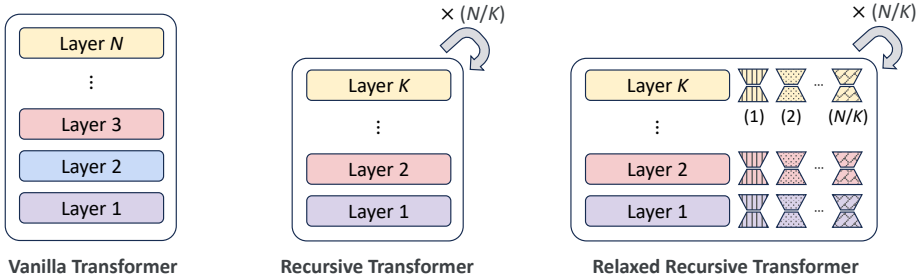
ICLR 2025 | arXiv:2410.20672v3

Paper discussion prepared September 28, 2026

Reuse a pretrained layer block; recover depth-specific differences with low-rank residuals.

Versioned paper; arXiv revision: Feb. 28, 2025. PDF: 49 pages.

Convert a pretrained model into a shared backbone



- Keep the original unfolded depth, while storing fewer unique layers.
- Add a separate low-rank residual at every unfolded depth.
- Shared computation also enables a new batching dimension.

Paper, Fig. 1, PDF p. 2; Secs. 2.2–2.4. Original author figure.

Parameter storage and computational depth are different

What sharing directly changes

$$P_{\text{rec}} \approx P_{\text{emb}} + \frac{P_{\text{blocks}}}{B}$$

Fewer independent weights are stored.

What still needs computation

- All L layers, unless tokens exit early.
- Distinct hidden states across loops.
- Distinct KV states across depths.

Key point

More room for batching is possible, but depends on the memory bottleneck. Parameter sharing alone does not guarantee lower latency.

Architectural interpretation of Eq. (2), p. 3; KV caveat explicitly discussed in App. K, p. 45.

A shared block is applied repeatedly

Let L be total depth, B the loop count, and $K = L/B$.

$$j(\ell) = ((\ell - 1) \bmod K) + 1, \qquad \mathbf{h}_t^{\ell} = f(\mathbf{h}_t^{\ell-1}; \Phi'_{j(\ell)}).$$

Gemma example

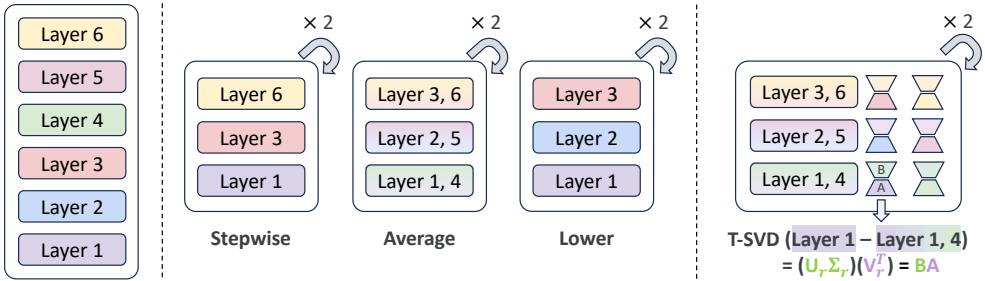
18 unfolded layers $\longrightarrow$ 9 unique layers, executed twice:

$$[\Phi'_1, \ldots, \Phi'_9] \longrightarrow [\Phi'_1, \ldots, \Phi'_9].$$

- This is the **CYCLE** tying strategy.
- Linear weights and normalization parameters are shared.
- Attention context and KV caches are implicit in f.

Paper, Sec. 2.2, Eq. (2), p. 3; App. B, p. 25.

The shared block can inherit pretrained knowledge



- **Stepwise:** select spaced layers, retaining first/last sources.
- **Average:** average weights at corresponding loop positions.
- **Lower:** retain the first K layers.

Paper, Fig. 2, p. 4. Six-layer example; two loops. Original author figure.

Each unfolded depth gets its own low-rank residual

For $W'_{j(\ell)} \in \mathbb{R}^{d\times k}$ and $\mathbf{x} \in \mathbb{R}^k$:

$$\mathbf{z} = W'_{j(\ell)}\mathbf{x} + B_\ell A_\ell \mathbf{x}, \qquad A_\ell \in \mathbb{R}^{r\times k}, \quad B_\ell \in \mathbb{R}^{d\times r}.$$

- The same base matrix serves multiple loop iterations.
- The adapter is selected by **unfolded depth** ℓ.
- Adapters cover attention and FFN linear matrices.
- **All parameters are trained**, including the shared backbone.

Key point

Rank controls the amount of depth-specific capacity. Each adapter costs $r(d + k)$ parameters.

Paper, Sec. 2.3, Eq. (3), pp. 4–5. Matrix B_ℓ is distinct from loop count B.

SVD initializes the residual toward the original layer

$$R_\ell = W_\ell - W'_{j(\ell)} \;\approx\; U_r^\ell \Sigma_r^\ell (V_r^\ell)^\top$$

$$B_\ell \leftarrow U_r^\ell \Sigma_r^\ell, \qquad A_\ell \leftarrow (V_r^\ell)^\top.$$

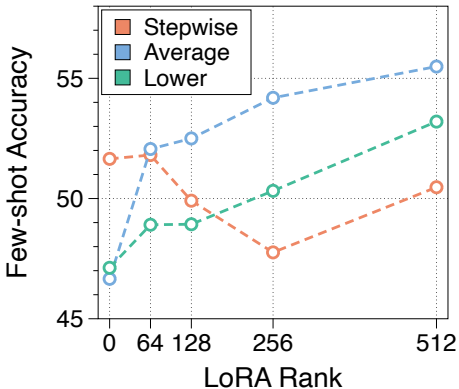
- The initial effective weight approximates the pretrained W_ℓ.
- $r = 0$: strict recursive model; larger r: richer reconstruction.
- If $R_\ell = 0$, use random A_ℓ and zero B_ℓ.

Key point

Matrix reconstruction improves with rank. Network accuracy need not improve monotonically; exact whole-model equivalence needs more conditions.

Paper, Eqs. (4)–(5), p. 5; App. C, p. 26. Last statement is a reading qualification.

The best initialization changes when LoRA is added



- Strict tying: Stepwise is a strong starting point.
- Relaxed tying: Average + SVD performs best here.
- Stepwise has a mid-rank accuracy dip.

Gemma, 15B tokens, $B = 2$:

$$\text{Average, } r = 512: \quad 49.0\% \rightarrow 55.5\%$$

when replacing zero initialization with SVD.

Original Fig. 7a, p. 11; Table G.2, p. 36. The 6.5-point gain is specific to Average initialization.

Three model families test the conversion recipe

Base model	Layers	N-emb	Emb/head	Pretrain
Gemma 2B	18	1.98B	0.52B	3T
TinyLlama 1.1B	22	0.97B	0.13B	105B
Pythia 1B	16	0.81B	0.21B	300B

- SlimPajama uptraining: 15B or 60B tokens; usually $B = 2$.
- Context 2048; batch about 2M tokens; 8 H100 GPUs.
- Seven-task mean accuracy $\uparrow$; three-corpus perplexity $\downarrow$.
- TinyLlama uses an **early checkpoint**, not its 3T checkpoint.

Table 1, p. 6; Apps. D–E, pp. 26–28. Later tables round Gemma N-emb to 1.99B.

The full-size reference changes with uptraining

Gemma checkpoint	Additional tokens	Mean accuracy
Original pretrained	0	**61.7%**
Full-size, uptrained	15B	60.1%
Full-size, uptrained	60B	**58.6%**
Relaxed, $r = 512$ + KD	60B	**58.4%**

Key point

The 0.2-point gap is to the uptrained full-size target. The gap to the original checkpoint is 3.3 points.

- The authors attribute deterioration to the new data distribution/quality.
- A scratch baseline does not inherit the same pretraining history.

Table 2, p. 7; Table I.2, p. 39. These tables qualify the wording in Sec. 3.3, p. 8.

Pretrained initialization drives a large part of the gain

Gemma, 0.99B N-emb, 15B training tokens	Accuracy
Non-recursive, trained from scratch	38.2%
Recursive, random initialization	37.6%
Recursive, Lower	47.1%
Recursive, Average	46.7%
Recursive, Stepwise	**51.7%**

- +13.5 points over the scratch non-recursive baseline.
- +14.1 points over random recursive initialization.
- Recursion by itself is insufficient in this setting.

Paper, Sec. 3.4; Table F.1, p. 30. Accuracy exceptions exist for other model/initialization pairs.

LoRA offers multiple points on the size–accuracy tradeoff

Gemma, 60B uptraining + KD	N-emb	Accuracy
Scratch non-recursive + KD	0.99B	44.9%
Recursive, Stepwise	0.99B	54.9%
Relaxed, Average + SVD, $r = 64$	1.07B	55.8%
Relaxed, $r = 128$	1.15B	56.6%
Relaxed, $r = 256$	1.30B	57.0%
Relaxed, $r = 512$	1.60B	**58.4%**
Full-size, 60B uptraining without KD	1.99B	58.6%

More capacity restores more accuracy, while reducing compression.

Paper, Fig. 4, p. 8; Table I.2, p. 39. These models have not yet received early-exit post-training.

Count embeddings and inherited training when quoting savings

Approximate total parameters

Gemma full-size: $1.99 + 0.52 = 2.51$B.

Strict recursive: $0.99 + 0.52 = 1.51$B.
About **40%** fewer total parameters.

Rank 512: $1.60 + 0.52 = 2.12$B.
About **16%** fewer total parameters.

Incremental training cost

60B tokens are 2% of Gemma's original 3T.

The conversion inherits those pretrained weights.

Online teacher inference and teacher preparation also cost compute.

Derived estimates from Table 1, p. 6 and Table I.2, p. 39; teacher definition in App. I, p. 38.

Longer uptraining and distillation both help

Recursive Gemma, Stepwise	Accuracy	Gain
15B, no KD	51.7%	–
60B, no KD	53.4%	+1.7 points
15B, forward KL	53.4%	+1.7 points
60B, forward KL	**54.9%**	+3.2 points

- Teacher: full-size model uptrained on 15B SlimPajama tokens.
- Forward KL constrains the student toward the teacher distribution.
- Reverse KL, JSD and TVD were also tested; small differences lack error bars.

Paper, Sec. 3.6, pp. 10–11; Tables I.1–I.2, pp. 38–39.

Early-exit training must protect the final prediction

$$\mathcal{L} = \sum_{i=1}^{B} \alpha_i \mathcal{L}_i$$

- Standard weighted CE: $\alpha_i = i / \sum_j j$.
- Preferred two-loop setting: final weight 1, intermediate weight 0.1.
- Distill from the **detached final output** to the intermediate exit.

Gemma: 15B uptrain + 15B post-train	Final	First loop
Weighted CE + self-KD	50.4%	49.9%
Aggressive 0.1 + self-KD	**52.6%**	48.0%

Paper, Table 3, p. 11; Sec. 3.7, p. 12. Accuracy is measured at fixed exits, not with a deployed dynamic gate.

Relaxation can weaken the intermediate exit

Gemma	Before exit training	Final after	First loop
Strict recursive	54.9%	54.0%	**48.8%**
Relaxed, $r = 512$	**58.4%**	56.2%	41.7%

Both receive 60B uptraining with KD, then 15B early-exit post-training.

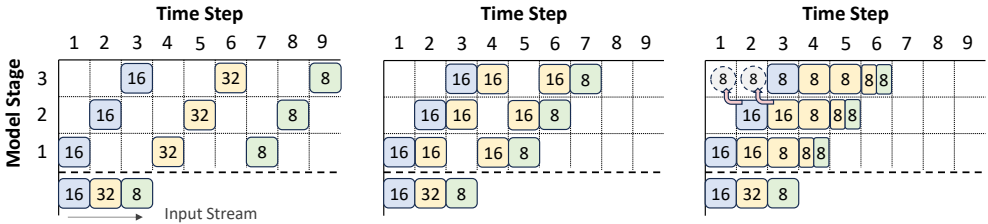
- Depth-specific adapters improve final capacity.
- They may specialize early representations for later processing.
- The authors' explanation is plausible, but not a proven mechanism.

Key point

The 58.4% accuracy result and the throughput model use different training states.

Paper, App. J and Table J.2, pp. 40–42.

Shared blocks let requests at different loops share a batch



(a) Vanilla Batching (b) Depth-wise Batching (c) With Early-exiting

- Horizontal: scheduler time; vertical: model stage.
- A loop-2 sample and a loop-1 sample use the same base block.
- Early exits free slots for queued work immediately.
- Relaxed models still need different adapters per sample.

Paper, Fig. 3, p. 5; Sec. 2.4. Original PDF crop retains all panel labels. Maximum batch = 32.

Throughput is estimated from timings and oracle trajectories

Measured component

- A100 40GB GPU.
- Dummy inputs and weights.
- Per-token component timings across batch sizes.
- Prefix 512; decode 2048.

Simulated component

- Queue of 20K samples.
- SlimPajama, RedPajama, PG19.
- Exit when an intermediate prediction matches the final one.
- Treat corpus tokens as generated.

Key point

This measures potential under ideal exit decisions. It is not an end-to-end serving benchmark.

Paper, Sec. 3.8, pp. 12–13; Apps. E/K, pp. 28, 44–45.

The 2.66× result becomes 1.88× against CSB

Gemma configuration	Acc.	Tokens/s	vs. vanilla	vs. CSB
Full-size, ordinary batch	57.3	1080	1.00×	0.71×
Full-size, CSB	57.3	1528	1.41×	1.00×
Recursive, CDB + oracle	54.0	**2877**	**2.66**×	**1.88**×
Relaxed $r = 64$, CDB + oracle	54.0	2157	2.00×	1.41×
Relaxed $r = 512$, CDB + oracle	56.2	1719	1.59×	1.13×

- Simulated, Transformer-block-only throughput; average of three corpora.
- Accuracy (%) is the final-exit seven-task mean after post-training.
- Model capacity and quality also differ across rows.

Paper, Fig. 8, p. 12 and Table K.2, p. 47. A100 40GB; prefix 512, decode 2048.

Deployment still has unresolved costs

1. **A real exit gate:** confidence estimation costs compute and can make errors.
2. **Embedding and output head:** omitted from final throughput comparisons.
3. **Missing deep KV states:** memory is counted, but refill computation is excluded.
4. **Multi-LoRA execution:** concatenation is measured, but wastes work; optimized kernels remain open.

Key point

Validate end-to-end throughput and output quality together, with realistic queues and a strong continuous-batching baseline.

Paper, App. K, pp. 44–45; Sec. 5, p. 15. Final sentence is a proposed validation criterion.

The evidence supports conversion at modest model scales

- Main experiments: roughly 1B–2B model families; smaller Pythia in the appendix.
- No 7B+ validation or evaluation of latent-reasoning gains.
- Seven few-shot tasks and perplexity give a limited capability profile.
- No reported multi-seed uncertainty supporting tiny accuracy differences.
- Rank, exit quality and adapter execution cost should be tuned together.

Question for discussion

Does a task-aware shared center plus residual decomposition improve the quality–memory–throughput tradeoff beyond separate SVD initialization?

Paper, Secs. 3 and 5; Apps. F–K. Discussion question is an analysis proposal.

Three points to retain

1. **Conversion matters.** Pretrained knowledge makes a compact recursive model competitive with same-size scratch baselines.
2. **Relaxation needs the right initialization.** Average + residual SVD gives a useful capacity control through LoRA rank.
3. **Serving remains an open systems problem.** Shared blocks enable depth-wise batching; the reported acceleration is hypothetical.

Bae et al., *Relaxed Recursive Transformers: Effective Parameter Sharing with Layer-wise LoRA*, ICLR 2025. arXiv:2410.20672v3. Main talk ends here; four backup slides follow.

Backup: separate inherited weights from recursion

Pythia 410M conversion	Non-recursive	Recursive
Random initialization	37.63%	38.02%
Pretrained Stepwise initialization	40.35%	**41.54%**

All four models: 150M non-embedding parameters; 15B Pile tokens.

- Pretrained weights contribute strongly.
- After inheriting weights, recursion adds 1.19 accuracy points.
- The recursive model executes more depth: this is not compute-matched.

Paper, Table F.3, p. 32. Main-model uptraining uses SlimPajama; this ablation uses Pile.

Backup: where to spend relaxation parameters

Gemma, Average + SVD, 15B tokens	N-emb	Accuracy
All components $r = 256$	1.30B	54.2%
KV $r = 128$; others $r = 256$	1.29B	54.1%
FFN $r = 128$; others $r = 256$	1.18B	53.3%

FFN rank reduction also removes many more parameters.

Alternative: layer-specific prefixes

At about 1.07B N-emb with Average initialization:

$$\text{prefix: } 47.6\% \qquad \text{SVD LoRA } r = 64 : \ 52.1\%.$$

This supports the value of recovering pretrained layer differences.

Paper, Tables G.1, G.2 and H.1, pp. 35–37.

Backup: three loops offer stronger compression

Gemma, 60B + KD, before exit training	N-emb	Accuracy
Strict recursive, $B = 2$	0.99B	54.9%
Strict recursive, $B = 3$	0.66B	53.3%
Relaxed, $B = 3$, $r = 256$	0.97B	56.1%
Relaxed, $B = 3$, $r = 512$	1.27B	57.7%

- All cases preserve the 18-layer unfolded depth.
- Strict $B = 3$ improves by 5.5 points over its 15B result.
- Three-loop relaxed exit training omits self-KD due to memory limits.

Paper, Table I.2, p. 39; Table J.2 and caption, p. 42.

Backup: a short-context table needs clarification

Table K.4 reports a 16GB limit with prefix 64 / decode 256.

Gemma rows as printed	SlimP	RedP	PG19
Full-size, ordinary batch	1740	3059	4796
Recursive, $B = 2$, CDB	3159	3050	2421

- The ratio of the printed mean throughputs is about 0.90.
- The same row is labeled 2.50×.
- The recursive row repeats values from Table K.2.

Key point

The discrepancy is present in both PDF and TeX. This talk uses Fig. 8 / Table K.2 for its main throughput numbers.

Reader verification of Table K.4, p. 49 versus Table K.2, p. 47. No corrected short-context multiplier is inferred.