Mixture-of-Depths

Dynamically allocating compute in transformer-based language models

Paper by David Raposo, Sam Ritter, Blake Richards,
Timothy Lillicrap, Peter Conway Humphreys, Adam Santoro

Google DeepMind; McGill University & Mila

Independent paper discussion

arXiv:2404.02258v1 · 2 April 2024
Reading prepared: 8 September 2026

Fix the budget; learn which tokens use it

Research question

Can a decoder spend less compute per forward pass while matching the best dense model at the same total training budget?

- Give each routed block exactly k token slots during training.
- Learn which tokens receive attention and MLP updates.
- Let other tokens pass unchanged to later blocks.

Two useful operating points: a larger model for lower loss, or a cheaper model for comparable quality.

Paper, Secs. 1 and 4.1, PDF pp. 1–2 and 7–10.

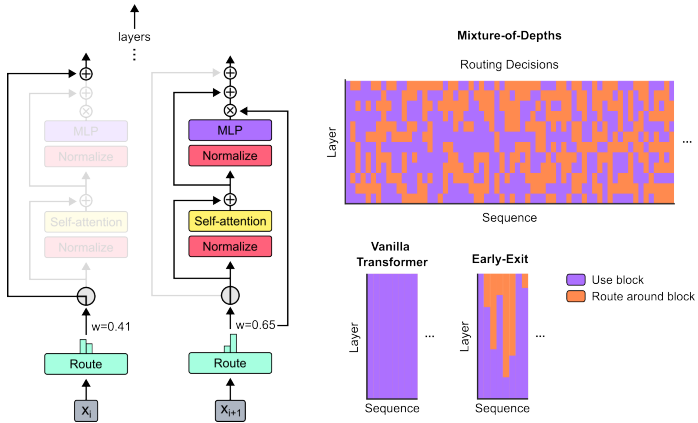
MoD changes effective depth at each token

Architecture	Routing decision	Consequence
Dense	Execute every block	Uniform block participation
Early exit	Stop processing a token	Cannot re-enter later blocks
Typical MoE	Choose MLP experts	More parameter choices per token
MoD	Execute block or bypass	Can skip, then re-enter; attention also changes

The choice affects both which tokens get updated and which tokens supply keys and values in that block.

Paper, Secs. 1–2, PDF pp. 2–3; conceptual comparison, not a benchmark of all methods.

Skipping a middle block does not end computation



Paper, Fig. 1, PDF p. 3. Original author figure.

- Purple: use block.
- Orange: bypass.
- Read vertically to follow one token.

All token states remain in the residual stream.

Capacity controls expensive tensor sizes

For sequence length S and capacity $C = k$, define $\rho = C/S$.

$$\underbrace{S \times d}_{\text{all token states}} \longrightarrow \underbrace{C \times d}_{\text{selected states}} \longrightarrow \underbrace{S \times d}_{\text{scatter updates back}}$$

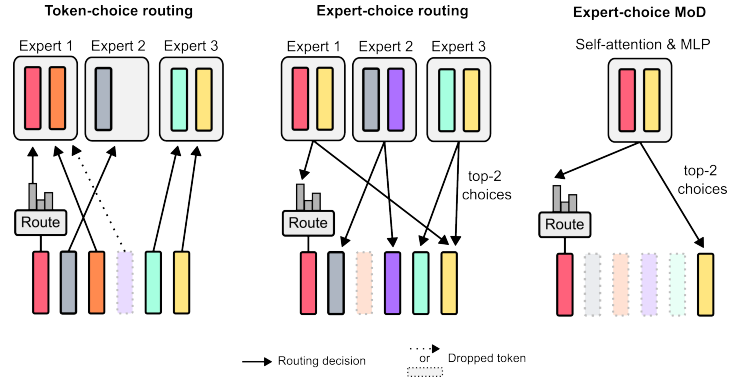
- Token projections and MLP work scale roughly with ρ.
- Token-pair attention work scales roughly with ρ^2.
- Routing, data movement, and other layers still cost compute.

Best reported configuration

Alternate full blocks with routed blocks; $S = 2048$, $C = 256$. Only the routed blocks bypass 87.5% of tokens.

Paper, Secs. 3.1–3.2, PDF pp. 4–5; Fig. 3, p. 8. Scaling explanation is a teaching derivation.

Expert-choice fills a fixed number of slots



One expensive path selects top-k tokens; every other token takes the residual path. No load-balancing loss is needed for this partition.

Paper, Sec. 3.3 and Fig. 2, PDF pp. 5–6. Original author figure.

The routing weight also carries a gradient

$$r_i^l = w_\theta^T x_i^l, \qquad \beta = 1 - C/S$$

$$x_i^{l+1} = \begin{cases} x_i^l + r_i^l f_i(\tilde{X}^l), & r_i^l > P_\beta(R^l), \\ x_i^l, & r_i^l < P_\beta(R^l). \end{cases}$$

- $x_i^l \in \mathbb{R}^d$: token state; r_i^l: scalar router score.
- $\tilde{X}^l$: the C selected states; f: attention and MLP update.
- Multiplying by r_i^l lets the language loss train the router.

The top-k index selection itself is still discrete.

Paper, Eq. (1), Sec. 3.4, PDF pp. 6–7. Equality/tie-breaking is omitted in the paper's equation.

Selected tokens keep their original causal order

Position	1	2	3	4	5	6	7	8
Score	.1	.9	.2	.3	.8	.4	.05	.6

For $C = 2$, positions 2 and 5 enter this block.

- Position 2 cannot attend to position 5; position 5 can attend to 2.
- Other positions keep their input states at this block.
- A later router may select position 3: bypassing is temporary.

Preserve original positions when gathering tokens and constructing masks.

Teaching example, not experimental data. Based on Secs. 3.2–3.4 and 5, PDF pp. 4–7 and 12.

Full-sequence top-k is non-causal

A future token can change an earlier decision

An early score of 0.9 may be in the top-2. If two future scores are 1.0 and 1.1, the early token loses its slot.

- Causal attention masks do not remove this ranking dependency.
- During generation, future router scores are unavailable.
- Training-time top-k therefore needs a causal substitute.

This is a property of the routing operation, even before considering the attention output.

Paper, Sec. 3.5, PDF p. 7. Numeric illustration is a teaching example.

Learn a local decision that imitates top-k

Auxiliary loss on the router

Use top-k membership as a binary target and r_i^l as the logit.

$$p_i = \sigma(r_i^l)$$

At generation: threshold the local probability at 0.5.

Reported effect on the main objective: approximately 0.2–0.3%.

Separate MLP predictor

Predict top-k membership from the same token state, with a stopped input gradient.

$$\hat{p}_i = g_\phi(\mathrm{sg}(x_i^l))$$

At generation: use the local predictor.

Authors report no effect on the main objective and little step-speed overhead.

Local predictions do not enforce exactly C selections per sequence.

Paper, Sec. 3.5, PDF p. 7. Predictor notation and capacity observation are explanatory.

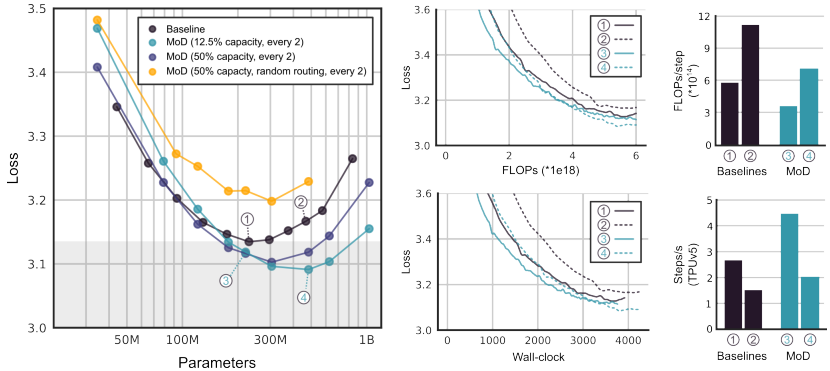
isoFLOP compares equal total training compute

Setting	Reported value
Batch / sequence length	128 / 2048
Schedule	Cosine over the full training run
Total FLOP budgets	6×10^{18}, 2×10^{19}, 10^{20}
Model size sweep	60M–3B parameters
Main control	Best dense model at each total FLOP budget
Causal evaluation data	256,000 held-out sequences; about 500M tokens

Equal total compute does not imply equal parameter counts, steps, or training tokens.

Paper, Sec. 3.6, p. 7; Fig. 4, p. 9; Sec. 4.2, p. 10. Data corpus and full optimizer settings are not reported.

Learned routing and interleaving preserve quality



At 6×10^{18} FLOPs: alternating 12.5%-capacity blocks works best; random routing performs much worse.

Paper, Sec. 4.1 and Fig. 3, PDF pp. 7–8. Random control shown at 50% capacity. Original author figure.

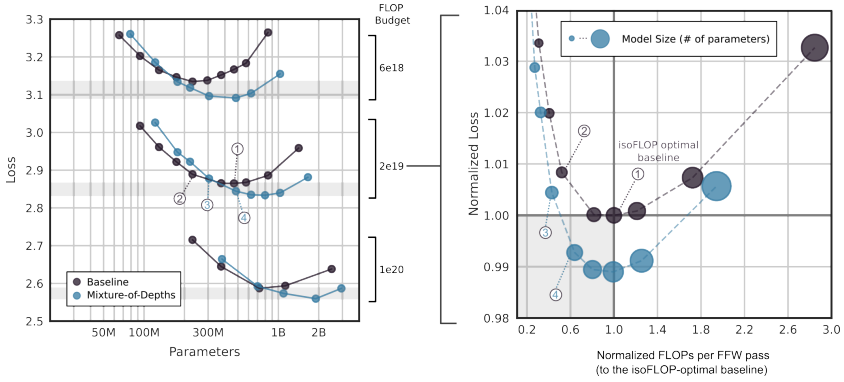
Faster steps can still mean similar total training time

Fig. 3 models #1 and #3	Dense baseline vs. MoD
Parameters	Both approximately 220M
Training budget	Both 6×10^{18} FLOPs
Language loss	Comparable; text describes MoD as slightly better
Training steps per second	MoD: +66% in the figure caption
Total training wall-clock	Approximately the same

A 66% step-rate gain means $t_{\mathrm{MoD}}/t_{\mathrm{dense}} \approx 1/1.66 = 0.602$: about 39.8% less time per step.

Paper, Sec. 4.1 / Fig. 3, PDF pp. 7–8; same-hardware comparison, axis labeled TPUv5. Time ratio is derived.

MoD offers lower-loss, lower-cost operating points



Left: optimal MoD models tend to be larger. Right: points below and left of $(1, 1)$ improve both loss and forward FLOPs.

Paper, Fig. 4, PDF p. 9. Right panel: middle training budget; bubble size denotes parameters. Original author figure.

The lowest-loss model is not the cheapest-step model

Optimize loss

Scale MoD up until forward FLOPs are roughly those of the optimal dense baseline.

The resulting MoD can have more parameters and a lower loss.

Optimize step cost

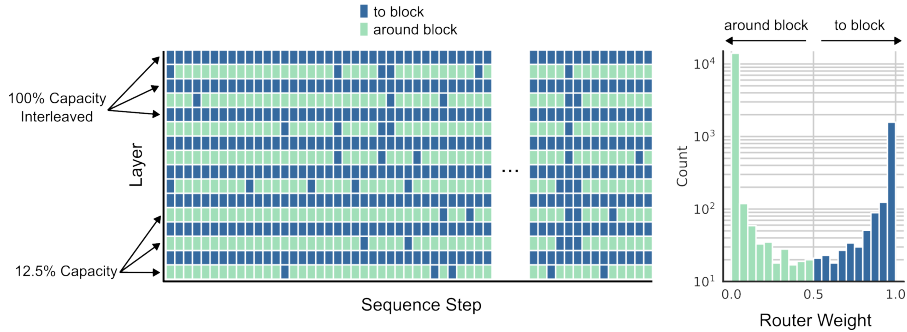
Choose a smaller MoD operating point while matching the best dense model's quality.

Smaller here need not mean fewer parameters than the dense baseline.

- Authors summarize up to 1.5% improvement in the language objective.
- Adding depth rather than width worked better in their experiments.

Paper, Sec. 1, p. 2; Sec. 4.1, pp. 8–9. Empirical design rules, not optimality guarantees.

Routing is structured, but its meaning is not settled

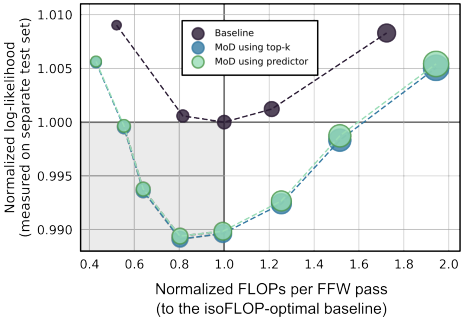


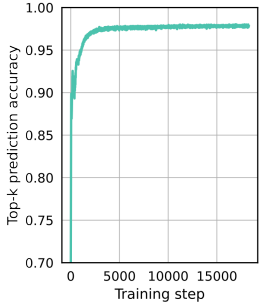
Full blocks form dense horizontal bands; some tokens recur across routed blocks. The authors report a preliminary correlation with prediction entropy.

This does not establish a causal measure of semantic importance.

Paper, Fig. 5 / Sec. 4.1, PDF pp. 9–10. Right-hand Count axis is logarithmic. Original author figure.

A causal predictor largely preserves held-out quality

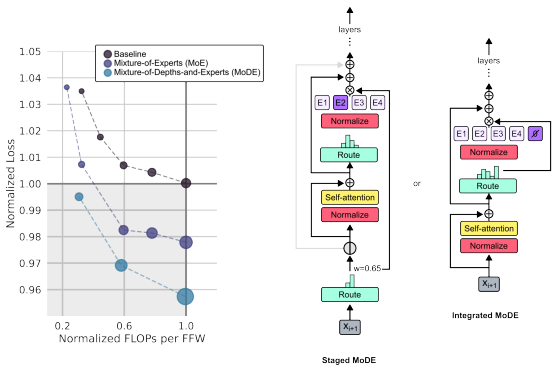




Predictor and top-k curves nearly overlap. Routing accuracy exceeds 97% early; the plotted plateau is about 98% (Sec. 3.5 summarizes 99%).

Paper, Sec. 4.2 / Fig. 6, PDF pp. 10–11. Left axis: normalized log-likelihood; lower is better here. Original author figure.

MoD can also combine with expert routing



Staged MoDE
Select tokens before attention, then route to MLP experts.

Integrated MoDE
Add a no-op option to MLP expert routing; attention still runs first.

MoDE's curve improves on MoE here.

Paper, Sec. 4.3 and Fig. 7, PDF pp. 10–12. The plot does not label separate staged/integrated curves. Original figure.

The evidence supports a compute trade-off, with limits

- **Reproducibility:** corpus, tokenizer, and full optimizer/model configurations are not reported.
- **Evaluation scope:** language modeling; no broad downstream task suite or multi-seed uncertainty analysis.
- **Deployment:** forward FLOPs do not fully specify decoding latency, batching, or KV-cache behavior.
- **Causal routing:** high classification accuracy does not guarantee identical decisions or exact per-sequence capacity.

At 12.5% capacity, always predicting "bypass" already gives 87.5% accuracy. Quality evaluation remains essential.

Reader analysis of reporting and method boundaries; Secs. 3.5–4.3, PDF pp. 7–12. The 87.5% baseline is derived.

A token may be useful as memory, even if easy to predict

Two reasons to spend compute at a position

Update its own representation; provide useful keys and values to later positions.

- This complicates an "easy token vs. hard token" interpretation.
- The authors propose decoupling query, key, and value routing.
- They also suggest memory and tool-use paths as future extensions.

The paper does not implement or validate those memory/tool systems.

Paper, Sec. 5, PDF pp. 12–13. Proposed extensions, not experimental results.

Take-home: allocate depth, then choose an operating point

1. **Mechanism:** a fixed training capacity with learned token participation reduces attention and MLP work.
2. **Evidence:** interleaved full/routed blocks improve the reported isoFLOP loss–compute frontier.
3. **Generation:** local predictors approximate non-causal top-k decisions with little observed quality loss.

Discussion question

Can fully causal routing retain both exact budget control and the quality benefits of full-sequence competition?

Synthesis of Secs. 3–5, PDF pp. 3–13.

Source and reading coverage

David Raposo, Sam Ritter, Blake Richards, Timothy Lillicrap, Peter Conway Humphreys, Adam Santoro.

Mixture-of-Depths: Dynamically allocating compute in transformer-based language models. 2024.

arXiv:2404.02258v1 (retrieved 8 September 2026)

- Read the author TeX and the same-version 14-page PDF.
- Covered all substantive sections, Eq. (1), and Figs. 1–7; no appendix.
- Figures are unchanged copies from the author source package.
- No author code was executed; no training experiments were reproduced.

Companion Chinese notes contain the detailed evidence map, metric caveats, and derivations.

Backup: binary supervision for local routing

Let $z_i = \mathbf{1}\{i \in \mathrm{TopK}(R^l, C)\}$ and $p_i = \sigma(r_i^l)$.

$$\mathcal{L}_{\mathrm{aux}} = -\frac{1}{S}\sum_{i=1}^{S}\left[z_i \log p_i + (1 - z_i)\log(1 - p_i)\right].$$

- This equation formalizes the prose in Sec. 3.5; it is not a numbered paper equation.
- Probability threshold $p_i > 0.5$ corresponds to raw logit $r_i^l > 0$.
- The paper's Fig. 5 uses a 0–1 "Router Weight" axis; do not silently interpret that as the raw linear score.
- The separate predictor uses a stopped input gradient; full loss coefficients and predictor dimensions are not reported.

Paper, Sec. 3.5, PDF p. 7; Fig. 5, p. 10. Equation and notation clarification are explanatory.

Backup: interleaving changes the savings calculation

For half full blocks and half routed blocks, with $\rho = 1/8$:

$$\text{Token-wise work ratio} \approx \tfrac{1}{2}(1+\rho) = 56.25\%,$$

$$\text{Token-pair work ratio} \approx \tfrac{1}{2}(1+\rho^2) = 50.78125\%.$$

- These are component-level estimates at fixed architecture.
- They omit routing, gathering/scattering, output layers, and hardware effects.
- Incremental decoding requires a separate cache and latency analysis.

Teaching derivation from Sec. 3.2, PDF pp. 4–5, and the configuration in Fig. 3, p. 8. Not measured speedups.

Backup: keep the metric denominators separate

Claim / metric	Comparison or denominator	Evidence
Up to 1.5% gain	Relative language objective, not accuracy points	Sec. 1, p. 2
+66% steps/s	Training models #3 vs. #1 on the same hardware	Fig. 3, p. 8
Normalized FLOPs	Forward pass of isoFLOP-optimal dense baseline	Fig. 4, p. 9
Predictor quality	Same 256,000-sequence held-out set	Sec. 4.2, p. 10
97–99% accuracy	Top-k membership prediction; reports vary	Sec. 3.5 / Fig. 6

Abstract also reports sampling step-speed gains above 50%; Fig. 6 does not provide a full per-configuration decoding latency table.