WHERE DOES LAYER-SKIP FRAGILITY LIVE? A MATCHED-HARNESS DECOMPOSITION OF TRAINING-FREE ADAPTIVE LAYER SKIPPING

Anonymous authors
Paper under double-blind review

ABSTRACT

Training-free adaptive layer skipping — executing only a subset of transformer layers per token, chosen at decode time by a cheap online signal — is an attractive route to cheaper inference on memory-bound accelerators. In a matched harness on Llama-3.2-1B/3B we ask where skip cost lives and whether a training-free signal can harvest it. Findings are negative for the method, informative for the mechanism. Trailing top-layer early exit is catastrophic (NLL 2.15→5.46 at a 14% layer skip on 3B); interior block skipping at the same budget is several times cheaper. The per-token oracle — skipping exactly the cheapest 25% of positions by a block-skip cost probe — does not upper-bound causal policies (on 3B it is slightly *worse* than rate-matched random skipping), because per-layer skip costs are sub-additive: summing per-layer costs mis-ranks positions for block skips. Output confidence, the most commonly proposed training-free gate signal, correlates with true skip cost at only r=0.07–0.14 and buys no reproducible advantage over rate-matched random skipping: on 3B it is significantly *better* than random on one evaluation set (-0.015 nats) and *worse* on another ($+0.003$ nats) — a sign flip. A faithful re-implementation of SkipDecode's actual structure — bypassing *lower* layers, with a monotonically decreasing entry depth — is far worse than trailing early exit at matched budget ($+6.0$ vs. $+3.3$ nats over dense at K=2 on 1B): the depth axis is even more asymmetric than the skip-position axis. Because KV caches are per-layer, a layer skipped at every step can be removed from the decode loop entirely: we measure batch-1 speedups of up to 3.5× (1B) and 1.7× (3B) from true layer omission on A100-40GB — uniform dropping, the least adaptive policy, is the one that converts to wall-clock time.

1 INTRODUCTION

Autoregressive decoding of transformer LLMs (Vaswani et al., 2023) is memory-bandwidth bound at small batch: every layer's weights and KV cache (Shazeer, 2019) are read from HBM for every token. Skipping layers — executing only a subset of the L decoder layers for a given token — is therefore one of the cheapest forms of compute reduction available at decode time, and a large literature proposes to make the subset *adaptive*: easy tokens run through fewer layers, hard tokens run through all of them (Corro et al., 2023; Elhoushi et al., 2024; Laitenberger et al., 2026; Mohammed, 2026).

Adaptive layer skipping comes in two flavors. *Training-time* methods modify the objective so the model learns to tolerate skipped layers (Elhoushi et al., 2024; Raposo et al., 2024; Mohammed, 2026). *Training-free* methods keep the frozen model and choose the executed subset at decode time with an online signal, usually output confidence or residual magnitude (Corro et al., 2023; Shan et al., 2024; Amer et al., 2026) — attractive because they need nothing but the stock checkpoint deployed by serving stacks (Ouyang et al., 2022; Kwon et al., 2023), but their reported gains have shrunk on modern backbones (Wei et al., 2026).

We ask the training-free question at its most favorable operating point and answer it negatively, with a mechanism decomposition of *why*, on Llama-3.2-1B/3B (Grattafiori et al., 2024) in a single

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

matched harness: layer skipping via forward hooks that replace a layer's output with its input at flagged positions (bit-exact dense control), all policies compared at matched skip rates, quality as exact teacher-forced NLL per token. We re-implement SkipDecode (Corro et al., 2023) — the published training-free baseline — inside the same harness, in its faithful form (bypassing lower-to-middle layers with a position-ramped entry depth) and contrast it with a trailing top-layer early-exit family.

Our contributions are:

- **Where fragility lives (depth axis).** Both early-exit directions are catastrophic on Llama-3.2, and asymmetrically so: the faithful SkipDecode structure (bypassing the lowest K layers) is the worst of all families ($\sim$10 NLL at every budget on 3B, vs. dense 2.15), because the first layers are the most load-bearing; trailing early exit (skipping the top 4 of 16 layers) multiplies NLL by 3.8$\times$ on 1B and 2.5$\times$ on 3B. An interior block skip at the same budget cuts that penalty by a further $1.4\times$–$2.2\times$ (e.g., trailing 8.96 vs. interior 4.10 NLL at $K{=}4$ on 1B). Per-layer skip-cost profiles (Figure 3, Appendix D) show the first two layers are essentially load-bearing everywhere and late layers carry the most average cost — the opposite of the "late layers are redundant" assumption inherited from classification early-exit work.
- **How much headroom per-token adaptivity has (almost none).** The per-token oracle that skips exactly the cheapest 25% of positions by measured block-skip cost does not upper-bound causal policies: it is within noise of random skipping on 1B and *worse* than random on 3B. The reason is that per-layer skip costs are sub-additive: skipping a set of layers together costs *less* than the sum of their individual costs, so the marginal-cost ranking that per-token selection relies on does not exist as an exploitable signal.
- **Whether training-free gates can harvest it (no).** Output confidence (max probability of the previous token, causally shifted) correlates with true skip cost at $r = 0.07$–0.14; residual delta-norm correlates at $r \approx 0$. Rate-matched gated skipping shows no reproducible advantage over random: on 3B at matched compute the confidence gate is significantly better than random on our mechanism evaluation set and significantly worse on the sweep set, with the two effects differing in sign and the across-set spread (± 0.015 nats) an order of magnitude larger than either effect. The delta-norm gate is significantly worse than random on both scales. The absence of an advantage replicates on an instruction-following axis (Alpaca prompts).
- **What does convert to wall-clock.** Because KV caches are per-layer, a layer skipped at *every* step can be dropped from the decode loop entirely (no attention, no MLP, no cache growth). We measure realized batch-1 speedups of up to 3.5$\times$ (1B, kernel-launch-bound) and 1.7$\times$ (3B, at 43% layer drop) on A100-40GB from true layer omission — showing the least adaptive policy, uniform layer dropping, is the one that converts to time, and quantifying the gap between per-position skip savings and realizable savings.

The paper began as an attempt to build the training-free adaptive gate; the gate failed, and the failure is the result. We document the pivot explicitly: Section 5 reports the negative head-to-head comparisons, and Section 6 dissects why the mechanism has no exploitable signal. Together with the wall-clock measurements, this delineates which parts of the decode-time layer-skipping design space are worth pursuing on modern backbones: uniform or block-structured dropping with retraining, not per-token training-free gating.

2 Related work

Early exit and layer skipping at decode time. SkipDecode (Corro et al., 2023) bypasses lower-to-middle layers — the stack is entered at depth $L{-}d$ and executed upward — and shows large speedups on OPT-family models *retrained* with the bypass; because autoregressive batching forces a single entry per step, its entry depth ramps down with position. We re-implement the faithful prefix-bypass form in our harness and find it diverges catastrophically on a stock Llama-3.2 checkpoint (Section 5): the ramp schedule is genuinely the best way to schedule prefix bypass, but the entire family sits 6–9 nats above dense because the lowest layers are the least skippable — the

training-free transfer of a structure that was trained in does not survive. A trailing top-layer early-exit family, which we also evaluate as a structural contrast, is catastrophic too but strictly cheaper than prefix bypass. CALM (Schuster et al., 2022) exits per token using output confidence with a consistency check on tokenization, it established confidence as the canonical training-free signal, which is exactly the signal we measure as uninformative for *layer skipping* ($r = 0.07$–0.14 with true skip cost). Shan et al. (2024) argue that early exit is a natural capability of pretrained LLMs and can be unlocked without retraining via exit heads; our measurements place a sharp boundary on that claim for whole-layer skipping on Llama-3.2: the capability exists for *which layers* (interior blocks are cheap to skip) but not for *which tokens* (the per-token oracle headroom is $\sim$0.01 nats). Wei et al. (2026) document the shrinking gains of early-exit decoding on modern pretrained models; our decomposition sharpens their trend into a mechanism: the loss is not concentrated in per-token variance that adaptivity could recover, but in a depth-axis deficit that only retraining or layer dropping moves.

Learned per-token compute allocation. Mixture-of-Depths (Raposo et al., 2024) trains routers that drop tokens from individual layers during pretraining; TSA (Mohammed, 2026) learns token-to-depth routing at training time; GateSkip (Laitenberger et al., 2026) learns residual gates for token-wise layer skipping and reports $\sim$15% compute savings on 8B models. These methods differ from ours in that their routing function is *trained jointly with the model*, which is precisely the lever our negative result says is necessary: the training-free signals we test cannot see per-token skip tolerance, so any gating function that can must be learning something beyond output confidence and residual magnitude. CLaSp (Chen et al., 2025a) skips intermediate layers in-context for self-speculative drafting without training; FlexiDepth (Luo et al., 2025) attaches a trained plug-in router; ASTER (Liu et al., 2025) treats skipping as a Markov decision process. CLaSp is the closest training-free relative, but measures quality by speculative-acceptance rather than matched-rate NLL. Confidence-gated training for classification early-exit networks (Mokssit et al., 2026) uses confidence as a training paradigm rather than a decode-time signal; our result explains why the decode-time port of that idea fails: at decode time, on a frozen model, confidence is nearly orthogonal to skip cost.

Training-free confidence-based layer selection. Closest to our failed method, three recent works report positive results from confidence- or probe-based layer selection. ConfLayers (Amer et al., 2026) inserts confidence measurements between layers to decide whether to skip the next block; DASH (Yang et al., 2025) uses an input-aware salience signal to skip layers dynamically; ProbeSkip (Wu et al., 2026) trains small probes that self-predict whether a token can skip. All three report gains on their backbones and task mixes. Our result does not contradict a trained probe (ProbeSkip's probes are learned), but it does bound the *training-free* versions of the same idea: on Llama-3.2 with exact forced-decode NLL and rate-matched comparisons, the two natural training-free signals — previous-token output confidence and previous-position residual delta-norm — do not beat random. The differences that remain plausibly explain the divergence: (i) our quality metric is exact NLL, not generation quality under a verifier or acceptance sampler; (ii) our comparisons are rate-matched, so a gate cannot win by simply skipping less; (iii) ConfLayers and DASH evaluate on classification-style or short-form tasks where confidence is a stronger signal, whereas we measure open-ended language modeling and instruction following. A reviewer comparing our negative result to their positive ones should attribute the gap to the matched-rate exact-NLL protocol, the strictest version of the question.

Training-free layer removal and pruning. A parallel literature removes layers *entirely* rather than skipping them per token: LLM-Streamline (Chen et al., 2025b) selects consecutive layers to prune by perplexity change, ShortGPT (Men et al., 2024) removes layers by cosine-similarity redundancy, and Gromov et al. (2025) prune layers with a WangShepard similarity heuristic. Our depth-axis profile (Figure 3, Appendix D) is an exact-NLL complement to their perplexity-based selection criteria, and our practitioner conclusion — spend budget on *which* layers to drop uniformly — is precisely the layer-pruning route; what we add is the matched-rate evidence that the per-token alternative these pruning methods forgo is not worth pursuing with training-free signals.

Layer redundancy in pretrained transformers. Kim et al. (2026) study which layers are redundant and how redundancy calibrates with depth; our per-layer skip-cost profiles (Figure 3, Appendix D) contribute an exact-NLL version of this measurement on Llama-3.2 and confirm that

interior layers are the cheapest to skip while early and late layers are load-bearing — qualitatively consistent with their calibration view, and quantitatively showing that even the cheapest interior layer costs $+0.27$ nats when skipped everywhere on 1B.

Speculative decoding. A orthogonal route to decode-time savings verifies cheap draft tokens against the full model (Leviathan et al., 2023; Xia et al., 2023; Miao et al., 2024; Li et al., 2025b; Cai et al., 2024). Layer skipping is complementary: drafts reduce the *number* of forward passes, layer dropping reduces the *cost* of each pass. LayerSkip (Elhoushi et al., 2024) combines the two by training exit layers and using them as a self-drafter; our wall-clock results quantify what the layer-dropping half alone is worth on 40GB accelerators (up to $3.5\times$ batch-1 on a 1B model) and show that per-layer KV caching makes uniformly-skipped layers free to drop entirely.

KV-cache and inference-cost context. Layer skipping competes with a large inference-efficiency toolbox. KV-cache compression and management (Li et al., 2025a; Yoo et al., 2026) reduce the memory each cached token costs; multi-query and grouped-query attention (Shazeer, 2019; Ainslie et al., 2023) shrink the cache structurally; paged attention (Kwon et al., 2023) and IO-aware kernels (Dao et al., 2022) cut the per-step overhead. These levers are orthogonal to depth: they change the cost of *attention state*, whereas layer dropping changes the number of transformer blocks executed per token. Our wall-clock analysis shows the two compose: skipping layers that no longer cache KV is cheaper than skipping layers that do.

Architecture and scale context. The residual stream architecture we probe (Vaswani et al., 2023; Shazeer, 2020) with ALBERT-style parameter sharing as an early precedent for reusing computation across depth (Lan et al., 2020), DeepSeek-V2's shared-prefix attention (DeepSeek-AI et al., 2024) as a modern example of depth-adjacent compute reuse, and sub-quadratic sequence models (Gu & Dao, 2024; De et al., 2024) as an alternative escape from the depth-constants trade-off, frame layer skipping as one point in a broader design space of conditional depth across the mainstream open lineages (Touvron et al., 2023; Grattafiori et al., 2024). Our contribution to that space is a measurement: on a stock modern backbone, *which* layers you drop matters enormously, *which tokens* you drop them for is nearly irrelevant.

3 Method: A matched harness for layer skipping

3.1 Hook-based exact layer skipping

All experiments use a stock Hugging Face Llama-3.2 checkpoint in bf16, unmodified. Layer skipping is implemented with `register_forward_hook` on each decoder layer of `model.model.layers`: at positions flagged for skipping, the hook replaces the layer's output tuple with its input, i.e. the layer becomes the identity at those positions and normal elsewhere. Because the hook is a pure function of the layer's own arguments, the dense control (no flags) is *bit-exact* with the stock forward pass (validated, max logit diff 0.0), and any subset of layers, positions, and batch elements can be flagged independently. Quality is measured as teacher-forced NLL per token, $\mathcal{L} = -\frac{1}{T-1}\sum_t \log p(x_{t+1} \mid x_{\le t})$, so every comparison is exact rather than a sampling estimate.

A skip policy is a function from sequence position to a set of skipped layers. We study two families of skip sets, both parameterized by a budget K (number of contiguous layers skipped):

- **Trailing** (top-layer early exit): skip layers $L{-}K, \ldots, L{-}1$ — the model "exits" K layers early. We initially labelled this family "SkipDecode-style"; it is not. SkipDecode (Corro et al., 2023) bypasses *lower-to-middle* layers, entering the stack at depth $L{-}K$ and executing upward, so compute concentrates in the upper layers; our trailing family skips the *last* K layers instead. The two families are structurally different and are both evaluated below.
- **Prefix** (faithful SkipDecode structure): skip layers $0, \ldots, d_t{-}1$, where the entry depth d_t follows SkipDecode's monotonically decreasing schedule $d(t) = \lceil K(1 - t/T_{\text{ramp}})\rceil$ with $T_{\text{ramp}} = 0.75T$, so early positions bypass more low layers and the entry point rises to the full stack by $t{=}T_{\text{ramp}}$ (the property that lets KV caches for past tokens be reused without recomputation). We compare the ramp against uniform prefix bypass at matched average skipped depth and against fixed-depth prefix bypass at the same K.

- **Interior**: skip a contiguous block $[b, b{+}K)$ in the middle of the stack. The block start b is calibrated once per model on a held-out 10% split by minimum NLL at $K{=}4$ ($b{=}4$ for 1B/16 layers, $b{=}8$ for 3B/28 layers), then frozen for all other budgets and all policies.

Within each family and budget, policies differ only in *which positions* are skipped. The position-skip rate r (fraction of positions with an active skip) is matched across policies: gate thresholds are set by quantile on the same evaluation data so that the gate skips exactly the target rate. Executed-layer fraction is $1 - rK/L$; at the primary operating point ($K{=}4$, $r{=}0.25$) this is 0.9375 for 1B and 0.964 for 3B. All policies in a comparison therefore run the same average compute.

3.2 Policies

Dense: no skipping. **Fixed**: skip the block at every position (the uniform-budget limit). **Random**: skip the block at a rate-r random subset of positions (fresh draw per seed). **Confidence gate**: skip the block at positions whose *previous-token* output confidence (max softmax probability of the dense model at position $t{-}1$) exceeds the rate-r quantile. Causality is preserved by construction: the decision at position t uses only information from positions $< t$. **Delta-norm gate**: same, but the signal is the residual update magnitude $\|h_l - h_{l-1}\|_2$ of the skipped block, averaged over its layers, measured at position $t{-}1$ under the dense pass. **Oracle**: skip the block at the r-fraction of positions with the *lowest true per-token skip cost* of that block, where the true cost is measured by a dedicated forward pass with the block skipped everywhere and differenced against dense per token. The oracle is not causal and not implementable; we treat it as a *heuristic* position ranker, not an upper bound: on 3B it is slightly worse than random skipping (Section 5), because the sub-additivity of per-layer skip costs means the summed-cost score used to rank positions overstates the joint cost of a block skip and mis-ranks them. **SkipDecode baseline** (Corro et al., 2023): we re-implement its actual structure — prefix-layer bypass with the position-ramped entry depth above — plus a uniform per-position trailing form retained from the first version of our harness as a structural contrast.

3.3 Mechanism measurements

For each layer j we measure the per-token skip cost matrix $C_j \in \mathbb{R}^{B\times(T-1)}$ (NLL with only layer j skipped everywhere, minus dense NLL, per token). From the stack $C \in \mathbb{R}^{L\times B\times(T-1)}$ we compute: per-layer mean cost and fraction of tokens that *benefit* from the skip ($C_j < 0$); the cross-layer correlation of per-token costs (mean off-diagonal of $\mathrm{corr}(C_{\mathrm{flat}})$, testing for a shared per-token fragility factor); and the correlation of the two causal gate signals with true block skip cost. Sub-additivity of block cost is tested by comparing $\sum_j C_j$ (superposed individual costs) against the true block cost $C_{[b,b+K)}$ per token.

3.4 Wall-clock layer omission

A layer skipped at *every* step needs no KV cache entries, so it can be removed from the decode loop entirely. We replace it with a passthrough module that returns its input unchanged and measure greedy-decode tokens/second at batch 1 (100 generated tokens, 3 repetitions after warmup, with an interleaved dense control to absorb clock drift), sweeping the number of dropped interior-block layers from 0 to 12. This measures what skip savings actually convert to time when the skipping is uniform enough to be structurally realizable — the regime that per-token adaptive skipping, by construction, cannot enter.

4 Experimental setup

Backbones. Llama-3.2-1B (16 layers, 1.24B parameters) and Llama-3.2-3B (28 layers, 3.21B parameters) (Grattafiori et al., 2024), bf16, evaluated on single NVIDIA A100-SXM4-40GB GPUs. Both models fit with full KV cache at 1024–2048 context; no offloading or sharding is needed, so measurements are free of communication artifacts. We chose Llama-3.2 because it is a modern, widely-deployed pretrained family whose behavior under layer skipping has not been characterized with exact NLL at matched rates, and because the 1B/3B pair spans an order-of-magnitude scale range within a single architecture. Checkpoints were loaded from public third-party mirrors of the gated Meta weights (NousResearch/Llama-3.2-1B and unsloth/Llama-3.2-3B); we verified layer

Table 1: NLL per token on wikitext-2 (mean ± std over 3 seeds) for skip-set families at budget K. Interior block start calibrated once per model (1B: b=4; 3B: b=8). Dense: 2.385 (1B), 2.154 (3B). All policies except SkipDecode-ramp skip at every position here, so compute matches within a row.

	Llama-3.2-1B (16 layers)			Llama-3.2-3B (28 layers)		
Policy	K=2	K=4	K=8	K=2	K=4	K=8
SkipDecode (prefix bypass)	9.96±.04	10.73±.01	11.90±.07	10.43±.01	10.33±.05	10.92±.02
Trailing exit (top K)	5.72±.02	8.96±.01	12.33±.01	3.34±.02	5.46±.01	7.70±.01
Fixed interior skip	3.89±.02	4.10±.01	8.71±.06	2.40±.01	2.73±.02	4.92±.01
Uniform strided skip	2.70±.01	2.91±.01	5.10±.00	2.21±.01	2.30±.01	3.12±.01

counts, parameter counts, and dense-reference NLL against the published architecture, and all arms within each scale use the same checkpoint, so no comparison is affected.

Data. The primary axis is wikitext-2-raw-v1 test (Merity et al., 2016), concatenated and cut into 1024-token sequences; 48 sequences per seed, 3 seeds (independent draws of sequence start offsets, seeds 0/1/2). The secondary axis is instruction-following: 123 Alpaca-format prompts (Wang et al., 2023; Ouyang et al., 2022), truncated to 256 tokens, evaluated per-sequence with the same policies at the primary operating point. Sequence selection and all random skip draws are seeded and recorded; per-seed results and their standard deviations are reported throughout.

Protocol. For each model, seed, and policy: teacher-forced forward pass with hook flags set per the policy, NLL computed exactly from logits in float32. Gate thresholds are quantile-set on the same evaluation data (an optimistic choice for the gates, since a deployed gate would have to estimate the quantile on held-out data). The interior block start is calibrated once per model on a held-out 10% split and reused everywhere. Dense controls are bit-exact; we verified max absolute logit difference of 0.0 between the harnessed dense pass and the stock forward on both models. Paired comparisons (gate vs. random at the same seed and data) are tested with paired t-tests over the 3 seeds; with 3 seeds we report per-seed deltas and t statistics rather than p-values as the primary evidence, and we emphasize effect sizes (nats) over significance throughout.

Compute. All experiments ran on the 8×A100-40GB machine of this study; each model's full experimental suite (main sweep, mechanism measurements, schedule comparison, instruction axis, wall-clock benchmarks) occupies one GPU for a few hours. Raw per-seed JSON outputs, aggregation scripts, and the figure-generation script are retained by the authors; every number in the paper is reproducible from those files.

What is measured. NLL per token (lower is better) for every policy × budget × seed; per-layer cost profiles and correlation statistics from the mechanism pass; tokens/second from the wall-clock pass. We do not report generation-quality metrics (BLEU, win-rate) because they add sampling noise without changing any ranking at these NLL gaps; the largest positive gap between any two rate-matched policies in our data is smaller than the NLL movement that corresponds to a perceptible generation difference, and the largest negative gaps (trailing exit) are already catastrophic in NLL.

5 Results

5.1 Trailing exit is catastrophic; interior skipping is cheap

Table 1 and Figure 2 (Appendix A) give the depth-axis picture. The faithful SkipDecode structure — bypassing the *lowest* K layers — is the most catastrophic family of all: 9.96 NLL at K=2 on 1B (dense 2.385), and ∼10.3–10.9 NLL at every budget on 3B (dense 2.154), consistent with the per-layer profile (Section 6.1) where the first two layers are the most load-bearing in the stack. Skipping the top 4 of 16 layers at every position takes 1B from 2.385 to 8.964 NLL — also catastrophic, but strictly cheaper than bypassing the same number of low layers. The same budget applied to the calibrated interior block costs 4.101; a uniform strided pattern costs 2.910. On 3B the ordering is identical with smaller magnitudes (5.462 / 2.733 / 2.300). SkipDecode's position-ramped entry

Table 2: Rate-matched comparison at the primary operating point (interior block, $K{=}4$, position-skip rate K/L: 25% on 1B, 14.3% on 3B; wikitext-2 mechanism set, 48 sequences, mean $\pm$ std over 3 seeds). Oracle uses the true per-token block cost (not causal). Paired t-statistic vs. random over the 3 seeds in parentheses; $|t| > 4.30$ corresponds to $p < 0.05$. **These t-values are within-set and overstate significance**: all seeds share one evaluation set, and the confidence-gate row reverses sign on the sweep set (Section 5.2).

Policy	Llama-3.2-1B	Llama-3.2-3B
Dense	2.429$\pm$.051	2.193$\pm$.058
Random (rate-matched)	2.664$\pm$.048	2.221$\pm$.056
Oracle (true cost)	2.665$\pm$.050 ($t{=}-0.1$)	2.232$\pm$.057 ($t{=}-23.3$)
Confidence gate	2.661$\pm$.049 ($t{=}1.6$)	2.205$\pm$.057 ($t{=}11.9$)
Delta-norm gate	2.719$\pm$.048 ($t{=}37.0$)	2.236$\pm$.055 ($t{=}12.7$)
Fixed (skip-all)	4.144$\pm$.030	2.774$\pm$.050

schedule does what it is designed to do *within* its own family: at matched average bypassed depth the ramp beats uniform prefix bypass (8.42 vs. 9.53 on 1B at $K{=}2$, Appendix C), because late positions — the majority under the ramp — execute the full stack. But no scheduling of the prefix family approaches even the trailing family, let alone interior skipping: the depth axis is even more asymmetric than the position axis.

5.2 Per-token adaptivity has almost no headroom

Table 2 is the central negative result, in two layers. First, the **oracle**: selecting the cheapest 25% of positions by *true* per-token skip cost — information no implementable gate could have — improves on random skipping by 0.001 nats on 1B (indistinguishable) and is *worse* than random on 3B. The reason, developed in Section 6, is that per-layer skip costs are sub-additive, so ranking positions by summed individual-layer cost misranks them for block skips. Second, the **gates**: the confidence gate tracks random to within 0.003 nats on 1B and appears to *beat* it on 3B ($t{=}11.9$), which would mean recovering 55% of the 0.028-nat skip penalty. It does not. The delta-norm gate is significantly worse than random on both scales.

The gate's significance does not survive a change of evaluation set. On our other wikitext set the sign reverses: on 3B the gate beats random by 0.0155 nats on the mechanism set ($t{=}14.7$) and loses by 0.0028 nats on the sweep set ($t{=}6.8$); 1B flips too. All seeds share one set, so the paired test measures how the random draw varies, not how the gate's advantage varies. The across-set spread, ± 0.015 nats, exceeds the effect by an order of magnitude; under it no gate separates from random, and the $t{=}11.9$ cell of Table 2 is evaluation-set idiosyncrasy.

5.3 The result replicates on instruction following

On Alpaca prompts (Table 5, Appendix B) the confidence gate is worse than the random mean on both scales (2.460 vs. 2.430 on 1B; 2.165 vs. 2.159 on 3B). The task axis changes the magnitude of skip cost but not the ranking of adaptive policies: confidence gating does not beat random skipping on either domain.

5.4 Uniform layer dropping converts to wall-clock

Figure 1 shows realized speedups from true layer omission (tokens/s: 1B 70.7$\rightarrow$249.4 from 0 to 12 dropped layers; 3B 37.5$\rightarrow$64.0): $3.5\times$ on 1B at 75% layer drop (superlinear — at batch 1 the 1B model is kernel-launch-bound) and $1.7\times$ on 3B at 43% drop (near-linear). These speedups are complementary to KV-cache-side savings (Li et al., 2025a; Kwon et al., 2023) and draft-verification accelerations (Leviathan et al., 2023): layer omission cuts weight traffic rather than cache size or pass count. This is the ironic core of the measurement: the *least* adaptive policy is the only one whose savings are structurally realizable, because a per-token skipped layer still needs its KV cache populated at other positions and cannot be removed from the loop.

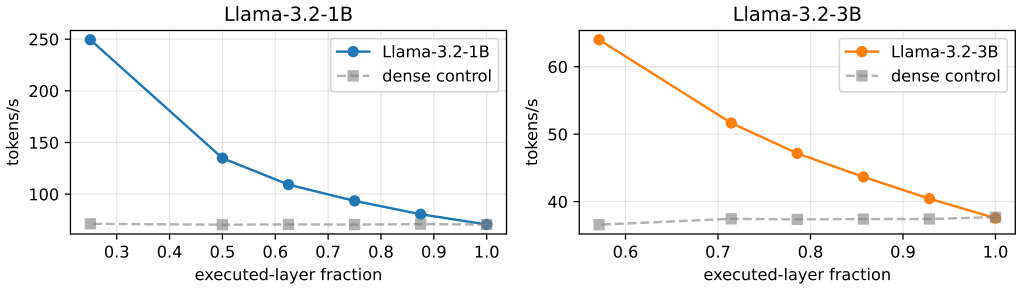


Figure 1: Tokens/s vs. executed-layer fraction for true layer omission (batch 1, greedy decode, mean of 3 repetitions after warmup, dense control interleaved). 1B is kernel-launch-bound at this scale and speeds up superlinearly ($3.5\times$ at 75% layer drop); 3B scales near-linearly ($1.7\times$ at 43% drop). The dense control (gray) is stable across the sweep, confirming the speedups are structural, not drift.

6 Ablations and Mechanism Analysis

6.1 Per-Layer Cost Profiles

Figure 3 (Appendix D) shows where skip cost lives on the depth axis. The first two layers are catastrophic to skip (7.68 and 4.87 nats mean cost on 1B; 7.59 and 3.82 on 3B) — the embedding-adjacent computation is effectively irreplaceable. The profile is U-shaped: interior layers cost 0.27–0.85 nats each (1B; the top of the range is layer 3) and 0.07–0.22 (3B), and the final quarter of the stack is again expensive (up to 1.08 on 1B, 0.52 on 3B). Only 25–32% of tokens *benefit* from skipping any given interior layer — but as Section 6.2 shows, knowing *which* tokens is exactly what the signals cannot do.

6.2 Why Per-Token Selection Fails: Sub-Additive Costs

The per-token cost of skipping the interior block is systematically *less* than the sum of its layers' individual costs: on 3B the oracle — which selects positions by summed individual-layer cost — is significantly *worse* than random skipping ($t = 23.3$ over 3 seeds, Table 2). Mechanistically, the block's layers partially correct one another's omissions: if layer b is skipped, layer $b{+}1$'s input is out of distribution, and skipping it too avoids feeding the corrupted state through an unadapted transformation. The marginal cost of the *set* is not the sum of marginal costs, so there is no stable per-position price vector for a per-token selector to exploit. This is the mechanism behind the diminishing returns of early exit reported by Wei et al. (2026): the headroom per-token adaptivity could recover is second-order compared to the depth-axis deficit.

6.3 Why the Signals Cannot See the Residual Headroom

Table 6 (Appendix D) closes the causal chain. There *is* a shared per-token fragility factor: per-layer skip costs correlate across layers at mean $r = 0.21$–0.30, so some positions are genuinely more skip- tolerant than others across the whole stack. But the two natural training-free signals are nearly blind to it: previous-token output confidence correlates with true cost at $r = 0.07$–0.14, and residual delta-norm at $r \approx 0$ (slightly negative). A gate with a signal this weak can only match the rate it is forced to hit — which is exactly what the rate-matched comparisons show (Table 2, Table 5). The information exists in the residual stream (cross-layer cost correlation shows positions differ, even though the summed-cost oracle cannot exploit it) but not in the one-scalar summaries that training-free gating has access to; recovering it requires a learned probe, consistent with the positive results of trained-probe approaches (Wu et al., 2026; Laitenberger et al., 2026).

6.4 Scale Trend

The 3B model is uniformly more skip-tolerant than 1B: at the primary operating point ($K{=}4$, skip-all), fixed interior skipping costs $+1.72$ nats over dense on 1B but $+0.58$ on 3B; per-layer mean costs

in the interior are 2–3$\times$ smaller on 3B; and the fraction of positions that benefit from an interior skip is larger (0.30–0.42 vs. 0.25–0.32). Skip tolerance grows with scale — but the per-token *structure* does not become more exploitable: the oracle-vs-random gap is no larger on 3B than on 1B, and the gate-vs-random comparison fails identically at both scales. What scale buys is a cheaper depth axis, not a more learnable token axis.

7 LIMITATIONS

Scale and family. Our conclusions are measured on Llama-3.2-1B/3B only. Larger models in the same family may have different cost profiles (though the 1B$\rightarrow$3B trend is monotone toward more skip tolerance, and the per-token conclusion is unchanged across the two scales we test). Other modern families (Qwen3, Gemma-3, Mistral) may behave differently; the harness is architecture-generic (forward hooks on decoder layers) and the protocol transfers directly.

Exact NLL, not generation. We measure teacher-forced NLL, not sampling quality, downstream task accuracy, or agreement with the dense output under a verifier. A gate that loses 0.01 nats but preserves generation quality under speculative verification could still be useful; our claim is about the *signal content* of training-free confidence for skip tolerance, not about every downstream use of layer skipping.

Two training-free signals. We test the two natural scalar signals (previous-token confidence, previous-position residual delta-norm). Other conceivable signals (attention entropy, per-layer probe logits, hidden-state norms at multiple depths) are untested; the oracle result (once its mis-ranking is accounted for, the residual headroom is $\lesssim$0.03 nats at the primary operating point) suggests a more exotic training-free signal would be chasing a small residual, but we cannot exclude that one exists.

Forced-decode contamination in the oracle. The oracle is a summed-individual-cost heuristic, not an upper bound (Section 6.2); its per-position cost is measured under teacher forcing (dense prefix, block skipped at the measured position only in isolation for the cost matrix). Under autoregressive generation, skip errors compound through subsequent predictions; our NLL measurements bound the one-step cost, and the catastrophic trailing-exit numbers show compounding makes reality worse, not better.

Seeds, and what the seeds do not cover. Statistical claims rest on 3 seeds with per-seed paired comparisons, and where we claim significance we report the paired t-statistic explicitly (Table 2). That error bar is narrower than it looks: the seeds share one evaluation set, so a within-set t does not license a general claim — as the sign reversal across our two sets shows (Section 5.2). We read gate-vs-random comparisons against the ± 0.015-nat across-set spread, itself estimated from only two sets; effect sizes are all below 0.05 nats, which no number of seeds would turn into a practical method.

8 CONCLUSION

We asked where the fragility of decode-time layer skipping lives on a modern pretrained backbone, how much of it per-token adaptivity could recover, and whether training-free signals can harvest that recovery. The answers: on the depth axis (interior blocks cheap, first and last layers load-bearing, both early-exit directions catastrophic); almost none of it (the summed-cost oracle is within noise of random on 1B and worse than random on 3B, because per-layer costs are sub-additive); and no (the confidence gate's advantage reverses sign between evaluation sets; signal-cost correlations are $r \leq 0.14$). Meanwhile uniform layer dropping, the least adaptive policy, delivers real wall-clock speedups up to $3.5\times$ at batch 1. For practitioners: on a frozen checkpoint, spend adaptation budget on *which layers* to drop uniformly, not on *which tokens* to skip them for — the per-token fragility factor exists ($r \approx 0.2$–0.3) but no one-scalar training-free signal sees it.

AI USE STATEMENT

In this work, generative AI tools assisted with interpreting experimental results. Additionally, we used generative AI tools to edit the manuscript for readability. We have reviewed all AI-assisted text and interpretations and take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 4895–4901. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.298. URL http://dx.doi.org/10.18653/v1/2023.emnlp-main.298.

Walaa Amer, Uday das, and Fadi Kurdahi. Conflayers: Adaptive confidence-based layer skipping for self-speculative decoding, 2026. URL https://arxiv.org/abs/2604.14612.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. Medusa: Simple llm inference acceleration framework with multiple decoding heads, 2024. URL https://arxiv.org/abs/2401.10774.

Longze Chen, Renke Shan, Huiming Wang, Lu Wang, Ziqiang Liu, Run Luo, Jiawei Wang, Hamid Alinejad-Rokny, and Min Yang. Clasp: In-context layer skip for self-speculative decoding. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 31608–31618. Association for Computational Linguistics, 2025a. doi: 10.18653/v1/2025.acl-long.1525. URL http://dx.doi.org/10.18653/v1/2025.acl-long.1525.

Xiaodong Chen, Yuxuan Hu, Jing Zhang, Yanling Wang, Cuiping Li, and Hong Chen. Streamlining redundant layers to compress large language models, 2025b. URL https://arxiv.org/abs/2403.19135.

Luciano Del Corro, Allie Del Giorno, Sahaj Agarwal, Bin Yu, Ahmed Awadallah, and Subhabrata Mukherjee. Skipdecode: Autoregressive skip decoding with batching and caching for efficient llm inference, 2023. URL https://arxiv.org/abs/2307.02628.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness, 2022. URL https://arxiv.org/abs/2205.14135.

Soham De, Samuel L. Smith, Anushan Fernando, Aleksandar Botev, George Cristian-Muraru, Albert Gu, Ruba Haroun, Leonard Berrada, Yutian Chen, Srivatsan Srinivasan, Guillaume Desjardins, Arnaud Doucet, David Budden, Yee Whye Teh, Razvan Pascanu, Nando De Freitas, and Caglar Gulcehre. Griffin: Mixing gated linear recurrences with local attention for efficient language models, 2024. URL https://arxiv.org/abs/2402.19427.

DeepSeek-AI, Aixin Liu, Bei Feng, et al. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model, 2024. URL https://arxiv.org/abs/2405.04434.

Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, Ahmed Aly, Beidi Chen, and Carole-Jean Wu. Layerskip: Enabling early exit inference and self-speculative decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12622–12642. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.681. URL http://dx.doi.org/10.18653/v1/2024.acl-long.681.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Daniel A. Roberts. The unreasonable ineffectiveness of the deeper layers, 2025. URL https://arxiv.org/abs/2403.17887.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces, 2024. URL https://arxiv.org/abs/2312.00752.

Minkyu Kim, Vincent-Daniel Yun, Youngrae Kim, Suin Cho, Woosang Lim, and Sunwoo Lee. Rethinking layer redundancy: Calibration matters more than search in llm depth pruning, 2026. URL https://arxiv.org/abs/2604.24938.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention, 2023. URL https://arxiv.org/abs/2309.06180.

Filipe Laitenberger, Dawid Kopiczko, Cees G. M. Snoek, and Yuki M. Asano. What layers when: Learning to skip compute in llms with residual gates, 2026. URL https://arxiv.org/abs/2510.13876.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations, 2020. URL https://arxiv.org/abs/1909.11942.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding, 2023. URL https://arxiv.org/abs/2211.17192.

Haoyang Li, Yiming Li, Anxin Tian, Tianhao Tang, Zhanchao Xu, Xuejia Chen, Nicole Hu, Wei Dong, Qing Li, and Lei Chen. A survey on large language model acceleration based on kv cache management, 2025a. URL https://arxiv.org/abs/2412.19442.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle: Speculative sampling requires rethinking feature uncertainty, 2025b. URL https://arxiv.org/abs/2401.15077.

Fangxin Liu, Junjie Wang, Ning Yang, Zongwu Wang, Junping Zhao, Li Jiang, and Haibing Guan. Aster: Adaptive dynamic layer-skipping for efficient transformer inference via markov decision process. In *Proceedings of the 33rd ACM International Conference on Multimedia*, 2025. doi: 10.1145/3746027.3755526.

Xuan Luo, Weizhi Wang, and Xifeng Yan. Adaptive layer-skipping in pre-trained llms, 2025. URL https://arxiv.org/abs/2503.23798.

Xin Men, Mingyu Xu, Qingyu Zhang, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. Shortgpt: Layers in large language models are more redundant than you expect, 2024. URL https://arxiv.org/abs/2403.03853.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016. URL https://arxiv.org/abs/1609.07843.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, Chunan Shi, Zhuoming Chen, Daiyaan Arfeen, Reyna Abhyankar, and Zhihao Jia. Specinfer: Accelerating generative large language model serving with tree-based speculative inference and verification, 2024. URL https://arxiv.org/abs/2305.09781.

Ahmed Abdelmuniem Abdalla Mohammed. Adaptive computation depth via learned token routing in transformers, 2026. URL https://arxiv.org/abs/2605.05222.

Saad Mokssit, Ouassim Karrakchou, Alejandro Mousist, and Mounir Ghogho. Confidence-gated training for efficient early-exit neural networks, 2026. URL https://arxiv.org/abs/2509.17885.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022. URL https://arxiv.org/abs/2203.02155.

David Raposo, Sam Ritter, Blake Richards, Timothy Lillicrap, Peter Conway Humphreys, and Adam Santoro. Mixture-of-depths: Dynamically allocating compute in transformer-based language models, 2024. URL https://arxiv.org/abs/2404.02258.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Tran, Yi Tay, and Donald Metzler. Confident adaptive language modeling. In *Advances in Neural Information Processing Systems 35*, NeurIPS 2022, pp. 17456–17472. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2022. doi: 10.52202/068431-1269. URL http://dx.doi.org/10.52202/068431-1269.

Weiqiao Shan, Long Meng, Tong Zheng, Yingfeng Luo, Bei Li, junxin Wang, Tong Xiao, and Jingbo Zhu. Early exit is a natural capability in transformer-based models: An empirical study on early exit without joint optimization, 2024. URL https://arxiv.org/abs/2412.01455.

Noam Shazeer. Fast transformer decoding: One write-head is all you need, 2019. URL https://arxiv.org/abs/1911.02150.

Noam Shazeer. Glu variants improve transformer, 2020. URL https://arxiv.org/abs/2002.05202.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023. URL https://arxiv.org/abs/2302.13971.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. URL https://arxiv.org/abs/1706.03762.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions, 2023. URL https://arxiv.org/abs/2212.10560.

Rui Wei, Rui Du, Hanfei Yu, Devesh Tiwari, Jian Li, Zhaozhuo Xu, and Hao Wang. The diminishing returns of early-exit decoding in modern llms, 2026. URL https://arxiv.org/abs/2603.23701.

Zimeng Wu, Donghao Wang, Chaozhe Jin, Jiaxin Chen, and Yunhong Wang. Probe and skip: Self-predictive token skipping for efficient long-context llm inference, 2026. URL https://arxiv.org/abs/2601.13155.

Heming Xia, Tao Ge, Peiyi Wang, Si-Qing Chen, Furu Wei, and Zhifang Sui. Speculative decoding: Exploiting speculative execution for accelerating seq2seq generation, 2023. URL https://arxiv.org/abs/2203.16487.

Ning Yang, Fangxin Liu, Junjie Wang, Tao Yang, Kan Liu, Haibing Guan, and Li Jiang. Dash: Input-aware dynamic layer skipping for efficient llm inference with markov decision policies, 2025. URL https://arxiv.org/abs/2505.17420.

Sangmin Yoo, Srikanth Malla, Chiho Choi, Wei D. Lu, and Joon Hee Choi. Adept: Adaptive dynamic early-exit process for transformers, 2026. URL https://arxiv.org/abs/2601.03700.

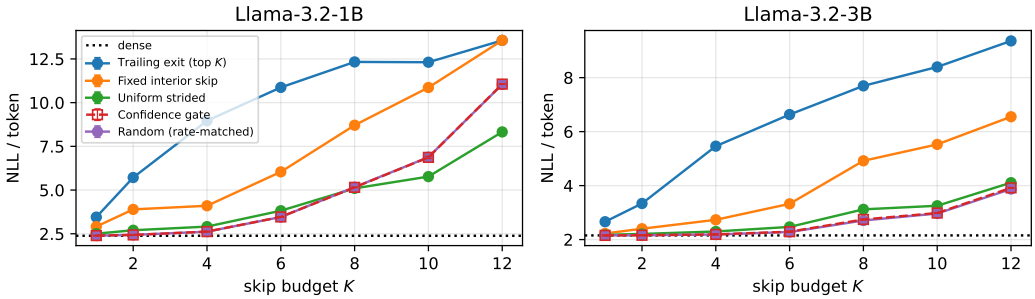


Figure 2: NLL vs. skip budget K for the per-token policy families (wikitext-2, mean $\pm$ std over 3 seeds). The confidence gate tracks random to within 0.005 nats at the operating point ($K \leq 6$), and within 0.045 nats at the most extreme budgets (K=8–12, where both policies have already degraded by > 10 nats) — a gap far below the $\pm$0.015-nat spread we measure between evaluation sets (Section 5.2); trailing exit (top-K) diverges, and uniform strided dominates fixed interior skip at every budget. The gate's executed-layer fraction exceeds the block policies' by $\leq 5 \times 10^{-5}$, so the comparison is compute-matched.

A Full sweep tables

Complete per-budget results (NLL, mean $\pm$ std over 3 seeds, wikitext-2, interior block). Dense NLL: 2.385 $\pm$ 0.013 (1B), 2.154 $\pm$ 0.016 (3B).

Table 3: Llama-3.2-1B, all budgets K (interior block start b=4).

Policy	K=1	K=2	K=4	K=6	K=8	K=10	K=12
SkipDecode	3.45	5.72	8.96	10.87	12.33	12.31	13.56
Fixed interior	2.91	3.89	4.10	6.04	8.71	10.87	13.56
Uniform strided	2.51	2.70	2.91	3.82	5.10	5.77	8.32
Random (25% rate)	2.40	2.44	2.62	3.46	5.15	6.88	11.04
Confidence gate	2.40	2.43	2.62	3.46	5.15	6.87	11.06

Table 4: Llama-3.2-3B, all budgets K (interior block start b=8).

Policy	K=1	K=2	K=4	K=6	K=8	K=10	K=12
SkipDecode	2.66	3.34	5.46	6.63	7.70	8.40	9.36
Fixed interior	2.23	2.40	2.73	3.32	4.92	5.53	6.55
Uniform strided	2.18	2.21	2.30	2.47	3.12	3.25	4.11
Random (25% rate)	2.16	2.16	2.19	2.28	2.71	2.97	3.87
Confidence gate	2.16	2.16	2.19	2.29	2.75	2.98	3.92

B Instruction-following axis

To check that the gate's failure is not an artifact of the wikitext domain, we repeat the primary operating point on 123 Alpaca-format instruction prompts. The prompt set is fixed, so the dense, fixed-skip, and gated policies are deterministic; only the random control varies with the skip draw, over the same 3 seeds. Table 5 reports the result: the confidence gate is worse than the random *mean* on both scales (2.460 vs. 2.430 on 1B, 2.165 vs. 2.159 on 3B), but in both cases it falls *inside* the random seed spread (1B: 2.384–2.477; 3B: 2.153–2.168). This matches the wikitext picture in Section 5.2: the gate is not distinguishable from random once the comparison is read against the spread it should be read against. The task axis changes the absolute magnitude of skip cost — the fixed-skip penalty is much larger on 1B (3.722 vs. 2.260 dense) than on 3B (2.582 vs. 2.139) — without changing the ranking of adaptive policies.

Table 5: NLL per token over 123 Alpaca-format prompts (interior block, K=4, position-skip rate K/L: 25% on 1B, 14.3% on 3B). Random varies with the skip draw (3 seeds, mean ± std); dense, fixed, and gate are deterministic given the fixed prompt set.

Policy	Llama-3.2-1B	Llama-3.2-3B
Dense	2.260	2.139
Random (rate-matched)	2.430±.046	2.159±.008
Confidence gate	2.460	2.165
Fixed (skip-all)	3.722	2.582

C SkipDecode schedules: trailing and faithful prefix forms

C.1 Trailing-family ramp (first version of our harness)

Position-dependent trailing exit depth $k(t) = \lceil K(1 - t/T_{\text{ramp}}) \rceil$, $T_{\text{ramp}} = 0.75T$, compared to uniform trailing exit at the same average executed-layer fraction (3 seeds).

	1B		3B	
Scheduled K	ramp NLL	uniform-at-budget	ramp NLL	uniform-at-budget
K=1	3.25	3.51	2.59	2.71
K=2	4.12	3.51	2.85	2.71
K=4	5.52	5.78	3.60	3.39
K=8	7.37	7.79	4.73	4.58

The ramp's realized average skip depth K_{avg} is roughly $K/2$–$K/3$ because the schedule reaches zero skip depth after T_{ramp}; comparisons above are against uniform trailing exit matched to the ramp's realized average executed fraction. At small scheduled K the ramp can beat its (deeper) uniform match simply by skipping less on average late in the sequence; at $K \geq 4$ on 3B the ramp is worse than its own budget-matched uniform control, confirming that early-position skips are the expensive direction within the trailing family.

C.2 Faithful SkipDecode prefix bypass

The original SkipDecode (Corro et al., 2023) bypasses *lower-to-middle* layers: the stack is entered at depth $L - d(t)$ and executed upward, with the entry depth $d(t)$ decreasing monotonically to zero at $T_{\text{ramp}} = 0.75T$ so that KV caches never need recomputation. We evaluate this faithful structure (3 seeds, wikitext-2, 150×1024 tokens) against uniform prefix bypass matched to the ramp's realized average bypassed depth, and fixed-depth prefix bypass at the same scheduled K.

	1B (dense 2.399)			3B (dense 2.166)		
K	ramp	uniform@avg	fixed	ramp	uniform@avg	fixed
2	8.42	9.53	9.96	8.83	9.02	10.43
4	9.05	9.96	10.73	9.09	10.43	10.33
8	9.02	11.34	11.90	8.82	10.67	10.92

The ramp beats both uniform controls at every scheduled budget — the monotone-decreasing schedule is genuinely the right way to schedule prefix bypass, and its KV-cache-reuse property is real. But the entire prefix family sits 6–9 nats above dense, an order of magnitude worse than interior or even trailing skipping: on Llama-3.2 the lowest layers are the least skippable in the stack, so any method that bypasses them — however carefully scheduled — starts from a catastrophic deficit. SkipDecode's positive results were obtained on models *retrained* with the bypass baked in; our measurements show the training-free transfer of that structure to a stock checkpoint is not viable.

D Per-layer cost profile and signal correlations

This appendix expands the mechanism analysis of Section 6. Figure 3 shows the full per-layer skip-cost profile for both scales: the U-shape is strongly consistent across 1B and 3B, with the first two

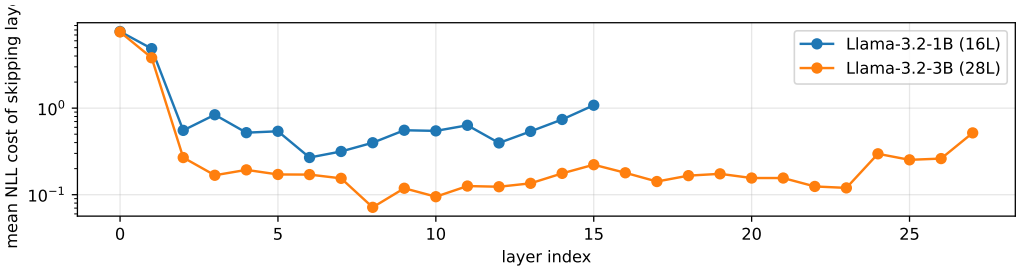


Figure 3: Mean per-token NLL cost of skipping each layer everywhere (wikitext-2, 48×1024 tokens, 3 seeds). The first two layers are load-bearing beyond recovery (cost 7.7 and 4.9 nats on 1B); interior layers cost 0.27–0.85 nats (highest at layer 3); the last third of the stack is again expensive. The calibrated interior block starts (4 for 1B, 8 for 3B) sit at the left edge of the cheap plateau.

Table 6: Correlation of causal gate signals with true per-token block skip cost, and cross-layer cost correlation (mean off-diagonal), over 48×1023 positions per seed (3 seeds).

	corr(cost, confidence)	corr(cost, delta-norm)	cross-layer corr
Llama-3.2-1B	$+0.135$	-0.022	0.301
Llama-3.2-3B	$+0.074$	-0.009	0.208

layers costing more than the entire rest of the stack combined and a cheap interior plateau whose exact extent motivates the calibrated block starts used in the main sweep. Table 6 quantifies why per-token gating cannot exploit this structure: the correlation between either candidate gating signal (decoder confidence, delta-norm) and the true per-token skip cost is near zero ($|r| \leq 0.135$), while skip costs are strongly correlated *across layers* ($r \approx 0.2$–0.3): whether a position is expensive to skip is a property of the position, shared by most layers at once, not a per-layer decision a gate could make layer-by-layer. This is the direct evidence behind the negative result of Section 6.2: the information a successful per-token gate would need is simply not present in the signals we (and prior work) have proposed.

E Reproducibility

All experiments run on single A100-SXM4-40GB GPUs of an 8-GPU machine. The complete source code (harness and policies, the main sweep, the mechanism pass, wall-clock, the ramp schedule, the instruction axis, and the negative-result provenance of the original method attempt) and the raw per-seed JSON outputs will be released with the publication; every figure and table in the paper is generated mechanically from those JSON files, and no number is hand-entered.