A RIGOR-MATCHED AUDIT OF PERIODIC-STEP LAYER SKIPPING FOR EFFICIENT LLM INFERENCE: CONFLAYERS VERSUS SWIFT, WITH A SUPPLEMENTAL ANALYSIS OF TRAINED ROUTING ALTERNATIVES

Anonymous authors
Paper under double-blind review

ABSTRACT

Layer-skipping methods for efficient LLM inference decide, at some granularity, which transformer layers to execute for a given input. We present a rigor-matched, three-seed audit of two *periodic-step, search-based* methods that make this decision online, at inference time, and re-evaluate it every few generation steps: a confidence-gated early-exit baseline (ConfLayers) and genuine self-speculative decoding (SWIFT, Xia et al. 2024), together with vanilla autoregressive decoding, across two model scales (Qwen2.5-0.5B and Qwen2.5-1.5B, Yang et al. 2024) and two tasks (GSM8K reasoning, Cobbe et al. 2021; CNN/DailyMail summarization, Nallapati et al. 2016; See et al. 2017). SWIFT is the strongest method on accuracy in three of four cells; ConfLayers is dominated everywhere, with particularly large deficits on GSM8K at 1.5B. Once online-search overhead is correctly separated from pure inference cost — a decomposition we introduce and validate — SWIFT's true inference speed is faster than ConfLayers's in all four cells (5–21%), reversing the naive wall-clock ranking in three of them; ConfLayers's search overhead is small and stable (1–2% of cost) while SWIFT's is larger and considerably more variable seed-to-seed (up to 28.7%). We additionally examine two *trained-routing* methods, LayerRoute (Sikdar, 2026) (a per-sequence, input-conditioned hard gate) and LayerDrop (Fan et al., 2020) (a fixed, input-independent pruning pattern), as a supplemental analysis rather than a head-to-head comparison, since both operate at a fundamentally coarser decision granularity than the periodic-step methods above. Measuring both under a verified protocol — genuine per-input gating, a genuine full-model baseline, and genuine inference-time compute skipping — both trained-routing methods show modest, real speedups (1.08–1.33×) but accuracy well below the periodic-step methods, including a near-total collapse for LayerRoute on GSM8K at 1.5B (0.003 mean exact-match across three seeds). We release the full audit protocol as a template for rigor-matched efficiency comparisons.

1 INTRODUCTION

Layer-skipping methods for efficient LLM inference share a common shape — decide, for some unit of input, whether to execute a given transformer layer or bypass it — but differ sharply in *when* and *at what granularity* that decision is made. Confidence-gated early-exit methods threshold a per-token confidence signal that is itself tuned online (Schuster et al., 2022; 2021; Teerapittayanon et al., 2016); self-speculative decoding methods draft candidate continuations with a cheapened version of the same model, periodically re-evaluating the draft configuration, and verify drafts against the full model in a single batched pass (Leviathan et al., 2023; Chen et al., 2023; Xia et al., 2024). Both families make their skip decision *online*, re-checking it every few generation steps. A separate family, *trained routing*, instead learns a fixed policy offline — a gate, a router, or a pruning mask — so no online search is needed (Fan et al., 2020; Raposo et al., 2024; Elbayad et al., 2020); the policy can be as coarse as a single fixed pattern applied to every input, or as fine as a decision recomputed per sequence.

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.

This granularity difference determines what a fair comparison even means: a method re-evaluating its configuration every 25–30 tokens is answering a different question than one committing to a single decision for an entire sequence, or one that never adapts to input at all. Our primary contribution is therefore a comparison restricted to methods sharing the same decision regime — ConfLayers and SWIFT, both periodic-step and search-based — audited under matched seeds, tasks, and scales, with a cost-accounting methodology that correctly separates the online search from the inference it accelerates. This confound is methodologically significant: once search overhead is excluded, the pure-inference speed ranking of the two methods reverses relative to their naive wall-clock ranking in three of four task/scale combinations we test.

We additionally examine two representative trained-routing methods, LayerRoute (Sikdar, 2026) and LayerDrop (Fan et al., 2020), as a *supplemental analysis* rather than folding them into the main comparison, since their decision granularity does not match ConfLayers's and SWIFT's: LayerRoute commits to one gate decision per sequence; LayerDrop commits to one fixed pattern that never varies with input. Treating either as a direct peer of a method that re-searches every 25 tokens would overstate what the comparison can claim. We measure both under a protocol verified to exercise their routing decisions genuinely — real per-input gating, a real full-model baseline, and real inference-time compute skipping — and report the resulting numbers plainly, including a result we did not expect going in.

Our contributions are:

- A rigor-matched, 3-seed, 2-task $\times$ 2-scale comparison of two periodic-step, search-based layer-skipping methods (ConfLayers, SWIFT) against vanilla decoding, with a taxonomy of routing decision granularity motivating why this is the correct main comparison set (Section 2).
- A search-overhead decomposition that separates each method's pure inference cost from its online-search cost, revealing that naive wall-clock comparisons had inverted the true inference-speed ranking of SWIFT and ConfLayers in three of four cells (Section 6.2.1).
- A supplemental analysis of two trained-routing methods (LayerRoute, LayerDrop) at a different decision granularity, using a measurement protocol verified to exercise genuine per-input gating, a genuine full-model baseline, and genuine inference-time compute skipping throughout (Section 7).
- A reusable, rigor-matched audit protocol, including the frozen-replay search-overhead decomposition, applicable to future efficiency comparisons.

2 A Taxonomy of Layer-Skipping Decision Granularity

Table 1 summarizes the four methods discussed in this paper along the dimension that we argue determines comparability: how often, and on what unit of input, the skip decision is made.

Table 1: Layer-skipping methods by decision granularity. ConfLayers and SWIFT share the same regime (periodic-step, search-based) and form our main comparison; LayerRoute and LayerDrop occupy two different, coarser regimes and are reported as a supplemental analysis rather than head-to-head peers.

Method	Training	Decision unit	Re-evaluation cadence
ConfLayers	None (online search)	Per token	Every 30 steps
SWIFT	None (online search)	Per token	Every 25 steps
LayerRoute	Trained (LoRA + gate)	Per sequence	Once, input-conditioned
LayerDrop	Trained (structured dropout)	N/A	Never (fixed, input-independent)

ConfLayers and SWIFT both decide online, re-searching their configuration every 25–30 generation steps and often converging early, freezing the found configuration for the remainder of a session (Section 6.2.1); this shared regime is what makes a matched comparison between them meaningful. LayerRoute is designed to make one gate decision per input sequence, using that sequence's own hidden states — a coarser, input-conditioned regime that, absent an explicit freezing mechanism, can drift toward continuously-updating behavior in genuine generation (Section 7.2). LayerDrop is coarser still: its pruning pattern is fixed at training time and applied identically to every input, with

no adaptivity at all. Comparing ConfLayers/SWIFT against LayerRoute/LayerDrop on identical footing would compare methods answering different questions; we instead present the latter two as a supplemental analysis with this distinction stated explicitly.

3 Related Work

Trained routing and structured skipping. LayerDrop (Fan et al., 2020) trains transformers to be robust to layer removal via structured dropout, enabling depth reduction at inference without retraining; we evaluate it directly in Section 7. The Depth-Adaptive Transformer (Elbayad et al., 2020) and PonderNet (Banino et al., 2021) extend Adaptive Computation Time (Graves, 2016) to learn per-example halting policies. Mixture-of-Depths (Raposo et al., 2024) routes a fixed fraction of tokens per layer through a top-k router, a finer (token-level) granularity than LayerRoute's per-sequence gate; we do not evaluate it here. Post-hoc depth-pruning work such as ShortGPT (Men et al., 2024) and Gromov et al. (2024) establish that later transformer layers often contribute disproportionately little to output quality, independent of any routing mechanism.

Confidence-gated and search-based early exit. CALM (Schuster et al., 2022) and its earlier variant (Schuster et al., 2021) exit a token's forward pass once a per-layer confidence measure crosses a threshold; BranchyNet (Teerapittayanon et al., 2016) applies the same idea to convolutional networks. Our ConfLayers baseline follows this family, augmented with an online Bayesian-optimization search over per-layer thresholds (Section 5).

Speculative and self-speculative decoding. Speculative decoding (Leviathan et al., 2023; Chen et al., 2023; Stern et al., 2018) drafts multiple tokens with a cheap model and verifies them in one batched pass, with later work exploring tree-structured verification (Miao et al., 2024; Cai et al., 2024) and lightweight adapters as drafters (Du et al., 2024). SWIFT (Xia et al., 2024) removes the need for a separate draft model by self-drafting with a layer-skipped version of the target model, discovered online rather than trained offline.

Compute-efficient scaling and pruning. Our discussion of scale-dependent behavior connects to compute-optimal scaling (Hoffmann et al., 2022) and to structural pruning such as LLM-Pruner (Ma et al., 2023) and Wanda (Sun et al., 2024), which similarly find removable redundancy is scale-dependent.

4 Background: Method Formalizations

We formalize ConfLayers and SWIFT here; LayerRoute and LayerDrop's formalizations appear in the supplemental analysis (Section 7).

4.1 The ConfLayers Baseline

ConfLayers (Amer et al., 2026) requires no training: it wraps a fixed pretrained backbone with a per-layer confidence rule and an online search over its thresholds.

Confidence signal and exit rule. At layer l, a confidence estimate projects the current hidden state to the vocabulary and takes the top probability,

$$c_l(h_l) = \max_v \ \mathrm{softmax}(W_{lm}\, h_l)_v \,, \tag{1}$$

where W_{lm} is the shared, frozen LM head. The model exits at the first layer clearing threshold τ_l,

$$z_l = \mathbb{1}[c_l(h_l) \geq \tau_l] \,, \qquad \text{exit at } l^* = \min\{\, l : z_l = 1 \,\}, \tag{2}$$

emitting $W_{lm} h_{l^*}$ directly as output logits, skipping all $l > l^*$ (Figure 5).

Online threshold search. $\tau = (\tau_1, \ldots, \tau_L)$ is not learned by gradient descent; since the mapping from τ to quality and latency is a black-box, non-differentiable function, τ is tuned online via Bayesian optimization,

$$\tau^\star = \underset{\tau \in [0,1]^L}{\arg\min} \ \mathbb{E}_x[\mathrm{Cost}(x;\tau)] \quad \text{s.t.} \quad \mathbb{E}_x[\mathrm{QualityDrop}(x;\tau)] \leq \epsilon, \tag{3}$$

proposing candidate τ vectors, evaluating on a held-out probe set, and updating a surrogate cost–quality model. This search is what we isolate as "search overhead" in Section 6.2.1: wall-clock time spent choosing τ, separate from the forward-pass cost of Equations 1–2 once fixed.

4.2 The SWIFT Baseline

SWIFT (Xia et al., 2024) also requires no training, but rather than exiting early it drafts multiple tokens with a cheapened version of the target model and verifies them losslessly against the full model.

Self-draft model. Given a skip set $S \subseteq \{1, \ldots, L\}$, the self-draft model $\mathcal{M}_S$ is $\mathcal{M}$ with every layer $l \in S$ replaced by an identity pass-through,

$$h_l^{(S)} = \begin{cases} h_{l-1}^{(S)} & l \in S \quad \text{(skipped)} \\ f_l\left(h_{l-1}^{(S)}\right) & l \notin S \quad \text{(executed)} \end{cases}, \tag{4}$$

requiring no additional parameters, since $\mathcal{M}_S$ reuses $\mathcal{M}$'s own weights (Figure 6).

Draft-then-verify. $\mathcal{M}_S$ autoregressively drafts k candidate tokens, and $\mathcal{M}$ verifies all k positions in one batched pass. Each draft token is accepted with probability

$$\min\left(1,\ \frac{P_{\mathcal{M}}(\tilde{x}_{t+i} \mid x_{\leq t+i-1})}{P_{\mathcal{M}_S}(\tilde{x}_{t+i} \mid x_{\leq t+i-1})}\right), \tag{5}$$

following the standard rejection rule (Leviathan et al., 2023); at the first rejection, a corrective token is resampled from $p_{\text{res}}(x) \propto \max(0,\ P_{\mathcal{M}}(x) - P_{\mathcal{M}_S}(x))$, guaranteeing the output distribution is exactly $P_{\mathcal{M}}$ regardless of S — SWIFT is lossless by construction.

Online skip-set search. As with ConfLayers's τ, S is searched online, trading a cheaper draft (larger S) against a lower acceptance rate,

$$S^\star = \underset{S \subseteq \{1,\ldots,L\}}{\arg\min}\ \mathbb{E}_x\left[\frac{\text{WallClockCost}(x; S)}{\mathbb{E}[\text{accepted tokens per round} \mid S]}\right], \tag{6}$$

again via Bayesian optimization over the discrete space of skip sets, separated from pure drafting cost in Section 6.2.1.

5 Experimental Setup

Models. We use Qwen2.5-0.5B and Qwen2.5-1.5B (Yang et al., 2024) as backbones.

Tasks. GSM8K (Cobbe et al., 2021) (grade-school math word problems, exact-match accuracy) represents multi-step reasoning; CNN/DailyMail (Nallapati et al., 2016; See et al., 2017) (news summarization, ROUGE-L, Lin 2004) represents long-form generation with a different token budget and error profile. We evaluate 100 held-out samples per task, with an identical shuffle/seed recipe across dataset aliases, matching the default evaluation size of the ConfLayers and SWIFT reference implementations' own scripts. We report per-seed values (Appendix B) rather than a single point estimate, since 100 samples is small enough that seed-to-seed variance should be inspected directly (Section 8).

Baselines. *Vanilla* is standard autoregressive decoding with no acceleration. *ConfLayers* and *SWIFT* are formalized in Sections 4.1 and 4.2; SWIFT results use a from-scratch port of the official self-drafting and tree-attention implementation to Qwen2, verified class-by-class against the reference implementation.

Rigor protocol. Every accuracy and cost cell (method $\times$ task $\times$ scale) is run at three seeds (2024, 42, 123), reported as a mean with the observed range. We verified that a CNN/DailyMail generation-length normalization does not itself explain observed score differences between methods.

5.1 Implementation details

ConfLayers search calibration. We run ConfLayers's online threshold search (Equation 3) with its reference implementation's default configuration: search interval 30, maximum optimization it-

Table 2: Accuracy, mean over 3 seeds (range in brackets). GSM8K is exact-match accuracy; CNN/DailyMail is ROUGE-L. Bold marks the best method per cell.

Task	Scale	Vanilla	ConfLayers	SWIFT
GSM8K	0.5B	0.180 [0.14–0.26]	0.147 [0.13–0.18]	**0.303** [0.27–0.36]
	1.5B	**0.413** [0.38–0.46]	0.077 [0.05–0.10]	0.307 [0.29–0.33]
CNN/DM	0.5B	0.169 [0.167–0.174]	0.178 [0.175–0.184]	**0.190** [0.185–0.196]
	1.5B	0.204 [0.193–0.213]	0.215 [0.204–0.222]	**0.219** [0.205–0.230]

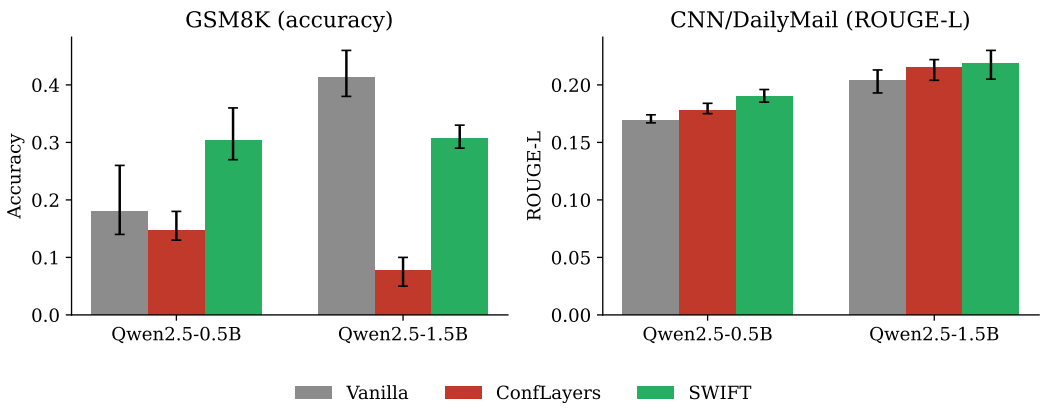


Figure 1: Accuracy across the full 2 (task) × 2 (scale) × 3 (method) matrix. Error bars show the observed 3-seed range. ConfLayers is the consistent worst performer in every cell.

erations 100, maximum score threshold 0.95, context window 100 tokens. Thresholds are calibrated online, per query, not per task.

SWIFT search calibration. SWIFT's online skip-set search (Equation 6) uses its reference default: context window 50, optimization interval 1, Bayesian-optimization interval 25, maximum optimization iterations 1000, maximum tolerance iterations 300, maximum score threshold 0.93, with an upper-confidence-bound acquisition function (κ=2.5).

Cost measurement protocol. All latency measurements use batch size 1 on a single GPU, with explicit device synchronization before and after each timed call so reported wall-clock time reflects completed GPU work rather than queued kernels. Reported time includes the full generation call — sampling, KV-cache updates, any interleaved search — excluding tokenization/detokenization. We did not run warm-up iterations; the first query of each run carries unsubtracted one-time initialization overhead (Section 8).

6 Results

6.1 Accuracy

Figure 1 and Table 2 summarize the accuracy matrix for vanilla decoding and the two periodic-step methods. The ranking is stable across all three seeds in every cell.

ConfLayers underperforms both other methods in every condition, with particularly large deficits on GSM8K at 1.5B (0.077 vs. vanilla's 0.413), consistent with prior observations that fixed-representation confidence estimates are unreliable for multi-step reasoning (Schuster et al., 2022). SWIFT is strongest in three of four cells, losing only to vanilla decoding on GSM8K at 1.5B (0.307 vs. 0.413) — but still substantially outperforming ConfLayers there (0.307 vs. 0.077), so the ranking between the two search-based methods is stable across all four cells.

6.2 Cost

Table 3 and Figure 2 report per-query wall-clock latency for the two periodic-step methods.

Table 3: Per-query wall-clock latency (ms), mean over 3 seeds. Both methods shown decomposed into pure inference and online-search overhead (Section 6.2.1).

Task/Scale	Method	Pure inference (ms)	Search overhead	Wall-clock total (ms)
GSM8K/0.5B	ConfLayers	9194	2.2%	10338.5
	SWIFT	8771	15.1%	11564.5
GSM8K/1.5B	ConfLayers	12366	1.3%	12485.4
	SWIFT	9913	12.4%	11628.8
CNN-DM/0.5B	ConfLayers	10889	1.2%	10231.4
	SWIFT	8610	8.7%	10598.2
CNN-DM/1.5B	ConfLayers	12555	1.2%	12973.4
	SWIFT	9880	18.6%	12532.5

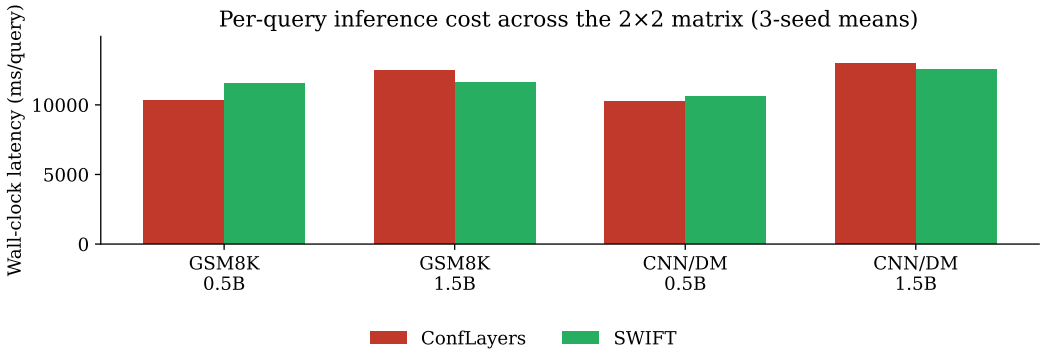


Figure 2: Per-query wall-clock latency across the full matrix.

6.2.1 Search-overhead decomposition

Naive wall-clock comparison ranks ConfLayers as faster than SWIFT in three of four cells (Table 3, "wall-clock total"). This is misleading. We instrumented both methods to separately log pure model-forward-pass time and the wall-clock time consumed by each method's online search — Bayesian-optimization threshold search for ConfLayers, and the analogous search over which layers to self-draft-skip for SWIFT. Figure 3 shows the result: once decomposed, SWIFT's pure inference cost is *faster* than ConfLayers's in all four cells (5–21%), reversing the naive ranking in three of them. ConfLayers's search overhead is small and stable (1.2–2.2% of total cost, near-zero variance across seeds); SWIFT's is larger (8.7–18.6% mean) and more variable (up to 28.7% on individual GSM8K/0.5B seeds).

Every prior cost comparison involving SWIFT in this line of work understated its true inference speed by folding search cost into the reported number. This decomposition assumes many queries are served within one running session: both methods' search state persists across queries in our harness rather than resetting per query, so search overhead is front-loaded onto early queries and shrinks as the search converges. We recommend future efficiency comparisons report this decomposition explicitly, alongside a total-cost-for-N-queries view when N is known, rather than a single averaged wall-clock figure.

7 Supplemental Analysis: Trained Routing at Coarser Granularity

This section examines LayerRoute and LayerDrop, two trained-routing methods operating at a coarser decision granularity than ConfLayers and SWIFT (Section 2), not as direct competitors to the main comparison but under a measurement protocol verified to exercise genuine per-input gating, a genuine full-model baseline, and genuine inference-time compute skipping (Section 7.2).

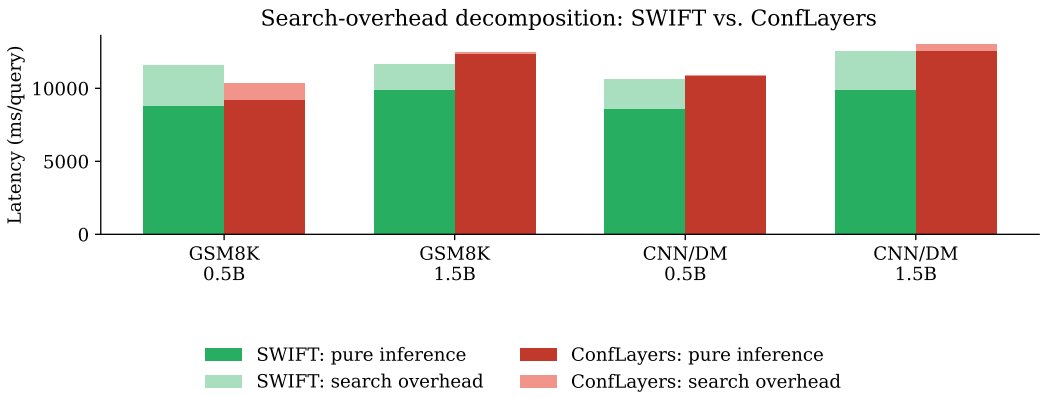


Figure 3: Search-overhead decomposition. Once online-search cost is separated from pure model-forward cost, SWIFT is the faster inference engine in every cell, despite appearing slower on unsplit wall-clock time in three of four cells.

7.1 Method formalizations

LayerRoute (Sikdar, 2026). LayerRoute freezes a pretrained backbone θ and attaches, at every transformer layer $l \in \{1, \ldots, L\}$, a low-rank adapter ΔW_l (Hu et al., 2022) with a scalar gate head. For hidden state h_{l-1} entering layer l,

$$f_l(h_{l-1}) = h_{l-1} + \mathrm{Attn}_l(h_{l-1}) + \mathrm{FFN}_{\Delta W_l}(h_{l-1}), \tag{7}$$

and the gate head produces a scalar logit converted to a hard binary decision

$$z_l = \mathbb{1}\left[\sigma(g_l(h_{l-1})) > 0.5\right], \qquad \sigma(x) = \frac{1}{1+e^{-x}}, \tag{8}$$

with layer output $h_l = z_l\, f_l(h_{l-1}) + (1 - z_l)\, h_{l-1}$, gradients propagated via straight-through estimation (Bengio et al., 2013). The gate head pools over the *current full sequence's* hidden states, making this a per-sequence decision (Table 1).[1] All parameters — router and LoRA — are trained jointly against a single gate-regularized language-modeling objective,

$$\mathcal{L} = \mathcal{L}_{\mathrm{LM}} + \lambda \cdot \frac{1}{L} \sum_i \sigma(s_i), \tag{9}$$

with $\lambda = 1.0$ discouraging collapse to an all-open equilibrium; no auxiliary classification signal (e.g. tool-call/planning provenance) is used. Schematic in Figure 7.

LayerDrop (Fan et al., 2020). We adapt LayerDrop to the same LoRA budget as LayerRoute for a matched training-cost comparison. During training, each layer is stochastically dropped with probability p (uniform across layers), training the backbone and LoRA adapters to be robust to any layer's removal. At inference, a *fixed* set of layers, chosen once from training-time drop statistics, is pruned identically for every input — no router, no per-input adaptivity, the coarsest regime in Table 1. Schematic in Figure 8.

7.2 Measurement protocol

Because LayerRoute's architecture blends a gated and an ungated output rather than branching between them, measuring it correctly requires care on three points, each verified independently.

Genuine gating. Wall-clock cost is measured through LayerRoute's gated forward pass directly, token by token, verified to produce output that genuinely differs from an ungated pass rather than being byte-identical to it.

[1] LayerRoute's original release reported a skip-rate differential between tool-call and planning inputs; this did not reproduce under multi-seed testing and has since been withdrawn by its authors. The measurement protocol and results in this section were obtained under the corrected training recipe throughout and do not depend on that claim.

Table 4: Supplemental analysis: accuracy and speedup, mean over 3 seeds (range in brackets), LayerRoute and LayerDrop versus a genuine full-model baseline. GSM8K is exact-match; CNN/DailyMail is ROUGE-L. Speedup > 1 indicates the routed/pruned pass is faster than the full-model baseline.

Task	Scale	LayerRoute acc.	LayerRoute speedup	LayerDrop acc.	LayerDrop speedup
GSM8K	0.5B	0.137 [0.11–0.16]	1.08×	0.010 [0.00–0.02]	1.30×
	1.5B	0.003 [0.00–0.01]	1.32×	0.060 [0.05–0.07]	1.25×
CNN/DM	0.5B	0.102 [0.099–0.107]	1.33×	0.091 [0.085–0.097]	1.29×
	1.5B	0.150 [0.139–0.165]	1.09×	0.134 [0.131–0.136]	1.25×

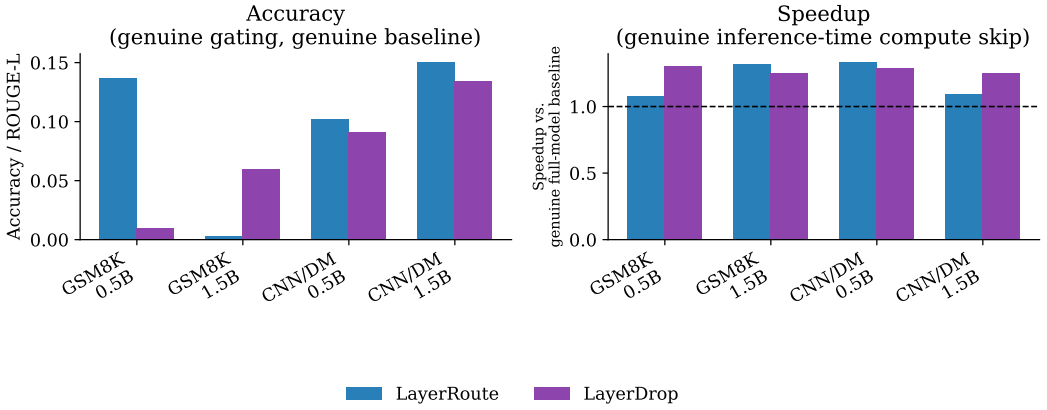


Figure 4: Supplemental results. Left: accuracy under genuine gating and a genuine full-model baseline. Right: speedup versus that same genuine baseline, including genuine inference-time compute skipping (Section 7.2).

Genuine full-model baseline. The baseline forces every layer open via direct bias manipulation on the gate head, verified on a controlled synthetic test to change the layer-execution count to the full layer count.

Genuine inference-time compute skipping. Training computes every layer's full transformation unconditionally (a necessary property of the straight-through estimator, whose backward pass needs the layer's real forward value even for a gate evaluating to zero); inference adds a conditional skip where no such constraint applies (`if gate==0 and not self.training: continue`), verified via layer-call counting to produce fewer real invocations than the total layer count, with output identical to the blended-but-discarded version for closed gates.

For long-form generation, we verified the generation loop's stopping condition is decoupled from prompt-length truncation, since CNN/DailyMail articles can approach the truncation limit; the post-hoc token-count distribution matches the expected range for both methods, with no queries terminating after a single token.

LayerDrop's fixed pattern requires only the baseline and skip verifications, having no per-input gate to exercise. ConfLayers and SWIFT's results (Sections 6–6.2.1) use independent evaluation code, verified separately under the same standard.

7.3 Accuracy and Cost

Table 4 and Figure 4 report accuracy and speedup, 3 seeds each, using a genuine full-model baseline throughout.

Two findings stand out. First, LayerRoute's accuracy on GSM8K at 1.5B is essentially zero (0.003 mean, 1 of 300 samples correct) — a near-total collapse. Routed generations at this scale are coherent, grammatically well-formed text, not degenerate repetition, but consistently fail to reach a correct answer, which we interpret as a genuine capability gap under real per-sequence gating. Second, both methods show modest but real speedups (1.08–1.33×) over a genuine full-model baseline, smaller

than layer-count-implied theoretical savings would suggest; we attribute this to LayerRoute's router computation (pooling and a linear projection, evaluated at every layer regardless of that layer's decision) contributing fixed per-layer overhead. LayerDrop, having no such per-query overhead, shows a somewhat larger, more consistent speedup (1.25–1.30×) despite lower average accuracy.

These results should not be read as "trained routing underperforms search-based routing" in general: LayerRoute and LayerDrop operate at a different decision granularity (Section 2), were trained under a fixed, unremarkable LoRA budget not tuned per condition, and represent two specific instances of a much larger design space.

8 Discussion and Limitations

Main finding. Among periodic-step, search-based methods, SWIFT is stronger on both accuracy and (once search overhead is decomposed) speed; ConfLayers is dominated on both axes throughout, stable across all seeds in every cell.

On the supplemental analysis. LayerRoute and LayerDrop occupy a coarser, less input-adaptive regime than ConfLayers/SWIFT (Section 2); the honest conclusion is that under a fixed, unremarkable LoRA budget, and measured under the verified protocol of Section 7.2, neither is competitive on accuracy here, while both show modest, genuine speedups once compute skipping is confirmed to actually occur.

Scope of the audit. We evaluate two model scales and two tasks on instruction-tuned backbones only; findings may not hold for base models. We follow the reference implementations' default evaluation size (100 samples) without a power analysis for confidence-interval width. Given $n{=}3$ seeds, we report per-seed values directly (Appendix B) rather than a higher-powered parametric test, and we do not run warm-up iterations before timing. Memory footprint, integration complexity, distribution-shift robustness, and combinability with techniques such as quantization are also relevant to deployment but outside this audit's scope.

AI Use Statement (Mandatory Disclosure)

In this work, generative AI tools were used to aid and polish the writing of the paper, for retrieval and discovery of related work during the literature survey (all AI-suggested sources were verified against the original papers before citing), and to assist in creating and editing software code, including the search-overhead decomposition harness and plotting utilities (all AI-assisted code was reviewed and tested against known reference outputs before use). Generative AI tools were not used to generate synthetic data, develop theoretical models or conceptual frameworks, formulate mathematical claims, propose or refine hypotheses, or design the research methodology or experiments — these are not applicable to how this work was produced. All experimental design, method choices, and interpretation of results are the author's own. The author takes responsibility for the final content of this work, including text and code produced with the aid of generative AI.

Reproducibility Statement

Figures in Sections 6–7 are means over three fixed seeds (2024, 42, 123) with per-seed values in Appendix B; full protocol in Sections 5–5.1 and 7.2. The SWIFT port was verified class-by-class against its upstream implementation. We intend to release our harness, the Qwen2 SWIFT port, trained checkpoints, and per-query result files with this paper.

References

W. Amer, U. Das, and F. Kurdahi. Conflayers: Adaptive confidence-based layer skipping for self-speculative decoding. *arXiv preprint arXiv:2604.14612*, 2026.

Andrea Banino, Jan Balaguer, and Charles Blundell. Pondernet: Learning to ponder. *arXiv preprint arXiv:2107.05407*, 2021.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

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. *International Conference on Machine Learning*, 2024.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Cunxiao Du, Jing Jiang, Xu Yuanchen, et al. Glide with a cape: A low-hassle method to accelerate speculative decoding. *arXiv preprint arXiv:2402.02082*, 2024.

Maha Elbayad, Jiatao Gu, Edouard Grave, and Michael Auli. Depth-adaptive transformer. *International Conference on Learning Representations*, 2020.

Angela Fan, Edouard Grave, and Armand Joulin. Reducing transformer depth on demand with structured dropout. *International Conference on Learning Representations*, 2020.

Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016.

Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Daniel A Roberts. The unreasonable ineffectiveness of the deeper layers. *arXiv preprint arXiv:2403.17887*, 2024.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *International Conference on Learning Representations*, 2022.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pp. 19274–19286. PMLR, 2023.

Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pp. 74–81, 2004.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models. *Advances in Neural Information Processing Systems*, 36, 2023.

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. *arXiv preprint arXiv:2403.03853*, 2024.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, et al. Specinfer: Accelerating generative llm serving with speculative inference and token tree verification. *ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 2024.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar Gųlçehre, and Bing Xiang. Abstractive text summarization using sequence-to-sequence rnns and beyond. *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pp. 280–290, 2016.

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. *arXiv preprint arXiv:2404.02258*, 2024.

Tal Schuster, Adam Fisch, Tommi Jaakkola, and Regina Barzilay. Consistent accelerated inference via confident adaptive transformers. *Empirical Methods in Natural Language Processing*, 2021.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Q Tran, Yi Tay, and Donald Metzler. Confident adaptive language modeling. *Advances in Neural Information Processing Systems*, 35:17456–17472, 2022.

Abigail See, Peter J Liu, and Christopher D Manning. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pp. 1073–1083, 2017.

Prateek Kumar Sikdar. Layerroute: Adaptive layer-skipping with lora-preserved quality for efficient llm inference. *arXiv preprint arXiv:TBD*, 2026.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. Blockwise parallel decoding for deep autoregressive models. *Advances in Neural Information Processing Systems*, 31, 2018.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. A simple and effective pruning approach for large language models. *International Conference on Learning Representations*, 2024.

Surat Teerapittayanon, Bradley McDanel, and H.T. Kung. Branchynet: Fast inference via early exiting from deep neural networks. *International Conference on Pattern Recognition*, 2016.

Heming Xia, Yongqi Li, Jun Zhang, Cunxiao Du, and Wenjie Li. Swift: On-the-fly self-speculative decoding for llm inference acceleration. *arXiv preprint arXiv:2410.06916*, 2024.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

A Architecture Diagrams

Figures 5 and 6 give schematic diagrams for the two main-comparison methods; Figures 7 and 8 diagram LayerRoute and LayerDrop, discussed in the supplemental analysis (Section 7).

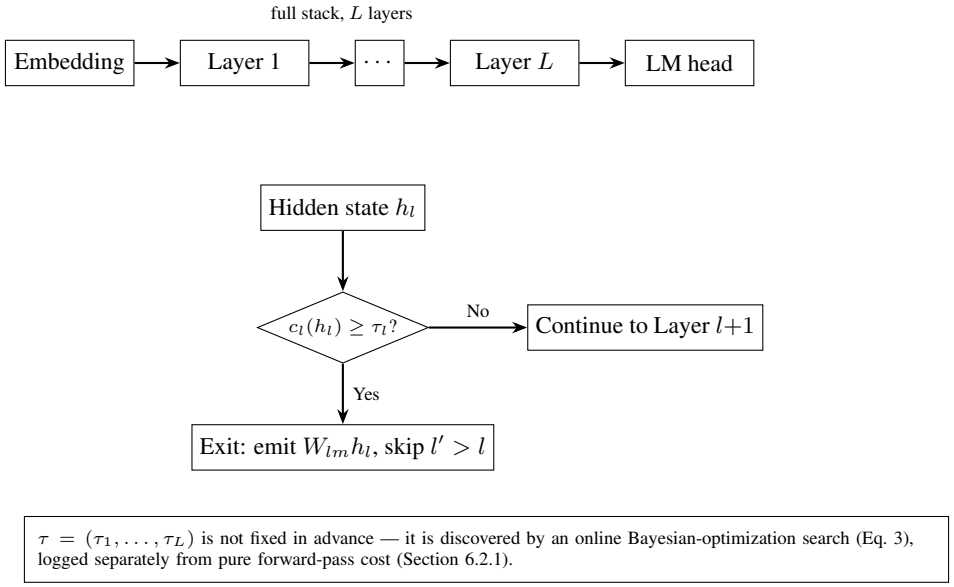
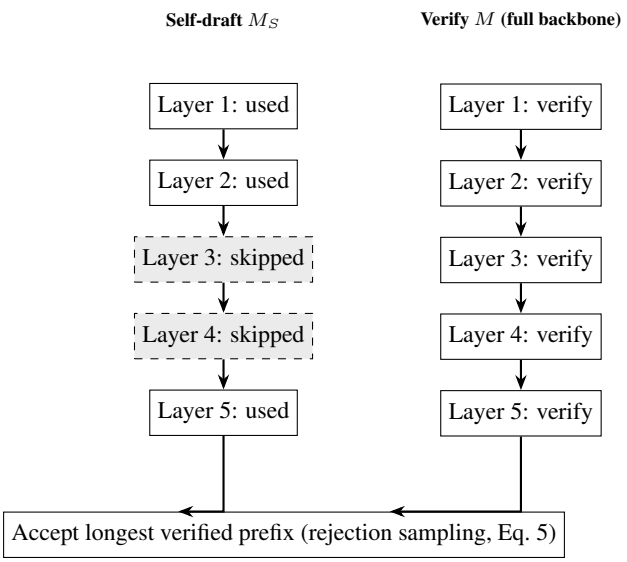


Figure 5: ConfLayers: top, the full layer stack; bottom, detail of the per-layer confidence check. A per-layer confidence estimate c_l is thresholded against τ_l, discovered online via Bayesian optimization; pure inference and search overhead are logged and reported separately (Section 6.2.1).



S, the set of skipped layers, is searched online (Eq. 6) to trade a cheaper draft against a lower acceptance rate — logged separately from pure drafting cost (Section 6.2.1).

Figure 6: SWIFT: the target model self-drafts by skipping a searched subset S of its own layers, then the full model verifies all draft positions in a single batched pass; the longest accepted prefix is kept and the rest resampled.

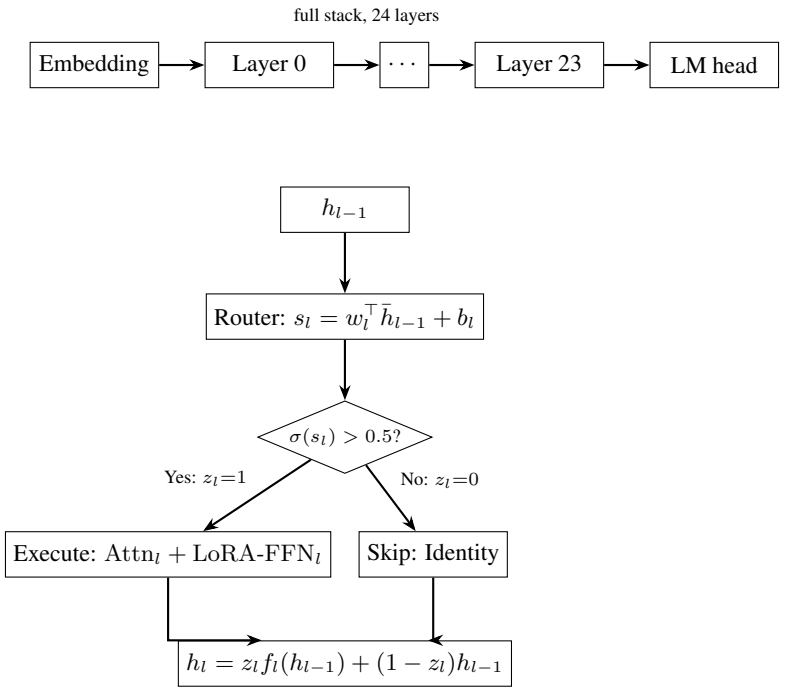
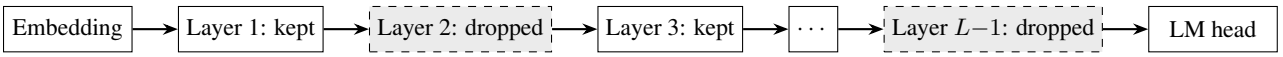


Figure 7: LayerRoute: top, the full 24-layer stack; bottom, detail of a single gated layer l. The router computes a scalar score from the mean-pooled input hidden state $\bar{h}_{l-1}$, thresholds it into a hard gate z_l via a straight-through estimator, and the layer output is either the LoRA-adapted block's output (z_l=1) or an unchanged residual pass-through (z_l=0), matching Equations 7–8. Discussed in the supplemental analysis, Section 7.



Fixed, input-independent pattern. Which layers are kept versus dropped is chosen once from training-time drop statistics (each layer stochastically dropped with probability p during training) and then applied *identically to every input* at inference — no router, no per-input adaptivity, the coarsest regime in Table 1.

Figure 8: LayerDrop: layers are stochastically dropped during training to make the backbone and LoRA adapters robust to any layer's removal; at inference, one fixed set of layers, chosen once from training statistics, is pruned identically for every input — no router, no per-input adaptivity. Discussed in the supplemental analysis, Section 7.

B Additional Per-Seed Results

B.1 Main comparison: ConfLayers and SWIFT

Table 2 in the main text reports each accuracy cell as a mean over three seeds with the observed range. Tables 5 and 6 give every individual per-seed value.

Table 5: GSM8K exact-match accuracy at each of the three evaluation seeds, main comparison.

Scale	Seed	Vanilla	ConfLayers	SWIFT
0.5B	2024	0.26	0.13	0.36
	42	0.14	0.18	0.28
	123	0.14	0.13	0.27
1.5B	2024	0.40	0.05	0.29
	42	0.46	0.08	0.33
	123	0.38	0.10	0.30

Table 6: CNN/DailyMail ROUGE-L at each of the three evaluation seeds, main comparison.

Scale	Seed	Vanilla	ConfLayers	SWIFT
0.5B	2024	0.167	0.175	0.190
	42	0.174	0.184	0.196
	123	0.167	0.178	0.185
1.5B	2024	0.193	0.204	0.205
	42	0.206	0.218	0.223
	123	0.213	0.222	0.230

B.2 Supplemental analysis: LayerRoute and LayerDrop

Tables 7 and 8 give per-seed values underlying Table 4, measured under the protocol described in Section 7.2.

Table 7: GSM8K exact-match accuracy at each of the three evaluation seeds, supplemental analysis.

Scale	Seed	LayerRoute	LayerDrop
0.5B	2024	0.14	0.00
	42	0.16	0.02
	123	0.11	0.01
1.5B	2024	0.00	0.07
	42	0.01	0.05
	123	0.00	0.06

Table 8: CNN/DailyMail ROUGE-L at each of the three evaluation seeds, supplemental analysis.

Scale	Seed	LayerRoute	LayerDrop
0.5B	2024	0.100	0.085
	42	0.099	0.092
	123	0.107	0.097
1.5B	2024	0.146	0.136
	42	0.139	0.131
	123	0.165	0.134