SSP: SELF-SPECULATIVE PREFILL

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-context inference with short outputs, common in document question answering, retrieval, and code completion, is often dominated by prefill. Existing key-value (KV) construction methods accelerate prefill by avoiding exact computation, but often make open-loop decisions without assessing whether the constructed cache entries are acceptable, creating a difficult quality-speed trade-off. This motivates identifying which draft KV entries can be safely used, rather than requiring every draft to be exact. Therefore, we introduce ***Self-Speculative Prefill*** **(SSP)**, a framework for verification-guided cache construction with four components: construction, verification, routing, and fallback. We analyze the theoretical boundaries of KV cache construction and verification. Our realization, **ACSSP**, constructs KV entries using low-rank residual KV maps with original model weights frozen, verifies drafts with a training-free attention-weighted verification score, uses token-level routing to select aggressive acceptance or continuation to a conservative anchor, and reuses verification-boundary states during continuation. Compared with POP, ACSSP improves average quality on LongBench and RepoBench by up to 7.47 points, delivers up to $1.25\times$ and $1.34\times$ TTFT speedups on RULER at 32K and 128K, respectively, and reduces mean prefill matmul FLOPs by up to 20.19% across seven downstream tasks on Llama-3.1-8B-Instruct. Code is available at `https://anonymous.4open.science/r/Zgrnf7sg`.

1 INTRODUCTION

Large language models (LLMs) (Brown et al., 2020; Grattafiori et al., 2024) increasingly serve long-context workloads with short outputs, including document question answering (Bai et al., 2024), retrieval, and repository-level code completion (Liu et al., 2024). In these workloads, prefill often dominates inference cost and latency. Native prefill propagates every prompt token through every Transformer layer (Vaswani et al., 2017) to construct the layer-wise key-value (KV) cache consumed by decoding (Kwon et al., 2023). For example, MInference (Jiang et al., 2024a) reports a prefill time of approximately 30 minutes for 1M tokens with Llama-3-8B-1M on a single A100 GPU. Much of the inference cost is therefore incurred before the first output token is generated.

Existing accelerations reduce this cost by avoiding parts of exact computation. Sparse attention (Jiang et al., 2024a; Lai et al., 2025) evaluates fewer query-key interactions, while prompt compression (Jiang et al., 2024b) and KV eviction (Li et al., 2024) reduce retained tokens or cache entries. KV construction methods such as POP (He et al., 2026) and SwiftKV (Qiao et al., 2025) preserve cache positions by constructing missing entries from earlier hidden states. However, these *constructor-only* methods make open-loop decisions and pass constructed KV entries directly to decoding without verifying their adequacy. Construction errors can consequently affect output quality, creating a difficult quality-speed trade-off.

This trade-off motivates identifying which drafts can be accepted and which require further computation. Speculative decoding (Leviathan et al., 2023) and self-speculative decoding (Elhoushi et al., 2024; Zhang et al., 2024) address a related question during generation by verifying draft tokens before accepting them. For prefill, executing the full exact suffix to verify draft KVs would eliminate the intended savings. Making this choice efficiently requires inexpensive signals for assessing whether draft KV entries are acceptable. we refer to these signals as evidence. Acceptable drafts can then be used, while others receive further computation that reuses completed work. Motivated by these requirements, we introduce *Self-Speculative Prefill* (SSP), which casts KV-cache construction as verification-guided compute allocation across the model lifecycle (Figure 1). Main contributions are:

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.

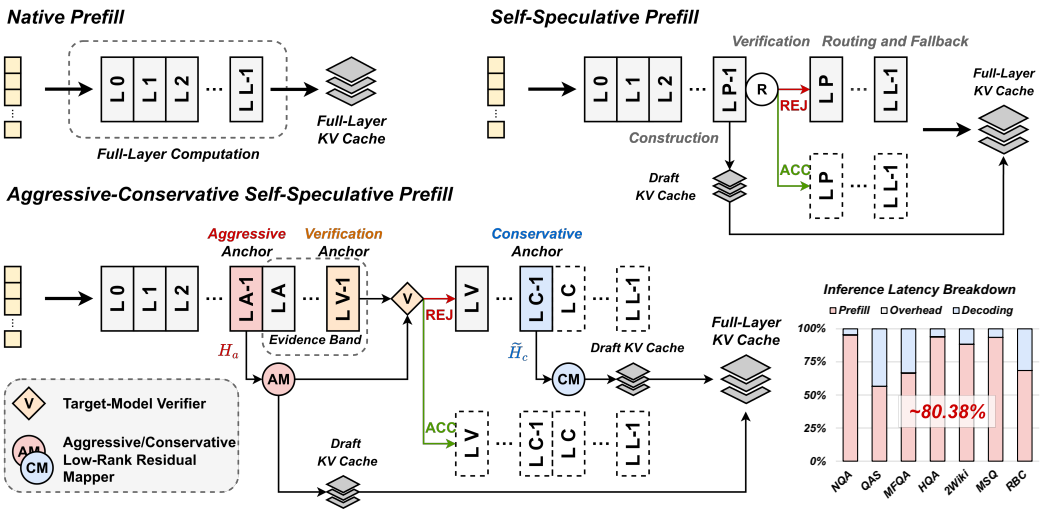

Figure 1: Overview of native prefill, SSP, and ACSSP. The bottom-right panel decomposes native inference latency into prefill, overhead, and decoding. Prefill accounts for approximately 80.38% of latency in long-context and short-output workloads shown.

1. **Verification-guided cache construction.** SSP casts KV-cache construction as a per-unit decision through *construction*, *verification*, *routing*, and *fallback*.
2. **Theoretical error boundaries.** We decompose construction error into an interface-determined information floor and a finite-capacity term, bound verification error by the distinguishability of the available evidence, and show that the two routing errors differ in whether fallback can repair them.
3. **A joint realization of all four components.** ACSSP constructs draft KV entries using low-rank residual KV maps over frozen base projections (construction), assesses the drafts using a training-free, attention-weighted verification score (verification), uses token-level routing to select aggressive acceptance or continuation to a conservative anchor (routing), and resumes rejected tokens from retained verification-boundary states without replaying the prefix (fallback).
4. **System implementation and comprehensive evaluation.** We implement ACSSP on vLLM (Kwon et al., 2023) and evaluate quality and efficiency under a consistent accounting boundary. Across evaluated models and profiles, ACSSP improves average quality on LongBench and RepoBench by up to 7.47 points over POP and achieves RULER TTFT speedups over POP of up to $1.25\times/1.34\times$ at 32K/128K. On Llama-3.1-8B-Instruct, it also reduces mean prefill matmul FLOPs by up to 20.19% relative to POP across seven downstream tasks.

2 Motivation and Related Work

2.1 Motivation

We examine opportunities for early exit, KV construction, and selective computation through three diagnostics (Figure 2). The *importance score* measures each layer's share of request-normalized squared token-gate gradients of continuation negative log-likelihood. The *similarity score* measures alignment between anchor states and later layer inputs. Finally, we measure how POP's full-suffix attention-weighted construction error is distributed across tokens. Appendix A.1 defines the diagnostics.

Insight 1: earlier layers carry most of the prefill importance. Importance concentrates in the first half of the model, motivating early exit. Model-dependent peaks caution against a universal exit depth. This local gradient diagnostic does not measure task loss from removing layers.

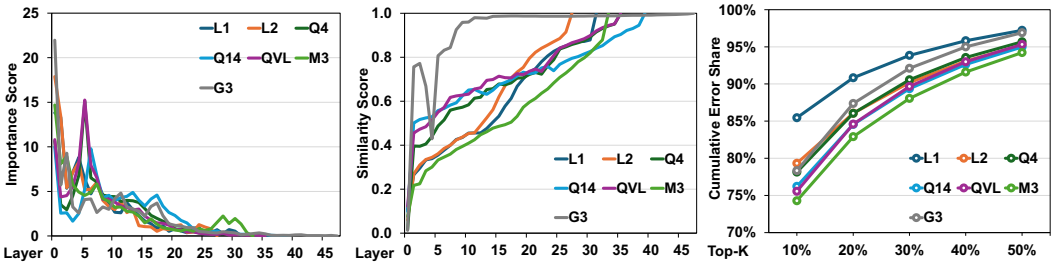

Figure 2: **Prefill diagnostics on seven models.** Layer importance (left), similarity to later layer inputs (middle), and cumulative share of full-suffix attention-weighted construction error contributed by the highest-error token fractions, averaged equally over requests (right). Llama-3.1-8B-Instruct (L1), Llama-3.2-3B-Instruct (L2), Qwen3-4B-Instruct-2507 (Q4), Qwen3-14B (Q14), Qwen3-VL-8B-Instruct (QVL), Ministral-3-8B-Instruct-2512-BF16 (M3), Gemma-3-12B-it (G3).

Insight 2: deeper states align more closely with later layer inputs. Similarity generally increases with depth, motivating KV construction from saved anchor states. Deeper anchors offer better alignment but less computation to skip. Normalization and projection still determine KV fidelity.

Insight 3: construction error is concentrated in a minority of tokens. Deeper anchors save more prefill computation but can reduce output quality, creating a quality-speed trade-off. We observe that the associated construction error is highly uneven across tokens. Measuring POP's full-suffix attention-weighted construction error against exact KVs, we rank nonfinal tokens within each request. Across seven models, the top 20% account for $>82\%$ of weighted error, and the top 50% account for $>94\%$. This concentration motivates verifying drafts and selectively allocating further computation to break the quality-speed trade-off. The ranking uses full-suffix information for diagnosis. Whether cheap evidence identifies useful routes is tested by the routing controls in Section 5.3.

2.2 Related Work

Model pruning and layer skipping. Wanda (Sun et al., 2024a), SliceGPT (Ashkboos et al., 2024), and ShortGPT (Men et al., 2025) remove weights, dimensions, or layers, while Router-Tuning (He et al., 2025) and LayerSkip (Elhoushi et al., 2024) support adaptive depth. KV Prediction (Horton et al., 2025), POP (He et al., 2026), and SwiftKV (Qiao et al., 2025) reduce prefill depth while constructing the missing K/V rows for the full-depth decoder. ACSSP adds verification-guided token routes and low-rank residual KV maps to pruning-based construction, keeping backbone weights frozen. Section 3.5 details constructor differences.

Token pruning and compression. Prompt compression (Jiang et al., 2023; 2024b; Pan et al., 2024) and KV compression (Zhang et al., 2023; Li et al., 2024) reduce inputs and cache entries. SpecPrefill (Liu et al., 2025) and its cross-family extension (Upasani et al., 2026) are draft-guided prompt compression methods in which the target processes only selected input tokens or text spans. The latter bridges different tokenizers through retokenization. FastKV (Jo et al., 2026) and DASH (Chen et al., 2026) limit token-state advancement. ACSSP instead constructs suffix K/V rows for exited tokens, preserving all positions.

Sparse attention. MInference (Jiang et al., 2024a), FlexPrefill (Lai et al., 2025), and sparse pattern sharing (Peng et al., 2025) reduce evaluated query-key interactions. DuoAttention (Xiao et al., 2025) separates retrieval and streaming heads. ACSSP instead decides whether to advance a token's hidden state. Exited tokens avoid deeper query and feed-forward computation, while every active query still attends over all positions.

Speculative inference. Speculative decoding (Leviathan et al., 2023; Sadhukhan et al., 2025) verifies draft tokens to preserve the target distribution. Self-speculative decoding (Elhoushi et al., 2024; Zhang et al., 2024) drafts from shallow layers and verifies at full depth. In SpecPrefill and its cross-family extension, 'speculation' refers to selecting prompt content, without draft-KV verification or rejection-triggered continuation. SSP verifies suffix-KV drafts using available evidence and supports continuation from retained states for units with rejected drafts. Its acceptance is approximate, with request-level quality assessed empirically.

3 SELF-SPECULATIVE PREFILL: FRAMEWORK AND BOUNDARIES

We define SSP and its theoretical boundaries. Appendix A gives proofs and specializations.

3.1 SSP MECHANISM

For a frozen L-layer decoder and request $x \sim \mathcal{D}$, let $H_m(x)$ be the exact state after m layers and $Y_m(x) = \mathrm{KV}_{m:L-1}(x)$ the native suffix cache. A routing unit is the smallest element assigned a separate construction or continuation decision. $\mathcal{I}(x)$ indexes these units and $\mathcal{A}(x)$ contains feasible route assignments. An SSP mechanism is

$$\mathfrak{S} = (G, V, \pi, F), \tag{1}$$

where G constructs draft KV entries from observations $U_m = \phi_m(H_m)$. Verification $V = (E, f_{\mathrm{ver}})$ uses E to obtain *evidence* $Z = (Z_u)_{u \in \mathcal{I}(x)}$, signals used to assess whether draft KV entries are acceptable, and computes scores or labels via a measurable $f_{\mathrm{ver}}(Z)$. Routing π uses these assessments and permitted evidence to assign execution paths $\mathbf{r}(x) \in \mathcal{A}(x)$. Fallback F specifies continuations that reuse pre-routing computation. Evidence must be available at the routing decision and may come from fixed computations or learned mechanisms. Verification need not certify exactness. Constant routes recover fixed depth reduction, one unit per request recovers request-level routing, and all-exact execution recovers Native. Appendix A.2 defines the quality-cost objective using execution cost c_π and request-level quality drop d_π.

3.2 KV CONSTRUCTION BOUNDARIES

Fix observations U_m, targets Y_m, and a constructor family $\mathcal{G}$. Their interface must match. Token-row maps observe one row of H_m and predict its suffix KVs. Condition on prompt length and unit index to fix target dimensions. Assume square-integrable Y_m and define $\mu_m(U_m) = \mathbb{E}[Y_m \mid U_m]$. Matrix variances use vectorization, and matrix norms are Frobenius.

Theorem 1 (Information and capacity of a suffix KV draft)**.** *For any nonempty family $\mathcal{G}$ of square-integrable measurable constructors,*

$$\underbrace{\inf_{g \in \mathcal{G}} \mathbb{E}\|Y_m - g(U_m)\|_F^2}_{\mathcal{E}^\star_{\mathcal{G}}(m)} = \underbrace{\mathbb{E}\,\mathrm{tr}[\mathrm{Var}(Y_m \mid U_m)]}_{\mathcal{E}_{\mathrm{info}}(m)} + \underbrace{\inf_{g \in \mathcal{G}} \mathbb{E}\|\mu_m(U_m) - g(U_m)\|_F^2}_{\mathcal{E}_{\mathrm{cap}}(\mathcal{G},m)}. \tag{2}$$

If, additionally, $Y_m = f_m(H_m)$ with f_m being L_f-Lipschitz, and a measurable state reconstruction ψ and $g \in \mathcal{G}$ satisfy $(\mathbb{E}\|H_m - \psi(U_m)\|_F^2)^{1/2} \le \varepsilon_U$ and $(\mathbb{E}\|f_m(\psi(U_m)) - g(U_m)\|_F^2)^{1/2} \le \varepsilon_{\mathcal{G}}$, then

$$\boxed{\mathcal{E}_{\mathrm{info}}(m) \le \mathcal{E}^\star_{\mathcal{G}}(m) \le (L_f \varepsilon_U + \varepsilon_{\mathcal{G}})^2.} \tag{3}$$

The decomposition separates missing information from restricted constructor capacity. Increasing rank can reduce $\mathcal{E}_{\mathrm{cap}}$ but cannot remove a positive $\mathcal{E}_{\mathrm{info}}$. A token-row interface need not have a positive floor if its observation determines the target. Proposition 5 specializes capacity to low-rank residual KV maps, whose empirical feature-span residual is not an estimate of the information floor. The upper bound requires the stated reconstruction and stability assumptions and concerns cache error, not task loss. Appendix A.3 gives the proof. Increasing rank enlarges the constructor family at a fixed interface, whereas changing the anchor changes the observation and target suffix. These are distinct design choices.

3.3 EVIDENCE-BASED VERIFICATION BOUNDARIES

Fix a constructor g and evidence extractor E, possibly fitted independently of the verifier sample. For $W = (x, u) \sim \mathcal{Q}_g$, let $Z = E(W)$ and define local adequacy $S = \mathbf{1}\{e_g(W) \le \tau\}$ for a specified discrepancy and tolerance, distinct from request-level drop d_π. Write $\nu_s = \Pr(S = s) > 0$ and $P_s = \mathcal{L}(Z \mid S = s)$. Including any score thresholding, a binary verifier has error $R_{\mathrm{ver}}(f_{\mathrm{ver}}) = \Pr[f_{\mathrm{ver}}(Z) \ne S]$ and empirical error $\widehat{R}_{\mathrm{ver}}(f_{\mathrm{ver}})$ on n i.i.d. labeled instances.

Theorem 2 (Distinguishability and verifier complexity). *For a measure μ dominating P_0, P_1 and densities $p_s = dP_s/d\mu$, the optimal error over all measurable verifiers is*

$$\epsilon_Z^\star := \inf_{f_{\mathrm{ver}}} \Pr[f_{\mathrm{ver}}(Z) \neq S] = \int \min\{\nu_0 p_0(z), \nu_1 p_1(z)\}\, d\mu(z). \tag{4}$$

With equal priors, $\epsilon_Z^\star = \frac{1}{2}(1 - \mathrm{TV}(P_0, P_1))$, where TV *denotes the total variation distance. For a nonempty declared class $\mathcal{V}$ of measurable binary verifiers, let $\widehat{f}_{\mathrm{ver}} \in \arg\min_{f_{\mathrm{ver}} \in \mathcal{V}} \widehat{R}_{\mathrm{ver}}(f_{\mathrm{ver}})$, and let $\Gamma_n(\mathcal{V}, \delta)$ simultaneously bound $|R_{\mathrm{ver}}(f_{\mathrm{ver}}) - \widehat{R}_{\mathrm{ver}}(f_{\mathrm{ver}})|$ for every $f_{\mathrm{ver}} \in \mathcal{V}$ with probability at least $1 - \delta$. Then, with that probability,*

$$\boxed{\epsilon_Z^\star \leq R_{\mathrm{ver}}(\widehat{f}_{\mathrm{ver}}) \leq \epsilon_Z^\star + \epsilon_{\mathrm{app}}(\mathcal{V}, Z) + 2\Gamma_n(\mathcal{V}, \delta),} \tag{5}$$

where $\epsilon_{\mathrm{app}}(\mathcal{V}, Z) = \inf_{f_{\mathrm{ver}} \in \mathcal{V}} R_{\mathrm{ver}}(f_{\mathrm{ver}}) - \epsilon_Z^\star \geq 0$.

Overlapping class-conditional evidence creates a Bayes error floor that verifier capacity alone cannot remove. Richer evidence cannot increase this floor (Proposition 4), but the maximum used to form a verification score need not become more informative with a wider window (Appendix B.2). Appendix A.4 gives finite-class bounds and sampling conditions. Correlated tokens are not independent instances. The ERM upper bound applies to labeled error minimization, not acceptance-rate calibration. The latter selects a deployment route fraction whose request-level quality must be evaluated separately. Constructor fidelity and evidence informativeness are therefore separate objectives. Lower mean cache error need not make the remaining inadequate drafts easier to identify. The map, random-routing, and verification comparisons in Section 5.3 examine these design consequences.

3.4 Routing, Fallback, and Error Consequences

False acceptance leaves an inadequate draft in the cache, whereas false rejection incurs continuation cost. Exact fallback avoids additional approximation error only with exact inputs and cache dependencies. ACSSP instead uses conservative maps and hybrid attention, so rejection does not guarantee recovery to Native. Neither construction error nor local verification error alone certifies coupled request-level quality, which Section 5.3 evaluates alongside execution cost.

3.5 Existing Methods in the SSP Framework

Architecture and pre-training methods. YOCO (Sun et al., 2024b) constructs shared global KVs from self-decoder outputs. CED (DeepSeek-AI et al., 2026) projects causal-encoder outputs into global KVs. HySparse2 (Wei et al., 2026) projects self-decoder layer inputs using cross-decoder K/V weights and shares the results with sparse layers. Pretraining adapts each model to these native cache interfaces without online draft verification.

Post-training methods. POP (He et al., 2026) constructs suffix KVs from earlier states using frozen projections, without construction-specific adaptation or online verification. SwiftKV (Qiao et al., 2025) uses earlier states with adapted QKV projections, relying on self-distillation to preserve output quality. Both can be viewed as special cases of SSP that directly accept all constructed KV entries without verification or fallback. ACSSP adds verification-guided token routes to low-rank residual KV mapping with frozen original weights.

4 ACSSP: Aggressive-Conservative Self-Speculative Prefill

ACSSP saves an aggressive anchor at a and gathers native evidence for verification through v. It then routes nonfinal tokens to aggressive mapping or continuation to a conservative anchor c, retaining all cache positions for the original decoder.

4.1 Low-Rank Residual KV Mapping

Layers are indexed $0, \ldots, L-1$. We use integer depths $0 \leq a < v \leq c < L$, with H_m denoting the native state after m layers. For token set $\mathcal{J}$, a low-rank residual KV map constructs missing K/V

rows from the saved anchor state $H_{m,\mathcal{J}}$. Let $\widetilde{H}_{\ell|m,\mathcal{J}} = \mathrm{Norm}_\ell(H_{m,\mathcal{J}})$ apply target layer ℓ's frozen attention normalization. Each KV map adds a low-rank residual correction to the target layer's frozen base projection:

$$\widehat{K}^{(m)}_{\ell,\mathcal{J}} = \mathrm{RoPE}_\ell\Big(\widetilde{H}_{\ell|m,\mathcal{J}}[W_{K,\ell} + U^{(m)}_{K,\ell} V^{(m)\top}_{K,\ell}]\Big),$$
$$\widehat{V}^{(m)}_{\ell,\mathcal{J}} = \widetilde{H}_{\ell|m,\mathcal{J}}[W_{V,\ell} + U^{(m)}_{V,\ell} V^{(m)\top}_{V,\ell}]. \tag{6}$$

RoPE preserves original token positions. Each residual correction has rank at most r. The sum of the base projection and its correction need not be low-rank. Only the low-rank factors are fitted, while the backbone, normalizations, positional encoding, and base projections remain frozen. Proposition 5 in Appendix B.1 characterizes the family's empirical least-squares capacity. Actual training uses teacher-forced trajectory distillation over H continuation steps:

$$\mathcal{L}_{\mathrm{map}} = \frac{1}{H}\sum_{t=1}^{H} \mathrm{KL}\big(p_t^{\mathrm{native}} \,\|\, p_t^{\mathrm{map}}\big). \tag{7}$$

Both distributions use the same continuation prefix. Maps are fitted on exact native anchors (Appendix C.2). Conservative deployment substitutes hybrid $\widetilde{H}_c$ for H_c, a shift included in evaluation. Zero residuals recover the base projection on the anchor, not native suffix KVs.

4.2 Training-Free Verification Band

Native Q/K/V tensors from $[a, v)$ and draft discrepancies provide token evidence Z_j, summarized by $s_j = f_{\mathrm{ver}}(Z_j)$. For prompt length n, score rows $\mathcal{R} = \{1, \ldots, n-1\}$ using the final $\min\{W, n\}$ query positions $\mathcal{Q}_W$. Query head h reads KV head $\kappa(h)$, with $\kappa(h) = h$ for ordinary multi-head attention. Let $A^{(\ell,h)}_{qj}$ denote native causal attention and λ_q the visible KV count. With $\Delta K = \widehat{K}^{(a)} - K$ and $\Delta V = \widehat{V}^{(a)} - V$, define

$$s_j = \max_{\ell\in[a,v),\ q\in\mathcal{Q}_W,\ h} \lambda_q A^{(\ell,h)}_{qj} \left(\frac{\|\Delta K_{\ell j\kappa(h)}\|_2}{\mathrm{RMS}(K_{\ell\kappa(h)})} + \frac{\|\Delta V_{\ell j\kappa(h)}\|_2}{\mathrm{RMS}(V_{\ell\kappa(h)})}\right). \tag{8}$$

Here $\mathrm{RMS}(X_{\ell\kappa}) = (\|X_{\ell,1:n,\kappa}\|_F^2/(nd_h) + \varepsilon_{\mathrm{num}})^{1/2}$, with head dimension d_h and positive numerical floor $\varepsilon_{\mathrm{num}}$. The maximum retains discrepancies salient to any observed head, query, or layer. λ_q compensates for uniform attention's $1/\lambda_q$ scale without ensuring length invariance. Thresholds are recalibrated when the evidence configuration changes.

Attention sensitivity. Both key and value perturbations affect attention output through the row's attention weight. The score retains this weighting but omits direction, query norm, and value spread. It is a sensitivity proxy, not a finite-error bound (Appendix B.2).

Scoring reuses native tensors, adding $O(Wn)$ query-key interactions per verification layer, draft construction, and row reductions without materializing full attention. It has no learned parameters, and exact H_v states are retained for continuation.

4.3 Aggressive-Conservative Routing

A frozen profile $\theta = (a, v, c, r, W, \tau_s)$ specifies anchors, rank, query window, and threshold, allowing a predeclared threshold schedule by input length. Token j follows

$$\mathrm{route}_\theta(j) = \begin{cases} A \text{ (exit at } v), & j < n,\ s_j \le \tau_s, \\ C \text{ (continue to } c), & j < n,\ s_j > \tau_s, \\ \text{full-depth execution}, & j = n. \end{cases} \tag{9}$$

Let $\mathcal{I}_A$ and $\mathcal{I}_C$ denote the aggressive and conservative nonfinal tokens. After routing, aggressive rows replace their native KVs in $[a, v)$ with the scored drafts and use mapped KVs thereafter. In $[v, c)$, queries $\mathcal{I}_C \cup \{n\}$ execute against the full hybrid cache with original positions and causal masks. At c, conservative tokens switch to mapped KVs. Only token n continues to produce the first logits. Active queries shrink from n to $|\mathcal{I}_C| + 1$ and finally one, while cache coverage remains complete. Continuation starts from exact states H_v but becomes approximate through hybrid attention.

Coverage calibration chooses the threshold. The conditional decision-theoretic interpretation appears in Appendix A.2.

The depths a, v, c set the construction representation, verification boundary, and continuation endpoint. Aggressive routing saves native token updates only in $[v, c)$ relative to an all-conservative route with the same band. Net savings require avoided work to exceed mapping, verification, routing, and data movement. Appendix B.3 derives $D(x) > O(x)$.

Table 1: LongBench and RepoBench quality across models. F, B, and S denote the Fast, Balanced, and Safe profiles of ACSSP, respectively. NarrativeQA (NQA), Qasper (QAS), MultiFieldQA-en (MFQA), HotpotQA (HQA), 2WikiMultihopQA (2Wiki), MuSiQue (MSQ), and RepoBench-C (RBC). Appendix Table 9 gives the full seven-model comparison.

Method	NQA	QAS	MFQA	HQA	2Wiki	MSQ	RBC	Avg.
Llama-3.1-8B-Instruct								
Native	27.54	44.73	56.08	57.55	49.63	33.01	66.07	47.80
POP	27.90	43.19	53.93	54.09	47.13	29.02	58.56	44.83
SwiftKV	29.15	45.76	55.58	56.06	43.74	31.30	65.96	46.79
ACSSP-F	27.94	42.53	53.80	56.92	46.49	26.86	60.09	44.95
ACSSP-B	29.37	42.03	53.94	57.81	47.51	31.94	61.88	46.35
ACSSP-S	29.80	44.12	53.06	56.73	49.49	31.20	63.48	46.84
Llama-3.2-3B-Instruct								
Native	25.09	39.99	50.69	53.09	39.15	25.06	62.79	42.27
POP	20.12	33.07	44.44	45.76	33.24	19.07	55.42	35.87
ACSSP-F	19.97	33.03	45.11	47.26	32.10	21.56	56.24	36.47
ACSSP-B	21.49	37.50	48.86	52.99	37.35	24.01	62.12	40.62
ACSSP-S	22.99	39.54	48.43	52.51	39.16	21.70	61.88	40.89
Qwen3-14B								
Native	28.02	45.68	53.65	63.53	54.32	40.47	70.46	50.88
POP	25.95	44.36	46.71	59.05	44.83	32.70	66.17	45.68
ACSSP-B	25.53	47.01	47.84	57.73	42.16	38.21	68.71	46.74
ACSSP-S	27.88	46.37	51.27	61.95	49.70	39.39	70.48	49.58
Ministral-3-8B-Instruct-2512-BF16								
Native	27.34	45.47	44.98	61.28	54.74	38.71	69.86	48.91
POP	22.82	36.50	38.70	47.32	43.14	28.62	52.62	38.53
ACSSP-B	27.23	40.78	43.95	53.65	47.58	34.61	59.51	43.90
ACSSP-S	29.47	41.79	44.46	57.64	51.47	34.61	62.59	46.00

5 Experiments

5.1 Setup

Models. We evaluate seven models spanning 3B-14B parameters. Table 1 reports quality results for Llama-3.1-8B-Instruct (Grattafiori et al., 2024), Llama-3.2-3B-Instruct, Qwen3-14B (Yang et al., 2025), and Ministral-3-8B-Instruct-2512-BF16 (Liu et al., 2026). Qwen3-4B-Instruct-2507, Qwen3-VL-8B-Instruct (Bai et al., 2025), and Gemma-3-12B-it (Team et al., 2025) extend the comparison to additional sizes and architectures. Appendices C.1 and C.4 give checkpoint details and complete results.

Methods. We compare ACSSP with full-depth prefill (Native), POP (He et al., 2026), and SwiftKV (Qiao et al., 2025). POP constructs missing KVs using frozen projections at a fixed exit depth. SwiftKV uses distilled attention projections, evaluated through its released SingleInputKV checkpoint based on Llama-3.1-8B-Instruct. ACSSP adapts only low-rank residual KV maps, keeping all original model weights, including the decoder, unchanged. SwiftKV reports approximately 1.36B cumulative training tokens, substantially more than the 134.25M prompt tokens sampled in each ACSSP map-fitting run (Appendix C.2).

Benchmarks. We evaluate six LongBench (Bai et al., 2024) QA tasks using F1 and the RepoBench-C 32K subset (Liu et al., 2024) using edit similarity (ES), totaling 1,450 examples. Average quality is the unweighted mean of the seven task scores. Single-needle RULER (Hsieh et al., 2024) measures TTFT

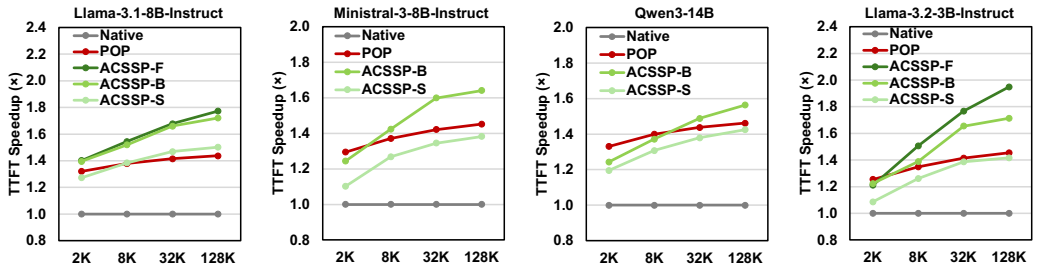

Figure 3: RULER TTFT speedup over Native. Ministral-3-8B-Instruct denotes Ministral-3-8B-Instruct-2512-BF16. See Appendix Table 11 for all models and input token throughput.

scaling at 2K/8K/32K/128K. Adapted RULER QA_2 evaluates accuracy and TTFT at 32K/128K, with accuracy scored by case-insensitive answer substring matching. Downstream timing covers all seven tasks, with HotpotQA (HQA) and RepoBench-C (RBC) highlighted in the main text. TTFT speedup divides matched Native mean latency by method mean latency, and cross-task speedup averages task-wise ratios. Detailed benchmark configurations and scoring protocols appear in Appendices C.1 and C.3.

Implementation details. ACSSP uses KV maps with rank-64 residual corrections. Fast, Balanced, and Safe profiles vary construction anchors, verification query windows, and routing thresholds. Calibration uses disjoint design traces, with one frozen threshold schedule shared by quality and timing within each experiment group. Our vLLM implementation runs on NVIDIA A800 GPUs (80 GB) with BF16 weights and KV caches, batch size 1, and matched tensor parallelism. End-to-end timing includes mapping, verification, routing, active-row compaction, cache materialization, and tensor-parallel synchronization. Appendices C.2 and C.3 detail implementation settings.

5.2 Main Results

Quality gains across model families. ACSSP-B improves the seven-task average over POP on all seven models. Gains on the four main models are 1.52 points for Llama-3.1-8B-Instruct, 4.75 for Llama-3.2-3B-Instruct, 1.06 for Qwen3-14B, and 5.37 for Ministral-3-8B-Instruct-2512-BF16 (Table 1). Safe further raises Ministral-3-8B-Instruct-2512-BF16's score to 46.00, a 7.47-point gain over POP. These gains extend across model families (Appendix Table 9).

Prefill acceleration across context lengths. On the single-needle workload, ACSSP-B exceeds POP's speedup at 32K and 128K on all four main models (Figure 3). Its 128K speedups are 1.721×, 1.713×, 1.564×, and 1.641× on Llama-3.1-8B-Instruct, Llama-3.2-3B-Instruct, Qwen3-14B, and Ministral-3-8B-Instruct-2512-BF16. Fast reaches 1.767× at 32K and 1.948× at 128K on Llama-3.2-3B-Instruct.

Downstream quality and speed. ACSSP-B improves both HotpotQA (HQA) and RepoBench-C (RBC) quality and TTFT over POP on Llama-3.1-8B-Instruct, Llama-3.2-3B-Instruct, and Ministral-3-8B-Instruct-2512-BF16 (Table 2). Across all seven tasks listed in Table 1, ACSSP-B improves both mean quality and mean task-wise speedup over POP on five of seven models (Appendix Tables 9 and 10). On Qwen3-14B and Gemma-3-12B-it, ACSSP improve mean quality at lower mean speedup.

Table 2: HQA F1, RBC edit similarity (ES), and TTFT speedup (×).

	HQA		RBC	
Method	F1	×	ES	×
Llama-3.1-8B-Instruct				
Native	57.55	1.000	66.07	1.000
POP	54.09	1.394	58.56	1.404
SwiftKV	56.06	1.867	65.96	1.885
ACSSP-F	56.92	1.583	60.09	1.615
ACSSP-B	57.81	1.532	61.88	1.476
ACSSP-S	56.73	1.376	63.48	1.394
Llama-3.2-3B-Instruct				
Native	53.09	1.000	62.79	1.000
POP	45.76	1.377	55.42	1.396
ACSSP-F	47.26	1.589	56.24	1.477
ACSSP-B	52.99	1.510	62.12	1.554
ACSSP-S	52.51	1.277	61.88	1.314
Qwen3-14B				
Native	63.53	1.000	70.46	1.000
POP	59.05	1.416	66.17	1.423
ACSSP-B	57.73	1.429	68.71	1.381
ACSSP-S	61.95	1.326	70.48	1.338
Ministral-3-8B-Instruct-2512-BF16				
Native	61.28	1.000	69.86	1.000
POP	47.32	1.392	52.62	1.407
ACSSP-B	53.65	1.496	59.51	1.453
ACSSP-S	57.64	1.295	62.59	1.315

Long-context QA. On adapted RULER QA_2 at 32K/128K, ACSSP improves accuracy by up to 19.00 percentage points and achieves up to $1.32\times$ TTFT speedup over POP across evaluated models and profiles (Appendix Table 12). At 128K, ACSSP-B improves both reported metrics on five models and matches POP's accuracy with higher speed on Llama-3.1-8B-Instruct.

5.3 Mechanism Analyses and Ablations

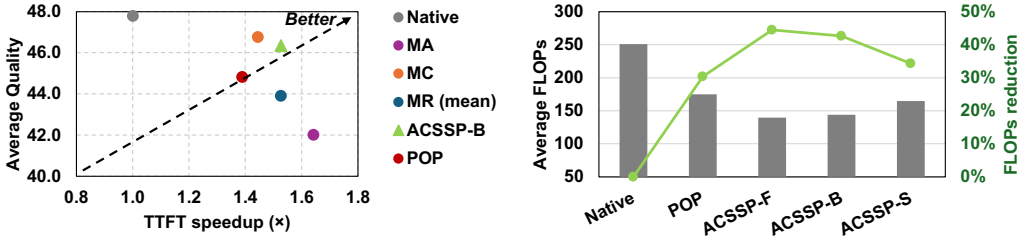

Figure 4: Llama-3.1-8B-Instruct seven-task analysis. **Left:** mean quality versus mean task-wise TTFT speedup. **Right:** mean prefill matmul FLOPs (bars, TFLOPs/request) and reduction relative to Native (line), including mapping and verification.

Verification-guided routing. Figure 4 (left) compares ACSSP-B with aggressive-only (MA), conservative-only (MC), and random routing (MR). At matched per-request aggressive counts, ACSSP-B gains 2.44 quality points over MR (46.35 versus 43.91) at the same rounded mean speedup ($1.526\times$). It recovers 4.33 points over MA and raises speedup over MC from $1.444\times$ to $1.526\times$ for a 0.42-point quality decrease. ACSSP-B also improves both averages over POP. These controls support verification-guided token selection (Appendix Tables 13 and 14).

Compute savings after overhead. Figure 4 (right) shows mean FLOPs reductions of 44.42%/42.59%/34.32% for ACSSP-F/B/S relative to Native, versus POP's 30.36%. ACSSP-F reduces POP's 174.791 TFLOPs/request to 139.495, saving 20.19% after including mapping and verification matmuls. Appendix C.6 details the accounting and operator breakdown.

Table 3: Effects of low-rank residual KV mapping and verification-guided routing on Llama-3.1-8B-Instruct. We report seven-task quality and RULER TTFT speedup ($\times$) over Native at 32K/128K.

Method	NQA	QAS	MFQA	HQA	2Wiki	MSQ	RBC	Avg.	32K	128K
Native	27.54	44.73	56.08	57.55	49.63	33.01	66.07	47.80	1.000	1.000
POP-m_{22}	27.90	43.19	53.93	54.09	47.13	29.02	58.56	44.83	1.414	1.439
POP-W24-Native	25.84	44.32	55.08	57.06	49.45	30.14	57.28	45.60	1.244	1.223
MAP-m_{22}	29.56	44.46	54.41	55.85	50.77	30.40	63.79	47.03	1.379	1.424

Low-rank residual KV mapping and verification. At anchor 22, low-rank residual KV mapping improves all seven task scores over POP, raising mean quality from 44.83 to 47.03 and RBC by 5.23 points (Table 3). With frozen projections, POP-W24-Native improves five task scores and raises the mean to 45.60, including gains of 2.97 on HQA and 2.32 on 2Wiki. Both learned construction and verification-guided continuation improve quality, with different costs. MAP retains $1.424\times$ RULER speedup at 128K, whereas POP-W24-Native achieves $1.223\times$ and lowers RBC quality. ACSSP combines learned maps with selective continuation to offer a faster operating point.

6 Conclusion

SSP formalizes self-speculative prefill through construction, verification, routing, and reusable fallback, assessing cache drafts before acceptance and selectively allocating further computation. ACSSP makes this framework practical through low-rank residual KV maps and token-level verification and routing, preserving original model weights and full cache coverage. Across seven models, it improves mean quality over POP, with measured latency and FLOP reductions demonstrating opportunities for better quality-speed trade-offs. These results establish SSP as a principled foundation for efficient long-context inference through self-speculative prefill.

AI USE STATEMENT

We used generative AI tools to assist with writing and polishing the manuscript, retrieving and discovering relevant literature, and developing proofs of mathematical claims. Writing assistance focused on clarity, organization, and consistency of terminology. Retrieval assistance supported the identification of related work. The authors checked the cited sources and their relevance. Mathematical assistance contributed to drafting the proofs of Proposition 3 and Proposition 4 in Appendices A.2 and A.4. The authors independently re-derived these proofs and checked each step against the stated assumptions. We also used AI tools to assist with code editing and debugging, with the resulting code tested against experimental records.

The authors reviewed and revised all AI-assisted content, verified references and technical claims against the original sources and experimental results, and take full responsibility for the accuracy, originality, and integrity of the final manuscript.

ETHICS STATEMENT

This work studies efficient inference using pretrained language models and established public benchmarks, without recruiting human participants or collecting new user data. Public availability does not imply that the underlying corpora are free of personal information, copyrighted material, or harmful content. Use of these resources remains subject to their applicable licenses and access conditions. Prefill acceleration can lower inference costs and broaden access, but can also facilitate misuse of the underlying models. Freezing the original model weights does not remove inherited biases or guarantee unchanged safety behavior, because approximate KV caches can alter generated outputs. Our evaluations measure task quality and latency rather than safety, fairness, or suitability for high-stakes deployment. Such uses require separate evaluation of both the base model and the accelerated inference path.

REPRODUCIBILITY STATEMENT

Our experiments use publicly available pretrained models and the PG-19, LongBench, RepoBench, and RULER datasets or benchmark generators. Section 5.1 summarizes the experimental setup, and Appendix C.1 specifies model variants, data selection, prompt processing, decoding, and scoring. Appendix C.2 details map fitting, adaptation budgets, profile settings, and threshold calibration. Appendix C.3 describes hardware, runtime settings, warmup, repetition, and timing aggregation. Quality and timing use the same frozen configuration within each evaluated variant. Appendices C.4 and C.5 provide complete result tables, fixed-depth comparisons, component controls, and routing-seed statistics. Theoretical assumptions and proofs appear in Appendices A and B.

REFERENCES

Saleh Ashkboos, Maximilian Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. SliceGPT: Compress Large Language Models by Deleting Rows and Columns. In *International Conference on Learning Representations*, volume 2024, pp. 11682–11701, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/316648eb8b4ffb6010f531b07848c300-Paper-Conference.pdf.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-vl technical report, 2025. URL https://arxiv.org/abs/2511.21631.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench: A Bilingual, Multitask Benchmark for Long Context Understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3119–3137, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.172. URL https://aclanthology.org/2024.acl-long.172/.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901, 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Yujie Chen, Tailai Chen, Yifeng Gao, Zoe Wanying He, Yijue Xu, Shaobo Wang, and Linfeng Zhang. Stability Implies Redundancy: Delta Attention Selective Halting for Efficient Long-Context Prefilling. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 26830–26848, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.1235. URL https://aclanthology.org/2026.acl-long.1235/.

DeepSeek-AI, :, Anyi Xu, B. Li, Bangcai Lin, Bing Xue, BingCheng Xian, Bingzheng Xu, Bochao Wu, Bowei Zhang, Boyi Deng, C. C. Yu, Chao Jin, Chaofan Lin, Chen Dong, Chenbing Wang, Chenfan Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chengyuan Zhang, Chenhao Xu, Chenqi Zhao, Chenze Shao, Chuhao Wang, Chuqi Zhang, Damai Dai, Dejian Yang, Deli Chen, Di Huang, Di Wu, Donghao Li, Erhang Li, Eric Fu, F. Zhou, Fangwei Zhou, Fangyun Lin, Fangzhou Yuan, Feiyu Xia, Fucong Dai, Guangbo Hao, Guanglin Li, Guanting Chen, Guoai Cao, Guofan Fan, Guolai Meng, Guowei Li, Haichuan Zhang, Haiyang Ma, Haiyang Shen, Han Li, Han Yu, Han Zhang, Hangyuan Deng, Hanwei Xu, Hanxiang Xu, Hanxun Zhong, Hao Guo, Hao Jiang, Hao Li, Hao Qin, Haodong Wen, Haofen Liang, Haofeng Huang, Haohua Liu, Haoling Zhang, Haoming Luo, Haoran Yang, Haotian Xu, Haotian Yuan, Haoting Huang, Haowen Luo, Haoyang Cai, Haoyu Chen, Haozhe Ji, Hengran Zhang, Hengrui Wang, Hengxu Wu, Honghui Ding, Hongxuan Tang, Huadong Wang, Huanqi Cao, Huazuo Gao, Hui Qu, Hui Zeng, J. Yang, J. H. Jin, J. H. Zhang, J. X. Zou, Jia Yu, Jiahui Zhou, Jiajun Chen, Jialiang Huang, Jialin Zhao, Jiamin Tang, Jian Zhou, Jianan Tong, Jianwen Li, Jiaqi Zhu, Jiarui Wang, Jiasheng Ye, Jiashi Li, Jiaxin Xu, Jiaying Ding, Jibai Lu, Jiewen Hu, Jin Yan, Jincheng Zhai, Jingchang Chen, Jingcheng Hu, Jingli Zhou, Jingsheng Xu, Jingting Xiang, Jingyan Yun, Jingyang Yuan, Jingyuan Cheng, Jinhua Zhu, Jinpeng Wang, Jinyi Chen, Jinyi Hu, Jiping Yu, Jueliang Guo, Junbo Pei, Junbo Sun, Junguang Jiang, Junjie Qiu, Junkang Zhou, Junqi Liu, Junren Li, Junxian Li, Junxiao Song, Junyi Guo, Kai Dong, Kaifeng Chen, Kaige Gao, Kang Guan, Kangdong Yuan, Ke Hong, Ke Xu, Kefan Zhao, Kexin Ji, Kexin Zhang, Kexing Zhou, Kuai Yu, Lan Zhang, Lean Wang, Lecong Zhang, Lei Wang, Letian Gao, Liang Zhao, Liansheng Xu, Lihua Guo, Lingxiao Luo, Lingyue Fu, Litao Deng, Litong Wang, Liyue Zhang, Longhao Chen, Lu Chen, Luotian Huang, Luyao Ma, Luyao Wang, M. S. Di, Max Mei, Menghao Ye, Miao Cui, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingjing Zhang, Mingqi Wei, Mingshu Chen, Mingxing Liu, Mingxu Zhou, Mingyu Xu, Mingyu Yang, Mingze Wang, Muyang Chen, Ni Shentu, Ning Wang, Niufang Ning, Panpan Huang, Peixin Cong, Peiyi Wang, Peiyuan Xin, Pengfei Ren, Pengfei Yan, Pengle Zhang, Qi Kang, Qi Tang, Qiancheng Wang, Qiang Li, Qihao Zhu, Qingyang Li, Qinyu Chen, Qiushi Du, Qizhou Guo, Rongxian Xu, Rui Ding, Rui Hu, Rui Tian, Rui Yu, Ruidong Zhu, Ruifan Xu, Ruihan Yang, Ruihang Xia, Ruijie Lu, Ruilin Geng, Ruipeng Hong, Ruiqi Ge, Ruisong Zhang, Ruize Sun, Ruizhe Pan, Runji Wang, Runqian Chen, Runxin Xu, Ruohong Tian, Ruomeng Shen, Ruoyu Zhang, Ryan X., S. H. Liu, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaofei Cai, Shaoheng Nie, Shaoyuan Chen, Shengding Hu, Shengkai Lin, Shengwen Ran, Shengyu Liu, Shengyuan Jia, Shi Bai, Shi Feng, Shicheng Xu, Shichun Liu, Shiqiang Hu, Shirong Ma, Shiyu Wang, Shiyuan Feng, Shufan Gong, Shuhan Lin, Shuiping Yu, Shunfeng Zhou, Shuo Yang, Shuomeng Wang, Shuting Guo, Shuting Pan, Shuying Yu, Sinuo Cao, Siyi Lin, Sizhe Chen, Songyang Chen, Songyang Zhou, Tao Ni, Tao Yun, Tian Jin, Tian Pei, Tian Ye, Tianle Lin, Tianran Ji, Tianyi Cui, Tianyuan Yue, Tingting Yu,

Tongrui Xiong, Wangding Zeng, Wei Liu, Wei Zhang, Weibin Xu, Weihao Zeng, Weilin Zhao, Wen Liu, Wenfeng Liang, Wenjie Pang, Wenjing Luo, Wenjing Yao, Wenjun Gao, Wenkai Shao, Wenkai Yang, Wenli Zhang, Wenlu Wang, Wenlve Huang, Wenqian Yan, Wentao Zhang, Xi Gao, Xiang He, Xiang Li, Xiangli Li, Xiangwen Wang, Xiangying Zhang, Xiankui Wei, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojian Qu, Xiaokang Chen, Xiaokang Zhang, Xiaotao Nie, Xiaoyao Zou, Xiaoyuan Li, Xicheng Guo, Xieting Chu, Xin Cheng, Xin Liu, Xin Xie, Xinbo Xu, Xingchao Liu, Xingchen Liu, Xingkai Yu, Xingyou Li, Xintong Yao, Xinyang Chen, Xinyong Jiang, Xinyu Yang, Xinyu Yang, Xu Chen, Xuanyu Wang, Xubei Zhong, Xuecheng Su, Xuejie Liu, Xuheng Lin, Xujie Fan, Xuncheng Zhao, Xuwei Fu, Y. C. Yan, Y. H. Jiang, Y. T. Wu, Y. W. M., Y. Z. Wang, Yafei Gao, Yang Yang, Yang Zhang, Yanru Ma, Yanwen Huang, Yao Li, Yao Li, Yao Meng, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yaoyang Ye, Yehang Yin, Yexinrui Wu, Yi Qian, Yi Tao, Yi Yu, Yichao Zhang, Yichen Jiang, Yicheng Wang, Yifan Ding, Yifan Shi, Yifeng Peng, Yifeng Zhai, Yijia Wu, Yiliang Xiong, Yilun Wang, Ying He, Ying Zhou, Yingjia Luo, Yinmin Zhong, Yiping Wang, Yisong Wang, Yixiang Zhang, Yixiao Chen, Yixuan Tan, Yixuan Wei, Yiyang Ma, Yiyao Yang, Yiyuan Liu, Yizai Cai, Yizhen Wei, Yizhi Wang, Yonglun Yang, Yongqi Zhuo, Yongqiang Guo, Yongtong Wu, Yu Wu, Yu Zhang, Yuan Bian, Yuan Cheng, Yuan Ou, Yuan Sun, Yuanfan Xu, Yuanhang Sun, Yuanhao Li, Yuchen Liu, Yuchen Yao, Yudong Han, Yuduan Wang, Yuhan Wu, Yuhao Meng, Yuheng Zou, YuKun Li, Yunchuan Wang, Yunfan Xiao, Yunfan Xiong, Yupeng Chen, Yuqian Cao, Yuqian Wang, Yuqing Chen, Yushun Zhang, Yutong Lin, Yuwei Xiao, Yuxian Gu, Yuxiang Chen, Yuxiang Huang, Yuxiang Luo, Yuxiang You, Yuxin Chen, Yuxin Xiang, Yuxuan Liu, Yuxuan Zhou, Yuyang Zhou, Yuzhe Guo, Yuzhen Huang, Yuzhuo Bai, Z. Y. Z., Zanlin Ni, Zehao Wang, Zehua Zhao, Zehui Ren, Zejun Zhao, Zhangli Sha, Zhanying Wang, Zhaochen Zhang, Zhaoshuai Du, Zhe Fu, Zhean Xu, Zhenda Xie, Zheng Liu, Zhengyan Zhang, Zhenhua Dong, Zhewen Hao, Zhibang Wang, Zhibin Gou, Zhicheng Ma, Zhihao Li, Zhihong Shao, Zhihuan Huang, Zhijie Li, Zhirui Lu, Zhixian Huang, Zhixuan Chen, Zhixuan Chen, Zhixuan Pan, Zhiyu Wu, Zhizhou Ren, Zhu He, Zhuoshu Li, Zhuping Zhang, Zian Xu, Zihao Wang, Zihui Gu, Zijia Zhu, Zili Zhang, Zilin Li, Zilong Hou, Zilong Lyu, Ziqiao Wang, Ziwei Xie, Ziya Zhang, Ziyi Gao, Zizheng Pan, Zonglin Li, Zongqing Yao, Zui Chen, Zuofan Wu, Chenchen Ling, Chengyu Hou, Chong Chen, D. Li, Di Qi, Dongjie Ji, Fang Wei, Fanyi Xia, Fei Xie, Feiyi Tan, Hailong Guo, Haiyan Zhai, Hui Zhou, Huihui Tan, Huijie Li, Jia Luo, Jia Song, Jialu Cai, Jian Liang, Jiangting Zhou, Jiaqi Gao, Jiayi Shao, Jie Chen, Jieyu Yang, Jin Chen, Jingde Zhang, Jingzi Zhou, Jinqian Wang, Jinyang Liu, JinZhao Sun, Junhua Ling, Junmin Zheng, Kaicheng Yang, Ke Xu, Le Su, Leyi Xia, Liangfeng Ding, Lin Zhuo, Linwang Ma, Linyan Zhu, Liyu Cai, Luqi Yao, M. K. Zhang, Meng Li, Miao Lin, Miaojun Wang, Min Zhang, Mingming Li, Mingming Wang, Mingze Yin, Minmin Han, Nan Cao, Ning Wang, Ningxin Ma, Panpan Wang, Peihan Lin, Peng Sun, Peng Zhang, Qian Ying, Qiang Xiang, Qiao Wang, Qingmiao Mao, Qiwei Jiang, Rongli Jin, Ruyi Chen, Sha Tao, Shangmian Sun, Shaoqing Wu, Shichao Zou, Si Lei, Tianyang Zhang, Tianyu Sun, Tingting Yin, W. L. Xiao, Wei An, Wei Li, Wei Wang, Weiwei Lin, Wenqing Hou, X. Lin, Xiangfei Meng, Xianzhu Huang, Xiao Peng, Xiaoqian Li, Xiaoting Zhang, Xiaowen Sun, Xiaoxiang Wang, Xiaoyu Ye, Xinrou Zhang, Xinyu Zhang, Xue Cao, Xueyin Chen, Yanan Zhou, Yanhong Xu, Yao Xia, Yao Xu, Yi Shao, Yihong Zhang, Yiling Ma, Ying Tang, Yining Lou, Yiru Chen, Yishi Piao, Yixuan Chen, Yong Xiong, Yuchen Xuan, Yuehan Yang, Yuer Xu, Yukun Zha, Yunxian Ma, Yuping Lin, Yuting Yan, Yutong Xie, Yuwen Sheng, Yuxuan Zhu, Zekai Zhang, Zhe Ju, Zhenzhen Lin, Zheren Gao, Zheyang Sun, Zhigang Yan, Zhongyu Wu, Zi Wang, Zihua Qu, Ziling Yan, and Ziyi Wan. DeepSeek-V4.1-Flash: Pushing the Limits of KV Cache Compression, 2026. URL https://arxiv.org/abs/2609.19969.

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, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.681. URL https://aclanthology.org/2024.acl-long.681/.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak,

Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle

Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The Llama 3 Herd of Models, 2024. URL https://arxiv.org/abs/2407.21783.

Junhui He, Zhihui Fu, Jun Wang, and Qingan Li. POP: Prefill-Only Pruning for Efficient Large Model Inference. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 5864–5877, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.291. URL https://aclanthology.org/2026.findings-acl.291/.

Shwai He, Tao Ge, Guoheng Sun, Bowei Tian, Xiaoyang Wang, and Dong Yu. Router-Tuning: A Simple and Effective Approach for Dynamic Depth. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 1925–1938, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.99. URL https://aclanthology.org/2025.emnlp-main.99/.

Maxwell Horton, Qingqing Cao, Chenfan Sun, Yanzi Jin, Sachin Mehta, Mohammad Rastegari, and Moin Nabi. KV Prediction for Improved Time to First Token. In *First Workshop on Scalable Optimization for Efficient and Adaptive Foundation Models*, 2025. URL https://openreview.net/forum?id=Bni1wy49vY.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. RULER: What's the Real Context Size of Your Long-Context Language Models? In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=kIoBbc76Sy.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. LLMLingua: Compressing Prompts for Accelerated Inference of Large Language Models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 13358–13376, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.825. URL https://aclanthology.org/2023.emnlp-main.825/.

Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H. Abdi, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. MInference 1.0: Accelerating Pre-filling for Long-Context LLMs via Dynamic Sparse Attention. In *Advances in Neural Information Processing Systems*, volume 37, pp. 52481–52515, 2024a. doi: 10.52202/

079017-1663. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/5dfbe6f5671e82c76841ba687a8a9ecb-Paper-Conference.pdf.

Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. LongLLMLingua: Accelerating and Enhancing LLMs in Long Context Scenarios via Prompt Compression. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1658–1677, Bangkok, Thailand, August 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.91. URL https://aclanthology.org/2024.acl-long.91/.

Dongwon Jo, Jiwon Song, Yulhwa Kim, and Jae-Joon Kim. FastKV: Decoupling of Context Reduction and KV Cache Compression for Prefill-Decoding Acceleration. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 32167–32186, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.1610. URL https://aclanthology.org/2026.findings-acl.1610/.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, pp. 611–626. Association for Computing Machinery, 2023. ISBN 9798400702297. doi: 10.1145/3600006.3613165. URL https://doi.org/10.1145/3600006.3613165.

Xunhao Lai, Jianqiao Lu, Yao Luo, Yiyuan Ma, and Xun Zhou. FlexPrefill: A Context-Aware Sparse Attention Mechanism for Efficient Long-Sequence Inference. In *International Conference on Learning Representations*, volume 2025, pp. 963–989, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/03645743ea35690f30d795d6bac149a5-Paper-Conference.pdf.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast Inference from Transformers via Speculative Decoding. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 19274–19286. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/leviathan23a.html.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. SnapKV: LLM Knows What You are Looking for Before Generation. In *Advances in Neural Information Processing Systems*, volume 37, pp. 22947–22970, 2024. doi: 10.52202/079017-0722. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/28ab418242603e0f7323e54185d19bde-Paper-Conference.pdf.

Alexander H. Liu, Kartik Khandelwal, Sandeep Subramanian, Victor Jouault, Abhinav Rastogi, Adrien Sadé, Alan Jeffares, Albert Jiang, Alexandre Cahill, Alexandre Gavaudan, Alexandre Sablayrolles, Amélie Héliou, Amos You, Andy Ehrenberg, Andy Lo, Anton Eliseev, Antonia Calvi, Avinash Sooriyarachchi, Baptiste Bout, Baptiste Rozière, Baudouin De Monicault, Clémence Lanfranchi, Corentin Barreau, Cyprien Courtot, Daniele Grattarola, Darius Dabert, Diego de las Casas, Elliot Chane-Sane, Faruk Ahmed, Gabrielle Berrada, Gaëtan Ecrepont, Gauthier Guinet, Georgii Novikov, Guillaume Kunsch, Guillaume Lample, Guillaume Martin, Gunshi Gupta, Jan Ludziejewski, Jason Rute, Joachim Studnia, Jonas Amar, Joséphine Delas, Josselin Somerville Roberts, Karmesh Yadav, Khyathi Chandu, Kush Jain, Laurence Aitchison, Laurent Fainsin, Léonard Blier, Lingxiao Zhao, Louis Martin, Lucile Saulnier, Luyu Gao, Maarten Buyl, Margaret Jennings, Marie Pellat, Mark Prins, Mathieu Poirée, Mathilde Guillaumin, Matthieu Dinot, Matthieu Futeral, Maxime Darrin, Maximilian Augustin, Mia Chiquier, Michel Schimpf, Nathan Grinsztajn, Neha Gupta, Nikhil Raghuraman, Olivier Bousquet, Olivier Duchenne, Patricia Wang, Patrick von Platen, Paul Jacob, Paul Wambergue, Paula Kurylowicz, Pavankumar Reddy Muddireddy, Philomène Chagniot, Pierre Stock, Pravesh Agrawal, Quentin Torroba, Romain Sauvestre, Roman Soletskyi, Rupert Menneer, Sagar Vaze, Samuel Barry, Sanchit Gandhi, Siddhant Waghjale, Siddharth Gandhi, Soham Ghosh, Srijan Mishra, Sumukh Aithal, Szymon Antoniak, Teven Le Scao, Théo Cachet, Theo Simon Sorg, Thibaut Lavril, Thiziri Nait Saada, Thomas Chabal, Thomas Foubert, Thomas Robert, Thomas Wang, Tim Lawson, Tom Bewley, Tom Bewley, Tom Edwards, Umar Jamil, Umberto Tomasini, Valeriia Nemychnikova, Van Phung, Vincent Maladière,

Virgile Richard, Wassim Bouaziz, Wen-Ding Li, William Marshall, Xinghui Li, Xinyu Yang, Yassine El Ouahidi, Yihan Wang, Yunhao Tang, and Zaccharie Ramzi. Ministral 3, 2026. URL https://arxiv.org/abs/2601.08584.

Jingyu Liu, Beidi Chen, and Ce Zhang. Speculative Prefill: Turbocharging TTFT with Lightweight and Training-Free Token Importance Estimation. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 38188–38209. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/liu25g.html.

Tianyang Liu, Canwen Xu, and Julian McAuley. RepoBench: Benchmarking Repository-Level Code Auto-Completion Systems. In *International Conference on Learning Representations*, volume 2024, pp. 47832–47850, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/d191ba4c8923ed8fd8935b7c98658b5f-Paper-Conference.pdf.

Xin Men, Mingyu Xu, Qingyu Zhang, Qianhao Yuan, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. ShortGPT: Layers in Large Language Models are More Redundant Than You Expect. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 20192–20204, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.1035. URL https://aclanthology.org/2025.findings-acl.1035/.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Dongmei Zhang. LLMLingua-2: Data Distillation for Efficient and Faithful Task-Agnostic Prompt Compression. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 963–981, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.57. URL https://aclanthology.org/2024.findings-acl.57/.

Dan Peng, Zhihui Fu, Zewen Ye, Zhuoran Song, and Jun Wang. Accelerating Prefilling for Long-Context LLMs via Sparse Pattern Sharing, 2025. URL https://arxiv.org/abs/2505.19578.

Aurick Qiao, Zhewei Yao, Samyam Rajbhandari, and Yuxiong He. SwiftKV: Fast Prefill-Optimized Inference with Knowledge-Preserving Model Transformation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 25734–25753, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1306. URL https://aclanthology.org/2025.emnlp-main.1306/.

Ranajoy Sadhukhan, Jian Chen, Zhuoming Chen, Vashisth Tiwari, Ruihang Lai, Jinyuan Shi, Ian Yen, Avner May, Tianqi Chen, and Beidi Chen. MagicDec: Breaking the Latency-Throughput Tradeoff for Long Context Generation with Speculative Decoding. In *International Conference on Learning Representations*, volume 2025, pp. 6835–6850, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/13f972adf12bdf886583d48cd528002f-Paper-Conference.pdf.

Mingjie Sun, Zhuang Liu, Anna Bair, and Zico Kolter. A Simple and Effective Pruning Approach for Large Language Models. In *International Conference on Learning Representations*, volume 2024, pp. 4942–4964, 2024a. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/14c856c7a41297804de4c4890e846b25-Paper-Conference.pdf.

Yutao Sun, Li Dong, Yi Zhu, Shaohan Huang, Wenhui Wang, Shuming Ma, Quanlu Zhang, Jianyong Wang, and Furu Wei. You Only Cache Once: Decoder-Decoder Architectures for Language Models. In *Advances in Neural Information Processing Systems*, volume 37, pp. 7339–7361, 2024b. doi: 10.52202/079017-0235. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/0df38cd13520747e1e64e5b123a78ef8-Paper-Conference.pdf.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, Gaël Liu, Francesco Visin, Kathleen Kenealy, Lucas Beyer, Xiaohai Zhai, Anton Tsitsulin, Robert Busa-Fekete, Alex Feng, Noveen Sachdeva, Benjamin Coleman, Yi Gao, Basil Mustafa, Iain Barr, Emilio Parisotto, David Tian, Matan Eyal, Colin Cherry, Jan-Thorsten Peter, Danila Sinopalnikov, Surya Bhupatiraju, Rishabh Agarwal, Mehran Kazemi, Dan Malkin, Ravin Kumar, David Vilar, Idan Brusilovsky, Jiaming Luo, Andreas Steiner, Abe Friesen, Abhanshu Sharma, Abheesht Sharma, Adi Mayrav Gilady, Adrian Goedeckemeyer, Alaa Saade, Alex Feng, Alexander Kolesnikov, Alexei Bendebury, Alvin Abdagic, Amit Vadi, András György, André Susano Pinto, Anil Das, Ankur Bapna, Antoine Miech, Antoine Yang, Antonia Paterson, Ashish Shenoy, Ayan Chakrabarti, Bilal Piot, Bo Wu, Bobak Shahriari, Bryce Petrini, Charlie Chen, Charline Le Lan, Christopher A. Choquette-Choo, CJ Carey, Cormac Brick, Daniel Deutsch, Danielle Eisenbud, Dee Cattle, Derek Cheng, Dimitris Paparas, Divyashree Shivakumar Sreepathihalli, Doug Reid, Dustin Tran, Dustin Zelle, Eric Noland, Erwin Huizenga, Eugene Kharitonov, Frederick Liu, Gagik Amirkhanyan, Glenn Cameron, Hadi Hashemi, Hanna Klimczak-Plucińska, Harman Singh, Harsh Mehta, Harshal Tushar Lehri, Hussein Hazimeh, Ian Ballantyne, Idan Szpektor, Ivan Nardini, Jean Pouget-Abadie, Jetha Chan, Joe Stanton, John Wieting, Jonathan Lai, Jordi Orbay, Joseph Fernandez, Josh Newlan, Ju yeong Ji, Jyotinder Singh, Kat Black, Kathy Yu, Kevin Hui, Kiran Vodrahalli, Klaus Greff, Linhai Qiu, Marcella Valentine, Marina Coelho, Marvin Ritter, Matt Hoffman, Matthew Watson, Mayank Chaturvedi, Michael Moynihan, Min Ma, Nabila Babar, Natasha Noy, Nathan Byrd, Nick Roy, Nikola Momchev, Nilay Chauhan, Noveen Sachdeva, Oskar Bunyan, Pankil Botarda, Paul Caron, Paul Kishan Rubenstein, Phil Culliton, Philipp Schmid, Pier Giuseppe Sessa, Pingmei Xu, Piotr Stanczyk, Pouya Tafti, Rakesh Shivanna, Renjie Wu, Renke Pan, Reza Rokni, Rob Willoughby, Rohith Vallu, Ryan Mullins, Sammy Jerome, Sara Smoot, Sertan Girgin, Shariq Iqbal, Shashir Reddy, Shruti Sheth, Siim Põder, Sijal Bhatnagar, Sindhu Raghuram Panyam, Sivan Eiger, Susan Zhang, Tianqi Liu, Trevor Yacovone, Tyler Liechty, Uday Kalra, Utku Evci, Vedant Misra, Vincent Roseberry, Vlad Feinberg, Vlad Kolesnikov, Woohyun Han, Woosuk Kwon, Xi Chen, Yinlam Chow, Yuvein Zhu, Zichuan Wei, Zoltan Egyed, Victor Cotruta, Minh Giang, Phoebe Kirk, Anand Rao, Kat Black, Nabila Babar, Jessica Lo, Erica Moreira, Luiz Gustavo Martins, Omar Sanseviero, Lucas Gonzalez, Zach Gleicher, Tris Warkentin, Vahab Mirrokni, Evan Senter, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, Yossi Matias, D. Sculley, Slav Petrov, Noah Fiedel, Noam Shazeer, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Jean-Baptiste Alayrac, Rohan Anil, Dmitry, Lepikhin, Sebastian Borgeaud, Olivier Bachem, Armand Joulin, Alek Andreev, Cassidy Hardin, Robert Dadashi, and Léonard Hussenot. Gemma 3 technical report, 2025. URL https://arxiv.org/abs/2503.19786.

Shubhangi Upasani, Ravi Shanker Raju, Bo Li, Mengmeng Ji, John Long, Chen Wu, Urmish Thakker, and Guangtao Wang. Cross-Family Speculative Prefill: Training-Free Long-Context Compression with Small Draft Models, 2026. URL https://arxiv.org/abs/2603.02631.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Jianyu Wei, Yizhao Gao, Qihao Zhang, Shimao Chen, Zhengju Tang, Yu Cheng, Shengjie Zhou, Zihan Jiang, Yifan Song, Hailin Zhang, Liang Zhao, Bo Yang, Gang Wang, Shijie Cao, and Fuli Luo. HySparse2: Hybrid Sparse Attention with Two-Level KV Sharing, 2026. URL https://arxiv.org/abs/2609.26368.

Guangxuan Xiao, Jiaming Tang, Jingwei Zuo, Junxian Guo, Shang Yang, Haotian Tang, Yao Fu, and Song Han. DuoAttention: Efficient Long-Context LLM Inference with Retrieval and Streaming Heads. In *International Conference on Learning Representations*, volume 2025, pp. 37228–37253, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/5c1ddd2e59df46fd2aa85c833b1b36ed-Paper-Conference.pdf.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu,

Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 Technical Report, 2025. URL https://arxiv.org/abs/2505.09388.

Jun Zhang, Jue Wang, Huan Li, Lidan Shou, Ke Chen, Gang Chen, and Sharad Mehrotra. Draft & Verify: Lossless Large Language Model Acceleration via Self-Speculative Decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11263–11282, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.607. URL https://aclanthology.org/2024.acl-long.607/.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang "Atlas" Wang, and Beidi Chen. H2O: Heavy-Hitter Oracle for Efficient Generative Inference of Large Language Models. In *Advances in Neural Information Processing Systems*, volume 36, pp. 34661–34710, 2023. doi: 10.52202/075280-1506. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/6ceefa7b15572587b78ecfcebb2827f8-Paper-Conference.pdf.

APPENDIX

A SSP Framework Support 20
A.1 Definitions of the Motivation Scores 20
A.2 Threshold Structure of a Separable Routing Surrogate 22
A.3 KV Construction Boundaries 23
A.4 Evidence-Based Verification Boundaries 24
B ACSSP Method Support 25
B.1 Low-Rank Residual KV Map Capacity 25
B.2 Attention Sensitivity and Verification Limitations 27
B.3 Execution Cost and Design Trade-offs 28
C Supplementary Experiments 30
C.1 Models, Data, and Metrics 30
C.2 Map Training and Profile Calibration 31
C.3 Runtime and Timing Protocol 33
C.4 Complete Quality and Efficiency Results 35
C.5 Fixed-Depth and Component Comparisons 39
C.6 Prefill FLOPs and Component Breakdown 42
D Qualitative Examples of Improvements over POP 43
D.1 HotpotQA: Answering with the Requested Entity 43
D.2 RepoBench: Completing the Evaluator Return 44
D.3 HotpotQA: Comparing Species Counts 45

A SSP FRAMEWORK SUPPORT

A.1 DEFINITIONS OF THE MOTIVATION SCORES

For an L-layer decoder, let $H_{i,m,t}$ denote token t's native hidden state after m layers on request i. Layers are numbered $l = 1, \ldots, L$ in the importance diagnostic. Anchor depths are numbered $m = 0, \ldots, L-1$. The importance and similarity diagnostics use 128 and 256 PG-19 requests per model, respectively, averaged equally over 4K/8K/16K/32K contexts. Both exclude the final prompt position, which produces the first output token.

Importance score: request-normalized token sensitivity. For request i with N_i prompt tokens, a gate $g_{l,t}$ scales both the attention and MLP residual outputs of layer l at position $t \in \{1, \ldots, N_i - 1\}$. The final prompt token is protected. Let $\mathcal{L}_i$ be the mean negative log-likelihood of a fixed 32-token continuation sampled from the native model. Evaluating all gates at one, define $A_{i,l,t} = (\partial \mathcal{L}_i / \partial g_{l,t}|_{g=1})^2$ and

$$I_{i,l} = \frac{\sum_{t=1}^{N_i-1} A_{i,l,t}}{\sum_{k=1}^{L} \sum_{t=1}^{N_i-1} A_{i,k,t}}, \qquad l = 1, \ldots, L. \tag{10}$$

For a nonzero denominator, $\sum_l I_{i,l} = 1$. The score measures a layer's share of total sensitivity within a request. Separate token gates avoid cancellation between token gradients before squaring, while a shared request-level denominator preserves their relative contributions. This refines the scalar-gate diagnostic of POP (He et al., 2026) for token-level analysis. It does not equate local sensitivity with the loss caused by removing an entire layer. The final layer's prefill-gate importance is zero under this protected-final-token protocol. That endpoint is not evidence that the layer is dispensable during full-model inference.

Similarity score. Let $H_{i,m,t}$ be the exact hidden state after m layers, hence the input to the next layer. For nonzero states and each later layer input $H_{i,j,t}$, define

$$\theta_{i,m,j,t} = \arccos\left(\operatorname{clip}\left[\frac{\langle H_{i,m,t}, H_{i,j,t} \rangle}{\|H_{i,m,t}\|_2 \|H_{i,j,t}\|_2}, -1, 1 \right] \right). \tag{11}$$

For $\mathcal{T}_i$, a set of 1,024 approximately equally spaced positions among the first $N_i - 1$ prompt tokens, our similarity score is

$$S_{i,m} = \frac{1}{|\mathcal{T}_i|} \sum_{t \in \mathcal{T}_i} \cos\left(\sqrt{\frac{1}{L-1-m} \sum_{j=m+1}^{L-1} \theta_{i,m,j,t}^2} \right), \qquad m = 0, \ldots, L-2. \tag{12}$$

SwiftKV (Qiao et al., 2025) averages cosine similarities to subsequent layer inputs. Our RMS angular aggregation gives large angular deviations more weight before taking the cosine. Equation 12 applies when at least one later layer input exists. We extend the displayed score with $S_{i,L-1} = 1$ for the final-layer endpoint. For the 32-layer Llama-3.1-8B-Instruct model, the suffix comparisons use $m = 0, \ldots, 30$ and end at $H_{i,31,t}$. The final layer output $H_{i,32,t}$ is not included. Runtime versions vary across the model runs in Figure 2.

Token-wise reconstruction error concentration. We use 128 fixed-seed PG-19 test documents excluded from map training, with 4K/8K/16K/32K prefixes from each document (512 requests per model). Each of the seven models uses its evaluation POP anchor. One Native prefill per request supplies reference states and attention. No output is generated. All nonfinal tokens are evaluated, with FP32 error reductions.

For a request of length n and POP anchor m, use zero-based suffix layers $\ell = m, \ldots, L-1$ and KV-head index g. POP constructs $\widehat{K}, \widehat{V}$ from H_m. Native supplies K, V. Define

$$D_{\ell,X} = \frac{1}{n-1} \sum_{t=1}^{n-1} \sum_{g} \|X_{\ell t g}\|_2^2 + \varepsilon, \quad X \in \{K, V\}, \quad \varepsilon = 10^{-12}, \tag{13}$$

$$d_{\ell j g} = \frac{\|\widehat{K}_{\ell j g} - K_{\ell j g}\|_2^2}{D_{\ell,K}} + \frac{\|\widehat{V}_{\ell j g} - V_{\ell j g}\|_2^2}{D_{\ell,V}}. \tag{14}$$

Each denominator is shared across tokens and heads within a request, layer, and K/V type, excluding the final token. Keys are compared after native key normalization, where applicable, and RoPE.

Let $\mathcal{Q}_8$ contain the final $\min\{8, n\}$ queries. With Native attention probabilities $A_{qj}^{(\ell,h)}$, query-to-KV-head assignment $\kappa(h)$, and visible-position count $\lambda_{\ell q}$, define

$$w_{\ell jg} = \max_{q \in \mathcal{Q}_8,\ h:\,\kappa(h)=g} \lambda_{\ell q} A_{qj}^{(\ell,h)}, \qquad e_j^{\mathrm{AW}} = \sum_{\ell=m}^{L-1} \sum_{g} w_{\ell jg} d_{\ell jg}. \tag{15}$$

Attention uses Native Q/K and causal or local masks. Masked positions have zero weight. The visible count includes the query's own position when visible. This is an attention-weighted reconstruction proxy, not attention-output error. Unlike the deployed verification score, it uses the full suffix, squared discrepancies, and a sum over layers and KV heads.

For request i, sort its $n_i - 1$ errors in descending order. With $k_i(p) = \lceil p(n_i - 1) \rceil$, define

$$C_i(p) = \frac{\sum_{r=1}^{k_i(p)} e_{i,(r)}^{\mathrm{AW}}}{\sum_{j=1}^{n_i-1} e_{ij}^{\mathrm{AW}}}, \qquad \overline{C}(p) = \frac{1}{M} \sum_{i=1}^{M} C_i(p). \tag{16}$$

The ratio is defined for positive-total-error requests. Table 4 reports request-equal averages, not pooled errors. The top 20% account for 82.91%-90.84% of weighted error. This offline concentration motivates selective computation. It does not establish downstream loss concentration or the predictive accuracy of online verification.

Table 4: Cumulative share (%) of POP's full-suffix attention-weighted normalized squared KV reconstruction error, ranked within each request and averaged equally across requests. Top-p selects the highest-error fraction p of eligible tokens.

Model	Top 10%	Top 20%	Top 30%	Top 40%	Top 50%
Llama-3.1-8B-Instruct	85.46	90.84	93.84	95.82	97.22
Llama-3.2-3B-Instruct	79.32	86.06	90.19	93.08	95.24
Qwen3-14B	76.22	84.56	89.37	92.62	94.97
Ministral-3-8B-Instruct-2512-BF16	74.27	82.91	88.06	91.61	94.22
Qwen3-4B-Instruct-2507	78.13	86.05	90.57	93.56	95.68
Qwen3-VL-8B-Instruct	75.56	84.60	89.67	92.98	95.32
Gemma-3-12B-it	78.34	87.38	92.10	94.99	96.89

A.2 Threshold Structure of a Separable Routing Surrogate

Objective. Fix a request distribution, evaluation protocol, and maximum supported prompt length. Let $\ell_\pi(x), \ell_{\mathrm{Native}}(x) \in [0,1]$ be normalized task losses and define the signed quality drop $d_\pi(x) = \ell_\pi(x) - \ell_{\mathrm{Native}}(x) \in [-1,1]$. This permits improvements to offset degradations, as in the reported average-score differences. The task mixture and within-task weights must match the evaluation protocol. For integrable execution cost $c_\pi(x)$ and resource use $m_\pi(x)$, define $C(\pi) = \mathbb{E}[c_\pi(x)]$, $R(\pi) = \mathbb{E}[d_\pi(x)]$, and $M(\pi) = \sup_{x \in \mathrm{supp}(\mathcal{D})} m_\pi(x)$. The quality-constrained objective is

$$\inf_{\pi \in \Pi} C(\pi) \quad \text{s.t.} \quad R(\pi) \le \rho, \qquad M(\pi) \le M_{\max}. \tag{17}$$

This is an evaluation objective, not an optimization problem solved by coverage calibration. If degradation without compensation is desired, replace d_π by its positive part and report that different risk explicitly. Full cache coverage fixes the number of persistent KV positions at each request length, not total memory, since saved states, drafts, and routing buffers can depend on the policy. The resource constraint therefore remains unless separately verified.

Separable surrogate. The following result concerns a two-action surrogate with additive per-unit losses, fixed unit weights, and no coupling constraints between decisions. It does not assume that ACSSP's request-level loss decomposes this way. Let $\alpha(z) \in [0,1]$ be the probability of choosing A from evidence z. Assume all such measurable policies are allowed after resource feasibility is separately established. Let $\delta_A(z), \delta_C(z)$ be integrable conditional expected losses of the two actions, independent of other route choices, and define

$$R_s(\alpha) = \mathbb{E}[\delta_C(Z) + \alpha(Z)\Delta(Z)], \qquad C_s(\alpha) = \mathbb{E}[c_C - \alpha(Z)k],$$

where $\Delta = \delta_A - \delta_C$ and $c_A < c_C$ are constant action costs, with $k = c_C - c_A > 0$.

Proposition 3 (Threshold structure of a separable relaxation)**.** *For $\lambda \ge 0$, the relaxation $\mathcal{L}_s(\alpha, \lambda) = C_s(\alpha) + \lambda(R_s(\alpha) - \rho)$ is minimized by*

$$\alpha_\lambda(z) = \begin{cases} 1, & \lambda\Delta(z) < k, \\ 0, & \lambda\Delta(z) > k, \\ \textit{any value in } [0,1], & \lambda\Delta(z) = k. \end{cases} \tag{18}$$

If $\Delta(z) = h(s(z))$ for a measurable nondecreasing h, strict acceptance and rejection are ordered by the scalar score, with arbitrary randomization on the indifference set. This set may span a score interval. If such a minimizer also satisfies $R_s(\alpha_\lambda) \le \rho$ and $\lambda(R_s(\alpha_\lambda) - \rho) = 0$, it solves the constrained surrogate $\min_\alpha C_s(\alpha)$ subject to $R_s(\alpha) \le \rho$.

Proof. Expanding the relaxation gives

$$\mathcal{L}_s(\alpha, \lambda) = c_C + \lambda(\mathbb{E}[\delta_C(Z)] - \rho) + \mathbb{E}[\alpha(Z)(\lambda\Delta(Z) - k)]. \tag{19}$$

The policy class permits independent measurable choices at each evidence value. Minimizing the affine integrand over $[0,1]$ gives Equation 18. The resulting rule is measurable. At $\lambda = 0$, it chooses A everywhere. For $\lambda > 0$, monotonicity of h orders the strict sublevel and superlevel sets. Ties permit mixing. For any risk-feasible α, feasibility, complementary slackness, and global minimization of the relaxation over all policies give

$$C_s(\alpha_\lambda) = \mathcal{L}_s(\alpha_\lambda, \lambda) \le \mathcal{L}_s(\alpha, \lambda) \le C_s(\alpha).$$

This proves the conditional constrained-optimality claim. □

An active constraint alone does not establish the existence of a suitable multiplier or a deterministic threshold solution. For example, with constant evidence, $c_A = 0$, $c_C = 1$, $\delta_A = 1$, $\delta_C = 0$, and $\rho = 1/2$, the constrained optimum mixes the two routes equally. Strong duality with attained primal and dual optima is one way to obtain a supporting multiplier. We do not assert these conditions for the coupled SSP problem.

Conditional comparative statics. At fixed Δ and $\lambda > 0$, increasing k expands the acceptance region. At fixed Δ and k, increasing λ contracts it. These are weak set inclusions, not necessarily strict threshold changes. Under a fixed quality budget, the supporting multiplier and conditional losses can change with context length, so no monotonic length-threshold relation follows. Coverage calibration controls an acceptance fraction. It neither identifies a quality budget or multiplier nor establishes the required loss monotonicity. Length-specific thresholds must be assessed through the complete request-level policy.

A.3 KV Construction Boundaries

Theorem 1 uses a fixed target shape by conditioning on prompt length and, for token-level reconstruction, the unit index. The same argument applies to a declared finite padding and masking convention. All expectations below use this conditional population. Bounds can be averaged over length or unit strata when the resulting quantities are integrable. We use the ordinary Frobenius norm. Relative KV error and attention-output error are different objectives.

Proof of the decomposition. Suppress m and let $\mu(U) = \mathbb{E}[Y \mid U]$. Square integrability makes $\mu(U)$ the L_2 orthogonal projection of Y onto the space of $\sigma(U)$-measurable random matrices. For any $g \in \mathcal{G}$, the cross term vanishes by the tower property:

$$\mathbb{E}\langle Y - \mu(U), \mu(U) - g(U)\rangle_F = \mathbb{E}\langle \mathbb{E}[Y - \mu(U) \mid U], \mu(U) - g(U)\rangle_F = 0. \tag{20}$$

Expanding the square therefore gives

$$\mathbb{E}\|Y - g(U)\|_F^2 = \mathbb{E}\|Y - \mu(U)\|_F^2 + \mathbb{E}\|\mu(U) - g(U)\|_F^2. \tag{21}$$

Writing $y = \mathrm{vec}(Y)$, the trace of its conditional covariance equals $\mathbb{E}[\|Y - \mu(U)\|_F^2 \mid U]$. Taking expectations identifies the first term as $\mathcal{E}_{\mathrm{info}}$. It is independent of g, so taking the infimum over $\mathcal{G}$ proves Equation 2. Nonnegativity of the second term gives the lower bound. The optimal value equals the information floor precisely when $\mu(U)$ belongs to the L_2 closure of $\{g(U) : g \in \mathcal{G}\}$. A minimizing constructor need not exist.

Proof of the conditional upper bound. Define $\|Z\|_{L_2(F)} = (\mathbb{E}\|Z\|_F^2)^{1/2}$. For the feasible g and ψ in the theorem, Minkowski's inequality and the assumed Lipschitz property give

$$\begin{aligned}(\mathcal{E}_{\mathcal{G}}^{\star})^{1/2} &\le \|f(H) - g(U)\|_{L_2(F)} \\ &\le \|f(H) - f(\psi(U))\|_{L_2(F)} + \|f(\psi(U)) - g(U)\|_{L_2(F)} \\ &\le L_f\|H - \psi(U)\|_{L_2(F)} + \varepsilon_{\mathcal{G}} \le L_f\varepsilon_U + \varepsilon_{\mathcal{G}}.\end{aligned} \tag{22}$$

Squaring and combining with the lower bound proves Equation 3. The Lipschitz inequality must hold for the native and reconstructed states involved. For token-row targets, f still acts on the full prompt state H. The upper bound requires $\psi(U)$ to reconstruct that state, not merely the observed row. No orthogonality between these two residuals is assumed, so their root-mean-square errors add. □

Interpretation. The information floor is zero exactly when Y is determined by U almost surely. Restricting a constructor to one hidden-state row does not by itself prove a positive floor, since that row might already identify the target on the relevant population. Limited linear or low-rank capacity can still cause error when this floor is zero. The upper bound requires the declared reconstruction and approximation errors. It does not supply numerical values for them or a downstream quality guarantee.

A.4 Evidence-Based Verification Boundaries

Proof of the Bayes boundary. Let $A = \{z : f_{\mathrm{ver}}(z) = 1\}$. Conditioning on S gives

$$R_{\mathrm{ver}}(f_{\mathrm{ver}}) = \int [\nu_0 p_0(z)\mathbf{1}_A(z) + \nu_1 p_1(z)\mathbf{1}_{A^c}(z)]\, d\mu(z). \tag{23}$$

The measurable set $A^\star = \{z : \nu_1 p_1(z) \geq \nu_0 p_0(z)\}$ minimizes the integrand pointwise, proving Equation 4. With equal priors, $\min(a,b) = (a+b-|a-b|)/2$ and $\int p_s\, d\mu = 1$ yield

$$\epsilon_Z^\star = \tfrac{1}{2}\int \min(p_0, p_1)\, d\mu = \tfrac{1}{2}\left(1 - \tfrac{1}{2}\int |p_0 - p_1|\, d\mu\right) = \tfrac{1}{2}(1 - \mathrm{TV}(P_0, P_1)). \tag{24}$$

Proof of the ERM bound. Work on the event where $\sup_{f_{\mathrm{ver}}\in\mathcal{V}} |R_{\mathrm{ver}}(f_{\mathrm{ver}}) - \widehat{R}_{\mathrm{ver}}(f_{\mathrm{ver}})| \leq \Gamma_n$. For any $\eta > 0$, select $f_{\mathrm{ver},\eta}$ with population error at most $\inf_{f_{\mathrm{ver}}\in\mathcal{V}} R_{\mathrm{ver}}(f_{\mathrm{ver}}) + \eta$. Empirical optimality gives

$$\begin{aligned} R_{\mathrm{ver}}(\widehat{f}_{\mathrm{ver}}) &\leq \widehat{R}_{\mathrm{ver}}(\widehat{f}_{\mathrm{ver}}) + \Gamma_n \leq \widehat{R}_{\mathrm{ver}}(f_{\mathrm{ver},\eta}) + \Gamma_n \\ &\leq R_{\mathrm{ver}}(f_{\mathrm{ver},\eta}) + 2\Gamma_n \leq \epsilon_Z^\star + \epsilon_{\mathrm{app}}(\mathcal{V}, Z) + \eta + 2\Gamma_n. \end{aligned} \tag{25}$$

Let $\eta \downarrow 0$. The event has probability at least $1-\delta$, and the lower bound follows because $\widehat{f}_{\mathrm{ver}}$ is a measurable verifier. This proves Theorem 2. □

A finite verifier family. For a nonempty finite class fixed independently of n i.i.d. labeled instances, Hoeffding's inequality and a union bound give

$$\Pr\left(\sup_{f_{\mathrm{ver}}\in\mathcal{V}} |R_{\mathrm{ver}}(f_{\mathrm{ver}}) - \widehat{R}_{\mathrm{ver}}(f_{\mathrm{ver}})| > t\right) \leq 2|\mathcal{V}|e^{-2nt^2}. \tag{26}$$

Setting the right side to δ gives $\Gamma_n = \sqrt{\log(2|\mathcal{V}|/\delta)/(2n)}$. The finite-class claim requires a predeclared collection of rules or thresholds. A class chosen from the same labels requires its selection to be included in the analysis. The general theorem instead permits any valid simultaneous deviation bound for the declared class.

Monotonicity of the identifiability floor under evidence refinement. The identifiability floor is a property of the evidence distribution, which makes it monotone in the evidence itself.

Proposition 4 (Refinement cannot raise the identifiability floor)**.** *For the same adequacy label S and population, let Z' refine Z, so that $\sigma(Z) \subseteq \sigma(Z')$. Then $\epsilon_{Z'}^\star \leq \epsilon_Z^\star$.*

Proof. Every verifier measurable with respect to $\sigma(Z)$ is measurable with respect to $\sigma(Z')$, so the feasible set whose infimum defines $\epsilon_Z^\star$ in Equation 4 is contained in the feasible set defining $\epsilon_{Z'}^\star$. Taking infima over a larger set cannot increase the value. □

The proposition concerns the evidence, not the statistic computed from it. Discarding information cannot lower the optimal verification error, while a lossy summary of finer evidence need not improve. ACSSP scores each (layer, query, head) pair and reduces the window with a maximum, and even for nested raw windows, the larger-window maximum need not determine the smaller-window maximum. At a fixed constructor and preprocessing, retaining all existing observations while widening W or adding native verification layers refines the raw evidence. The deployed scalar's ranking quality has no monotonicity guarantee, and thresholds must be recalibrated when the configuration changes.

Evidence and verification scores. The bounds apply to the declared evidence interface Z. For a fixed measurable assessment $T = f_{\mathrm{ver}}(Z)$, $\sigma(T) \subseteq \sigma(Z)$, so $\epsilon_Z^\star \leq \epsilon_T^\star$. Routing may use both Z and T without gaining information beyond Z. Routing restricted to T may lose information. In ACSSP, native tensors and draft discrepancies supply evidence, which is summarized by the verification score s_j. The counterexample in Appendix B.2 concerns this score, not the full native tensors from the verification band.

Sampling and calibration. Fit the constructor and any evidence representation independently of verifier-validation data, or analyze the entire data-dependent pipeline. Sampling one declared token per independent request from $\mathcal{Q}_g$ gives independent decision instances. Alternatively, a bounded request-average token loss permits request-level concentration. Its sample size is the number of requests, not tokens. Moreover, symmetric classification error does not encode the different quality and cost consequences of false acceptance and false rejection.

B ACSSP METHOD SUPPORT

B.1 LOW-RANK RESIDUAL KV MAP CAPACITY

Proposition 5 (Exact reduced-rank ACSSP constructor floor)**.** *Stack normalized token-anchor features as $X \in \mathbb{R}^{N\times d}$ and the native targets of one layer's K or V projection (all KV heads together) as $Y \in \mathbb{R}^{N\times q}$ on a fixed diagnostic sample. Key targets are before RoPE. Ordinary orthogonal RoPE preserves their row-wise error, whereas scaled positional transforms require their own metric. Let B_0 be the frozen native projection, $E_0 = Y - XB_0$, and $P_X = XX^\dagger$. For low-rank residual KV maps $X(B_0 + D)$ with $\mathrm{rank}(D) \le r$, the empirical squared-error optimum is*

$$\mathcal{E}^\star_{\mathrm{map},r} = \|(I - P_X)E_0\|_F^2 + \sum_{j>r} \sigma_j^2(P_X E_0). \tag{27}$$

Moreover, the best unrestricted linear map, the low-rank residual KV map with correction rank at most r, and the frozen original projection obey

$$\underbrace{\|(I - P_X)Y\|_F^2}_{\mathcal{E}^\star_{\mathrm{distill}}} \le \mathcal{E}^\star_{\mathrm{map},r} \le \underbrace{\|Y - XB_0\|_F^2}_{\mathcal{E}_{\mathrm{orig}}}. \tag{28}$$

The identity sums over independently parameterized layer/type maps with fixed features and targets. It does not impose a separate rank budget per head when factors are shared across heads.

Proof of Proposition 5. For any D with $\mathrm{rank}(D) \le r$, orthogonal projection onto $\mathrm{col}(X)$ gives

$$\begin{aligned}\|Y - X(B_0 + D)\|_F^2 &= \|E_0 - XD\|_F^2 \\ &= \|(I - P_X)E_0\|_F^2 + \|P_X E_0 - XD\|_F^2.\end{aligned} \tag{29}$$

The two terms are orthogonal because $(I - P_X)E_0 \in \mathrm{col}(X)^\perp$ and both $P_X E_0$ and XD lie in $\mathrm{col}(X)$. The set $\{XD : \mathrm{rank}(D) \le r\}$ equals the set of rank-at-most-r matrices whose columns lie in $\mathrm{col}(X)$. The forward inclusion is immediate, while any such Q satisfies $Q = XX^\dagger Q = X(X^\dagger Q)$ and $\mathrm{rank}(X^\dagger Q) \le \mathrm{rank}(Q) \le r$. Because $P_X E_0$ already lies in $\mathrm{col}(X)$, the Eckart-Young-Mirsky theorem applied to the second term yields $\sum_{j>r} \sigma_j^2(P_X E_0)$ and proves Equation 27.

For unrestricted linear distillation, least squares gives

$$\min_B \|Y - XB\|_F^2 = \|(I - P_X)Y\|_F^2 = \|(I - P_X)E_0\|_F^2, \tag{30}$$

where the final equality uses $(I - P_X)XB_0 = 0$. Moreover,

$$\|E_0\|_F^2 = \|(I - P_X)E_0\|_F^2 + \sum_{j\ge 1} \sigma_j^2(P_X E_0). \tag{31}$$

The rank-r spectral tail is nonnegative and no larger than the full spectral sum, proving both inequalities in Equation 28. A truncated SVD of $P_X E_0$ has columns in $\mathrm{col}(X)$ and is attained by $D = X^\dagger(P_X E_0)_r$, whose rank is at most r. Independent layer/type minima can therefore be summed. □

This is a fixed-sample least-squares capacity result, not the optimum of trajectory KL training in Equation 7. The feature-span residual is a restriction of linear prediction, not an estimate of $\mathcal{E}_{\mathrm{info}}$. Increasing rank cannot worsen this optimum at fixed features, but does not guarantee better generalization or latency. Conservative-map features depend on the routing policy. A comparison must hold those features fixed to invoke the identity.

Instantiating the constructor floor. Table 5 evaluates the reduced-rank least-squares oracle on 32 distinct PG-19 test documents excluded from map training, with eight each at 4K/8K/16K/32K tokens, with 512 approximately equally spaced nonfinal positions per document (16,384 rows). We save exact Native states at anchors m_{16} and m_{20} and the inputs to their suffix layers. The m_{20} features are not hybrid continuation states. Each target layer's frozen normalization is applied to both its anchor features and Native inputs. States and normalization use BF16. Linear projections and the oracle decomposition use FP64, with keys measured before RoPE. Squared errors are summed across suffix layers and K/V projections and divided by the corresponding total Native target squared norm.

At each fixed anchor, increasing rank reduces the spectral tail while leaving the feature-span residual unchanged. From rank 16 to 128, the normalized optimum falls from 0.089645 to 0.052313 at m_{16} and from 0.060927 to 0.035309 at m_{20}. These values characterize attainable KV reconstruction on the fixed sample, not the KV errors of the KL-trained checkpoints. The oracle is fitted to the diagnostic sample itself, so it is neither a held-out generalization guarantee nor an estimate of the information floor. Actual training behavior is assessed separately by the trajectory KL diagnostic in Figure 5. Lower KV least-squares optima do not imply lower trained-model KL.

Table 5: Llama-3.1-8B-Instruct reduced-rank KV capacity on a fixed diagnostic sample. All entries are normalized squared errors. The rank-independent feature-span residuals are 0.025179 (m_{16}) and 0.016961 (m_{20}). The corresponding frozen-projection errors are 0.269446 and 0.214499. The optimum equals the residual plus the spectral tail, up to rounding.

	Anchor m_{16}		Anchor m_{20}	
r	$\mathcal{E}^{\star}_{\mathrm{map},r}$	Spectral tail	$\mathcal{E}^{\star}_{\mathrm{map},r}$	Spectral tail
16	0.089645	0.064465	0.060927	0.043966
32	0.075597	0.050418	0.051237	0.034276
64	0.063711	0.038531	0.043050	0.026089
128	0.052313	0.027134	0.035309	0.018348

Late-training trajectory agreement. Figure 5 reports mean trajectory KL on 4K prefixes over the final 50 training updates, with sample SD across updates, in units of 10^{-3}. On Llama-3.1-8B-Instruct at rank 64, it decreases from 4.008 ± 1.812 at m_{12} to 0.718 ± 0.277 at m_{26}. Raising rank from 16 to 128 lowers it at every evaluated anchor. At m_{12} it falls from 5.920 ± 2.673 to 3.551 ± 1.584. Gains from additional rank shrink with depth. At each family's deepest anchor, rank-64 KL lies between 0.72 and 1.71. These results show how anchor depth and residual capacity affect training-time agreement. They are distinct from the sample-fitted KV oracle and downstream task quality.

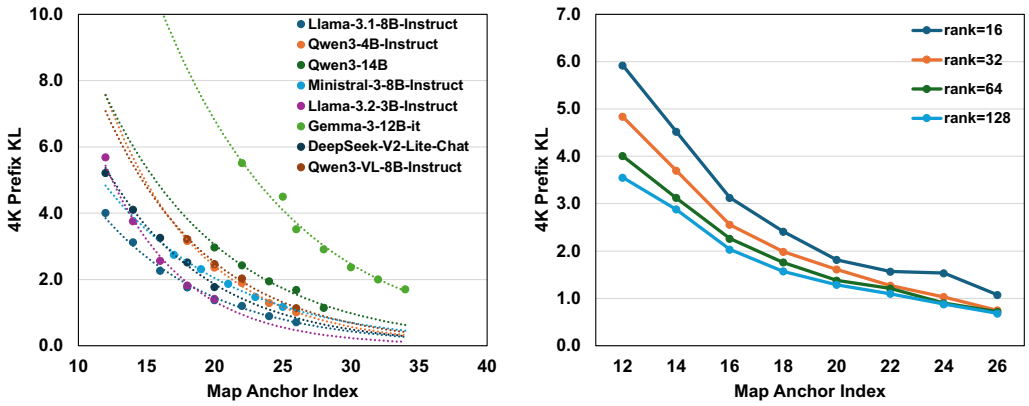

Figure 5: Final-50 4K-prefix KL (units of 10^{-3}) against map anchor depth. **Left:** rank-64 maps for the eight model families. **Right:** the Llama-3.1-8B-Instruct rank sweep at ranks $16/32/64/128$ over anchors m_{12}-m_{26}. Within each family and at each rank the statistic falls monotonically as the anchor deepens, and the improvement from additional rank shrinks with depth. This is a late-training trajectory diagnostic, not a measurement of KV reconstruction error or a consequence of the capacity bound in Equation 27. In the legend, Qwen3-4B-Instruct and Ministral-3-8B-Instruct denote Qwen3-4B-Instruct-2507 and Ministral-3-8B-Instruct-2512-BF16, respectively.

B.2 Attention Sensitivity and Verification Limitations

For a fixed query q, let $z_i = q^\top k_i/\sqrt{d_h}$, $\alpha = \mathrm{softmax}(z)$, and $o = \sum_i \alpha_i v_i$. Perturb only row j in directions $(\delta k_j, \delta v_j)$, holding the query and other rows fixed. Since $\delta z_j = q^\top \delta k_j/\sqrt{d_h}$, the softmax derivative gives $\delta\alpha_i = \alpha_i(\mathbf{1}\{i=j\} - \alpha_j)\delta z_j$. Substitution into $\delta o = \sum_i \delta\alpha_i v_i + \alpha_j \delta v_j$ yields

$$\delta o = \alpha_j \delta v_j + \alpha_j \frac{q^\top \delta k_j}{\sqrt{d_h}}(v_j - o). \tag{32}$$

For finite inputs, $o(t) = o(0) + t\delta o + O(t^2)$ as $t \to 0$. Thus both first-order effects contain α_j, motivating attention weighting. Key sensitivity also depends on error direction, query norm, and value spread. The verification score retains normalized error magnitudes and α_j. Its RMS normalization and visible-length scaling are design choices, not a finite-error bound.

Proposition 6 (Information lost by the verification score)**.** *There exist a bounded native cache and two draft KV rows with the same scalar verification score in Equation 8, but different effects on a future query. Consequently, no rule based only on this scalar can universally determine future-query adequacy.*

Proof. In dimension $d \geq 2$, take orthonormal u, w, a finite nonzero key k_0, and bounded $v_1 \neq v_2$. Let both native keys equal k_0 and let the two drafts change only the first key to $k_0 + tw$ or $k_0 + tu$, with finite $t \neq 0$. Values remain unchanged. Their key-error norms, native attention weights, RMS denominators, and visible-length factors coincide for every verification query. Their scalar scores therefore coincide. For $q_\star = u$, the first draft leaves both logits equal and produces $o_0 = (v_1 + v_2)/2$. The second produces

$$o_1 = \sigma(t/\sqrt{d})v_1 + [1 - \sigma(t/\sqrt{d})]v_2, \qquad o_1 - o_0 = [\sigma(t/\sqrt{d}) - 1/2](v_1 - v_2) \neq 0, \tag{33}$$

where σ is the logistic function. A tolerance between zero and $\|o_1 - o_0\|$ assigns opposite labels to identical scores. On a balanced population supported on this pair, a verifier observing only this scalar score has Bayes error $1/2$. □

The counterexample concerns information discarded by the score over permitted drafts. It is not an error lower bound for a particular fitted ACSSP map on its deployment distribution. Finite multi-row errors and changes to later queries also fall outside the one-row derivative. Transfer of the score to unobserved suffix layers and the combined routes are therefore assessed only through request-level experiments.

Adding verification layers delays exit for every token, while adding query rows increases verification overhead without adding native depth. Neither change guarantees monotonic improvement of the deployed score (Appendix A.4).

B.3 Execution Cost and Design Trade-offs

We account for ACSSP's work by partitioning its execution at v and c. Let $n_A = |\mathcal{I}_A|$, $n_C = |\mathcal{I}_C|$, and $n_A + n_C = n - 1$. Let $\mathcal{P} = \{1, \dots, n\}$ and $\mathcal{Q}_C = \mathcal{I}_C \cup \{n\}$. Denote by $\mathsf{B}_\ell(\mathcal{Q}, n)$ the cost of target layer ℓ for active query positions $\mathcal{Q}$ with n cache positions and the model's attention mask, and by $\mathsf{M}^A_\ell(s)$ and $\mathsf{M}^C_\ell(s)$ the aggressive and conservative map costs for s rows. In an additive cost model, excluding cache writes common to both methods,

$$\begin{aligned} C_{\mathrm{ACSSP}}(x) = &\sum_{\ell<v} \mathsf{B}_\ell(\mathcal{P}, n) + C_{\mathrm{ver}}(x) + C_{\mathrm{route}}(x) \\ &+ \sum_{\ell=v}^{c-1} \left[\mathsf{B}_\ell(\mathcal{Q}_C, n) + \mathsf{M}^A_\ell(n_A)\right] \\ &+ \sum_{\ell=c}^{L-1} \left[\mathsf{B}_\ell(\{n\}, n) + \mathsf{M}^A_\ell(n_A) + \mathsf{M}^C_\ell(n_C)\right]. \end{aligned} \tag{34}$$

Derivation by execution interval. Before v, every token executes the target layers, producing exact native states and KVs. Verification overhead C_{ver} adds aggressive draft KVs within $[a, v)$, $O((v-a)Wn)$ selected-query attention interactions, and error reductions. Native KVs support execution within the verification band and scoring before routing. After routing, aggressive rows replace their native KVs in $[a, v)$ with the already computed drafts. Conservative rows and the final token retain native KVs in the verification band. This replacement does not recompute or change the saved exact states H_v, but it changes the cache consumed by subsequent decoding. Route construction, replacement of KV entries in the verification band (including additional writes and data movement), saved-state handling, and active-row gather/scatter enter C_{route}. Construction of drafts within the verification band is charged only to C_{ver}, so reusing these drafts adds no second mapping charge. In $[v, c)$, the n_C conservative tokens and final token execute target layers, while the other n_A rows use aggressive maps. Beyond c, only the final token executes target layers. Both exited sets obtain their missing rows from their respective maps. For layers at or beyond v, the token sets partition the prompt, so each required KV row is supplied once by target execution or the selected map. Within the verification band, both native and draft rows are computed, with construction and replacement charged as above. Summing these operations gives Equation 34 without prefix replay.

Break-even condition. Native costs $\sum_{\ell=0}^{L-1} \mathsf{B}_\ell(\mathcal{P}, n)$ under the same accounting. Define the avoided target-layer work

$$\begin{aligned} D(x) = &\sum_{\ell=v}^{c-1} [\mathsf{B}_\ell(\mathcal{P}, n) - \mathsf{B}_\ell(\mathcal{Q}_C, n)] \\ &+ \sum_{\ell=c}^{L-1} [\mathsf{B}_\ell(\mathcal{P}, n) - \mathsf{B}_\ell(\{n\}, n)], \end{aligned} \tag{35}$$

and let $O(x)$ be $C_{\mathrm{ver}} + C_{\mathrm{route}}$ plus all mapping terms in Equation 34. Subtracting the two costs cancels the shared prefix and gives $C_{\mathrm{Native}}(x) - C_{\mathrm{ACSSP}}(x) = D(x) - O(x)$. Hence ACSSP saves cost in this model exactly when $D(x) > O(x)$. The common prefix is paid once by both methods. Reuse prevents paying for it again, rather than making execution within the verification band free.

Length scaling in a work surrogate. Fix the model, anchors, rank r, verification query window W, and aggressive fraction $p = n_A/(n-1)$ for $n > 1$. To examine scaling, replace position-dependent costs by a count-based surrogate with $q = |\mathcal{Q}|$: $\widetilde{\mathsf{B}}_\ell(q, n) = qn\beta_\ell + q\mu_\ell$, with nonnegative coefficients for attention interactions and per-row projections/feed-forward work. This idealization holds the effective interaction cost fixed. Causal positions, sparse or local attention, and kernel shapes can change it. The surrogate avoided work is

$$\widetilde{D}(n) = (n-1)(n\beta_1 + \mu_1) + p(n-1)(n\beta_2 + \mu_2), \tag{36}$$

where subscript 1 sums over $[c, L)$ and subscript 2 over $[v, c)$. Both terms can contribute quadratically. Writing $\beta_{\mathrm{eff}} = \beta_1 + p\beta_2$ and $u = \mu_1 + p\mu_2$ gives $\widetilde{D}(n) = \beta_{\mathrm{eff}} n^2 + (u - \beta_{\mathrm{eff}})n - u$. At $p = 0$,

conservative exit still contributes β_1 to the quadratic coefficient. Only the additional avoided native work relative to an all-conservative assignment with the same verification band,

$$\widetilde{D}_{\text{adaptive}}(n) = p(n-1)(n\beta_2 + \mu_2),$$

vanishes with p. Its net benefit must also subtract the difference in mapping, routing, and data-movement costs between the assignments.

For fixed dimensions and evidence budget, suppose the surrogate's complete additional overhead satisfies $0 \le \widetilde{O}(n) \le \kappa n + \kappa_0$ with $\kappa, \kappa_0 \ge 0$. This is an explicit assumption covering full base projections, low-rank residual corrections, verification scoring, routing, saved-state handling, replacement of KV entries in the verification band, and data movement. Then

$$\widetilde{D}(n) - \widetilde{O}(n) \ge \beta_{\text{eff}} n^2 + (u - \beta_{\text{eff}} - \kappa)n - (u + \kappa_0). \tag{37}$$

A positive right side is sufficient for savings in this surrogate. A nonpositive value does not establish a loss. If $\beta_{\text{eff}} > 0$ and these coefficients remain fixed, the lower bound is eventually positive. The linear coefficient has no prescribed sign, and this argument supplies neither a measured hardware break-even length nor a universal speedup curve. When p or other effective coefficients vary by length, the bound must be evaluated with those length-dependent values rather than one fixed polynomial.

Observed length signature. Across all seven ACSSP-B profiles, mean TTFT speedup is higher at 128K than at 2K. The ACSSP$-$POP margin is negative at 2K except on Llama-3.1-8B-Instruct, but positive at both 32K and 128K on six of seven models. Gemma-3-12B-it remains below POP. Qwen3-VL-8B-Instruct grows from $1.210\times$ at 2K to $1.441\times$ at 128K and exceeds POP from 8K onward, while the Qwen3-4B-Instruct-2507, Ministral-3-8B-Instruct-2512-BF16, and Llama-3.2-3B-Instruct results also exceed POP at 32K and 128K. Gemma-3-12B-it at 2K ($0.989\times$) is the only sub-unit ACSSP-B cell, consistent with an overhead-dominated short-context regime. These observations are consistent with length-dependent amortization of overhead but do not identify a break-even length or validate the surrogate coefficients. The additive model omits kernel efficiency, fusion, and memory traffic, as the next paragraph makes explicit.

Connection to measured latency. This decomposition specifies work and overhead, not a hardware-independent speedup. Layer and map costs depend on active shapes, kernels, and memory traffic. Fusion or overlap can invalidate a sum of separately timed components. End-to-end timing must include all cache writes, saved states, mapping, scoring, compaction, and data movement, with shared costs restored. Persistent KV length remains n, and temporary memory must be assessed separately. A larger exited-token fraction therefore need not produce a proportional latency reduction.

Relative to an all-conservative assignment with the same verification band, aggressive exits avoid native continuation only in $[v, c)$. At $v = c$, routing selects constructors but saves no native continuation depth against that assignment. Increasing τ_s enlarges the aggressive set for fixed scores, but neither task quality nor shape-dependent latency must vary monotonically. The anchors, threshold, and evidence budget must therefore be assessed jointly. Comparisons with a fixed-depth method that has no verification band must also include the extra verification overhead.

C SUPPLEMENTARY EXPERIMENTS

C.1 MODELS, DATA, AND METRICS

Models and baselines. We evaluate Llama-3.1-8B-Instruct, Qwen3-14B, Qwen3-4B-Instruct-2507, Qwen3-VL-8B-Instruct, Ministral-3-8B-Instruct-2512-BF16, Llama-3.2-3B-Instruct, and Gemma-3-12B-it. Qwen3-VL-8B-Instruct uses text-only prompts, and Qwen3-14B thinking is disabled. POP retains 22 layers for Llama-3.1-8B-Instruct, 19 for Llama-3.2-3B-Instruct, 27 for Qwen3-14B, 23 for Ministral-3-8B-Instruct-2512-BF16, 24 for Qwen3-4B-Instruct-2507, and 26 for Qwen3-VL-8B-Instruct. Prompts, decoding, and scoring are matched within each model. The checkpoint revisions for Llama-3.2-3B-Instruct and Qwen3-VL-8B-Instruct are `e443548a5da3c59ed14484f4bf4a3c61cccd7cab` and `0c351dd01ed87e9c1b53cbc748cba10e6187ff3b`, respectively.

Distilled-checkpoint baseline. We evaluate SwiftKV (Qiao et al., 2025) using its released distilled SingleInputKV weights based on Llama-3.1-8B-Instruct, without AcrossKV. SwiftKV distills affected QKV projections and uses a fixed prefill depth. Reproducing its distillation and evaluating it across multiple model families is difficult within our experimental resources, so this comparison uses the released checkpoint on Llama-3.1-8B-Instruct, included in the main results. It is not a comparison under matched adaptation budgets. Table 9 includes its seven-task quality scores. On HotpotQA it achieves 56.06 F1 and $1.867\times$ TTFT speedup against its matched Native control. Table 12 gives its RULER QA_2 quality and timing. Table 2 reports its HQA and RBC results.

Question answering. We use NarrativeQA, Qasper, MultiFieldQA-en, HotpotQA, 2WikiMultihopQA, and MuSiQue from LongBench, with 200 examples per task except 150 for MultiFieldQA-en, totaling 1,150 examples. Evaluation uses greedy decoding, batch size 1, and at most 32,640 prompt tokens. Table headers use NQA, QAS, MFQA, HQA, 2Wiki, MSQ, and RBC for NarrativeQA, Qasper, MultiFieldQA-en, HotpotQA, 2WikiMultihopQA, MuSiQue, and RepoBench-C, respectively.

Code completion. RepoBench-C uses the official v1.1 32K subset. We sample 50 fixed-seed examples from each cross-file-first, cross-file-random, and in-file setting in both Python and Java, totaling 300 shared examples. Raw-code prompts use no chat template and are capped at 32,640 model-tokenizer tokens. Actual length distributions appear in Table 8. Generation uses at most 64 tokens, temperature 0.2, top-$k = 1$, top-$p = 1$, and official first-line extraction. Edit similarity uses fuzzywuzzy 0.18.0's pure-Python backend. Setting weights follow the selected subset populations: 1,000:940:985 for Python and 1,000:1,000:1,000 for Java, with equal language weights. Exact match is auxiliary. Code quality uses A40 GPUs. Code timing uses A800 GPUs with the same checkpoints and frozen policies.

Aggregation. The main quality average equally weights the six QA scores and RepoBench edit similarity. Component ablations use the same seven-task metric and matched configurations. For MR, we compute each seed's seven-task average first, then report the mean and sample standard deviation across the three seeds. The single-anchor sweep in Table 15 also reports the seven-task average, including RepoBench.

C.2 MAP TRAINING AND PROFILE CALIBRATION

Table 6 lists the anchor depths and verification query windows. All KV maps use rank-64 residual corrections, and all backbone, normalization, and base projection weights remain frozen.

Threshold calibration. ACSSP-B uses disjoint design and validation traces from QA, code, and retrieval workloads in four predeclared length ranges (2-4K, 4-16K, 16-32,640, and 32,641-131,072). Target aggressive acceptance is 50% in the first two ranges and 75% in the latter two. One threshold is selected against the lowest-coverage design stratum in each range, including 128K. These thresholds form one frozen schedule for the experiment group. Validation measures realized coverage rather than imposing a task-quality constraint. Held-out 128K acceptance is 75.99%, 75.95%, 74.25%, and 78.30% for the four main models, and 74.84% for Qwen3-VL-8B-Instruct. The same calibrated threshold schedule remains frozen for both quality and timing within each experiment. Coverage is an execution target, not a certified quality budget.

Table 6: ACSSP anchor and verification settings (rank 64). The base coverage specifies the nominal profile. The ACSSP-B length-dependent calibration schedule is described in this subsection.

Profile	(a, v, c)	W	Base coverage
Llama-3.1-8B-Instruct			
ACSSP-F	$(16, 17, 18)$	8	75%
ACSSP-B	$(16, 17, 20)$	8	50%
ACSSP-S	$(18, 20, 22)$	16	50%
Llama-3.2-3B-Instruct			
ACSSP-F	$(12, 13, 14)$	8	50%
ACSSP-B	$(12, 13, 18)$	8	50%
ACSSP-S	$(16, 18, 20)$	8	50%
Qwen3-14B			
ACSSP-F	$(20, 21, 22)$	8	50%
ACSSP-B	$(22, 23, 26)$	8	50%
ACSSP-S	$(24, 26, 28)$	8	50%
Ministral-3-8B-Instruct-2512-BF16			
ACSSP-F	$(17, 18, 19)$	8	50%
ACSSP-B	$(17, 18, 23)$	8	50%
ACSSP-S	$(21, 23, 25)$	8	50%
Qwen3-4B-Instruct-2507			
ACSSP-F	$(18, 19, 20)$	8	50%
ACSSP-B	$(18, 19, 24)$	8	50%
ACSSP-S	$(22, 24, 26)$	8	50%
Qwen3-VL-8B-Instruct			
ACSSP-B	$(22, 23, 26)$	8	50%
Gemma-3-12B-it			
ACSSP-B	$(28, 30, 32)$	8	50%

Map fitting. Maps minimize the teacher-forced distillation objective in Equation 7, with the shared settings in Table 7. Training uses 32 documents from the official PG-19 train split. For six models, eight source records are drawn from each of the 4K/8K/16K/32K source buckets. Qwen3-14B uses 32 source records from the 4K bucket. All records are clipped to 4K before prefix sampling, so source buckets determine document provenance rather than optimization sequence length. Each update draws two records and four prefixes per record, of lengths $\{2, u, 2049, 4096\}$ with deterministic sampling of $u \in [3, 4095]$. The resulting batch contains eight sequences, averaging 2,048.54 prompt tokens each. Conservative maps are fitted on exact native H_c states. Deployment uses the hybrid continuation described in Section 4.

Training-budget comparison. The count of 134,253,265 in Table 7 is the cumulative number of sampled prompt tokens over 8,192 updates, with repeated prefixes from 32 documents. The additional

Table 7: Shared map-training settings. Only the low-rank residual factors are optimized.

Setting	Value
Training data	32 PG-19 training documents; prefixes up to 4,096 tokens
Residual rank	64
Distillation horizon H	4 teacher-forced continuation tokens
Optimizer / initial learning rate	AdamW / 2×10^{-4}
Optimization updates	8,192
Global batch	Eight variable-length sequences per update
Sampled prompt tokens	134,253,265 (plus 262,144 continuation tokens)
Anchor-state source	Exact native states for both maps

262,144 tokens are teacher-forced continuations. It is not the size of a unique-token corpus. SwiftKV reports a roughly 680M-token mixture of Ultrachat, SlimOrca, and OpenHermes-2.5 and two training epochs (Qiao et al., 2025), yielding a nominal cumulative exposure of $680\mathrm{M} \times 2 \approx 1.36\mathrm{B}$ training tokens. This estimate counts repeated dataset passes. The paper does not separate prompt-only tokens from response tokens, so it is not an exact prompt-only count. These cumulative token exposures describe the training recipes rather than matched compute. Actual GPU-hours also depend on hardware, sequence lengths, objectives, and the number of separately fitted maps. The released-checkpoint comparison does not isolate the effect of training budget.

Constructor capacity. For one projection of shape $d \times q$, a rank-r residual correction trains $r(d+q)$ factor entries while freezing the dq base entries. On the same fixed features and squared reconstruction objective, the nested constructor families satisfy $\mathcal{E}^{\star}_{\mathrm{full}} \leq \mathcal{E}^{\star}_{r} \leq \mathcal{E}_{\mathrm{frozen}}$. Thus low-rank adaptation can improve attainable fidelity over frozen projections, but unrestricted adaptation has at least as much capacity. This ordering does not rank the task quality of trained ACSSP and SwiftKV checkpoints, which use different adaptation recipes.

C.3 Runtime and Timing Protocol

Runtime. Timing uses NVIDIA A800 GPUs (80 GB), vLLM 0.11.0, PyTorch 2.8.0, BF16 weights and KV caches, batch size 1, eager execution, disabled prefix caching, and a 0.70 memory budget. Qwen3-14B and Gemma-3-12B-it use tensor parallelism 2, while Qwen3-VL-8B-Instruct uses tensor parallelism 1. Each method is compared with a Native reference on the same device or matched tensor-parallel configuration. Qwen3-14B uses YaRN factor 4 at 128K with original context length 32,768. SwiftKV runs in the adapted vLLM runtime. Its RULER and HotpotQA timings use workload-specific matched Native controls. RepoBench timing compares SwiftKV and Native on the same GPU.

RULER workload and metrics. We adapt RULER (Hsieh et al., 2024) `niah_single_1`, using deterministic split-specific keys and insertion depths at 2K/8K/32K/128K lengths. Prompts reserve 128 context positions and fit whole filler units without truncating the needle or query. Actual token counts vary slightly with the tokenizer. Each method processes the same 20 requests per length, generates one token, and is measured in three repetitions with four excluded warmups per repetition. TTFT speedup is matched mean Native TTFT divided by mean method TTFT. Input token throughput (tokens/s) is total prompt tokens divided by generation-call wall time within a repetition, averaged across repetitions.

Downstream timing. We replay all 200 HotpotQA and 300 RepoBench examples using the quality checkpoints, prompt processing, anchors, verification settings, and frozen thresholds. Timing measures the first generated token, while quality uses full generation. Each method uses three repetitions with four excluded warmups and a matched Native control. Table 8 reports actual lengths and reference latencies.

Table 8: Downstream prompt-length distributions and Native TTFT references.

Model	Task	N	Median	P90	TTFT (ms)
Llama-3.1-8B-Instruct	HQA	200	14314	16348	1045.4
Llama-3.1-8B-Instruct	RBC	300	21080	23878	1935.4
Llama-3.1-8B-Instruct (SwiftKV)	HQA	200	14314	16348	1047.0
Llama-3.1-8B-Instruct (SwiftKV)	RBC	300	21080	23878	1935.4
Llama-3.2-3B-Instruct	HQA	200	14314	16348	511.0
Llama-3.2-3B-Instruct	RBC	300	21080	23878	977.4
Qwen3-14B	HQA	200	15033	17037	1106.4
Qwen3-14B	RBC	300	21220	24081	1904.8
Ministral-3-8B-Instruct-2512-BF16	HQA	200	15654	17685	1286.6
Ministral-3-8B-Instruct-2512-BF16	RBC	300	22942	26122	2403.4
Qwen3-4B-Instruct-2507	HQA	200	15029	17033	817.8
Qwen3-4B-Instruct-2507	RBC	300	21220	24081	1510.4
Qwen3-VL-8B-Instruct	HQA	200	15029	17029	1182.4
Qwen3-VL-8B-Instruct	RBC	300	21220	24081	2086.7
Gemma-3-12B-it	HQA	200	14945	16973	829.5
Gemma-3-12B-it	RBC	300	25709	29450	1668.2

Policy matching and uncertainty. Each experiment group, including component, verification, and routing-seed comparisons, uses a shared calibration fixed before evaluation. Within each evaluated variant, quality and timing use identical checkpoints, anchors, verification settings, and the frozen threshold schedule by input-length range. Ablations change only their specified components. Quality and timing do not trigger separate calibration. Method order is interleaved across repetitions. Timing reports use 95% percentile intervals from 10,000 paired crossed request/repetition bootstrap draws. No outliers are removed. Measurements include mapping, verification, routing, compaction, cache writes, and tensor-parallel synchronization.

Long-context QA protocol. Table 12 pairs adapted RULER QA_2 quality with TTFT speedup at 32K and 128K, separately from the single-needle timing workload. We use the RULER HotpotQA reader and prompt template, preserve every original context document, and add deterministically shuffled distractor documents up to the tokenizer-specific budget, reserving 128 tokens for generation.

Inputs are shared across methods within each model and length. No answer-dependent document selection is used. Quality uses 200 requests, while timing uses the first 20 requests, with the next four as excluded warmups in each of three repetitions. Both evaluations use the same checkpoints and frozen policies on two A800 GPUs. Timing generates one token, whereas quality permits up to 128. These speedups refer to TTFT, not full-response latency. Each reported timing mean includes all three repetitions.

Seven-task timing protocol. Table 10 uses A800 GPUs, batch size 1, and one output token. Qwen3-14B and Gemma-3-12B-it use TP=2, while all other models use TP=1. Each QA repetition replays the full task inputs before four excluded warmups and formal measurement of all 200 requests (150 for MFQA). The three-repetition QA comparisons use same-device Native controls. RBC timing uses 300 requests, three repetitions, matched Native controls, and four excluded warmups per repetition. QA and RBC latencies are aggregated separately before computing task-wise speedups.

C.4 Complete Quality and Efficiency Results

Downstream quality. Table 9 shows higher mean quality for ACSSP-B than POP on every model. Safe raises the largest gain to 7.47 points.

Table 9: Seven-model quality comparison on the same 1,450-example protocol. ACSSP-F, ACSSP-B, and ACSSP-S denote the evaluated profile choices.

Method	NQA	QAS	MFQA	HQA	2Wiki	MSQ	RBC	Avg.
Llama-3.1-8B-Instruct								
Native	27.54	44.73	56.08	57.55	49.63	33.01	66.07	47.80
POP	27.90	43.19	53.93	54.09	47.13	29.02	58.56	44.83
SwiftKV	29.15	45.76	55.58	56.06	43.74	31.30	65.96	46.79
ACSSP-F	27.94	42.53	53.80	56.92	46.49	26.86	60.09	44.95
ACSSP-B	29.37	42.03	53.94	57.81	47.51	31.94	61.88	46.35
ACSSP-S	29.80	44.12	53.06	56.73	49.49	31.20	63.48	46.84
Llama-3.2-3B-Instruct								
Native	25.09	39.99	50.69	53.09	39.15	25.06	62.79	42.27
POP	20.12	33.07	44.44	45.76	33.24	19.07	55.42	35.87
ACSSP-F	19.97	33.03	45.11	47.26	32.10	21.56	56.24	36.47
ACSSP-B	21.49	37.50	48.86	52.99	37.35	24.01	62.12	40.62
ACSSP-S	22.99	39.54	48.43	52.51	39.16	21.70	61.88	40.89
Qwen3-14B								
Native	28.02	45.68	53.65	63.53	54.32	40.47	70.46	50.88
POP	25.95	44.36	46.71	59.05	44.83	32.70	66.17	45.68
ACSSP-B	25.53	47.01	47.84	57.73	42.16	38.21	68.71	46.74
ACSSP-S	27.88	46.37	51.27	61.95	49.70	39.39	70.48	49.58
Ministral-3-8B-Instruct-2512-BF16								
Native	27.34	45.47	44.98	61.28	54.74	38.71	69.86	48.91
POP	22.82	36.50	38.70	47.32	43.14	28.62	52.62	38.53
ACSSP-B	27.23	40.78	43.95	53.65	47.58	34.61	59.51	43.90
ACSSP-S	29.47	41.79	44.46	57.64	51.47	34.61	62.59	46.00
Qwen3-4B-Instruct-2507								
Native	27.97	45.00	50.30	59.12	43.43	24.31	68.65	45.54
POP	24.64	41.79	49.97	55.58	43.29	21.20	64.79	43.04
ACSSP-B	25.64	41.92	49.17	56.33	40.93	21.88	66.92	43.26
ACSSP-S	26.55	42.89	50.92	57.37	38.62	22.84	66.86	43.72
Qwen3-VL-8B-Instruct								
Native	33.38	47.71	53.55	65.78	56.63	37.30	67.96	51.76
POP	32.49	47.80	53.60	66.62	54.88	38.21	63.77	51.05
ACSSP-B	31.37	48.11	53.76	66.36	56.87	37.07	65.56	51.30
Gemma-3-12B-it								
Native	25.12	42.13	56.88	59.84	53.17	36.95	67.12	48.74
POP	23.54	41.22	56.36	59.28	52.09	34.34	65.57	47.49
ACSSP-F	24.27	42.81	57.67	59.34	54.63	35.48	66.74	48.70
ACSSP-B	22.87	42.29	56.97	59.21	54.16	36.24	67.26	48.43

Downstream efficiency. ACSSP-B improves mean task-wise TTFT speedup over POP on five of seven models (Table 10), including $1.512\times$ versus $1.389\times$ on Llama-3.1-8B-Instruct. Together with the quality table, this establishes joint improvements in the two reported means on those five models.

Context-length scaling. ACSSP-B exceeds POP at both 32K and 128K on six of seven models (Table 11). All ACSSP-B profiles have higher speedup at 128K than at 2K. HQA/RBC columns report the downstream comparisons from the main text.

Table 10: Seven-task downstream TTFT speedup (×) for Native, POP, and ACSSP across seven models. Each task reports the ratio of three-repetition mean Native TTFT to method TTFT. Avg. is the unweighted arithmetic mean of the seven task speedups, not a pooled-latency speedup. Warmup and device-matching protocols are specified in Appendix C.3. Native is normalized to one.

Method	NQA	QAS	MFQA	HQA	2Wiki	MSQ	RBC	Avg.
Llama-3.1-8B-Instruct								
Native	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
POP	1.407	1.367	1.375	1.391	1.385	1.397	1.404	1.389
ACSSP-F	1.639	1.497	1.531	1.579	1.535	1.589	1.615	1.569
ACSSP-B	1.597	1.466	1.488	1.531	1.490	1.539	1.476	1.512
ACSSP-S	1.421	1.315	1.345	1.378	1.352	1.381	1.394	1.369
Llama-3.2-3B-Instruct								
Native	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
POP	1.407	1.339	1.353	1.379	1.352	1.385	1.396	1.373
ACSSP-F	1.722	1.463	1.508	1.607	1.519	1.630	1.477	1.561
ACSSP-B	1.585	1.397	1.406	1.513	1.435	1.518	1.554	1.487
ACSSP-S	1.344	1.205	1.223	1.276	1.231	1.283	1.314	1.268
Qwen3-14B								
Native	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
POP	1.433	1.393	1.403	1.417	1.404	1.420	1.423	1.413
ACSSP-B	1.471	1.354	1.374	1.431	1.389	1.452	1.381	1.407
ACSSP-S	1.362	1.274	1.292	1.328	1.298	1.339	1.338	1.319
Ministral-3-8B-Instruct-2512-BF16								
Native	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
POP	1.409	1.360	1.369	1.390	1.374	1.395	1.407	1.386
ACSSP-B	1.521	1.380	1.410	1.489	1.431	1.514	1.453	1.457
ACSSP-S	1.324	1.238	1.253	1.290	1.258	1.301	1.315	1.282
Qwen3-4B-Instruct-2507								
Native	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
POP	1.443	1.372	1.388	1.415	1.392	1.421	1.432	1.409
ACSSP-B	1.521	1.355	1.384	1.482	1.412	1.511	1.400	1.438
ACSSP-S	1.344	1.225	1.250	1.300	1.261	1.315	1.331	1.289
Qwen3-VL-8B-Instruct								
Native	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
POP	1.347	1.311	1.319	1.334	1.322	1.336	1.343	1.330
ACSSP-B	1.401	1.305	1.323	1.362	1.327	1.372	1.359	1.350
Gemma-3-12B-it								
Native	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
POP	1.415	1.367	1.380	1.399	1.382	1.405	1.418	1.395
ACSSP-F	1.511	1.361	1.393	1.460	1.404	1.484	1.491	1.444
ACSSP-B	1.365	1.255	1.290	1.335	1.297	1.347	1.363	1.322

Table 11: Cross-model end-to-end TTFT speedup and input token throughput for the same selected profiles as Table 9. RULER and downstream speedups are ratios of the matched mean Native TTFT to the method mean over repeated measurements. Throughput uses the same requests. Mapping, verification, routing, cache writes, and TP synchronization are included.

	RULER TTFT speedup, ×				Input token throughput (tokens/s)				TTFT speedup, ×	
Method	2K	8K	32K	128K	2K	8K	32K	128K	HQA	RBC
Llama-3.1-8B-Instruct										
Native	1.000	1.000	1.000	1.000	14616	13718	9394	3792	1.000	1.000
POP	1.321	1.380	1.416	1.438	19312	18924	13299	5454	1.394	1.404
ACSSP-F	1.403	1.545	1.678	1.773	20503	21191	15760	6723	1.583	1.615
ACSSP-B	1.396	1.520	1.660	1.721	20382	20842	15596	6527	1.532	1.476
ACSSP-S	1.273	1.385	1.469	1.502	18591	18992	13803	5695	1.376	1.394
Llama-3.2-3B-Instruct										
Native	1.000	1.000	1.000	1.000	32968	29547	17745	6300	1.000	1.000
POP	1.254	1.348	1.414	1.454	41276	39796	25082	9162	1.377	1.396
ACSSP-F	1.210	1.505	1.767	1.948	39863	44576	31634	12363	1.589	1.477
ACSSP-B	1.223	1.388	1.654	1.713	40241	40972	29336	10795	1.510	1.554
ACSSP-S	1.085	1.260	1.386	1.416	36359	37831	25116	9131	1.277	1.314
Qwen3-14B										
Native	1.000	1.000	1.000	1.000	13907	13306	9902	4496	1.000	1.000
POP	1.331	1.400	1.438	1.462	18487	18621	14239	6575	1.416	1.423
ACSSP-B	1.244	1.372	1.489	1.564	17286	18248	14739	7032	1.429	1.381
ACSSP-S	1.196	1.308	1.380	1.425	16513	17107	13517	6358	1.326	1.338
Ministral-3-8B-Instruct-2512-BF16										
Native	1.000	1.000	1.000	1.000	13067	12409	8518	3553	1.000	1.000
POP	1.294	1.371	1.421	1.452	16891	17008	12107	5159	1.392	1.407
ACSSP-B	1.243	1.423	1.599	1.641	16229	17652	13621	5831	1.496	1.453
ACSSP-S	1.101	1.268	1.345	1.382	14251	15494	11457	4872	1.295	1.315
Qwen3-4B-Instruct-2507										
Native	1.000	1.000	1.000	1.000	22900	19939	11208	3774	1.000	1.000
POP	1.299	1.389	1.448	1.489	29722	27693	16227	5619	1.417	1.432
ACSSP-B	1.223	1.400	1.565	1.625	27969	27905	17543	6133	1.486	1.400
ACSSP-S	1.064	1.269	1.374	1.410	24388	25681	15692	5378	1.303	1.331
Qwen3-VL-8B-Instruct										
Native	1.000	1.000	1.000	1.000	13872	12921	8641	3427	1.000	1.000
POP	1.267	1.322	1.349	1.374	17562	17074	11655	4707	1.333	1.343
ACSSP-B	1.210	1.344	1.428	1.441	16778	17357	12341	4937	1.361	1.359
Gemma-3-12B-it										
Native	1.000	1.000	1.000	1.000	14688	16429	15312	11261	1.000	1.000
POP	1.312	1.377	1.421	1.468	19250	22619	21762	16526	1.399	1.418
ACSSP-F	1.050	1.461	1.559	1.524	15417	23865	23638	16971	1.460	1.491
ACSSP-B	0.989	1.313	1.381	1.433	14532	21573	21142	16138	1.334	1.363

Long-context QA. At 128K, ACSSP-B improves accuracy and TTFT speedup over POP on five models, and matches accuracy with higher speed on Llama-3.1-8B-Instruct (Table 12). Llama-3.2-3B-Instruct gains 14.5 accuracy points.

Table 12: Adapted RULER QA_2 (HotpotQA) accuracy (Acc., %) and TTFT speedup (×) on two A800 GPUs (TP=2). Quality uses 200 requests, greedy decoding, and at most 128 output tokens, scored by case-insensitive answer substring. Timing uses 20 requests from the same workload, one output token, and three repetitions with four excluded warmups each. Speedup is the matched mean Native TTFT divided by the method mean. These are QA_2 timings, not the single-needle timings in Figure 3.

	32K		128K	
Method	Acc.	×	Acc.	×
Llama-3.1-8B-Instruct				
Native	54.5	1.000	41.5	1.000
POP	50.0	1.402	39.5	1.433
SwiftKV	52.5	1.884	40.5	1.949
ACSSP-F	52.5	1.590	43.0	1.729
ACSSP-B	53.5	1.559	39.5	1.677
ACSSP-S	53.5	1.389	43.5	1.432
Llama-3.2-3B-Instruct				
Native	47.5	1.000	42.0	1.000
POP	26.0	1.396	23.0	1.444
ACSSP-F	41.5	1.640	28.0	1.909
ACSSP-B	45.0	1.555	37.5	1.771
ACSSP-S	45.0	1.299	38.5	1.341
Qwen3-14B				
Native	57.0	1.000	51.5	1.000
POP	51.5	1.440	41.0	1.465
ACSSP-B	52.0	1.506	47.0	1.583
ACSSP-S	57.5	1.379	50.5	1.387
Ministral-3-8B-Instruct-2512-BF16				
Native	53.5	1.000	45.5	1.000
POP	51.5	1.407	30.0	1.449
ACSSP-B	56.0	1.497	40.0	1.452
ACSSP-S	49.0	1.295	37.5	1.299
Qwen3-4B-Instruct-2507				
Native	58.0	1.000	45.0	1.000
POP	59.5	1.435	42.5	1.480
ACSSP-B	65.5	1.508	45.5	1.653
ACSSP-S	61.0	1.318	46.5	1.376
Qwen3-VL-8B-Instruct				
Native	52.0	1.000	41.5	1.000
POP	54.0	1.343	37.0	1.370
ACSSP-B	52.0	1.359	39.0	1.389
Gemma-3-12B-it				
Native	50.0	1.000	30.5	1.000
POP	49.5	1.422	31.0	1.468
ACSSP-F	50.5	1.537	29.0	1.343
ACSSP-B	51.0	1.380	29.5	1.431

C.5 Fixed-Depth and Component Comparisons

Component protocol. MA, MC, and MR use the same aggressive and conservative maps and anchors $(a, v, c) = (16, 17, 20)$ with $W = 8$. MA selects the aggressive route for every eligible token, MC selects the conservative route, and MR randomizes token identities at ACSSP-B's per-request aggressive counts. The final token remains protected. Tables 13 and 14 report quality and timing, respectively. RULER and downstream speedups refer to their own workloads.

Table 13: Llama-3.1-8B-Instruct component ablation: seven-task quality and RULER TTFT speedup (×). MR averages three seeds.

Method	NQA	QAS	MFQA	HQA	2Wiki	MSQ	RBC	Avg.	32K	128K
Native	27.54	44.73	56.08	57.55	49.63	33.01	66.07	47.80	1.000	1.000
MA	26.19	39.97	50.53	51.68	37.43	27.97	60.39	42.02	1.722	1.817
MC	31.28	42.42	55.22	57.65	46.88	32.24	61.68	46.77	1.495	1.557
MR (mean)	28.36	40.74	52.16	53.83	43.10	28.38	60.80	43.91	1.602	1.599
ACSSP-B	29.37	42.03	53.94	57.81	47.51	31.94	61.88	46.35	1.657	1.723

Table 14: Llama-3.1-8B-Instruct component-ablation TTFT speedup (×). Each task divides three-repetition mean Native TTFT by method TTFT. Avg. is the arithmetic mean of seven task ratios. MR averages the three device-matched seed speedups.

Method	NQA	QAS	MFQA	HQA	2Wiki	MSQ	RBC	Avg.
Native	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
MA	1.696	1.573	1.607	1.651	1.615	1.664	1.689	1.642
MC	1.484	1.401	1.425	1.449	1.426	1.457	1.467	1.444
MR (mean)	1.597	1.465	1.487	1.529	1.488	1.537	1.579	1.526
ACSSP-B	1.596	1.467	1.494	1.531	1.493	1.539	1.562	1.526

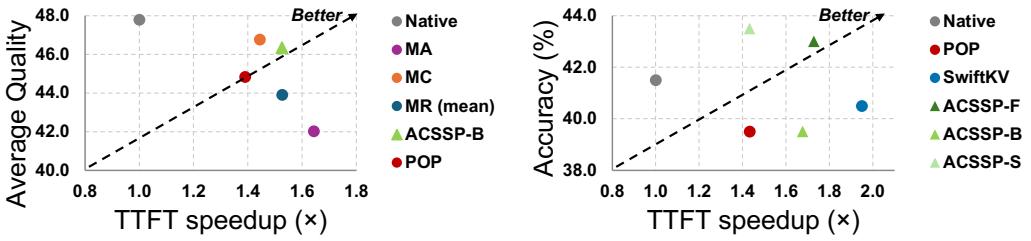

Figure 6: Llama-3.1-8B-Instruct quality-speed trade-offs. **Left:** seven-task mean quality versus the arithmetic mean of task-level TTFT speedups. MR averages three seeds. **Right:** 128K RULER QA_2 accuracy versus TTFT speedup on that workload. Higher values on both axes are better.

Quality-speed trade-offs. Figure 6 shows that ACSSP-B gains 2.44 quality points over MR at the same rounded mean task-wise speedup. Compared with MC, it increases mean speedup by 5.7% with a 0.42-point quality decrease, recovering most of the conservative route's quality at a faster operating point.

On 128K QA_2, Fast improves over POP in both accuracy (43.0% versus 39.5%) and speedup (1.729× versus 1.433×). Safe reaches 43.5% at 1.432×, while ACSSP-B matches POP's accuracy at 1.677×. SwiftKV is faster (1.949×) but scores 40.5%, below Fast and Safe. Fast also exceeds ACSSP-B on this workload, so profile ordering depends on the task. These measured points demonstrate distinct quality-speed choices.

Verification-control protocol. Table 3 fixes the construction anchor at m_{22}. POP-W24-Native uses verification boundary m_{24}, $W = 8$, and frozen routing thresholds. Its fused route-map merge is disabled because it is incompatible with frozen-projection mappers. All controls are evaluated on the same seven-task quality suite. RULER timings include their construction and verification costs.

Fixed-anchor construction. The rank-64 sweep retains all KV positions and full-depth decoding, varying only the exit anchor without verification or routing. From m_{12} to m_{20}, mean quality rises from 27.79 to 46.77 as mean speedup falls from $2.053\times$ to $1.442\times$ (Tables 15-16).

Table 15: Llama-3.1-8B-Instruct rank-64 single-anchor quality. Nonfinal tokens exit at the listed anchor. The final token and decoder execute at full depth. RepoBench-C uses the official v1.1 32K subset. Avg. is the unweighted seven-task mean.

Method	NQA	QAS	MFQA	HQA	2Wiki	MSQ	RBC	Avg.
Native	27.54	44.73	56.08	57.55	49.63	33.01	66.07	47.80
MAP-m_{12}	18.33	27.52	37.71	32.57	15.84	11.33	51.22	27.79
MAP-m_{14}	24.09	34.73	49.21	40.86	32.74	22.65	55.20	37.07
MAP-m_{16}	26.79	39.76	51.02	52.17	39.32	28.96	61.01	42.72
MAP-m_{18}	28.96	43.14	53.31	58.04	48.02	27.19	60.47	45.59
MAP-m_{20}	31.28	42.42	55.22	57.65	46.88	32.24	61.72	46.77
MAP-m_{22}	29.56	44.46	54.41	55.85	50.77	30.40	63.79	47.03
MAP-m_{24}	29.88	44.07	54.21	55.99	47.67	31.61	64.58	46.86
MAP-m_{26}	29.74	44.76	54.60	55.88	48.10	31.21	65.10	47.06

Table 16: Llama-3.1-8B-Instruct single-anchor TTFT speedup ($\times$) on the seven downstream tasks, corresponding to Table 15. Each task reports the ratio of three-repetition mean Native TTFT to method TTFT. Avg. is the unweighted arithmetic mean of the seven task speedups, not a pooled-latency speedup. The timing protocol follows Appendix C.3.

Method	NQA	QAS	MFQA	HQA	2Wiki	MSQ	RBC	Avg.
Native	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
MAP-m_{12}	2.190	1.923	1.975	2.066	1.979	2.091	2.144	2.053
MAP-m_{14}	1.955	1.764	1.799	1.868	1.802	1.885	1.922	1.856
MAP-m_{16}	1.769	1.625	1.655	1.702	1.656	1.716	1.743	1.695
MAP-m_{18}	1.611	1.500	1.525	1.558	1.529	1.571	1.595	1.556
MAP-m_{20}	1.483	1.405	1.422	1.446	1.420	1.452	1.469	1.442
MAP-m_{22}	1.366	1.307	1.323	1.339	1.327	1.351	1.362	1.339
MAP-m_{24}	1.275	1.235	1.244	1.257	1.244	1.261	1.269	1.255
MAP-m_{26}	1.193	1.165	1.171	1.179	1.170	1.184	1.189	1.178

Routing-seed sensitivity. Across seeds 42, 1234, and 3407, MR's seven-task means range from 43.80 to 44.05, all below ACSSP-B's 46.35 (Table 17). The MR mean is 43.91 ± 0.13, where $\pm$ denotes sample SD across seed-level averages. Each seed matches ACSSP-B's per-request aggressive counts, supporting a benefit from score-guided token selection across the tested random allocations. Averages use unrounded code scores. Table 18 reports seed-level RULER timing means.

Table 17: MR quality by routing seed. QA and RepoBench share the same frozen control configuration. Avg. equally weights the seven tasks. Its sample SD is computed across seed-level averages.

Seed	NQA	QAS	MFQA	HQA	2Wiki	MSQ	RBC	Avg.
42	29.16	40.01	52.61	54.54	42.99	28.05	60.99	44.05
1234	28.04	41.44	51.97	54.12	43.32	28.01	60.27	43.88
3407	27.89	40.78	51.90	52.82	43.00	29.07	61.14	43.80
Mean	28.36	40.74	52.16	53.83	43.10	28.38	60.80	43.91
Sample SD	0.69	0.72	0.39	0.90	0.19	0.60	0.47	0.13
ACSSP-B	29.37	42.03	53.94	57.81	47.51	31.94	61.88	46.35

MR RULER speedups use the shared Native reference from the MA/MC component-control measurements (3501.27 ms at 32K and 34541.31 ms at 128K), each averaged over three repetitions. We divide these reference means by MR's TTFT averaged over three repetitions and three routing seeds,

Table 18: MR TTFT by routing seed (ms). Each entry averages three timing repetitions of 20 requests. Mean and sample standard deviation summarize these seed-level means, rather than individual request or repetition variability.

Seed	2K	8K	32K	128K
42	93.95	385.17	2163.83	21416.91
1234	94.09	385.33	2167.53	21430.23
3407	96.86	395.84	2223.48	21939.78
Mean	94.97	388.78	2184.95	21595.64
Sample SD	1.64	6.11	33.42	298.11

yielding $1.602\times$ and $1.599\times$, respectively. These ratios use a shared baseline, not a separately paired Native run for each seed. RepoBench speedups are included in the downstream tables above.

C.6 Prefill FLOPs and Component Breakdown

Table 19 summarizes Llama-3.1-8B-Instruct prefill matrix-multiplication FLOPs on all 1,450 downstream requests. Native and POP counts use the actual prompt lengths. Each ACSSP profile uses measured token routes, including the original positions of conservative queries, rather than nominal acceptance fractions. We count one prompt prefill through the first-token LM-head logits, with one fused multiply-add counted as two FLOPs. Subsequent decoding, warmups, and repeated timing runs are excluded. Within each task, requests receive equal weight. Avg. equally weights the seven task means. Reduction is $100(1 - \overline{F}_{\mathrm{method}}/\overline{F}_{\mathrm{Native}})$, not the average of task-wise percentage reductions.

Table 19: Llama-3.1-8B-Instruct prefill matmul FLOPs (TFLOPs/request) and reduction relative to Native. (a) Task-level means and their unweighted average. (b) Additive operator breakdown of the same seven-task average. LR denotes the low-rank residual correction. Extra is the sum of FLOPs for base KV mapping, low-rank residual corrections, and extra verification QK divided by the method's total FLOPs, expressed as a percentage. Reduction includes mapping and verification matmuls. Entries are rounded independently.

(a) Seven-task FLOPs and reduction

Method	NQA	QAS	MFQA	HQA	2Wiki	MSQ	RBC	Avg.	Red. (%)
Native	519.741	80.160	113.127	227.180	117.288	283.318	416.013	250.975	0.00
POP	361.412	55.975	78.951	158.355	81.849	197.414	289.583	174.791	30.36
ACSSP-F	287.253	44.945	63.375	126.793	65.683	158.076	230.337	139.495	44.42
ACSSP-B	295.421	46.683	65.872	131.518	68.243	164.119	236.667	144.075	42.59
ACSSP-S	338.452	53.527	75.179	150.103	77.745	187.178	271.646	164.833	34.32

(b) Operator breakdown of the seven-task average

Method	Native QKV+O	Native MLP	Attn. QK+AV	Base KV map	LR corr.	Verif. QK	LM head	Total	Extra (%)
Native	35.825	150.465	64.685	0.000	0.000	0.000	0.001	250.975	0.00
POP	24.630	103.448	44.473	2.239	0.000	0.000	0.001	174.791	1.28
ACSSP-F	19.372	81.364	34.966	3.514	0.275	0.002	0.001	139.495	2.72
ACSSP-B	20.051	84.214	36.164	3.379	0.264	0.002	0.001	144.075	2.53
ACSSP-S	23.079	96.932	41.583	2.997	0.234	0.007	0.001	164.833	1.96

Accounting boundary. Native QKV+O and MLP include all executed token-layer projections, including native layers within the verification band and continuation. Attention QK+AV counts visible causal query-key pairs at the actual active query positions. Base KV mapping and low-rank residual correction costs include verification-band drafts for all eligible tokens, counted once and reused after routing. Verification QK counts both extra dense scoring passes in the fused implementation, including products before masking. It excludes native execution within the verification band already counted in the other columns. The final prompt token remains active through all layers. RMSNorm, RoPE, softmax, activations, scalar arithmetic and reductions, comparisons, indexing, data movement, kernel padding, and synchronization are outside this matmul-only count. Their runtime costs are not zero, so these reductions are neither hardware-instruction measurements nor latency predictions.

Compute savings after overhead. ACSSP-F, ACSSP-B, and ACSSP-S reduce mean matmul FLOPs by 44.42%, 42.59%, and 34.32% relative to Native, compared with POP's 30.36%. Relative to POP, the reductions are 20.2%, 17.6%, and 5.7%, respectively, and every profile uses fewer FLOPs on each of the seven tasks. For ACSSP-B, mapping and verification matmuls account for 2.53% of its total, while the largest absolute savings over POP come from native MLP and attention operations. Thus the measured routes retain a net arithmetic saving after their construction and verification overhead is included.

D QUALITATIVE EXAMPLES OF IMPROVEMENTS OVER POP

We present three examples selected from the Llama-3.1-8B-Instruct evaluations in which ACSSP-B improves on POP, including two from HotpotQA and one from RepoBench. Each compares Native, POP, and ACSSP-B on the same request. Inputs are excerpted for readability, while evaluation uses the full processed prompts. Outputs are reproduced from the evaluation records, with RepoBench restricted to the extracted first line used for scoring. These cases illustrate specific output differences. Their selection does not support estimates of overall improvement frequency or claims about the internal routing mechanism.

D.1 HOTPOTQA: ANSWERING WITH THE REQUESTED ENTITY

Input length: 14,036 tokens. **Sample ID:**

`e4df9576e5cd102024e81baf94a43325335d0ba2d8892041`.

Input excerpts. The following sentences are drawn from two passages in the original input, with intervening material omitted.

> [. . .]
> "We Found Love" is a song by Barbadian singer Rihanna from her sixth studio album, Talk That Talk (2011).
> [. . .]
> The song debuted at number two on the UK Singles Chart, behind Rihanna's "We Found Love".
> [. . .]

The second excerpt is from the passage describing The Wanted's "Lightning."

Question. The song that came out ahead of The Wanted's "Lightning" on the UK Singles Chart was featured on which album?

Table 20: HotpotQA outputs for the album-identification example.

Source	Answer
Reference	Talk That Talk
Native	Talk That Talk
POP	Rihanna's "We Found Love"
ACSSP-B	Talk That Talk

Observation. The question requires linking a chart position to a song and then to its album. POP returns the song, whereas Native and ACSSP-B return the requested album and match the reference answer.

D.2 REPOBENCH: COMPLETING THE EVALUATOR RETURN

Input length: 23,563 tokens. **Setting:** Python, cross-file-first. **Sample ID:**

`python-cross_file_first-001-ab3cd10a5686`.

The target file is `tools/train_net.py` in the benchmark repository `u2seg/U2Seg`. All three methods receive the same processed prompt.

Input excerpt. The function builds a list of dataset evaluators. The visible branches handle empty and single-element lists. Earlier repository context and function statements are omitted. Long code statements are wrapped for display only. The POP output below is a single extracted line in the evaluation record.

```
    # ... preceding input omitted ...
    if len(evaluator_list) == 0:
        raise NotImplementedError(
            "no Evaluator for the dataset {} with the type {}"
            .format(
                dataset_name, evaluator_type)
        )
    elif len(evaluator_list) == 1:
        return evaluator_list[0]
```

Reference next line:

```
    return DatasetEvaluators(evaluator_list)
```

Native:

```
    return DatasetEvaluators(evaluator_list)
```

POP:

```
    if comm.is_this_rank_for_precisebn(
        cfg, len(comm.get_world_size())):
```

ACSSP-B:

```
    return DatasetEvaluators(evaluator_list)
```

Observation. Native and ACSSP-B exactly match the reference next line, which returns the combined evaluators. POP introduces an additional condition instead. The difference is substantive rather than purely typographic, but matching the reference line does not establish execution-based correctness.

D.3 HotpotQA: Comparing Species Counts

Input length: 9,732 tokens. **Sample ID:**

`7e525340152c2f4b26d7fba85c680dde7b75af77356a7c22`.

Input excerpts. The following excerpts retain the species counts from the two relevant passages.

> [. . .]
> Pistacia is a genus of flowering plants in the cashew family, Anacardiaceae. It contains 10 to 20 species [. . .]
> [. . .]
> The orchid genus Dracula, abbreviated as Drac in horticultural trade, consists of 118 species native to Mexico, Central America, Colombia, Ecuador and Peru.
> [. . .]

Question. Whiich genus has more species, Dracula or Pistacia? The spelling of the question is retained from the input.

Table 21: HotpotQA outputs for the species-count comparison.

Source	Answer
Reference	Dracula
Native	Dracula
POP	Pistacia
ACSSP-B	Dracula

Observation. The passages give 118 species for Dracula and 10-20 for Pistacia. Native and ACSSP-B select the genus with the larger count, whereas POP selects the other genus.