SPECROUTER: REUSABLE PAYOFF PREDICTION FOR COST-AWARE SPECULATIVE DECODING

Anonymous authors
Paper under double-blind review

ABSTRACT

Speculative decoding can obtain draft tokens in very different ways—copying from the context, n-gram lookup, a small draft model, or a learned tree drafter—and which drafter is fastest depends both on the text being generated and on the hardware and batch size that execute it. We introduce SpecRouter, which separates these two factors. A 0.58M-parameter predictor reads the target model's hidden state and estimates how many tokens each drafter would commit; a separately calibrated cost model prices each drafter on the current hardware; only the drafter with the lowest predicted cost per committed token runs. The predictor needs no exploration to train: with deterministic drafters and greedy verification, the output is the same whichever drafter runs, so every drafter's committed length at every prefix can be computed exactly offline. We evaluate on sixteen workloads with 100 held-out requests each. Averaged over workloads, SpecRouter speeds up decoding by $2.75\times$ on Llama-3.1-8B and $2.95\times$ on Qwen3-8B, which is 30% and 27% higher than the speedup of EAGLE-3, the strongest fixed drafter. It also exceeds the best fixed drafter chosen separately for each workload in hindsight. Comparing costs explicitly is the largest single source of this gain. SpecRouter also outperforms online methods that learn drafter choice during generation, reaching $4.34\times$ versus $3.96\times$ on a pool of domain-specialized drafters. A single predictor, never retrained, is the fastest policy on A40 and H200 GPUs at batch sizes 1, 4, and 16; only its cost model is recalibrated.

1 INTRODUCTION

Speculative decoding accelerates autoregressive generation by drafting tokens cheaply and verifying them in one target pass (Leviathan et al., 2023). Drafts can come from a standalone language model, a learned tree drafter such as EAGLE, or text copied from the prompt and the generated prefix (Li et al., 2024a; 2025; He et al., 2024; Hu et al., 2024). We call each such proposal mechanism in a fixed configuration, or target-only decoding, an *action*. Actions exploit different regularities: copying commits long spans when text repeats, while neural drafters handle new text but pay for every proposal. The fastest action changes across workloads and within a single generation (Appendix D.3), and it also depends on what each action costs to run.

Choosing an action therefore requires two kinds of knowledge that vary for different reasons. The *payoff* of an action—the number of tokens verification would commit if it proposed now—depends on the continuation. Inside a copied span SAM commits long runs, while in new text EAGLE-3 commits more. An action's *cost*—the time to propose, verify, and maintain state—depends on the hardware, the batch, and the cache state: at larger batch sizes, verifying a tree of drafted tokens is no longer nearly free. Existing selectors either switch between copying and a neural drafter with a threshold (SuffixDecoding, Oliaro et al., 2025; contemporaneous HVD, Su et al., 2026) or learn drafter choice online from verification feedback (BanditSpec, Hou et al., 2025; HedgeSpec, Liu et al., 2026). We ask a different question: can every action's payoff be predicted once from the target's hidden state, and then priced separately wherever the model is deployed?

The answer rests on a simple observation: committed tokens can be computed exactly offline, without exploration. With deterministic proposals and greedy verification, the target's output is the same whichever action runs, so at every prefix of a training generation we can evaluate every action and record how many tokens it would commit (Proposition 1). Verified trajectories already supply

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.

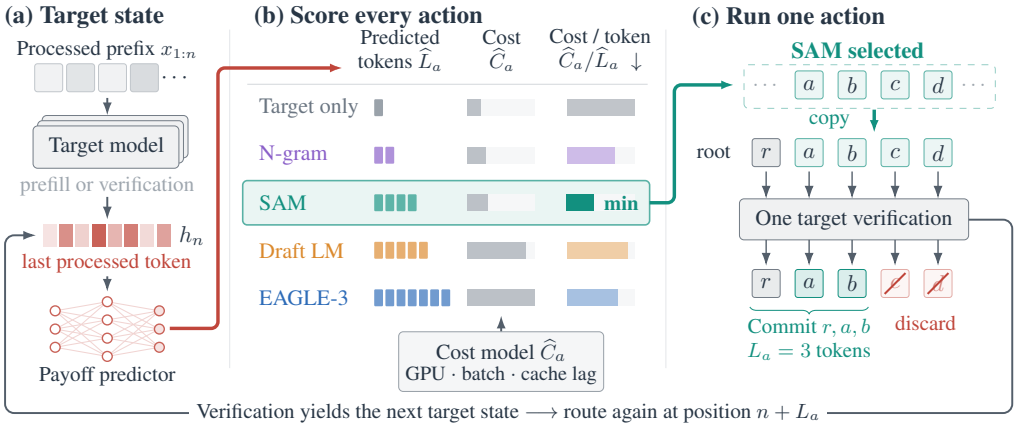


Figure 1: **SpecRouter scores every action and runs one.** (a) A small predictor maps the target's hidden state h_n to predicted committed tokens for every action. (b) A separately calibrated cost model prices each action, and the router picks the lowest predicted cost per committed token. (c) Only that action proposes; one target pass verifies it and yields the next state. Bars and tokens are illustrative.

counterfactual feedback for online drafter selection and replay labels for cache verification (Liu et al., 2026; Su et al., 2026); we use the property in its strongest form, as a complete label vector for supervised learning. SpecRouter trains a compact predictor on these labels. It reads the hidden state the target has already computed and estimates the committed tokens of every action in the pool—n-gram lookup, SAM copying, a standalone draft model, EAGLE-3, and target-only decoding. A separate cost model estimates each action's execution time in the current deployment, and only the action with the lowest predicted cost per committed token proposes (Figure 1).

Our contributions are:

1. **A formulation that separates payoff from cost.** An action's payoff—the tokens it would commit—depends on the continuation; its cost depends on the deployment. Under deterministic greedy verification, every action's committed tokens at every prefix are an exact offline label (Proposition 1), and the cost-per-committed-token rule is invariant to uniform cost rescaling (Section 2).
2. **Cost-aware routing that beats tuned hybrids and online selectors.** A 0.58M-parameter predictor over target hidden states, paired with a cost model, reaches $2.75\times$ on Llama-3.1-8B and $2.95\times$ on Qwen3-8B. The strongest threshold hybrid reaches $2.55\times$ and $2.76\times$. SpecRouter also exceeds the hindsight best fixed action on 15 of 16 Llama and 13 of 16 Qwen workloads (Tables 1 and 2), and it beats HedgeSpec on HedgeSpec's own expert pool ($4.34\times$ versus $3.96\times$; Table 3). Explicit cost comparison is the largest single factor in this gain, and target states bring additional actions into use (Table 4, Figure 2).
3. **A reusable payoff predictor.** A single checkpoint, never retrained, is the fastest policy on A40 and H200 GPUs at batch sizes 1, 4, and 16. Costs fitted at batch size one carry over to batch size four and to the other GPU; at batch size 16, recalibrating costs alone shifts the policy from EAGLE-3 to copying (Figure 3).

2 Payoff, Cost, and Exact Counterfactual Labels

After the target has processed a prefix $x_{1:n}$, its logits determine the next greedy token $r = x_{n+1}$, which begins the next verification block, and its hidden state h_n is available. An action a—a proposal mechanism in a fixed configuration, or target-only decoding—proposes a continuation after r. One target pass verifies r and the proposal, committing r together with the longest accepted prefix of the proposal. The number of committed tokens is the action's payoff, $L_a(x_{1:n}) \geq 1$; target-only decoding has $L = 1$. Let s denote the prefix together with execution state such as cache lag and batch composition, and let $C_a(s)$ be the time to propose, verify, and update state. The local objective is

cost per committed token,

$$q_a(s) = \frac{C_a(s)}{L_a(x_{1:n})}. \tag{1}$$

A long accepted draft is valuable only relative to the time needed to produce and verify it, so a short copied continuation can beat a longer neural proposal.

In general, an action reveals its committed tokens only after it has drafted and been verified, so the committed tokens of unselected actions are unobserved. Greedy verification removes this obstacle.

Proposition 1. *Suppose verification accepts exactly the proposal tokens that agree with the target's greedy continuation, and that each action's proposal is a deterministic function of the committed prefix, with drafter state synchronized to that prefix before proposing. Then every routing policy generates the same output, the greedy continuation $y^\star$, and at every prefix of $y^\star$ each action's payoff is determined by $y^\star$ and the action alone.*

Proof. Each block commits the target's greedy token r followed by proposal tokens that agree with the greedy continuation, so each block extends a prefix of $y^\star$ by tokens of $y^\star$. By induction over blocks, every policy generates $y^\star$. At a prefix $y^\star_{1:n}$, the proposal of action a is fixed by assumption, and its payoff is one plus the length of its longest agreement with $y^\star_{n+2:}$, which does not depend on the policy that reached the prefix. □

The proposition turns the estimation of committed tokens into supervised learning: a greedy generation per training request, with every action evaluated at each of its prefixes, yields the full vector $\mathbf{L}(x_{1:n})$, including actions a behavior policy would rarely select. It also separates committed tokens from execution history: a neural drafter's cache may lag because other actions were chosen, and catching up affects $C_a(s)$ but not L_a. Policies still reach different decision points and incur different maintenance costs, so minimizing Equation 1 at each decision is a local rule rather than a sequence-level optimum.

Given predicted committed tokens $\widehat{L}_a$ and costs $\widehat{C}_a$, SpecRouter chooses

$$a_t = \arg\min_{a \in \mathcal{A}} \frac{\widehat{C}_a(s_t)}{\max\big(1, \widehat{L}_a(x_{1:n_t})\big)}. \tag{2}$$

The rule compares actions with different proposal structures in a common unit, time per committed token, and requires no comparability between their drafts. It is also invariant to a common rescaling of costs: for any $k > 0$,

$$\arg\min_a \frac{k\,\widehat{C}_a(s)}{\widehat{L}_a(x_{1:n})} = \arg\min_a \frac{\widehat{C}_a(s)}{\widehat{L}_a(x_{1:n})}. \tag{3}$$

A faster GPU that accelerates every action by the same factor therefore leaves every decision unchanged. Only changes in relative cost alter decisions—batching, for instance, makes tree verification more expensive relative to target-only decoding—and these enter through $\widehat{C}_a$ without changing the payoff predictor. Figure 5 in Appendix C illustrates this geometry.

3 SpecRouter

SpecRouter has two components: a payoff predictor shared by all actions and a cost model for the current deployment. The predictor $f_\theta(h_n, z_n)$ outputs one committed-token estimate per action. For Llama, it reads the inputs to target layers 16 and 29, each projected to 32 dimensions. It also reads eight prefix statistics z_n, such as n-gram match length and generation position, and a two-hidden-layer MLP maps all inputs to the outputs (583,237 parameters; Appendix C). The inputs are by-products of target execution and prefix bookkeeping, so scoring the pool requires no additional forward pass, and no candidate proposal or future verification result is an input. A $1 + \mathrm{softplus}$ output link enforces $\widehat{L}_a \geq 1$, and the target-only output is fixed to one.

Offline collection evaluates every action at shared prefixes of the training requests and records the vector of committed tokens alongside causal features. The loss combines regression on committed tokens with a term that weights errors by the decisions they induce. With costs C_a, predicted ratios

$\widehat{q}_a = C_a / \max(1, \widehat{L}_a)$, their mean $\bar{q}$, and soft selections $\pi_a = \mathrm{softmax}_a(-\widehat{q}_a/(\tau\bar{q}))$, the loss for a labeled prefix is

$$\mathcal{L}(\theta) = \frac{\lambda_{\text{reg}}}{|\mathcal{A}|} \sum_a \mathrm{Huber}_\delta(\widehat{L}_a, L_a) + \sum_a \pi_a \frac{C_a}{\max(1, L_a)}. \tag{4}$$

The second term is the realized cost per token of the soft decision, following the predict-then-optimize view that predictions should be judged by the decisions they produce (Elmachtoub & Grigas, 2022); the first keeps outputs interpretable as token counts. Because a policy that commits k tokens decides again k positions later, it reaches only a policy-dependent subset of the labeled prefixes; two causal refitting rounds reweight training toward the decision points the current policy reaches, as in dataset aggregation (Ross et al., 2011).

An action's cost comprises proposal, verification, cache catch-up, and routing overhead. Unselected neural drafters keep their weights resident and catch up on the committed prefix when next selected, and this catch-up is charged to their cost. A routing charge common to all actions stays in each numerator; divided by different lengths, it does not cancel. In the primary executor, costs are linear in context length and cache catch-up, with shared slopes and per-workload intercepts fitted on training requests. A constant cost per action gives the same speedup ($2.75\times$, Table 4), and the held-out-workload experiment uses no workload information. The batched transfer executor fits each action's cost by non-negative least squares on context length, cache catch-up, and proposal length, with no workload identity.

In synchronous batches, requests keep separate tokens, caches, and verification state, while neural calls run as true tensor batches. One action is chosen per batch block by minimizing

$$\frac{\widehat{C}_a(s_{\mathcal{B}})}{\sum_{i \in \mathcal{B}} \widehat{L}_a\big(x^{(i)}_{1:n_i}\big)} \tag{5}$$

over the active requests $\mathcal{B}$, with the predictor evaluated per request and unchanged. Adding an action, or another configuration of an existing one, requires its labels, one output head, and a cost; the existing heads are unaffected.

4 Experimental Setup

Targets and workloads. We evaluate Llama-3.1-8B-Instruct and Qwen3-8B (Grattafiori et al., 2024; Yang et al., 2025) on the sixteen workloads and prompt sources of HVD (Su et al., 2026), which span document editing, code, tool use, SQL, multi-hop QA, long-document summarization, and chat. Each target is evaluated on 100 test requests per workload (HVD reports 20), 1,600 in total, disjoint from the requests used for fitting and validation. Decoding is greedy with a 256-token output limit. The Llama predictor is trained on 10,000 source-balanced requests. Appendix B gives the hardware, precision, checkpoints, and predictor layers for both targets.

Action pool. The pool contains request-local n-gram lookup (N-gram, up to eight tokens) and a request-local suffix copier in the style of SAM (SAM, up to 64 tokens). It also contains a small standalone draft model proposing eight tokens (Draft LM), EAGLE-3 with a $32/8/4$ tree, and target-only decoding. Fixed policies and routers share action implementations.

Baselines. EAGLE-3 is the fastest single action on average for both targets. The SuffixDecoding-style threshold uses SAM when its match score exceeds a validation-tuned threshold and EAGLE-3 otherwise (Oliaro et al., 2025). The HVD-style threshold follows HVD's decision structure—a threshold on a learned copy payoff with an EAGLE fallback (Su et al., 2026)—using SpecRouter's SAM head.[1] UCB and EXP3 follow BanditSpec's published equations, with committed length as reward and statistics reset per request (Hou et al., 2025). HedgeSpec (Liu et al., 2026) is compared on its own released expert pool.

Measurement campaigns. Results come from four campaigns, each internally matched; we compare numbers only within a campaign. (i) The primary campaign (Tables 1, 2, and 4; Figure 2) measures decode-only speedup at batch size one with a 256-token cap. For workload w and policy p, $S_{w,p} =$

[1] HVD's implementation was not public when our experiments were run, and its reported results use a different runtime on H100 GPUs, so we compare decision structures within one executor.

Table 1: **SpecRouter is the fastest deployable selector on both targets and exceeds the per-workload best fixed action chosen in hindsight.** Primary campaign: macro decode speedup over target-only on 16 workloads, with 95% confidence intervals. Italic rows use test-time hindsight and are not deployable (Section 4).

Method	Llama-3.1-8B	Qwen3-8B
EAGLE-3 (best single action)	2.12 [2.10, 2.14]	2.33 [2.31, 2.35]
SuffixDecoding-style threshold	2.52 [2.50, 2.55]	2.74 [2.73, 2.74]
HVD-style threshold	2.55 [2.22, 2.26]	2.76 [2.71, 2.79]
SpecRouter	**2.75** [2.73, 2.78]	**2.95** [2.92, 2.98]
Workload-best fixed (hindsight)	*2.53*	*2.71*
Local hindsight reference	*3.08*	*3.31*

$T_{w,\mathrm{AR}}/T_{w,p}$, where T sums decode time over the workload's requests, including routing, proposal, verification, and state maintenance and excluding prefill. We report the unweighted mean over workloads (macro); Appendix A gives geometric means and medians. Three training seeds each receive three timing repetitions, and 95% confidence intervals reflect both sources of variation. (ii) A separately remeasured online-selection campaign (Table 3, first row) runs UCB, EXP3, EAGLE-3, and SpecRouter together in the same five-action Llama executor. (iii) The expert-pool campaign (Table 3, second row) uses HedgeSpec's released EAGLE experts, its seven-domain data, and a 512-token cap. (iv) The deployment campaign (Figure 3) runs a fixed Llama checkpoint in a synchronous batched executor on A40 and H200 GPUs at batch sizes 1, 4, and 16. It measures pooled request-inclusive speedup, including prefill and setup, on the same 1,600 requests with three timing repetitions.

Hindsight references. Two references use test-time information and are not deployable. *Workload-best fixed* picks, for each workload, the fastest single action after measuring all of them. The *local hindsight reference* picks, at each block it reaches, the action with the lowest realized cost per committed token; it knows realized payoffs but not future costs, so it measures remaining local headroom rather than a sequence-level upper bound.

5 Results

5.1 SpecRouter is the fastest deployable selector

SpecRouter is the fastest deployable selector on both targets (Table 1). It reaches $2.75\times$ on Llama-3.1-8B and $2.95\times$ on Qwen3-8B. Fixed EAGLE-3, the strongest single action, reaches $2.12\times$ and $2.33\times$; SpecRouter's macro speedup is 30% higher on Llama and 27% higher on Qwen. The strongest hybrid, the HVD-style threshold, reaches $2.55\times$ and $2.76\times$. The SuffixDecoding-style threshold alone roughly matches the hindsight workload-best action on Llama ($2.52\times$ versus $2.53\times$), so the threshold hybrids are strong baselines. SpecRouter beats it on 15 of 16 Llama and 13 of 16 Qwen workloads. The advantage over the hindsight workload-best action also holds for the geometric mean, for the median, and without FineEdit (Appendix A). Scoring the pool costs 0.37 ms per block (0.66% of decode time), and all timings include it.

It exceeds the best fixed action chosen in hindsight, most where actions are complementary (Table 2). The fastest fixed action changes across workloads: on Llama it is SAM on six workloads, the Draft LM on one, and EAGLE-3 on nine. SpecRouter beats this per-workload choice on 15 of 16 Llama and 13 of 16 Qwen workloads. The gains are concentrated: 14–43% on six Llama workloads (Delegate-52, FineEdit, RepoBench, SWE-Bench, BFCL, MuSiQue), where several actions are competitive. On the other ten, one action dominates, and SpecRouter matches it with a gain below 5%; on Qwen, too, the largest gains are on editing, code, and tool use. Routing also avoids EAGLE-3's long-context degradation (Lee et al., 2025): on RepoBench, InfiniteBench, and GovReport, fixed EAGLE-3 reaches $1.00\times$, $0.89\times$, and $0.84\times$ on Llama. SpecRouter reaches $2.15\times$, $1.19\times$, and $1.73\times$ on the same workloads. Averaging per-workload gains over EAGLE-3, rather than comparing macro speedups, gives 51% on Llama, but the median gain is 9%: SpecRouter more than doubles EAGLE-3's speedup on FineEdit, Delegate-52, RepoBench, and GovReport (Table 13). On Qwen, the mean and median gains are 27% and 5%.

Table 2: **No fixed action is best everywhere; SpecRouter exceeds the best fixed action on 15 of 16 Llama and 13 of 16 Qwen workloads.** Primary campaign: decode speedup over target-only for each workload; the fastest fixed action is underlined. The local hindsight reference is not deployable (Section 4).

	Llama-3.1-8B						Qwen3-8B					
Workload	N-gram	SAM	Draft LM	EAGLE-3	SpecRouter	Local hindsight	N-gram	SAM	Draft LM	EAGLE-3	SpecRouter	Local hindsight
Delegate-52	1.52	1.88	1.71	1.23	**2.68**	3.17	0.95	1.40	1.26	2.44	**2.61**	2.94
FineEdit	2.93	4.74	2.10	1.42	**5.85**	6.49	3.14	8.05	0.82	2.45	**10.11**	11.18
RepoBench	1.43	1.87	1.64	1.00	**2.15**	2.53	1.04	1.57	1.31	1.47	**1.94**	2.46
SWE-Bench	1.21	1.32	1.32	1.87	**2.13**	2.50	1.05	1.23	1.02	2.12	**2.23**	2.51
Magicoder	1.29	1.29	1.95	3.14	**3.27**	3.62	1.15	1.22	1.44	2.58	**2.64**	2.98
MBPP	1.30	1.30	2.22	3.52	**3.59**	3.88	1.36	1.36	1.79	2.80	**3.00**	3.35
BFCL	0.90	1.32	1.38	1.06	**1.63**	1.90	0.74	1.29	1.43	1.96	**2.25**	2.62
Spider	1.09	1.13	1.90	2.04	**2.10**	2.54	1.08	1.13	1.71	1.37	**1.87**	2.21
HotpotQA	1.65	1.79	1.23	3.25	**3.35**	3.60	1.48	1.72	1.08	2.89	**3.00**	3.40
MuSiQue	1.38	1.67	1.29	1.40	**1.93**	2.32	1.07	1.58	1.00	2.70	**2.75**	3.06
2WikiMHQA	1.59	1.61	1.37	3.05	**3.16**	3.35	1.61	1.65	1.39	2.81	**2.92**	3.31
InfiniteBench	0.72	1.16	0.96	0.89	**1.19**	1.46	0.64	1.18	0.89	2.00	**2.21**	2.42
CNN/DM	1.23	1.38	1.20	2.51	**2.54**	2.82	1.12	1.28	0.80	2.13	**2.13**	2.41
GovReport	1.13	1.66	1.49	0.84	**1.73**	1.98	0.69	1.37	0.72	2.64	**2.62**	2.83
Alpaca	1.12	1.12	1.65	3.40	**3.36**	3.49	1.10	1.11	1.07	2.63	**2.61**	2.77
MT-Bench	1.22	1.26	1.75	3.26	**3.34**	3.63	1.11	1.14	1.00	2.34	**2.38**	2.55
Mean	1.36	1.66	1.57	2.12	**2.75**	3.08	1.21	1.77	1.17	2.33	**2.95**	3.31

Table 3: **SpecRouter outperforms online selectors, including HedgeSpec on its own expert pool.** Macro decode speedup on Llama-3.1-8B; each row is a separately matched campaign (Section 4). HedgeSpec's domain-matched expert, which needs domain labels, reaches 4.36 on its pool. †Not defined for this pool (App. E).

Action pool	EAGLE-3	UCB	EXP3	HedgeSpec	SpecRouter	*Local hindsight*
Five actions (16 workloads)	2.16	1.80	1.73	n/a†	**2.77**	*3.09*
HedgeSpec's experts (7 domains)	3.47	3.10	2.99	3.96	**4.34**	*4.42*

It also outperforms online selectors, including HedgeSpec on its own expert pool (Table 3). In the remeasured pool, UCB and EXP3 reach $1.80\times$ and $1.73\times$, below fixed EAGLE-3 ($2.16\times$). With statistics reset at every request, they spread their blocks almost evenly over the actions: EAGLE-3 receives only 21% of them, and the Draft LM, which commits the most tokens per block but is the most expensive action, receives the most (Appendix E). They maximize committed length, following BanditSpec, so the comparable SpecRouter variant is the cost-blind one in Table 4 (longest predicted commitment, $2.31\times$): under the same objective, target-state prediction still outperforms online feedback, and comparing costs supplies the remaining gain. (This comparison spans two campaigns, which differ by about 2% on fixed EAGLE-3.) On HedgeSpec's released experts and data, SpecRouter reaches $4.34\times$ versus $3.96\times$ for HedgeSpec and wins on six of seven domains; Python is the exception. It comes within $0.02\times$ of the domain-matched expert, although its payoff predictor uses no domain labels.

5.2 Cost comparison drives the gain

Comparing costs is the largest single factor (Table 4). Choosing the action with the longest predicted commitment, ignoring cost, drops speedup by 16%, from $2.75\times$ to $2.31\times$; no other change costs as much. Deciding with a threshold on the predicted SAM commitment, as the HVD-style baseline does, instead of comparing costs, loses 7.3%. The threshold is also slower on Qwen ($2.76\times$ versus $2.95\times$; Table 1).

Routing must repeat within a request. Keeping the first decision for the whole request gives $2.46\times$ ($2.66\times$ on Qwen), below the hindsight workload-best fixed action.

Target states sharpen routing and bring more actions into use (Figure 2). With prefix statistics alone, the router learns copy-versus-EAGLE arbitration ($2.59\times$) but never selects N-gram or the

Table 4: **Comparing costs matters most, and routing must repeat within a request; target states and both loss terms help, while a constant cost per action suffices.** Primary campaign, Llama-3.1-8B: macro decode speedup and its change relative to SpecRouter. Extension rows test generalization rather than ablation (per-workload values in App. D).

Component	Change from SpecRouter	Speedup	Change
—	SpecRouter (5 actions, target states + statistics, cost rule)	**2.75**	—
Decision rule	Longest predicted commitment (ignores cost)	2.31	−16.0%
	HVD-style threshold on the SAM head	2.55	−7.3%
Granularity	Route once per request	2.46	−10.5%
Predictor inputs	Prefix statistics only	2.59	−5.8%
	Target states only	2.69	−2.2%
Cost model	Static per-action costs	2.75	−0.1%
	No workload intercepts	2.74	−0.4%
Training	Regression loss only	2.68	−2.5%
	Decision loss only	2.63	−4.4%
	No causal refitting	2.66	−3.3%
Extension	Workload held out of training (workload-agnostic costs)	2.59	−5.8%
	+ EAGLE-3 60/7/10 tree as a sixth action	2.81	+2.2%

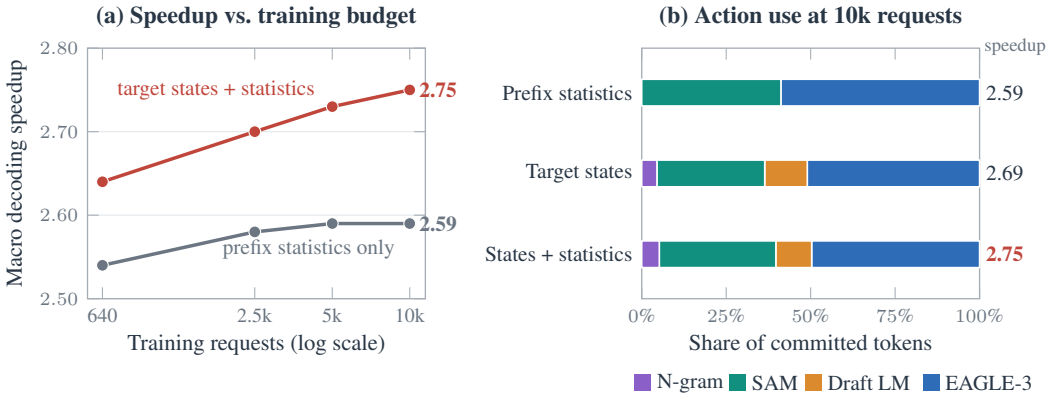


Figure 2: **Target states sharpen routing and bring N-gram and the Draft LM into use.** Primary campaign, Llama-3.1-8B; the states-plus-statistics model is SpecRouter. (a) Macro decode speedup versus training requests. (b) Share of committed tokens by action at 10,000 training requests, with macro speedup at right.

Draft LM. With target states added, those actions commit 16% of tokens and speedup reaches 2.75× (Figure 2b). The advantage holds at every training budget, while the statistics-only predictor levels off after 2,500 requests (Figure 2a). Prefix statistics resemble HVD's cache and position features, so the statistics-only row also approximates HVD's inputs under our cost rule.

The cost model can be simple, and both loss terms help. A constant cost per action matches the fitted model (2.75×), and removing workload intercepts gives 2.74×, so neither workload identity nor state-dependent costs are needed in this executor. Regression alone and the decision term alone lose 2.5% and 4.4%. Without causal refitting, speedup falls 3.3%, and one refitting round already recovers most of it (2.73×; Appendix D).

The predictor extends beyond its training workloads and action pool. Trained with each workload held out in turn and with workload-agnostic costs, SpecRouter reaches 2.59×, still above the hindsight workload-best fixed action (2.53×). Adding EAGLE-3's larger 60/7/10 tree as another action raises speedup to 2.81×; the tree alone reaches 2.16×. The extension needs only the new action's labels, an output head, and a cost, and it leaves the existing heads unchanged. Masking actions at serving time shows that their value is unequal: removing N-gram barely matters, while removing SAM or EAGLE-3 costs much more (Appendix D.2).

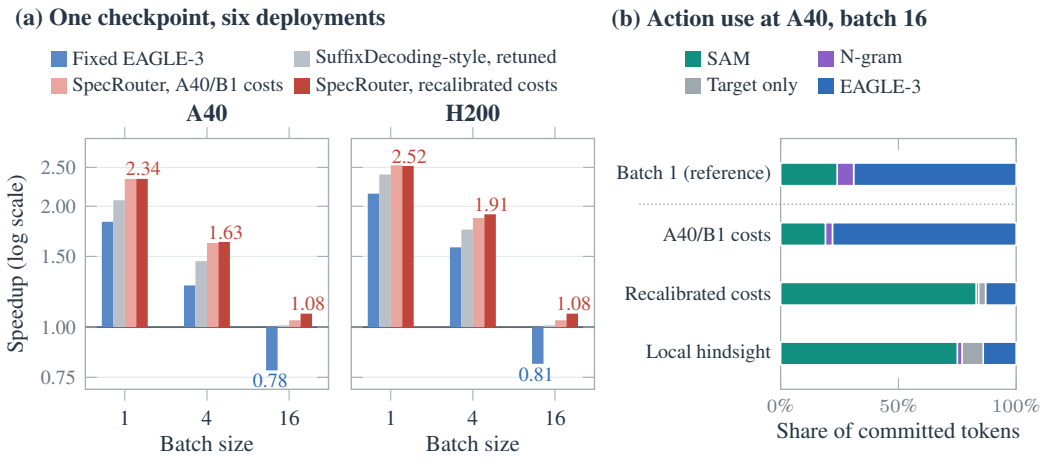


Figure 3: **One payoff predictor, never retrained, is the fastest policy in all six deployments; at batch size 16, recalibrating costs alone shifts the policy toward SAM.** Deployment campaign: Llama-3.1-8B, pooled request-inclusive speedup over target-only at the same GPU and batch size. (a) Bars grow from $1\times$, the best fixed policy at batch size 16. (b) Share of committed tokens at A40, batch size 16. Full grid: App. F.

5.3 One payoff predictor transfers across deployments

The payoff predictor transfers unchanged, and costs carry over when relative costs are stable (Figure 3a). With either cost fit, SpecRouter is faster than the best fixed action and the retuned SuffixDecoding-style threshold in every setting. Costs fitted once on the A40 at batch size one also serve batch size four and the H200: recalibrating them changes speedup by at most 2.1%. Equation 3 predicts this. At these batch sizes, the ratio of the fitted EAGLE-3 and target-only cost intercepts stays between 1.50 and 1.76 on both GPUs, so decisions barely change.

When relative costs change, recalibrating costs alone changes the policy (Figure 3b). At batch size 16, verification is no longer nearly free: the cost ratio rises to 2.30 on the A40 and 2.25 on the H200, and fixed EAGLE-3 becomes slower than target-only decoding. With the A40/B1 costs, SpecRouter still speeds up decoding ($1.04\times$) but, on the A40, keeps committing 78% of tokens through EAGLE-3. With recalibrated costs, the same payoff predictions shift 83% of tokens to SAM, close to the local hindsight reference's 75%. They reach $1.08\times$ on both GPUs, versus $1.01\times$ for the retuned SuffixDecoding-style threshold.

6 Related Work

Draft models, Medusa, and EAGLE learn to propose (Leviathan et al., 2023; Cai et al., 2024; Li et al., 2024a;b; 2025), while reference-based decoding, REST, SAM, and lookahead decoding reuse or construct continuations without a separate model (Yang et al., 2023; He et al., 2024; Hu et al., 2024; Fu et al., 2024). Token trees (Miao et al., 2024; Chen et al., 2024b) and adaptive draft lengths and trees (Mamou et al., 2024; Huang et al., 2025; Zhang et al., 2024) shape speculation within one mechanism. All of these can supply actions to SpecRouter, including several configurations of one drafter.

SuffixDecoding and SAM decoding fall back from copying to a neural drafter when the match is weak (Oliaro et al., 2025; Hu et al., 2024), and cascade drafting chains draft models of different sizes (Chen et al., 2024c). Contemporaneous HVD learns the accepted length of cached proposals and thresholds that estimate to arbitrate cache verification against EAGLE (Su et al., 2026). SpecRouter instead predicts every action's committed tokens from shared target state and compares them through separately supplied execution costs; HVD also studies multiple GPUs and a serving prototype, whereas our transfer study holds predictor weights fixed.

BanditSpec, MetaSD, and HedgeSpec select drafters online from verification feedback (Hou et al., 2025; Kim et al., 2026; Liu et al., 2026), and HedgeSpec observes that a verified trajectory can supply counterfactual information about unselected drafters. SpecRouter amortizes related information offline into a predictor conditioned on current target state, so only one action executes at inference. The distinction is the information and cost model available at decision time, not a restriction of online methods to homogeneous pools. An earlier method also named SpecRouter selects multi-level draft/verifier chains (Wu et al., 2025); ours selects a proposal mechanism for a fixed target.

Model routers such as FrugalGPT and RouteLLM trade response quality against inference cost (Chen et al., 2024a; Ong et al., 2025). Speculative routing keeps the target fixed, so quality is never at stake; the decision recurs at every block, and its payoff can be supervised exactly. Our objective follows the predict-then-optimize view that predictors should be trained for the decisions they inform (Elmachtoub & Grigas, 2022).

7 Limitations

Our primary study uses greedy decoding, 8B targets, outputs of up to 256 tokens (512 in the expert pool), request-local copying, and synchronous batches that share a single action per block. Under sampling, committed lengths become random and labels become expected committed tokens; cross-request suffix caches, continuous batching with per-request actions, and larger tensor-parallel targets are natural extensions of the same interface. Exact labels and transfer rely on proposals and verification decisions that are reproducible across backends: a change that alters an action's proposals requires relabeling that action (Appendix F), and a numerical change that alters the target's greedy decisions changes the reference continuation itself. Offline collection evaluates every action at every training prefix: the Llama predictor uses 1,648,541 dense counterfactual rows from 10,000 requests, about 120 A40 GPU-hours. Keeping neural drafters resident adds about 3.3 GB of weights at inference, and cost recalibration uses 160 training requests per setting. The ablation, held-out-workload, and deployment analyses use Llama only; the bandit baselines use BanditSpec's committed-length reward, which does not account for cost; and results come from four separately timed campaigns (Section 4).

8 Conclusion

SpecRouter separates what an action would commit, which depends on the continuation, from what it costs, which depends on the deployment. Deterministic greedy verification makes every action's committed tokens an exact offline label, so one target-state predictor scores every action while only one runs. With explicit cost comparison, the resulting router is faster than tuned threshold hybrids and online selectors on two targets and sixteen workloads. One unchanged payoff predictor is also the fastest policy across six deployments, with only its costs recalibrated.

Reproducibility Statement

Section 4 and Appendices B and C specify the workloads, action configurations, predictor architecture, training hyperparameters, cost models, timing definitions, and baseline implementations. The four measurement campaigns use distinct, explicitly defined aggregation protocols, and every reported speedup includes the work performed by the evaluated selector.

AI Use Statement

AI-assisted tools supported software development, literature checking, manuscript review, writing, and polishing. The authors are responsible for the resulting code, evidence, citations, and claims.

References

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. Medusa: Simple llm inference acceleration framework with multiple decoding heads. *arXiv preprint arXiv:2401.10774*, 2024.

Lingjiao Chen, Matei Zaharia, and James Zou. FrugalGPT: How to use large language models while reducing cost and improving performance. *Transactions on Machine Learning Research*, 2024a. URL https://arxiv.org/abs/2305.05176.

Zhuoming Chen, Avner May, Ruslan Svirschevski, Yuhsun Huang, Max Ryabinin, Zhihao Jia, and Beidi Chen. Sequoia: Scalable, robust, and hardware-aware speculative decoding. *arXiv preprint arXiv:2402.12374*, 2024b. URL https://arxiv.org/abs/2402.12374.

Ziyi Chen, Xiaocong Yang, Jiacheng Lin, Chenkai Sun, Kevin Chen-Chuan Chang, and Jie Huang. Cascade speculative drafting for even faster LLM inference. In *Advances in Neural Information Processing Systems*, 2024c.

Adam N. Elmachtoub and Paul Grigas. Smart "predict, then optimize". *Management Science*, 68(1): 9–26, 2022.

Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. Break the sequential dependency of llm inference using lookahead decoding. *arXiv preprint arXiv:2402.02057*, 2024.

Aaron Grattafiori et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Zhenyu He, Zexuan Zhong, Tianle Cai, Jason D. Lee, and Di He. REST: Retrieval-based speculative decoding. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics*, 2024.

Yunlong Hou, Fengzhuo Zhang, Cunxiao Du, Xuan Zhang, Jiachun Pan, Tianyu Pang, Chao Du, Vincent Y. F. Tan, and Zhuoran Yang. BanditSpec: Adaptive speculative decoding via bandit algorithms. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025.

Yuxuan Hu, Ke Wang, Xiaokang Zhang, Fanjin Zhang, Cuiping Li, Hong Chen, and Jing Zhang. SAM decoding: Speculative decoding via suffix automaton. *arXiv preprint arXiv:2411.10666*, 2024.

Kaixuan Huang, Xudong Guo, and Mengdi Wang. SpecDec++: Boosting speculative decoding via adaptive candidate lengths. In *Conference on Language Modeling*, 2025.

Taehyeon Kim, Hojung Jung, and Se-Young Yun. Multi-drafter speculative decoding with alignment feedback. *arXiv preprint arXiv:2604.05417*, 2026.

Jaeseong Lee, Seung-won Hwang, Aurick Qiao, Gabriele Oliaro, Ye Wang, and Samyam Rajbhandari. OWL: Overcoming window length-dependence in speculative decoding for long-context inputs. *arXiv preprint arXiv:2510.07535*, 2025. URL https://arxiv.org/abs/2510.07535.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE: Speculative sampling requires rethinking feature uncertainty. *arXiv preprint arXiv:2401.15077*, 2024a.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE-2: Faster inference of language models with dynamic draft trees. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 7421–7432, 2024b.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE-3: Scaling up inference acceleration of large language models via training-time test. *arXiv preprint arXiv:2503.01840*, 2025.

Hongyi Liu, Jiaji Huang, Zhen Jia, Youngsuk Park, and Yu-Xiang Wang. Not-a-bandit: Provably no-regret drafter selection in speculative decoding for llms. In *International Conference on Learning Representations*, 2026.

Jonathan Mamou, Oren Pereg, Daniel Korat, Moshe Berchansky, Nadav Timor, Moshe Wasserblat, and Roy Schwartz. Dynamic speculation lookahead accelerates speculative decoding of large language models. In *Proceedings of the 4th NeurIPS Efficient Natural Language and Speech Processing Workshop*, volume 262 of *Proceedings of Machine Learning Research*, pp. 456–467, 2024.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, Chunan Shi, Zhuoming Chen, Daiyaan Arfeen, Reyna Abhyankar, and Zhihao Jia. SpecInfer: Accelerating large language model serving with tree-based speculative inference and verification. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 2024.

Gabriele Oliaro, Zhihao Jia, Daniel Campos, and Aurick Qiao. SuffixDecoding: Extreme speculative decoding for emerging ai applications. In *Advances in Neural Information Processing Systems*, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/b7aea253ab34a773967f1e4cdea9e4fb-Abstract-Conference.html.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M. Waleed Kadous, and Ion Stoica. RouteLLM: Learning to route LLMs with preference data. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=8sSqNntaMr.

Stephane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pp. 627–635, 2011. URL https://proceedings.mlr.press/v15/ross11a.html.

Xin Su, Dawid Majchrowski, Fangyuan Yu, Vanshil Atul Shah, Sebastian Rogawski, Pawel Morkisz, Anahita Bhiwandiwalla, and Phillip Howard. Hybrid verified decoding: Learning to allocate verification in speculative decoding. *arXiv preprint arXiv:2606.01019*, 2026.

Hang Wu et al. SpecRouter: Adaptive routing for multi-level speculative decoding in large language models. *arXiv preprint arXiv:2505.07680*, 2025.

An Yang et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Nan Yang, Tao Ge, Liang Wang, Binxing Jiao, Daxin Jiang, Linjun Yang, Rangan Majumder, and Furu Wei. Inference with reference: Lossless acceleration of large language models. *arXiv preprint arXiv:2304.04487*, 2023.

Situo Zhang, Hankun Wang, Da Ma, Zichen Zhu, Lu Chen, Kunyao Lan, and Kai Yu. AdaEAGLE: Optimizing speculative decoding via explicit modeling of adaptive draft structures. *arXiv preprint arXiv:2412.18910*, 2024.

Appendix roadmap. Appendix A gives alternative summaries (Table 5) and a log-scale view (Figure 4) of the per-workload results in Table 2. Appendix B documents data, implementation, and measurement: workload provenance (Table 6) and the Llama and Qwen configurations (Tables 7 and 8). Appendix C illustrates the decision rule (Figure 5) and details the predictor, the threshold and route-once controls, and the cost models of Section 3. Appendix D supports Table 4 and Figure 2: per-workload controls and the six-action extension (Table 9), predictor inputs and training budget (Tables 10 and 11), held-out workloads (Table 12), prediction diagnostics, selection opportunity by granularity (Table 13), serving-time action masks (Table 14), and routing traces (Figure 6). Appendix E supports Table 3: the matched five-action run (Tables 15 and 16) and the per-domain expert-pool results (Table 17). Appendix F supports Figure 3 with the full transfer grid (Table 18) and a backend that changes proposals.

A Complete Primary Workload Grid

Table 2 in Section 5.1 reports the primary-campaign fixed-action and routing results for every workload on both targets; Table 1 and the summaries below are computed from it. Macro denotes an unweighted mean over workloads, not a pooled throughput ratio. The local hindsight reference is defined in Section 4. The fastest fixed action is SAM on six Llama workloads, the Draft LM on one, and EAGLE-3 on nine; on Qwen, EAGLE-3 leads on thirteen workloads, SAM on two, and the Draft LM on one.

Aggregation across workloads. Table 5 summarizes the workload entries in Table 2. The geometric mean is $\exp(|\mathcal{W}|^{-1}\sum_w \log S_{w,p})$; medians are taken across workload speedups. These summaries describe the workload grid, computed at recorded precision, rather than uncertainty across training seeds. Workload-best fixed takes the fastest fixed action separately on each workload before aggregation. The router exceeds that comparator on 15 of 16 Llama workloads and 13 of 16 Qwen workloads; Qwen has one tie at displayed precision and two losses. The median workload-level gain over workload-best fixed is 3.81% on Llama and 4.55% on Qwen. Over the same grid, the geometric mean of the local hindsight reference is $2.90\times$ on Llama and $3.02\times$ on Qwen, and that of the SuffixDecoding-style threshold on Llama is $2.31\times$.

Table 5: **SpecRouter's advantage over workload-best fixed holds under every summary, including without FineEdit.** Primary campaign; decode speedup over target-only, computed from Table 2.

Target	Summary	EAGLE-3	Workload-best fixed	SpecRouter
Llama	Arithmetic mean	2.12	2.53	2.75
	Geometric mean	1.87	2.35	2.56
	Median	1.96	2.28	2.61
	Mean excluding FineEdit	2.16	2.38	2.54
Qwen	Arithmetic mean	2.33	2.71	2.95
	Geometric mean	2.28	2.50	2.68
	Median	2.45	2.51	2.61
	Mean excluding FineEdit	2.33	2.35	2.48

B Data, Implementation, and Measurement

The evaluation combines actions with different proposal structures and cache requirements. This section specifies what is shared, what is measured, and how repeated test measurements are aggregated.

B.1 Data provenance and split

We use the HVD workload suite and prompt sources (Su et al., 2026). Each target uses 100 test requests per workload, for 1,600 requests in total. Requests are fixed by identifier, not selected by latency, and stop at the first EOS or the 256-token limit. Upstream dataset identifiers and source-specific prompt construction follow the sources listed below.

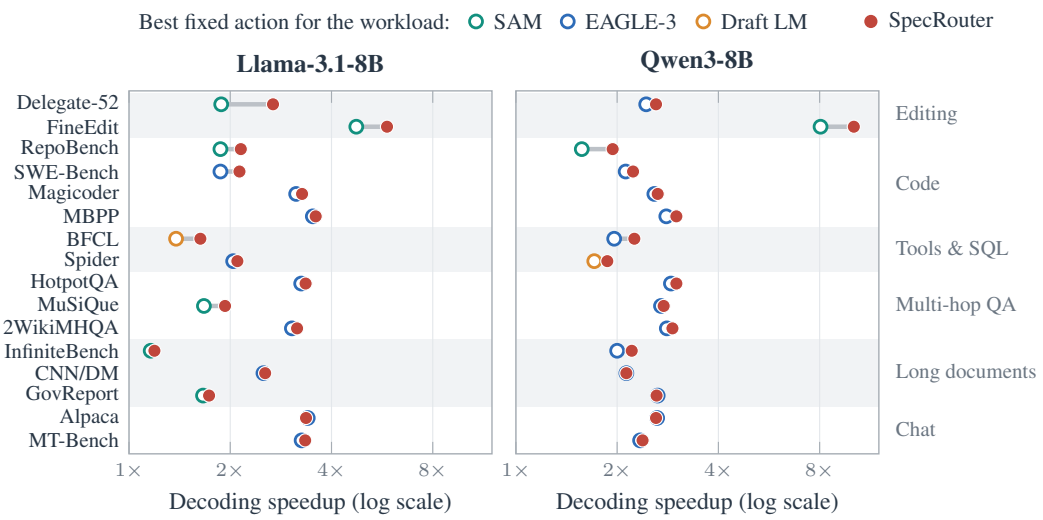


Figure 4: **No fixed action is best everywhere, and SpecRouter exceeds the best one on most workloads.** Log-scale view of Table 2 (primary campaign). Hollow markers: fastest fixed action per workload, colored by action; filled markers: SpecRouter.

Table 6: **Every workload contributes 100 test requests per target.** Workload provenance and test population (primary campaign). Identifiers name upstream datasets; the HVD source manifests fix the prompt construction.

Workload	Upstream dataset identifier	Requests
Delegate-52	microsoft/delegate52	100
FineEdit	YimingZeng/FineEdit_bench	100
RepoBench	tianyang/repobench_python_v1.1	100
SWE-Bench	SWE-bench/SWE-bench_Verified	100
Magicoder	ise-uiuc/Magicoder-Evol-Instruct-110K	100
MBPP	google-research-datasets/mbpp	100
BFCL	llamastack/bfcl_v3	100
Spider	NBAmine/xlangai-spider-with-context	100
HotpotQA, MuSiQue, 2WikiMHQA	HVD multi-hop QA source manifests	100 each
InfiniteBench, CNN/DM, GovReport	HVD long-document source manifests	100 each
Alpaca, MT-Bench	HVD instruction/chat source manifests	100 each

The Llama predictor is fitted on 10,000 source-balanced requests, with 1,648,541 dense counterfactual rows available and 424,446 selected training rows in the seed-7 configuration. A separate source-stratified validation fold supplies checkpoint-selection data, with 32,713 selected validation rows. Test requests are excluded from fitting and checkpoint selection. Training, validation, and test requests are drawn from the same workload distributions but are disjoint.

B.2 Action implementations and runtime

Table 7: **Llama-3.1-8B runtime and action configuration.** Primary campaign. Fixed policies and the router use the same action implementations.

Component	Configuration
Target	Llama-3.1-8B-Instruct, FP16, greedy, batch one
Hardware/software	A40 (48 GB); PyTorch 2.6.0, Transformers 4.53.3, CUDA 12.4; Python 3.10.20
EAGLE-3	EAGLE3-LLaMA3.1-Instruct-8B, FP16; 32-token proposal budget, depth parameter 8, top-k 4
Draft LM	Llama-3.2-1B, BF16, eight draft tokens
N-gram	Request-local unindexed scan; maximum proposal length eight
SAM	Native request-local adaptive suffix-match copying; linear proposal, cap 64, speculation factor 1, offset 0, minimum token probability 0.1
Prefill	Draft-LM prompt prefill overlaps target prefill when active
Predictor inputs	Inputs to target layers 16 and 29 (each 4,096 $\rightarrow$ 32) and eight prefix statistics

Table 8: **Qwen3-8B runtime and action configuration.** Primary campaign, in the same format as Table 7. Fixed policies and the router use the same action implementations.

Component	Configuration
Target	Qwen3-8B, BF16, greedy, batch one
Hardware/software	A40 (48 GB); PyTorch 2.6.0, Transformers 4.53.3, CUDA 12.4; Python 3.10.20
EAGLE-3	AngelSlim/Qwen3-8B_eagle3, BF16; 32-token proposal budget, depth parameter 8, top-k 4
Draft LM	Qwen3-0.6B, BF16, eight draft tokens
N-gram	Request-local unindexed scan; maximum proposal length eight
SAM	Native request-local adaptive suffix-match copying; linear proposal, cap 64, speculation factor 1, offset 0, minimum token probability 0.1
Prefill	Draft-LM prompt prefill overlaps target prefill when active
Predictor inputs	Inputs to target layers 18 and 33 (each 4,096 $\rightarrow$ 32) and eight prefix statistics; 10,000 source-balanced training requests

The SAM action is a suffix-copy implementation, not a reproduction of every published SAM or SuffixDecoding variant. EAGLE uses the stated shared tree configuration; the fixed baseline is that implementation rather than the best workload-specific tree.

B.3 Generation and verification

Each policy runs the complete generation loop with native target verification. The target verifies the selected proposal and determines the accepted continuation and next root. Generation stops at EOS or the output limit. Wall-clock measurements include the proposal, verification, and state updates performed along each policy's generation.

B.4 Timing and aggregation

Each action path and policy receives an eight-token warm-up pass that is excluded from timing. Policies are paired by request in rotating order, and request boundaries are synchronized. Decode cost sums recorded decision, proposal, verification, commit, and postprocessing intervals. These host intervals include enqueue and implicit wait time; component values are not isolated GPU-kernel latencies. Request-inclusive time also charges target/draft prefill, setup, and cleanup. Loading, compilation, and warm-up are excluded.

Primary Llama timing workers each use an individually checked A40. Continuous sampling rejects attempts with foreign GPU occupancy; all valid measurements in the retained attempt are kept. No

fastest-repetition selection is performed. Each of the three training seeds is evaluated in three timing repetitions; reported seed-level values average those repetitions. Confidence intervals account for both training-seed variation and repeated timing measurements. Because devices differ, fixed actions are remeasured alongside new selectors rather than combined from different runs.

For fixed policy a, $T_{w,a} = \sum_{i \in w} T_{i,a}$. Workload-best fixed has cost $\min_a T_{w,a}$, including target-only among the available actions. Seed-level statistics average the three repetitions, and the three-seed summary averages the resulting seed-level statistics.

C Predictor and Integration Details

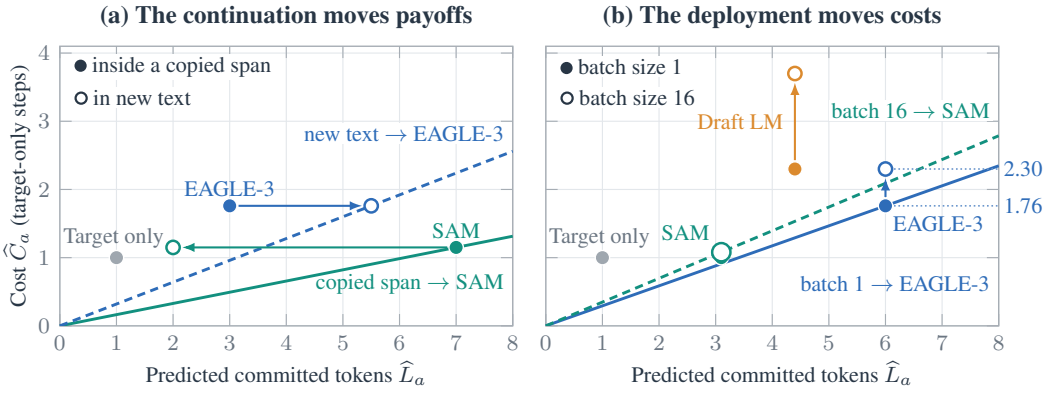


Figure 5: **The cost-per-committed-token rule as geometry: the lowest ray from the origin wins.** Each action is a point at (predicted committed tokens, cost in target-only steps). (a) The continuation moves points horizontally; (b) the deployment moves them vertically. EAGLE-3's costs are measured on the A40; other coordinates are illustrative.

After the target processes $x_{1:n}$, logits at n determine root x_{n+1}. The router consumes layer inputs at position n. The next target pass processes the root and chosen continuation; after committing L tokens, the next router input is at $n + L$. No representation of an unprocessed root is assumed. Pending target features for EAGLE are retained across other actions, and cache catch-up is included in the cost before an action is selected.

The following architecture and training settings describe the Llama predictor. The eight scalars are N-gram match length and its $\log(1 + x)$, a repetition indicator, generated-prefix length and its $\log(1 + x)$, context length and its $\log(1 + x)$, and prompt length. They are standardized on fitting data. Extra executor fields for recent acceptance are not predictor inputs. Target-layer inputs at layers 16 and 29 are each projected from 4,096 to 32 dimensions with per-layer fusion and LayerNorm. Two width-512 ReLU hidden layers feed five outputs, using $1 + \mathrm{softplus}$ and zero dropout. The target-only output is fixed to one. The model has 583,237 parameters.

Training uses AdamW, learning rate 0.003, weight decay 0.001, batch size 8,192, at most 60 epochs per round, patience 20, and two causal refits. Huber delta is 2, regression weight 16, routing temperature 0.8, and the routing term averages ratios as in Equation 4. Refitting uses deterministic offsets from the training seed; the seed-7 configuration's last refit uses seed 27. Checkpoint selection minimizes hard causal routing cost at the decision points reached by the current policy, with their execution state.

Threshold and route-once controls. The SuffixDecoding-style threshold uses one SAM match-score threshold per target, tuned on validation requests, with EAGLE-3 as the alternative. The HVD-style threshold uses the SAM head of the payoff predictor and chooses SAM when $\widehat{L}_{\mathrm{SAM}} \geq k$, otherwise EAGLE-3. Validation searches $k \in \{2, 4, 6, 8, 12, 16\}$ and selects $k = 2$. The route-once control uses the first block's prediction to choose an action and holds that choice throughout the request.

Cost-model variants. The primary cost model uses shared slopes with workload-specific intercepts. Static costs replace its state dependence with a calibrated scalar per action. Workload-agnostic costs remove the workload intercepts. In the deployment campaign, each non-negative least-squares model includes an intercept, context length, neural-cache catch-up, and proposal-token terms, fitted on training requests. The A40/B1 and recalibrated variants differ only in which deployment provides the cost fit. One common action is selected per synchronous batch block, while requests retain separate tokens, masks, accepted paths, and drafter state. In this executor, the eager Draft LM costs approximately three to five times target-only and receives no selected tokens.

D Representation, Action-Set, and Extension Analyses

This section gives the per-workload and supplementary evidence behind Table 4 and Figure 2. Table 9 gives the workload-level threshold and representation controls and the larger-tree experiment. The $60/7/10$ EAGLE-3 tree uses the same weights, precision, and flags as $32/8/4$. Fixed-tree measurements run on two A40s in opposite order. Its addition as a sixth action supplies one new set of counterfactual labels, one output head, and one calibrated cost; the existing output heads are not retrained. The six-action policy assigns 21% of committed tokens to the larger tree, concentrated on Alpaca, MT-Bench, MBPP, and Magicoder.

Table 9: **With the $60/7/10$ tree as a sixth action, SpecRouter exceeds every fixed action, including that tree, on every workload.** Primary campaign, Llama-3.1-8B; per-workload decode speedups for the threshold and input controls of Table 4 and the pool extension.

Workload	SuffixDecoding-style	Statistics only	States only	EAGLE-3 $60/7/10$	SpecRouter (6)
Delegate-52	1.98	2.28	2.56	1.21	2.69
FineEdit	4.95	5.05	5.65	1.38	5.87
RepoBench	1.85	1.97	2.08	0.96	2.16
SWE-Bench	1.93	1.90	2.08	1.89	2.14
Magicoder	3.16	3.30	3.23	3.27	3.30
MBPP	3.50	3.47	3.54	3.67	3.72
BFCL	1.33	1.42	1.59	1.05	1.64
Spider	2.02	1.98	2.08	2.06	2.11
HotpotQA	3.27	3.27	3.26	3.23	3.36
MuSiQue	1.69	1.83	1.90	1.38	1.94
2WikiMHQA	3.07	3.10	3.09	3.11	3.17
InfiniteBench	1.02	1.10	1.17	0.85	1.19
CNN/DM	2.53	2.50	2.49	2.55	2.58
GovReport	1.33	1.64	1.70	0.80	1.74
Alpaca	3.39	3.36	3.34	3.73	3.78
MT-Bench	3.27	3.27	3.30	3.46	3.50
Macro	2.52	2.59	2.69	2.16	2.81

Table 10: **Target states bring N-gram and the Draft LM into use.** Primary campaign, Llama; macro speedup ($\pm$ seed standard deviation) and committed-token shares (%) at 10,000 training requests; all variants share the architecture, budget, loss, and refitting schedule. Target-only shares are below 0.05%.

Inputs	Speedup	N-gram	SAM	Draft LM	EAGLE-3
Prefix statistics	2.59 ± 0.02	0.00	41.20	0.00	58.80
Target states	2.69 ± 0.01	4.50	31.90	12.60	51.00
States + statistics	**2.75**± 0.01	5.20	34.50	10.60	49.70

Table 11: **The advantage of target states holds at every training budget.** Primary campaign, Llama; macro decode speedup when training on subsets of the existing labels (Figure 2a).

Training requests	640	2,500	5,000	10,000
Prefix statistics only	2.54	2.58	2.59	2.59
Target states + statistics	2.64	2.70	2.73	**2.75**

Training recipe. Table 4 reports the loss and refitting ablations. With one causal refitting round instead of two, speedup is $2.73\times$, so most of the refitting gain comes from the first round, consistent with correcting the mismatch between labeled prefixes and the decision points the policy reaches.

Held-out workloads. Each leave-one-workload-out predictor uses the main training recipe on the other fifteen workloads, with workload-agnostic costs, and is evaluated on the excluded workload's test requests. Aggregating these sixteen evaluations gives $2.59\times$, compared with $2.75\times$ for the within-distribution predictor. The held-out policy exceeds EAGLE-3 on fifteen workloads; Alpaca is the exception, with a 2.9% relative deficit. Its largest deficit relative to workload-best fixed selection is 4.6% on Magicoder. Representative speedups are shown in Table 12. This protocol tests transfer across workload sources rather than merely splitting requests within each source.

Table 12: **Among representative workloads, holding the workload out of training costs most on the structured-output workloads BFCL and Spider.** Primary campaign; representative Llama workloads under leave-one-workload-out training. Both columns use the same test requests; the training population changes.

Workload	In-distribution training	Workload held out
Delegate-52	2.68	2.55
FineEdit	5.85	5.62
BFCL	1.63	1.41
Spider	2.10	1.97
Sixteen-workload macro	2.75	2.59

Prediction diagnostics. At test decision points reached by the Llama router, committed-length mean absolute errors are 0.58, 1.73, 1.06, and 0.84 tokens for N-gram, SAM, Draft LM, and EAGLE-3, respectively. The selected action agrees with the local hindsight reference on 66.8% of blocks, and on 81.3% of blocks its realized length and calibrated cost give a cost per token within 5% of that reference. These are local decision diagnostics, distinct from the end-to-end speedup metric. An alternative rule minimizing $\widehat{C}_a - \widehat{\rho}\widehat{L}_a$, with $\widehat{\rho}$ a per-request running estimate of cost per token, reaches $2.76\times$. Its similar performance supports the simpler ratio rule.

D.1 Selection opportunity at workload, request, and block scale

Table 13 gives a supplemental Llama diagnostic of the granularity of selection. Workload-best fixed chooses one fixed action per workload; request-best fixed chooses the fastest complete fixed-action run independently for every request with hindsight and pays no selection overhead. For fixed-run times $T_{i,a}$, these comparators have workload costs $\min_a \sum_{i \in w} T_{i,a}$ and $\sum_{i \in w} \min_a T_{i,a}$, respectively, including target-only among the available actions.

Table 13: **Block-level routing exceeds even the best fixed action chosen per request in hindsight.** Primary campaign, Llama; router gain is the mean of workload-relative gains, not a ratio of macro means. Hindsight choices pay no selection overhead.

Policy	Macro speedup	Router gain
Global fixed EAGLE-3	2.117	+50.68%
Workload-best fixed	2.526	+9.56%
Request-best fixed	2.679	+2.10%
SpecRouter block routing	**2.750**	—
Local hindsight reference	3.079	—

Request-level heterogeneity explains a substantial fraction of the available routing gain, while block-level routing retains additional local opportunity. This hindsight diagnostic asks how close the best fixed action per request could get; the learned route-once control in Table 4 instead asks whether the router can reuse its first prediction for the whole request. SpecRouter supplies one learned payoff interface for variation across requests and within a continuation.

D.2 Sensitivity to action availability

A deployed action may become unavailable. We mask one output at serving time while leaving the payoff predictor unchanged (Table 14). Target-only remains available, and the workload-best fixed comparator is recomputed over the available pool. This tests the existing predictor under a changed inventory, not a retrained smaller pool.

Table 14: **Action value is unequal: masking N-gram barely matters, while masking SAM or EAGLE-3 costs much more.** Primary campaign, Llama; one action masked at serving time with an unchanged predictor. Masked routers use their own target-only normalization, so these are descriptive cross-run comparisons.

Available pool	Router macro	Workload-best fixed	Mean gain
Full pool	2.7505	2.5261	+9.56%
Without N-gram	2.7495	2.5261	+9.23%
Without Draft LM	2.7222	2.5220	+7.88%
Without SAM	2.5636	2.3504	+11.63%
Without EAGLE-3	1.9534	1.8716	+4.40%

Action value is highly nonuniform. N-gram is nearly redundant in aggregate in this pool; removing Draft LM produces a modest measured reduction, while removing SAM or EAGLE-3 is much more consequential. These differences also show why token shares are not marginal speedup contributions.

D.3 Within-generation routing traces

Figure 6 shows selected blocks on four example requests, comparing SpecRouter with the local hindsight reference. The examples visualize decisions at successive committed prefixes; the route-once control in Table 4 quantifies the benefit of repeated selection.

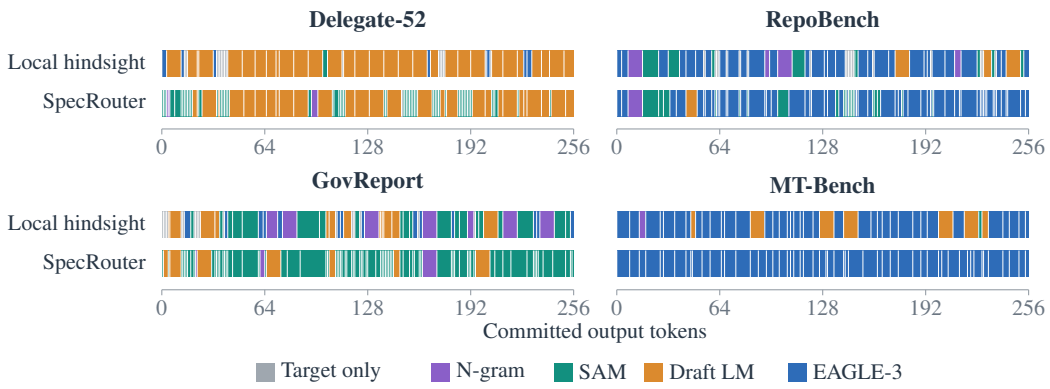


Figure 6: **Both the local hindsight reference and SpecRouter switch among actions as the continuation evolves.** Primary campaign, Llama; four illustrative requests. Horizontal extent is committed output tokens; color is the selected action. Data: `provenance/trace-blocks.csv`.

E Online Selection and Expert-Pool Comparison

E.1 Matched five-action comparison

Every policy in Table 15 is remeasured together on Llama-3.1-8B using the same five-action executor, action implementations, and sixteen workload types. This is a separate matched campaign: its fixed-action and router values are not spliced into Table 1a. UCB and EXP3 use the published BanditSpec equations (Hou et al., 2025), with committed length as reward, $\delta = 0.5$, maximum speculation-length parameter 32, random seed 42, and statistics reset at each request.

Table 15: **In the matched five-action campaign, both bandits fall below fixed EAGLE-3, while SpecRouter exceeds it.** Online-selection campaign, Llama-3.1-8B; macro decode speedup, all policies remeasured together (four decimals).

Policy	Macro speedup
Target only	1.0000
N-gram	1.2823
Draft LM	1.5689
SAM	1.6525
EAGLE-3	2.1586
UCB	1.7969
EXP3	1.7337
SpecRouter	**2.7714**
Local hindsight reference	3.0851

These acceptance-oriented, request-reset policies recover less of the measured opportunity than the state-conditioned payoff predictor in this executor. Their arm-selection logs show why (Table 16): each spreads its blocks almost evenly across the five actions and gives the largest share to the Draft LM, which commits the most tokens per block under their choices but also takes about twice as long per block as EAGLE-3.

Table 16: **With per-request resets, UCB and EXP3 spread their blocks almost evenly over the five actions and favor the expensive Draft LM.** Online-selection campaign, Llama; shares of routed blocks pooled over workloads, from the run's action-usage logs. Last two rows: UCB's tokens per block and mean block time by action.

	Target only	N-gram	SAM	Draft LM	EAGLE-3
UCB, share of blocks (%)	17.2	18.5	18.9	24.3	21.2
EXP3, share of blocks (%)	18.3	19.5	19.4	22.0	20.8
SpecRouter, share of blocks (%)	0.1	3.9	50.8	6.5	38.7
UCB, committed tokens per block	1.00	2.04	3.05	5.05	3.77
UCB, mean block time (ms)	39.9	57.0	42.5	134.7	66.4

HedgeSpec in the five-action pool. HedgeSpec's full-information updates evaluate all drafters on the verified tokens; lightweight EAGLE prefills make this inexpensive (Liu et al., 2026). Feedback for copying and the Draft LM needs action-specific implementations, and running every proposal would add drafting costs and be an unfair proxy. We therefore compare with HedgeSpec on its own EAGLE expert pool.

E.2 Specialist EAGLE expert pool

The expert-pool comparison uses HedgeSpec's released EAGLE checkpoints and seven-domain data. Training uses 560 requests (448 fitting and 112 validation), with 280 test requests. The payoff outputs correspond to the experts plus target-only. Every policy uses a 512-token output cap and the same target-only normalization; HedgeSpec follows its published algorithm and released implementation. The domain-matched expert uses domain identity, while the router selects from target-state payoff predictions without domain labels at inference.

Table 17: **SpecRouter beats HedgeSpec on six of seven domains of HedgeSpec's own expert pool and nearly matches the domain-matched expert.** Expert-pool campaign (Llama-3.1-8B, 512-token cap); mean decode speedup. Bold marks the best selector per domain; the last two columns are references.

Domain	EAGLE-3	UCB	EXP3	HedgeSpec	SpecRouter	Domain-matched expert	Local hindsight
Python	4.08	3.45	3.23	**4.44**	4.36	4.40	4.48
Math	3.61	3.51	3.43	4.22	**4.75**	4.77	4.80
Biology	3.65	3.27	3.07	4.15	**4.35**	4.36	4.43
Chemistry	3.14	3.02	2.95	3.84	**4.46**	4.47	4.54
MedQA	3.11	2.74	2.66	3.59	**3.98**	4.03	4.13
CNN/DM	2.98	2.20	2.14	2.82	**3.35**	3.35	3.44
SQL	3.72	3.53	3.44	4.62	**5.12**	5.12	5.13
Mean	3.47	3.10	2.99	3.96	**4.34**	4.36	4.42

SpecRouter exceeds HedgeSpec on six of seven domains. Python is the exception: HedgeSpec reaches $4.44\times$ versus $4.36\times$ for SpecRouter. The router's mean of $4.34\times$ is close to the domain-matched expert's $4.36\times$ and the local hindsight reference's $4.42\times$. This result shows that the payoff interface is not specific to heterogeneous action pools: with specialist EAGLE experts, it nearly recovers domain-matched selection without using domain labels at inference.

F Proposal Semantics and Deployment Transfer

Payoff transfer assumes that each action proposes the continuation described by its labels and that verification preserves the target's greedy decisions. The eager batched executor of the deployment campaign (Section 4) passed proposal-equivalence checks in every tested deployment. Table 18 gives all transfer controls behind Figure 3. At A40, batch size 16, SpecRouter with A40/B1 costs commits 78% of tokens through EAGLE-3, versus 69% at batch size one; recalibrating its costs adds 3.8%

speedup there. With recalibrated costs, 83% of tokens come from SAM, 13% from EAGLE-3, 3% from target-only, and 1% from N-gram, close to the local hindsight reference (75% SAM and 14% EAGLE-3).

Table 18: **With identical predictor weights in all six settings, SpecRouter is fastest in every deployment with either cost fit.** Deployment campaign: pooled request-inclusive speedup over target-only at the same GPU and batch size. The SuffixDecoding-style threshold is frozen at A40/B1 or retuned per setting with a target-only fallback.

			SuffixDecoding-style		SpecRouter	
Setting	Best fixed	Frozen EAGLE-3	Frozen	Retuned	A40/B1 costs	Recalibrated
A40 / B1	1.83	1.83	2.07	2.07	**2.34**	**2.34**
A40 / B4	1.27	1.27	1.44	1.46	1.62	**1.63**
A40 / B16	1.00	0.78	0.90	1.01	1.04	**1.08**
H200 / B1	2.15	2.15	2.38	2.40	**2.53**	2.52
H200 / B4	1.58	1.58	1.73	1.75	1.87	**1.91**
H200 / B16	1.00	0.81	0.93	1.01	1.04	**1.08**