SPECFIT: NOT ALL DRAFT TOKENS COST THE SAME IN OFFLOADED MOE INFERENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Mixture-of-experts (MoE) models activate only a few experts per token, but when their expert weights are offloaded to host memory, host-to-device (H2D) expert transfers enter the critical path of decoding. Speculative decoding verifies several draft tokens in one forward pass, yet in an MoE model each verified token activates its own experts, so a larger draft tree transfers more experts. Because of this H2D cost, speculative decoding commits several tokens per verification cycle yet is sometimes even slower than autoregressive decoding. Based on this observation, we propose SpecFit, which chooses the verified draft subtree to maximize predicted throughput rather than acceptance length. SpecFit estimates the expected number of committed tokens from draft path probabilities and predicts cycle time from the node count and the per-layer union of the experts the subtree activates; this union is a monotone submodular coverage function, so each node pays only for the experts it adds. An exact search over ancestor-closed subtrees returns the subtree with the highest predicted throughput; run after each draft level, the same search also decides when to stop drafting, and the unmodified target verifies the result. On CNN/DailyMail summarization and a GSM8K subset with Qwen3-30B-A3B, SpecFit decodes 21.6% and 7.0% faster than EAGLE-3 with the same expert cache and 1.97× and 1.72× as fast as vanilla EAGLE-3.

1 INTRODUCTION

Mixture-of-experts (MoE) layers route each token to a few expert feed-forward networks, decoupling the parameters a model stores from the computation it spends per token (Shazeer et al., 2017; Fedus et al., 2022), and have become a central design for recent large language models (Jiang et al., 2024; DeepSeek-AI et al., 2024; Yang et al., 2025). Sparse activation reduces computation, but not memory. When GPU memory cannot hold every expert, inference systems keep expert weights in host memory and cache, prefetch, or compute on the CPU the experts they need (Eliseev & Mazur, 2023; Xue et al., 2024; Kamahori et al., 2025), and host-to-device (H2D) expert transfers enter the critical path of decoding. Speculative decoding is a natural remedy. A small drafter proposes several tokens and the target verifies them together in one forward pass (Leviathan et al., 2023; Chen et al., 2023), so the weights that forward reads serve every committed token. This is especially attractive when those weights must first be copied from host memory (Svirschevski et al., 2024).

Existing work improves speculative decoding along two lines. Tree-based methods draft candidate branches and choose which to verify by their expected acceptance, ranking branches by draft confidence (Li et al., 2024; 2025), maximizing the expected acceptance length (Wang et al., 2025), or sizing the tree against measured hardware cost (Chen et al., 2024). Methods for MoE models act on the experts, prefetching the experts predicted during drafting (Chen et al., 2025), limiting the experts loaded during verification (McDanel et al., 2026), or training a resident draft expert (Han, 2026).

Tree selection, however, treats verification cost as a function of how many tokens are verified, which holds in a dense model, where every token reads the same weights. An MoE model routes each verified token independently, so a draft tree activates the union of its tokens' experts, including experts needed only by branches that will be rejected. A larger tree commits more tokens per cycle but also transfers more experts, so much of the speedup that its acceptance length promises is lost. Verification cost depends not only on how many draft tokens are verified but also on which ones, and a selector that values trees by acceptance and size cannot see this difference.

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.

We propose SpecFit, which selects the verification subtree that maximizes predicted throughput rather than acceptance length. It scores each ancestor-closed subtree of an unmodified EAGLE-3 candidate pool by the ratio of a draft-probability reward to a cycle-time estimate that grows with the number of distinct experts the subtree activates in each layer, with coefficients fitted once on calibration queries. An exact depth-first search with incremental expert bitmaps then decides both which nodes and how many to verify. Run after each draft level, the same search also decides when to stop drafting. The target is untouched: its router is unchanged, and every routed expert of every selected token is executed.

The design rests on four properties. The expected committed length of a subtree is one plus the sum of its nodes' commit probabilities, with no independence assumption; the expert-union cost is monotone and submodular; a smaller subtree raises throughput if and only if it saves a larger fraction of cycle time than of committed tokens; and selecting by the per-cycle ratio stays within a computable margin of the throughput-optimal rule. On 40 summarization articles and an 80-question GSM8K subset, SpecFit decodes 21.6% and 7.0% faster than EAGLE-3 with the same expert cache and 1.97× and 1.72× as fast as vanilla EAGLE-3. The gains come from cheaper verification and drafting rather than longer accepted paths, and hold on GSM8K, where EAGLE-3 already accepts long draft paths.

In summary, we make the following contributions:

- We identify and quantify a conflict between speculative decoding and expert offloading in MoE models: verifying a draft tree enlarges the union of activated experts, so that verification cost depends on which draft tokens are verified, not only on how many.
- We propose SpecFit, which selects the verification subtree and the drafting depth by the ratio of a draft-probability reward to an expert-union cost and solves this selection exactly, without retraining the drafter or altering the target's routing and expert execution.
- We characterize when a smaller subtree raises throughput, bound the distance of per-cycle ratio selection from the throughput-optimal rule, and prove the size-limited selection NP-hard; on summarization and GSM8K, SpecFit improves decoding throughput over both vanilla EAGLE-3 and EAGLE-3 with an expert cache.

2 BACKGROUND

This section introduces the two techniques that SpecFit combines and shows why, together, they make verification cost depend on which draft tokens are verified.

MoE inference with expert offloading. An MoE model has L MoE layers. Layer ℓ contains N expert feed-forward networks and a router that sends each token x to the set $E_\ell(x)$ of the k experts with the highest gate scores; the layer output combines those experts' outputs with the gate weights (Shazeer et al., 2017; Fedus et al., 2022). When the experts do not fit in GPU memory, an offloading runtime keeps them in host memory and uses a limited number of GPU slots per layer as an expert cache (Eliseev & Mazur, 2023; Xue et al., 2024; Song et al., 2024). A forward pass over a query Q must execute, in every layer, each expert in $\bigcup_{x \in Q} E_\ell(x)$. An expert missing from the cache is copied host-to-device before the layer can finish and evicts another expert. The cache avoids a transfer when a later forward needs an expert that is still in the cache, but the per-layer union over the query remains the demand that each forward must meet.

Tree-based speculative decoding. In EAGLE-2 and EAGLE-3 (Li et al., 2024; 2025), the drafter starts from the pending root token s and expands its most likely continuations for several steps into a candidate pool V. Each node $v \in V$ has a parent $\mathrm{pa}(v) \in V \cup \{s\}$ and a path probability p_v, the product of draft conditional probabilities from the root to v. A subset $S \subseteq V$ is then verified: the target processes $S \cup \{s\}$ in one forward with a tree attention mask and, under greedy verification, accepts the longest root-to-node path whose tokens match its own predictions. The cycle commits the root and the accepted path, and the target's next prediction becomes the new root. EAGLE-3 verifies the nodes with the highest p_v up to a fixed budget. If a cycle commits A tokens in time T, long-run decoding throughput is $\mathbb{E}[A]/\mathbb{E}[T]$, so choosing S trades committed tokens against cycle time.

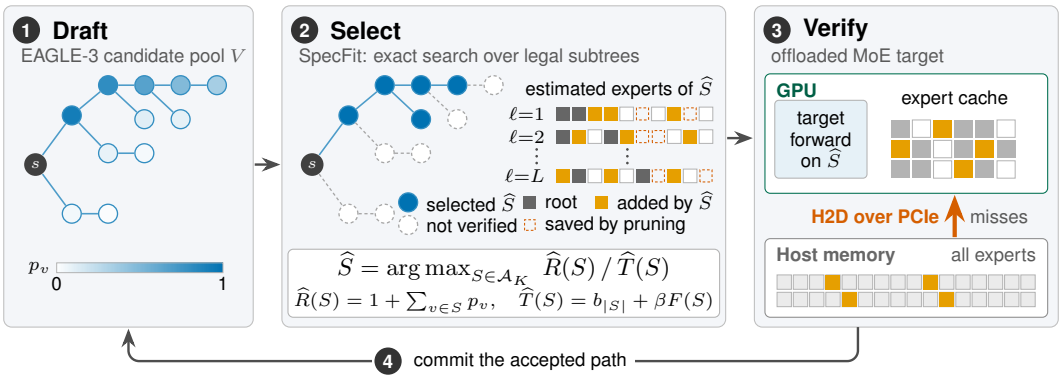


Figure 1: One SpecFit decoding cycle. (1) EAGLE-3 drafts a candidate pool V with path probabilities p_v level by level; drafting stops once the selection takes no node from the newest level. (2) SpecFit searches the ancestor-closed subtrees exactly and selects the one with the largest $\widehat{R}/\widehat{T}$; experts needed only by unselected nodes are excluded from $F(\widehat{S})$. (3) The offloaded target verifies $\widehat{S}$. Its own router decides which experts run, and experts missing from the expert cache are copied from host memory. (4) The accepted path is committed.

Why the verified set matters. Combining the two, write $E_\ell(v)$ for E_ℓ of the token of node v in the tree forward, which depends on the node's ancestors as well as on its own token. A forward that verifies $S \cup \{s\}$ must execute

$$D(S) = \sum_{\ell=1}^{L} \left| E_\ell(s) \cup \bigcup_{v \in S} E_\ell(v) \right| \tag{1}$$

experts. Draft tokens at different positions and on different branches are routed separately, so the union grows with the tree, and a rejected branch pays for its experts as fully as an accepted one.

Choosing the verified set under this cost is difficult for four reasons. First, the reward is uncertain: acceptance is revealed only by verification, so selection must rely on estimates available before the target runs. Second, the cost is not additive: tokens that share an expert in a layer pay for it once, so the expert union is a coverage function and a node's marginal cost depends on the rest of the tree. Third, the choices are structured: a token can be committed only with its prefix, so only ancestor-closed subsets are meaningful. Fourth, the objective is a ratio: maximizing acceptance ignores cost, minimizing expert demand is trivially achieved by verifying no draft token, and the best tree size varies from cycle to cycle.

The decision also lies on the critical path of every cycle, so it must be cheap, and it should not buy speed by pruning or substituting routed experts, which would change the target's computation. Existing selectors value trees by acceptance and, where they model cost, by size or depth (Li et al., 2025; Wang et al., 2025; Chen et al., 2024); none accounts for the expert union. Section 3 formulates and solves this selection.

3 Method

In each speculative cycle, SpecFit decides how deep EAGLE-3 drafts and which draft tokens the target verifies. While the drafter builds the candidate pool level by level, SpecFit selects the legal subtree with the largest ratio of predicted reward to predicted cycle time and stops drafting once that subtree takes no node from the newest level; the target then verifies the selected subtree and commits an accepted path (Figure 1). Section 3.1 defines the legal subtrees. Section 3.2 predicts a subtree's reward from draft probabilities and its cycle time from the experts its nodes activate. Section 3.3 solves the resulting objective exactly, uses it to decide the drafting depth, relates it to the throughput-optimal rule, and describes the route information and target execution.

3.1 Selecting a Verification Subtree

With the root s, candidate pool V, and parents $\mathrm{pa}(v)$ of Section 2, a selected node requires its ancestors, which defines the legal family of subtrees with at most K draft nodes,

$$\mathcal{A}_K = \{S \subseteq V : \ |S| \le K, \quad v \in S \Rightarrow \mathrm{pa}(v) \in S \cup \{s\}\}. \tag{2}$$

Our candidate generator uses top-2 expansion, depth at most 5, and at most 18 nodes, with $K = \min(7, |V|)$; the pool becomes smaller near the generation limit or when SpecFit stops drafting early (Section 3.3).

3.2 Estimating Reward and Expert Cost

A useful subtree must retain likely accepted paths while limiting the work spent verifying all its nodes. Let $I_v(S)$ indicate that draft node v is committed when S is verified. By linearity of expectation, the expected number of tokens committed per cycle is

$$R(S) = 1 + \sum_{v \in S} \Pr\big(I_v(S) = 1\big) \ \approx \ 1 + \sum_{v \in S} p_v = \widehat{R}(S), \tag{3}$$

in which the root contributes one and each draft node its commit probability. The first equality holds for any dependence among the commit events, with each probability allowed to depend on the whole tree. Because these probabilities are revealed only by verification, SpecFit estimates them with the path probabilities p_v of Section 2, which the drafter already computes while building the pool.

Let $\widehat{E}_{\ell,v}$ be the estimate of $E_\ell(v)$ supplied to the selector for node v in layer ℓ. Substituting these estimates into D of Equation 1 gives the union feature F, and SpecFit predicts cycle time from it and the node count,

$$F(S) = \sum_{\ell=1}^{L} \left| \widehat{E}_{\ell,s} \cup \bigcup_{v \in S} \widehat{E}_{\ell,v} \right|, \qquad \widehat{T}(S) = b_{|S|} + \beta F(S). \tag{4}$$

The node-count intercepts $b_0, \ldots, b_7$ and the shared coefficient β are fitted once by nonnegative least squares to cycle times measured on GSM8K calibration questions, then frozen (Appendix A). The fitted cycle time covers drafting, target verification, and commit; the selector's own time is measured and included in the reported throughput.

$F(S)$ estimates $D(S)$, the expert demand that the expert cache serves; every selected branch contributes to it, whether or not it is later accepted. SpecFit with the expert cache uses coefficients fitted with the cache active, so β is the average time per expert of demand with cache hits folded in; without the cache, it uses coefficients fitted with the cache cleared before every target forward. Either way, $\widehat{T}$ needs only the routes of the pool. For fixed estimates, F is a coverage function. Adding v to S increases it by $\sum_\ell |\widehat{E}_{\ell,v} \setminus \widehat{U}_\ell(S)|$, where $\widehat{U}_\ell(S)$ is the layer-ℓ union in Equation 4; since $\widehat{U}_\ell(S)$ only grows with S, F is monotone and submodular. A node therefore pays only for experts new to the subtree, and its marginal cost can only fall as the subtree grows.

3.3 Selection, Verification, and Available Information

SpecFit selects

$$\widehat{S} = \arg\max_{S \in \mathcal{A}_K} \frac{\widehat{R}(S)}{\widehat{T}(S)}. \tag{5}$$

This objective jointly chooses node identities and cardinality. For positive rewards and times, a subtree with (R, T) outperforms one with (R_0, T_0) exactly when

$$\frac{R}{T} > \frac{R_0}{T_0} \quad \Longleftrightarrow \quad \frac{T_0 - T}{T_0} > \frac{R_0 - R}{R_0}. \tag{6}$$

Pruning nodes therefore raises throughput if and only if it saves a larger fraction of cycle time than of committed tokens.

Equation 5 maximizes the ratio within each cycle, whereas throughput is a ratio of expectations over cycles (Section 2). To relate the two, let the cycle states, each a pool with its probabilities

and routes, be independent and identically distributed and independent of earlier selections, and let $\Theta(\pi) = \mathbb{E}[\widehat{R}(S_\pi)]/\mathbb{E}[\widehat{T}(S_\pi)]$ be the predicted throughput of a rule π that selects S_π in each cycle. The throughput-optimal rule selects $S^* = \arg\max_{S \in \mathcal{A}_K} \widehat{R}(S) - \lambda^* \widehat{T}(S)$ and attains the largest value $\lambda^* = \max_\pi \Theta(\pi)$ (Dinkelbach, 1967). This rule needs λ^*, which depends on the distribution of all cycles; Equation 5 needs only the current pool.

Proposition 1. *Let $\rho = \widehat{R}(\widehat{S})/\widehat{T}(\widehat{S})$ be the largest ratio in a cycle, and let Θ be the predicted throughput of Equation 5. Then*

$$0 \le \lambda^* - \Theta \le \frac{\mathbb{E}\big[(\rho - \lambda^*)\big(\widehat{T}(S^*) - \widehat{T}(\widehat{S})\big)\big]}{\mathbb{E}\big[\widehat{T}(\widehat{S})\big]}, \tag{7}$$

and the quantity inside the expectation is nonnegative in every cycle.

The margin vanishes when every cycle has the same largest ratio, and it is large only when cycles differ widely in their largest ratio and the two rules then select trees of widely different time. Appendix B proves the proposition and evaluates the margin on recorded pools. The same argument shows that, for fixed intercepts, a larger β can only lower the selected tree's predicted expert demand per committed token, $F(\widehat{S})/\widehat{R}(\widehat{S})$.

For general pools, the maximization in Equation 5 is hard.

Proposition 2. *Deciding whether some $S \in \mathcal{A}_K$ reaches a given ratio $\widehat{R}(S)/\widehat{T}(S)$ is NP-complete, even for a pool of depth one, one MoE layer, and equal intercepts. Without the size limit K and with intercepts affine and nondecreasing in $|S|$, the largest ratio is computable in polynomial time by parametric minimum cut (Picard, 1976; Gallo et al., 1989).*

SpecFit's pools have at most 18 nodes and $K \le 7$, so the legal family is small, and SpecFit solves Equation 5 exactly by enumeration (Algorithm 1 in Appendix B). A depth-first traversal reaches every subtree in $\mathcal{A}_K$ exactly once and carries the per-layer expert unions as bitmaps from parent states. Equal ratios prefer larger reward, smaller union, fewer draft nodes, and then lexicographically smaller node IDs, so the returned subtree is the unique exact maximizer of Equation 5 over $\mathcal{A}_K$.

Adaptive drafting depth. The same objective also decides how deep EAGLE-3 drafts. The drafter adds one level of the pool per draft forward; let $V_{\le d}$ be the nodes of the first d levels. After drafting level d, SpecFit solves Equation 5 with $V_{\le d}$ in place of V. If the selected subtree contains a level-d node, SpecFit drafts the next level; otherwise it stops drafting and verifies that subtree. At the maximum depth it selects over the full pool. The stop rests on the tree structure: a deeper node can be selected only together with its level-d ancestor and adds no more reward than that ancestor, which the selection has already left out. The stopped selection equals the full-pool selection whenever the latter stays within the first d levels (Lemma 2 in Appendix B).

Route information. In our experiments, $\widehat{E}$ comes from a route atlas, a separate target query over the full candidate pool on SpecFit's reference prefix, prepared outside the request timer. During measurement, each cycle's prefix and pool are checked against the atlas. Selection uses only the draft probabilities, the atlas routes, and the frozen coefficients.

Target execution. After selection, the target verifies $\widehat{S}$ with its own router and executes every routed expert of every selected token; experts missing from the expert cache are copied from host memory.

4 Experiments

We compare SpecFit with EAGLE-3 and autoregressive (AR) decoding, each without and with the expert cache, on summarization and GSM8K, decompose the gain with verification and transfer counters, vary the cache capacity and the GPU, and account for route prediction and selection cost.

4.1 SETUP

The target is Qwen3-30B-A3B-Instruct-2507 (Yang et al., 2025; Qwen Team, 2025) with its RedHatAI EAGLE-3 drafter (Red Hat AI, 2025), and the workloads are 40 CNN/DailyMail articles (Nallapati et al., 2016) and 80 GSM8K questions (Cobbe et al., 2021) from Spec-Bench (Xia et al., 2024). All measurements run at request batch size one, on a single A100 PCIe 40GB GPU except in Section 4.5. Every configuration, including AR, runs on our offload backend, which keeps all expert weights in pinned host memory. With the expert cache, each layer holds 48 experts in GPU slots with LRU replacement; without it, the cache is cleared before every target forward. Table 1 lists the compared methods; SpecFit and EAGLE-3 use the same drafter and differ only in drafting depth and in which and how many nodes are verified. EAGLE-3 without the cache is the vanilla EAGLE-3 of the introduction; the cached pair is our primary comparison, and SpecFit without qualification denotes its cached configuration. The cost coefficients are fitted on GSM8K calibration questions, with and without the cache, and used unchanged on both workloads. Speedups are relative to AR without the expert cache, and committed tokens are the tokens a verification cycle appends to the output, namely the root and the accepted draft nodes (Section 2). Appendix A gives the backend, timing, calibration, and data details.

Table 1: Compared methods.

Method	Verification-node selection
AR	Serial target decoding, without a drafter
EAGLE-3	Highest cumulative probability, at most 7 nodes
SpecFit	Equation 5 with adaptive depth, at most 7 nodes

4.2 SUMMARIZATION

To test whether SpecFit gains by leaving out draft nodes that do not pay for their experts, we first use summarization, on which EAGLE-3 accepts short draft paths. Each method runs on the 40 CNN/DailyMail articles without and with the expert cache; Figure 2 reports decode throughput.

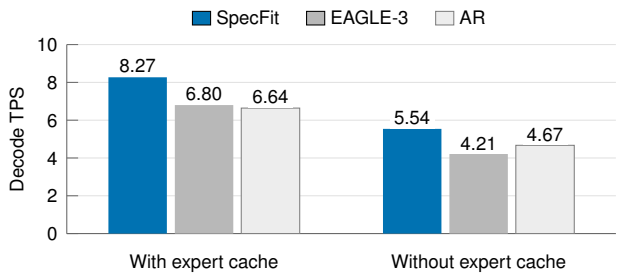


Figure 2: Results on the CNN/DailyMail summarization dataset.

SpecFit decodes 8.27 tokens/s, 21.6% above cached EAGLE-3 (19.9% including prefill) and 1.77× AR. Without the cache, it decodes 31.6% faster than vanilla EAGLE-3, which is itself slower than AR (0.90×). The verification and transfer counters in Table 2 show where the gain comes from. EAGLE-3 verifies seven draft nodes per cycle but accepts only 1.54 of them. SpecFit verifies 2.88 draft nodes instead. It stops drafting before the fifth level in 85.8% of cycles, which cuts drafting time per cycle by 44.9%. It commits 13.1% fewer tokens per cycle than cached EAGLE-3, but each cycle takes 28.5% less decode time, so it decodes 21.6% faster. Verification accounts for 78.3% of the time saved per cycle, and SpecFit transfers 22.1% fewer experts per output token. This is the trade the objective makes: a node with a low acceptance probability adds little to the expected commit but still adds its experts to the verified union, so leaving such nodes out shortens the cycle by a larger fraction than it reduces committed tokens.

Table 2: Verification and transfer counters on the CNN/DailyMail summarization dataset.

Method	Cache	Draft nodes	Committed	Cycles	ms/cycle	H2D/token
AR	No	–	1.00	9,652	214.2	400.7
EAGLE-3	No	7.00	2.54	3,823	600.9	486.5
SpecFit	No	2.29	2.08	4,702	373.6	347.7
AR	Yes	–	1.00	9,652	150.6	49.7
EAGLE-3	Yes	7.00	2.54	3,823	371.7	73.4
SpecFit	Yes	2.88	2.21	4,389	265.8	57.2

4.3 GSM8K

To test whether SpecFit still gains when EAGLE-3 accepts long draft paths, we run the same comparison on the 80 GSM8K questions; Figure 3 reports decode throughput.

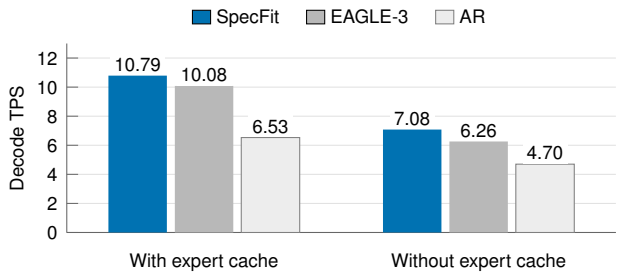


Figure 3: Results on the GSM8K dataset.

SpecFit decodes 10.79 tokens/s, 7.0% above cached EAGLE-3 (6.3% including prefill) and 2.29× AR; without the cache, it decodes 13.1% faster than vanilla EAGLE-3. The verification and transfer counters in Table 3 show where this gain comes from. With the cache, EAGLE-3 commits 4.35 tokens per cycle, against 2.54 on summarization, so fewer of its nodes are worth leaving out. SpecFit still verifies 5.09 draft nodes instead of 7.00 and commits 7.8% fewer tokens per cycle, but each cycle takes 13.8% less decode time, and it transfers 11.6% fewer experts per output token; adaptive depth stops drafting before the fifth level in 33.7% of cycles and cuts drafting time per cycle by 11.7%. The gain is smaller than on summarization but has the same source, shorter cycles rather than longer accepted paths: cycle time falls by a larger fraction than committed tokens, as Equation 6 requires.

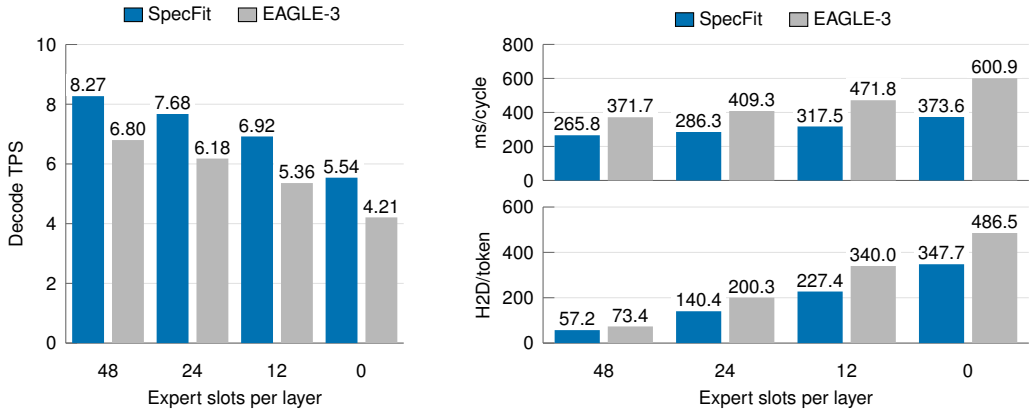
Table 3: Verification and transfer counters on the GSM8K dataset.

Method	Cache	Draft nodes	Committed	Cycles	ms/cycle	H2D/token
AR	No	–	1.00	17,791	212.5	398.2
EAGLE-3	No	7.00	4.35	4,097	692.3	345.7
SpecFit	No	4.27	3.81	4,662	536.7	294.1
AR	Yes	–	1.00	17,791	153.1	76.5
EAGLE-3	Yes	7.00	4.35	4,097	429.8	92.7
SpecFit	Yes	5.09	4.01	4,407	370.4	82.0

4.4 Sensitivity to Offload Pressure

To test whether SpecFit's gain grows as the expert cache shrinks, we rerun EAGLE-3 and SpecFit on the 40 summarization articles with 24 and 12 expert slots per layer and compare them with the 48 slots of the main runs and with the configurations without the cache (0 slots); Figure 4a reports decode throughput.

SpecFit's decoding gain over EAGLE-3 grows from 21.6% at 48 slots to 24.3% and 29.2% at 24 and 12 slots, and it is 31.6% without the cache. With the cache, SpecFit verifies 2.88 draft nodes and



(a) Decode throughput.

(b) Cycle time and transfers.

Figure 4: Results on CNN/DailyMail with different numbers of expert slots per layer.

commits 13.1% fewer tokens per cycle than EAGLE-3 at every capacity, so the growing gain comes from shorter cycles. As Figure 4b shows, at 48, 24, and 12 slots, its time per cycle is 28.5%, 30.1%, and 32.7% shorter, and it transfers 22.1%, 29.9%, and 33.1% fewer experts per output token.

Because the cache capacity does not enter selection, the trees at 48, 24, and 12 slots are the same; for these capacities, let a configuration spend decode time $T = B + \tau H$ on R decode tokens, a fixed part B plus H expert transfers of time τ each. For SpecFit and EAGLE-3 (subscripts S and E),

$$\frac{R_S/T_S}{R_E/T_E} = (1-w)\,\frac{R_S/B_S}{R_E/B_E} + w\,\frac{R_S/H_S}{R_E/H_E}, \qquad w = \frac{\tau H_S}{T_S}, \tag{8}$$

so the gain averages SpecFit's advantages in tokens per unit of fixed time and in tokens per transfer, weighted by the share w of its time spent on transfers. If B and τ do not depend on the capacity, a smaller cache raises w and moves the gain toward the second advantage, which is 28.3%, 42.7%, and 49.5% at 48, 24, and 12 slots (Figure 4b), above the measured gain at each capacity.

4.5 Another GPU

To test whether the gain carries over to a GPU with less memory and a slower host link, we repeat the GSM8K comparison on a single RTX 4090 24GB. It holds 32 expert slots per layer instead of 48, and with the cache cleared, the fitted time per loaded expert is 0.78 ms, against 0.41 ms on the A100. The cost coefficients are refitted on this GPU with the same protocol (Appendix A), and the comparison uses the 40 odd-numbered of the 80 questions; Table 4 reports the results.

Table 4: Results on the GSM8K dataset on a single RTX 4090 24GB.

Method	Cache	Decode TPS	Speedup	Draft nodes	Committed	Cycles	ms/cycle	H2D/token
AR	No	2.67	1.00×	–	1.00	8,714	374.4	398.6
EAGLE-3	No	3.29	1.23×	6.99	4.32	2,009	1,306.7	346.4
SpecFit	No	3.89	1.46×	3.78	3.68	2,339	942.8	288.0
AR	Yes	4.24	1.59×	–	1.00	8,714	236.1	126.9
EAGLE-3	Yes	6.05	2.27×	6.99	4.32	2,009	711.1	150.0
SpecFit	Yes	6.70	2.51×	4.53	4.00	2,162	593.7	128.3

With the expert cache, SpecFit decodes 6.70 tokens/s on this GPU, 10.8% above cached EAGLE-3 and 2.51× AR; without the cache, it decodes 18.1% faster than vanilla EAGLE-3. As on the A100, it verifies fewer draft nodes (4.53 instead of 6.99) and commits 7.5% fewer tokens per cycle, but each cycle takes 16.5% less decode time, and it transfers 14.5% fewer experts per output token. SpecFit thus makes the same trade on a GPU whose memory and transfer costs differ.

4.6 Route Prediction and Selection Cost

Our measurements take routes from the atlas, outside the request timer, so that selection is evaluated on its own; a deployment replaces the atlas with a route predictor. Lightweight expert predictors reach 84.7% average accuracy (Song et al., 2024) and 94.7% on Qwen3-30B-A3B (Zhu et al., 2025). SpecFit reads only the predicted expert sets $\widehat{E}$, and verification still executes every expert that the target's router chooses, so route errors can change only which nodes are selected. With a simulated predictor of 60% accuracy, which replaces each routed expert by a random expert outside the true top-8 with probability 40%, SpecFit still decodes 2.26× faster than AR on 40 GSM8K questions, against 2.27× with exact routes, because the correct predictions still show which draft tokens share experts, while wrong ones rarely coincide across nodes and add a nearly uniform cost per node. A linear-head predictor, which maps each draft node's hidden state to logits over all 48×128 experts, has 12.6M parameters, the memory of about three expert slots, and needs about 25 MFLOP per node; applied after each drafted level, it adds an estimated 0.1–0.2 ms per cycle. SpecFit's own selection, included in every measurement above, takes a median of 2.82 ms per cycle on GSM8K (0.7% of decode time) and 1.43 ms on summarization (0.6%), both small compared with the 59 ms by which a SpecFit cycle is shorter than a cached EAGLE-3 cycle on GSM8K. Training route predictors is left to future work.

5 Related Work

Expert offloading. Offloading systems keep MoE expert weights in host memory and cache and prefetch the experts they expect to need (Eliseev & Mazur, 2023; Xue et al., 2024; Song et al., 2024), or compute some experts on the CPU (Kamahori et al., 2025). SpecFit runs on an on-demand expert cache without prefetching; it changes which tokens create expert demand, not how it is served.

Draft-tree construction and selection. Leviathan et al. (2023) and Chen et al. (2023) develop draft-and-verify acceleration with target-distribution-preserving sampling rules under their assumptions. EAGLE-2 constructs context-dependent trees ranked by draft confidence, and EAGLE-3 develops the draft model and feature interface (Li et al., 2024; 2025). OPT-Tree maximizes the expected acceptance length (Wang et al., 2025), Sequoia chooses tree size and depth against measured hardware cost (Chen et al., 2024), and SpecExec verifies large trees to amortize weight loading for dense models with offloaded weights (Svirschevski et al., 2024). SpecFit keeps a fixed EAGLE-3 drafter and candidate generator and changes only how deep it drafts and which subset is sent to verification, valuing both by an expert-union cost rather than by size or shape.

Speculative decoding for MoE models. SP-MoE prefetches experts predicted during drafting and pipelines their execution (Chen et al., 2025). MoE-Spec limits the experts loaded during verification and thereby alters the target computation (McDanel et al., 2026), whereas SpecFit executes all routed experts for its selected tokens. DraftExpert combines a trained resident draft expert with expansion-aware truncation and prefetching for offloaded inference (Han, 2026). SpecFit instead evaluates a complete-subtree reward/cost ratio over a bounded feasible family, with cardinality included in the search, and trains no model component.

6 Conclusion

Under expert offloading, speculative verification in MoE models pays for every expert that the verified tree activates. SpecFit therefore selects each cycle's verification subtree and drafting depth by the ratio of a draft-probability reward to an expert-union cost, solved exactly over ancestor-closed subtrees, without changing the drafter, the router, or expert execution. With Qwen3-30B-A3B, it verifies fewer draft nodes when their expert cost outweighs their expected tokens. On summarization, where EAGLE-3 accepts short draft paths, this raises decoding throughput by 21.6% over EAGLE-3 with the same expert cache. On GSM8K, where EAGLE-3 accepts long paths, each cycle is still 13.8% shorter and the gain is 7.0%, to 2.29× that of AR decoding without the cache. As the expert cache shrinks from 48 to 12 slots per layer, the summarization gain grows to 29.2%, and on an RTX 4090, whose host link is slower, the GSM8K gain is 10.8%.

AI USE STATEMENT

We used AI assistants for software development, experiment inspection, mathematical exposition, literature search, and drafting of the manuscript. The authors reviewed all AI-assisted content and take full responsibility for the claims, references, and results in this paper.

ETHICS STATEMENT

This work speeds up decoding of existing MoE language models whose experts are offloaded to host memory. It uses a publicly released target model, drafter, and datasets, involves no human subjects, and collects no new data. SpecFit changes only which draft tokens are sent to verification, and the target model verifies them unmodified, so the method adds no capability or risk beyond those of the model it accelerates.

REPRODUCIBILITY STATEMENT

Section 4.1 and Appendix A give the model, the workloads, the expert cache configuration, the calibration protocol, and the fitted cost coefficients. Section 3 and Algorithm 1 specify the selection and the drafting-depth rule, and Appendix B states the assumptions and gives complete proofs of both propositions and both lemmas. The supplementary material contains the source code, with the configuration files and scripts that run the calibration and the comparisons in Section 4.

REFERENCES

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating Large Language Model Decoding with Speculative Sampling, 2023. URL https://arxiv.org/abs/2302.01318.

Liangkun Chen, Zijian Wen, Tian Wu, Xiaoxi Zhang, and Chuan Wu. SP-MoE: Speculative Decoding and Prefetching for Accelerating MoE-based Model Inference, 2025. URL https://arxiv.org/abs/2510.10302.

Zhuoming Chen, Avner May, Ruslan Svirschevski, Yuhsun Huang, Max Ryabinin, Zhihao Jia, and Beidi Chen. Sequoia: Scalable and robust speculative decoding. In *Advances in Neural Information Processing Systems*, volume 37, pp. 129531–129563, 2024. doi: 10.52202/079017-4116.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, et al. Training Verifiers to Solve Math Word Problems, 2021. URL https://arxiv.org/abs/2110.14168.

DeepSeek-AI, Aixin Liu, Bei Feng, et al. DeepSeek-V3 Technical Report, 2024. URL https://arxiv.org/abs/2412.19437.

Werner Dinkelbach. On Nonlinear Fractional Programming. *Management Science*, 13(7):492–498, 1967. doi: 10.1287/mnsc.13.7.492. URL https://doi.org/10.1287/mnsc.13.7.492.

Artyom Eliseev and Denis Mazur. Fast Inference of Mixture-of-Experts Language Models with Offloading, 2023. URL https://arxiv.org/abs/2312.17238.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022. URL https://jmlr.org/papers/v23/21-0998.html.

Giorgio Gallo, Michael D. Grigoriadis, and Robert E. Tarjan. A Fast Parametric Maximum Flow Algorithm and Applications. *SIAM Journal on Computing*, 18(1):30–55, 1989. doi: 10.1137/0218003. URL https://doi.org/10.1137/0218003.

Dengke Han. DraftExpert: Expansion-Aware Self-Speculative Decoding for End-Device MoE Inference, 2026. URL https://arxiv.org/abs/2607.24434.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, et al. Mixtral of Experts, 2024. URL https://arxiv.org/abs/2401.04088.

Keisuke Kamahori, Tian Tang, Yile Gu, Kan Zhu, and Baris Kasikci. Fiddler: CPU-GPU orchestration for fast inference of mixture-of-experts models. In *International Conference on Learning Representations*, pp. 56099–56115, 2025. URL https://openreview.net/forum?id=N5fVv6PZGz.

Richard M. Karp. Reducibility among Combinatorial Problems. In *Complexity of Computer Computations*, pp. 85–103. Springer US, Boston, MA, 1972. doi: 10.1007/978-1-4684-2001-2_9. URL https://doi.org/10.1007/978-1-4684-2001-2_9.

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, 2023. URL https://proceedings.mlr.press/v202/leviathan23a.html.

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. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-main.422. URL https://aclanthology.org/2024.emnlp-main.422/.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE-3: Scaling up inference acceleration of large language models via training-time test. In *Advances in Neural Information Processing Systems*, volume 38, pp. 136737–136756, 2025. doi: 10.52202/085713-4562.

Bradley McDanel, Steven Li, Sruthikesh Surineni, and Harshit Khaitan. MoE-Spec: Expert Budgeting for Efficient Speculative Decoding, 2026. URL https://arxiv.org/abs/2602.16052.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar Gülçehre, and Bing Xiang. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pp. 280–290. Association for Computational Linguistics, 2016. doi: 10.18653/v1/K16-1028. URL https://aclanthology.org/K16-1028/.

Jean-Claude Picard. Maximal Closure of a Graph and Applications to Combinatorial Problems. *Management Science*, 22(11):1268–1272, 1976. doi: 10.1287/mnsc.22.11.1268. URL https://doi.org/10.1287/mnsc.22.11.1268.

Qwen Team. Qwen3-30B-A3B-Instruct-2507 model card, 2025. URL https://huggingface.co/Qwen/Qwen3-30B-A3B-Instruct-2507.

Red Hat AI. Qwen3-30B-A3B-Instruct-2507-speculator.eagle3 model card, 2025. URL https://huggingface.co/RedHatAI/Qwen3-30B-A3B-Instruct-2507-speculator.eagle3.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=B1ckMDqlg.

Xiaoniu Song, Zihang Zhong, Rong Chen, and Haibo Chen. ProMoE: Fast MoE-based LLM Serving using Proactive Caching, 2024. URL https://arxiv.org/abs/2410.22134.

Ruslan Svirschevski, Avner May, Zhuoming Chen, Beidi Chen, Zhihao Jia, and Max Ryabinin. SpecExec: Massively parallel speculative decoding for interactive LLM inference on consumer devices. In *Advances in Neural Information Processing Systems*, volume 37, pp. 16342–16368, 2024. doi: 10.52202/079017-0522.

Jikai Wang, Yi Su, Juntao Li, Qingrong Xia, Zi Ye, Xinyu Duan, Zhefeng Wang, and Min Zhang. OPT-Tree: Speculative decoding with adaptive draft tree structure. *Transactions of the Association for Computational Linguistics*, 13:188–199, 2025. doi: 10.1162/tacl_a_00735. URL https://aclanthology.org/2025.tacl-1.8/.

Heming Xia, Zhe Yang, Qingxiu Dong, Peiyi Wang, Yongqi Li, Tao Ge, Tianyu Liu, Wenjie Li, and Zhifang Sui. Unlocking efficiency in large language model inference: A comprehensive survey of speculative decoding. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 7655–7671. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-acl.456. URL https://aclanthology.org/2024.findings-acl.456/.

Leyang Xue, Yao Fu, Zhan Lu, Chuanhao Sun, Luo Mai, and Mahesh Marina. MoE-Infinity: Efficient MoE Inference on Personal Machines with Sparsity-Aware Expert Cache, 2024. URL https://arxiv.org/abs/2401.14361.

An Yang, Anfeng Li, Baosong Yang, et al. Qwen3 Technical Report, 2025. URL https://arxiv.org/abs/2505.09388.

Shien Zhu, Samuel Bohl, Robin Oester, and Gustavo Alonso. Pre-Attention Expert Prediction and Prefetching for Mixture-of-Experts Large Language Models, 2025. URL https://arxiv.org/abs/2511.10676.

A Experimental Details

Model and data. The target model has 48 MoE layers with 128 experts per layer and top-8 routing. The summarization workload consists of the 40 even-numbered of the 80 CNN/DailyMail articles in Spec-Bench, with inputs of 156–1,496 tokens and a limit of 2048 input plus output tokens. The GSM8K workload is the 80-question subset of Spec-Bench, with a limit of 512 input plus output tokens.

Backend and baselines. The offload backend replaces the expert modules of the Hugging Face Transformers Qwen3-MoE model. With the expert cache, missing experts are loaded on demand, with at most two copies in flight. A configuration without the expert cache clears the cache before every target forward, including prefill, and the clearing is timed; an expert shared by tokens of one forward is still loaded once, and the KV cache is kept. EAGLE-3 drafts the full five-level pool with the same generator as SpecFit and verifies all nodes when the pool has fewer than seven nodes near the generation limit.

Timing and aggregation. Each method's request stream begins with an empty expert cache, whose contents carry over between questions, while KV and draft state are new for every question. Measured request time includes prefill, drafting, atlas conversion, selection, target verification, commit, and required synchronization; model loading, atlas acquisition, and post-request scoring are excluded. For output count N_i, request time t_i^{req}, and prefill time t_i^{pre}, the throughput measures are

$$\mathrm{TPS}_{\mathrm{dec}} = \frac{\sum_i (N_i - 1)}{\sum_i (t_i^{\mathrm{req}} - t_i^{\mathrm{pre}})}, \qquad \mathrm{TPS}_{\mathrm{req}} = \frac{\sum_i N_i}{\sum_i t_i^{\mathrm{req}}}. \tag{9}$$

The first output token is attributed to prefill. Draft nodes, committed tokens, and ms/cycle are means per target verification cycle; an AR cycle is one decode forward and commits one token. H2D/token counts expert-weight transfers per output token, including prefill.

Calibration. The probes use the states of GSM8K questions 401–403 at cycle boundaries 0, 4, and 8. In each of these nine states, two probe trees are measured for each size $k = 0, \ldots, 7$, a probability-prefix tree and a minimum-expert-union tree, twice each and without clearing the expert cache. Averaging the repeats gives $9 \times 8 \times 2 = 144$ points (k_i, F_i, y_i), where y_i is cycle time without selector time. The fit minimizes $\sum_i (y_i - b_{k_i} - \beta F_i)^2$ subject to $\beta, b_0, \ldots, b_7 \geq 0$. The same probe trees are also timed with the expert cache cleared before every target forward, and SpecFit without the cache uses the fit to these 144 points. The RTX 4090 of Section 4.5 repeats this protocol with 32 expert slots per layer. In milliseconds, $\beta = 0.210715513$ with the cache and 0.408001022 without it on the A100, and 0.206037121 and 0.781593955 on the RTX 4090; the intercepts b_k are listed below.

	k	0	1	2	3	4	5	6	7
A100	With cache	142.214	147.532	150.522	152.269	152.743	163.204	155.175	156.068
	Without cache	122.702	117.975	118.338	116.389	115.309	114.770	113.887	113.671
RTX 4090	With cache	245.640	293.125	319.436	343.284	367.087	386.660	401.359	416.368
	Without cache	151.724	158.827	164.033	164.883	169.941	171.869	173.418	175.833

The intercepts are fitted independently and need not increase with k. Reproduction uses the full-precision frozen coefficients.

Measurement runs. On GSM8K, AR and EAGLE-3 with the expert cache come from the main run on the 80 questions, and SpecFit with the cache from a second run; a third run measures AR and EAGLE-3 without the expert cache, and a fourth SpecFit without it. On summarization, AR without the cache and EAGLE-3 and SpecFit with the cache come from one run on the 40 articles, AR with the cache and EAGLE-3 without it from a second run, and SpecFit without the cache from a third run; the slot sweep runs EAGLE-3 and SpecFit once at 24 and once at 12 slots. On the RTX 4090, SpecFit with the cache comes from one run on the 40 questions, AR without the cache and EAGLE-3 with and without it from a second run, SpecFit without the cache from a third run, and AR with the cache from a fourth. Every configuration has 80 requests on GSM8K on the A100 and 40 otherwise.

B SELECTION THEORY

B.1 THROUGHPUT OF A SELECTION RULE

A cycle state comprises the pool with its probabilities, routes, and legal family $\mathcal{A}_K$, and a rule π selects $S_\pi \in \mathcal{A}_K$ from the current state. Suppose that the states of successive cycles are independent and identically distributed and independent of earlier selections. If $\widehat{R}$ and $\widehat{T}$ equal the expected committed tokens and cycle time given the state, the renewal–reward theorem gives the long-run throughput $\Theta(\pi) = \mathbb{E}[\widehat{R}(S_\pi)]/\mathbb{E}[\widehat{T}(S_\pi)]$; otherwise $\Theta(\pi)$ is the throughput that the proxies predict. Let

$$h(\lambda) = \mathbb{E}\Big[\max_{S \in \mathcal{A}_K} \big(\widehat{R}(S) - \lambda \widehat{T}(S)\big)\Big]. \tag{10}$$

The intercepts are positive, so $\widehat{T}$ is bounded below by a positive constant, and h is continuous and strictly decreasing, with $h(0) \geq 1$ and $h(\lambda) \to -\infty$; let λ^* be its root. Every rule satisfies $\mathbb{E}[\widehat{R}(S_\pi) - \lambda^* \widehat{T}(S_\pi)] \leq h(\lambda^*) = 0$, so $\Theta(\pi) \leq \lambda^*$, and the rule selecting S^* attains equality. Hence $\lambda^* = \max_\pi \Theta(\pi)$ (Dinkelbach, 1967).

Lemma 1. *In any cycle, let S_λ maximize $\widehat{R}(S) - \lambda \widehat{T}(S)$ over $\mathcal{A}_K$. Then $(\rho - \lambda)\big(\widehat{T}(S_\lambda) - \widehat{T}(\widehat{S})\big) \geq 0$. In particular, $\widehat{T}(S^*) \leq \widehat{T}(\widehat{S})$ when $\rho < \lambda^*$, and $\widehat{T}(S^*) \geq \widehat{T}(\widehat{S})$ when $\rho > \lambda^*$.*

Proof. Since ρ is the largest ratio, $\widehat{R}(S) - \rho \widehat{T}(S) \leq 0$ for every S, with equality at $\widehat{S}$, so $\widehat{S}$ maximizes $\widehat{R} - \rho \widehat{T}$. Adding $\widehat{R}(S_\lambda) - \lambda \widehat{T}(S_\lambda) \geq \widehat{R}(\widehat{S}) - \lambda \widehat{T}(\widehat{S})$ and $\widehat{R}(\widehat{S}) - \rho \widehat{T}(\widehat{S}) \geq \widehat{R}(S_\lambda) - \rho \widehat{T}(S_\lambda)$ gives the claim. □

Proof of Proposition 1. The lower bound holds because λ^* is the largest predicted throughput. Since $\widehat{R}(\widehat{S}) = \rho \widehat{T}(\widehat{S})$, $\lambda^* - \Theta = -\mathbb{E}[(\rho - \lambda^*)\widehat{T}(\widehat{S})]/\mathbb{E}[\widehat{T}(\widehat{S})]$. Since $\widehat{R}(S^*) \leq \rho \widehat{T}(S^*)$, $0 = h(\lambda^*) = \mathbb{E}[\widehat{R}(S^*) - \lambda^* \widehat{T}(S^*)] \leq \mathbb{E}[(\rho - \lambda^*)\widehat{T}(S^*)]$. Adding this nonnegative quantity to the numerator gives Equation 7, and Lemma 1 with $\lambda = \lambda^*$ gives the sign of each term. If ρ is the same in every cycle, $\widehat{R}(S) \leq \rho \widehat{T}(S)$ holds for every selection, so $\lambda^* = \rho = \Theta$. □

Expert demand and β. Fix the intercepts, and let $\widehat{S}_1$ and $\widehat{S}_2$ be the selections under $\beta_1 < \beta_2$. Write the inverse ratio as $\widehat{T}(S)/\widehat{R}(S) = x(S) + \beta y(S)$ with $x(S) = b_{|S|}/\widehat{R}(S)$ and $y(S) = F(S)/\widehat{R}(S)$. Each selection minimizes the inverse ratio under its own coefficient, so $x(\widehat{S}_1) + \beta_1 y(\widehat{S}_1) \leq x(\widehat{S}_2) + \beta_1 y(\widehat{S}_2)$ and $x(\widehat{S}_2) + \beta_2 y(\widehat{S}_2) \leq x(\widehat{S}_1) + \beta_2 y(\widehat{S}_1)$. Adding them gives $(\beta_2 - \beta_1)\big(y(\widehat{S}_2) - y(\widehat{S}_1)\big) \leq 0$, so $F(\widehat{S}_2)/\widehat{R}(\widehat{S}_2) \leq F(\widehat{S}_1)/\widehat{R}(\widehat{S}_1)$.

B.2 ADAPTIVE DEPTH

Lemma 2. *Let $\widehat{S}_d$ be the selection over $V_{\leq d}$, with $\widehat{S}_0$ the root-only choice. The legal families are nested in d, so the largest ratio is nondecreasing in d. Adaptive depth stops at the first level d with $\widehat{S}_d = \widehat{S}_{d-1}$, and its selection equals the full-pool selection $\widehat{S}$ unless $\widehat{S}$ contains a node deeper than d.*

Proof. A legal subtree of $V_{\leq d}$ has at most $\min(7, |V_{\leq d}|)$ nodes and is therefore a legal subtree of $V_{\leq d'}$ for every $d' > d$, which gives the nesting and the monotone largest ratio. The tie rule orders all subtrees independently of the pool, so a subtree that is optimal in a family is optimal in every subfamily containing it. If $\widehat{S}_d$ has no level-d node, it lies in the family of $V_{\leq d-1}$ and hence equals $\widehat{S}_{d-1}$; conversely, $\widehat{S}_{d-1}$ has no level-d node. Likewise, if $\widehat{S} \subseteq V_{\leq d}$, then $\widehat{S}_d = \widehat{S}$. □

In cycles whose largest ratio is below λ^*, the throughput-optimal rule selects no more predicted time than Equation 5 (Lemma 1), so a stopped tree with less predicted time than $\widehat{S}$ departs from Equation 5 in the same direction as that rule.

B.3 Complexity and Exact Search

Proof of Proposition 2. A subtree is a certificate checkable in polynomial time, so the problem is in NP. We reduce from CLIQUE (Karp, 1972): given a graph with n vertices, m edges, and an integer $2 \le r \le n$, decide whether some r vertices are pairwise adjacent. Let $k = r(r-1)/2$ and $p = 1/(m+1)$. The pool has one node per edge, each a child of the root. The single MoE layer has one expert per vertex; the node of edge $\{u, w\}$ routes to $\{u, w\}$, and the root routes to no expert. Set $p_v = p$ for every node, $b_j = b > 0$ for every j, $K = k$, $0 < \beta < bp/\big(n(1 + (k-1)p)\big)$, and threshold $\theta = (1 + kp)/(b + \beta r)$. Then $F(S)$ is the number of vertices covered by the edges in S, and the edges of an r-clique reach θ. Conversely, a subtree with fewer than k nodes has ratio at most $(1 + (k-1)p)/b < \theta$ by the choice of β, so a subtree reaching θ has k edges covering at most r vertices. Since q vertices span at most $q(q-1)/2$ edges, these edges cover exactly r vertices and are all pairs among them, an r-clique.

Without the size limit, let $b_j = b_0 + aj$ with $b_0 > 0$ and $a \ge 0$. Apart from the root's experts, $F(S)$ counts the pairs (ℓ, e) such that some $v \in S$ routes to expert e in layer ℓ. For fixed $\lambda \ge 0$, maximizing $\widehat{R}(S) - \lambda \widehat{T}(S)$ over ancestor-closed S is then a maximum-weight closure problem (Picard, 1976). Its network has a vertex for each node and each pair, an arc of capacity p_v from the source to each node, arcs of capacity λa from each node and $\lambda\beta$ from each pair to the sink, and infinite arcs from each node to its parent and to its pairs; the source side of a minimum cut is an optimal subtree with its pairs. Source capacities are constant and sink capacities are nondecreasing in λ, which after reversing all arcs is the form handled by Gallo et al. (1989): one parametric maximum flow yields nested minimum cuts for all λ, at most as many as the network has vertices. At λ equal to the largest ratio, the maximum of $\widehat{R} - \lambda\widehat{T}$ is zero and is attained only by subtrees of that ratio, so the best ratio among the nested cuts is the largest ratio. □

Exact search. Algorithm 1 gives the search of Section 3.3. Pool node IDs put parents before descendants, and the depth-first traversal adds nodes in increasing ID order, each only when its parent is present, so it reaches every subtree in $\mathcal{A}_K$ exactly once.

Algorithm 1 SpecFit selector

Require: Pool V, root s, atlas $\widehat{E}$, frozen coefficients (b, β)
1: Read cumulative draft probabilities p_v and encode expert bitmaps B_v
2: Initialize the incumbent to the root-only choice
3: **for all** ancestor-closed $S \subseteq V$, $|S| \le \min(7, |V|)$, in depth-first order **do**
4: Carry $B(S) = B_s$ OR $\bigvee_{v \in S} B_v$ from the parent state
5: $r \leftarrow 1 + \sum_{v \in S} p_v$; $f \leftarrow \mathrm{popcount}(B(S))$
6: $t \leftarrow b_{|S|} + \beta f$
7: Update incumbent by $(r/t, r, -f, -|S|)$ and the node-ID tie rule
8: **end for**
9: **return** Incumbent subtree

B.4 Evaluation on Recorded Pools

We evaluate these quantities on the candidate pools that SpecFit drafted in every cycle of its runs with the expert cache (Appendix A), with the coefficients fitted with the cache: 4,389 cycles on the 40 summarization articles and 4,407 cycles on the 80 GSM8K questions. The legal family of a cycle is that of its drafted pool, expectations are taken over the recorded cycles with the trajectory held fixed, and λ^* is computed by Dinkelbach's iteration (Table 5). The optimality gap of Equation 5 is below 1% of λ^* and lies within the bound of Equation 7, although the two rules select different trees in about a quarter of the cycles. With the recorded acceptances in place of $\widehat{R}$, S^* commits 0.5% and 0.7% more tokens per unit of predicted time than $\widehat{S}$. Doubling β lowers both the expert demand per committed token and the number of selected nodes. Drafting stops before level 5 in 85.8% of the summarization cycles and 33.7% of the GSM8K cycles.

Table 5: Selection theory evaluated on the candidate pools drafted by SpecFit, with the coefficients fitted with the cache. Percentages of λ^* refer to the predicted throughput.

Quantity	Summarization	GSM8K
Cycles	4,389	4,407
$\lambda^* - \Theta$ (% of λ^*)	0.74	0.95
Bound of Equation 7 (% of λ^*)	1.38	1.75
Cycles with $S^* \neq \widehat{S}$ (%)	26.6	24.5
Coefficient of variation of ρ	0.31	0.25
Gain of S^* with recorded acceptances (%)	0.53	0.69
Mean $F(\widehat{S})/\widehat{R}(\widehat{S})$ at β / 2β	365.5 / 351.5	305.7 / 298.0
Mean selected nodes at β / 2β	2.88 / 2.35	5.09 / 4.54
Mean drafted levels	2.86	4.39
Cycles stopped before level 5 (%)	85.8	33.7