



Goose: Anisotropic Speculation Trees for Training-Free Speculative Decoding

Tao Jin* Phuong Minh Nguyen Naoya Inoue
 Japan Advanced Institute of Science and Technology (JAIST)
 {morgan, phuongnm, naoya-i}@jaist.ac.jp
 🐦 [GOOSE_Speculative_Decoding](#)

Abstract

Speculative decoding accelerates large language model inference by drafting multiple candidate tokens and verifying them in a single forward pass. Candidates are organized as a tree: deeper trees accept more tokens per step, but adding depth requires sacrificing breadth (fallback options) under a fixed verification budget. Existing training-free methods draft from a single token source and shape their trees without distinguishing candidate quality across origins. We observe that two common training-free token sources— n -gram matches copied from the input context, and statistical predictions from prior forward passes—differ sharply in acceptance rate ($\sim 6\times$ median gap, range $2\text{--}18\times$ across five models and five benchmarks). We prove that when such a quality gap exists, the optimal tree is *anisotropic* (asymmetric): reliable tokens should form a deep chain while unreliable tokens spread as wide branches, raising the depth ceiling of balanced trees. We realize this structure in GOOSE, a training-free framework that builds an adaptive **spine tree**—a deep chain of high-acceptance context-matched tokens with wide branches of low-acceptance alternatives at each node. The resulting tree provably accepts at least as many tokens per step as either source alone. On five LLMs (7B–33B) and five benchmarks, GOOSE achieves $1.9\text{--}4.3\times$ lossless speedup, outperforming balanced-tree baselines by 12–33% under the same budget.

1 Introduction

Speculative decoding (Leviathan et al., 2023; Chen et al., 2023) accelerates autoregressive LLM inference by letting a drafter propose several candidate tokens at once. The target model verifies them in a single forward pass, accepting correct guesses at no extra cost while keeping the output provably identical to standard decoding (*lossless*; Leviathan et al., 2023).

The simplest draft is a single *chain*: efficient when the drafter is accurate, but a single rejection loses the entire continuation. *Tree-structured speculation* (Miao et al., 2024; Cai et al., 2024) hedges this risk with multiple candidates organized as a tree, whose **tree attention mask** restricts each candidate’s attention to its ancestors so the target model scores every root-to-leaf path in one forward pass. Designing a good *draft tree* then means allocating a fixed node budget B between *depth* (longer paths) and *breadth* (more fallbacks).

Among training-free methods, three representative approaches illustrate the design space and its limits. *Token Recycling* (TR; Luo et al., 2025) retrieves statistically likely next tokens from the target model’s own logits, producing wide, shallow trees that hedge broadly but sacrifice depth. *Prompt Lookup Decoding* (PLD; Saxena, 2023) copies n -gram-matched continuations from the input context, producing a deep chain accepted at high rates, but with no fallback when the match breaks or is absent. Sequoia (Chen et al., 2024) optimizes the depth–breadth trade-off by dynamic programming, but under a source-blind acceptance model: acceptance depends on a candidate’s rank among siblings, not on its provenance.

*Corresponding author.

Such a model cannot favor high-acceptance tokens when sources of different reliability coexist.

This source-blind assumption forfeits a key structural advantage once both sources coexist. We measure that context-matched (PLD) tokens are accepted $2\text{--}18\times$ more often than transition (TR) tokens (median $\sim 6\times$ across five models and five benchmarks; Section 3). This *acceptance heterogeneity* suggests placing high-acceptance tokens along a deep *spine* and low-acceptance tokens as wide *branches* at every spine node: the spine pushes depth while branches provide fallback when it breaks early. We instantiate this insight in GOOSE, a training-free framework that builds an adaptive **spine tree** (Figure 2): a deep chain of context-matched tokens, with wide branches of transition alternatives forking from every spine node, verified by a unified greedy walk in one forward pass. We prove that the spine tree’s expected yield (accepted tokens per cycle) is at least that of either source used alone (Proposition 4), and design runtime adaptations that preserve this property when one source is absent or dominant (Section 4.3).

Our contributions:

1. We formalize acceptance heterogeneity between two training-free token sources and prove that the optimal tree is *anisotropic*, allocating depth to reliable tokens and breadth to unreliable ones, which raises the depth ceiling of single-source trees (Section 3).
2. We design a confidence-adaptive, **training-free** tree construction that combines PLD and TR into a single spine tree with a non-degradation guarantee on expected yield (Section 4).
3. Experiments on five models (7B–33B) across five benchmarks show $1.9\text{--}4.3\times$ lossless speedup, outperforming isotropic-tree baselines by 12–33% under equal node budgets (Table 1).

2 Related Work

Speculative decoding (Leviathan et al., 2023; Chen et al., 2023; Stern et al., 2018) accelerates autoregressive LLM inference by guessing multiple future tokens and verifying them in a single forward pass; Xia et al. (2024) provide a comprehensive survey. A key extension is *tree-structured verification* (Miao et al., 2024; Cai et al., 2024): candidates are organized into a tree and checked simultaneously, after which the longest accepted path is selected and one bonus token is appended.

Draft-model methods. Representative approaches include EAGLE (Li et al., 2024a;b), which trains feature-level draft heads, and Medusa (Cai et al., 2024), which attaches parallel prediction heads; EAGLE-3 (Li et al., 2025) replaces feature prediction with direct token prediction and multi-layer feature fusion, achieving up to $\sim 6.5\times$ speedup. All such methods (Miao et al., 2024; Kim et al., 2023; Zhou et al., 2024, *inter alia*) require per-model training or auxiliary models, limiting deployment flexibility.

Training-free methods. Training-free drafters avoid auxiliary models entirely by mining candidate tokens from data already available at inference time. The simplest strategy copies tokens from reference documents (Yang et al., 2023) or from the prompt and prior output via n -gram matching (Saxena, 2023); such copies are cheap but limited to recurring patterns. A broader class retrieves candidates from external datastores—REST (He et al., 2024) uses a suffix array, SuffixDecoding (Oliaro et al., 2025) a suffix tree over the prompt and prior outputs (compared head-to-head in Section 5.2), while Lookahead Decoding (Fu et al., 2024) constructs n -grams on the fly through parallel Jacobi iterations. Other methods refine token selection (Somasundaram et al., 2025; Le et al., 2025) or reuse the model’s own early layers (Zhang et al., 2024; Elhoushi et al., 2024). All the above improve *which tokens to draft*. GOOSE asks the orthogonal question of *how the verification tree should be shaped* given sources with very different acceptance rates; better token selection would only widen the gap the spine tree exploits.

Tree topology optimization. Sequoia (Chen et al., 2024) formalizes the depth–breadth trade-off introduced above: under its *positional acceptance assumption*, acceptance depends only on a token’s rank among siblings, so its DP produces non-uniform trees while remaining *source-blind*—one acceptance vector for all candidates. We call a balanced tree with the same branching at every depth *isotropic* and use it as our equal-budget topology control (Section 5.3); we additionally compare against Sequoia’s official DP-optimized trees in Section 5.2.

EAGLE-2 (Li et al., 2024b) prunes low-confidence branches via the draft head’s per-token scores, growing deeper along well-predicted paths; this adapts the shape to within-drafter quality variation but requires a trained head and lacks optimality guarantees. TR (Luo et al., 2025) applies a similar principle within its transition lookup table: higher-ranked candidates receive more children than lower-ranked ones, producing non-uniform trees without training.

Despite producing non-uniform trees, these methods optimize within a single draft source, so the quality differences they exploit remain *within-source*. GOOSE exploits a qualitatively different axis: the *cross-source* gap between context-matched and transition tokens (2–18 $\times$; Section 3). Extending Sequoia’s DP to this heterogeneous setting yields an *anisotropic* optimum, in which reliable tokens form a deep chain while unreliable ones spread as wide branches, confirmed by 12–33% gains (Table 1).

3 Optimal Topology under Heterogeneous Acceptance

The optimal tree shape follows directly from the acceptance asymmetry between the two draft sources. We build on Sequoia’s framework (Chen et al., 2024), which models each draft token as independently accepted with a source-independent probability and optimizes the tree via dynamic programming; we extend the model to two sources with different acceptance rates. This section formalizes that asymmetry, derives the expected yield of the spine tree (Proposition 1), proves that branches should be concentrated near the root (Proposition 2), and shows that the resulting tree strictly dominates any single-source alternative (Proposition 3).

Heterogeneous acceptance model. Following this model, we associate each draft source with a per-token acceptance probability: p_s for context-matched (spine) tokens and p_t for transition tokens, with $p_s > p_t$. Each spine token is modeled as independently accepted with probability p_s . Consecutive tokens are positively correlated in practice; this leaves the bound in Equation (1) valid, only more conservative (validated in all 24 settings, Appendix E.4).

To ground the model, we measure acceptance rates from greedy verification logs across all five models and five benchmarks (Figure 1). The empirical spine acceptance $\hat{p}_s$ (fraction of context-matched draft tokens accepted by the target model) ranges from 0.07 to 0.60 (median 0.21), while the empirical transition acceptance $\hat{p}_t$ lies in 0.03–0.05 (median 0.033). The resulting heterogeneity ratio $\hat{p}_s/\hat{p}_t$ spans 2 $\times$ to 18 $\times$ (median $\approx 6\times$; Figure 1). Note that p_s and p_t are *population* rates, used in this yield analysis and in tree construction only: verification (Section 4.2) is pure argmax token-identity matching. When Proposition 3 “upgrades” a slot from p_t to p_s , this describes which source class occupies it in the model, not a runtime score; a branch continuing after a spine break keeps its own class rate p_t and inherits nothing from the spine.

Key quantities. Before stating the main results, we fix notation. The spine consists of m context-matched tokens forming a linear chain; at each spine node i ($0 \leq i \leq m-1$), w_i transition candidates fork off as branches. The total number of candidates—spine plus all branches—is bounded by the node budget B (i.e., $m + \sum_i w_i \leq B$; branch extension chains of up to $D-1$ nodes are idealized as budget-free here, while the deployed builder charges every node to B , Section 4.1). Let τ denote the number of tokens accepted in one verification cycle (Leviathan et al., 2023; Chen et al., 2024), which we call the *compression ratio*; the *expected yield* is then $\mathbb{E}[\tau]$. We also define two recurring terms. Given w_i branch candidates at spine node i , each accepted independently with probability p_t , the probability

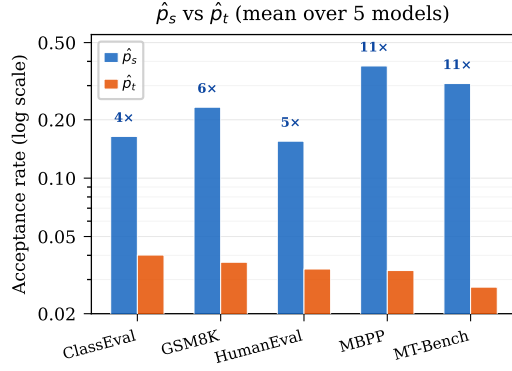


Figure 1: **Acceptance heterogeneity: context-matched spine tokens are accepted far more often than transition branch tokens.** $\hat{p}_s / \hat{p}_t$ = fraction of PLD spine / TR branch draft tokens accepted (greedy verification logs); bars: means over the five models, log scale; $N\times$: the ratio, which exceeds 1 in all 25 settings—the empirical basis for the anisotropic design.

that *at least one* is correct is $\phi_i = 1 - (1 - p_t)^{w_i}$. Intuitively, ϕ_i grows quickly with w_i even when p_t is small. Each accepted branch token can extend into a chain of further transition tokens (maximum depth $D=6$ in all experiments), reaching depth k with probability p_t^k ; summing gives the expected extension length: $\bar{\ell} = \sum_{k=1}^{D-1} p_t^k = p_t(1 - p_t^{D-1}) / (1 - p_t)$, which ranges from 0 (when $p_t \rightarrow 0$) to $D-1$ (when $p_t \rightarrow 1$).

We now bound $\mathbb{E}[\tau]$ via the *indicator random variable* (IRV) decomposition (Leviathan et al., 2023; Chen et al., 2024): the expected accepted token count equals the sum over all tree depths of the probability that the accepted path reaches and accepts that depth.

Proposition 1 (Spine Tree Expected Yield). *Under the heterogeneous model with spine length $m \geq 1$ and branch widths $\{w_i\}_{i=0}^{m-1}$ (total budget $m + \sum_i w_i \leq B$), the expected accepted path length satisfies:*

$$\mathbb{E}[\tau] \geq \underbrace{\sum_{i=1}^m p_s^i}_{\text{spine}} + \underbrace{\sum_{i=0}^{m-1} p_s^i (1-p_s) \phi_i (1+\bar{\ell})}_{\text{continuation synergy}} + 1 \quad (1)$$

with ϕ_i and $\bar{\ell}$ as defined above. The bound is tight when the branches are independent chains.

Proof sketch. By the IRV decomposition (Leviathan et al., 2023; Chen et al., 2024), $\mathbb{E}[\tau]$ is a sum over depths of the probability the accepted path reaches that depth: the spine term is $\sum_i p_s^i$; in the synergy term, $p_s^i (1-p_s)$ is the probability the spine breaks exactly at depth $i+1$, ϕ_i the probability some branch there succeeds, and $(1+\bar{\ell})$ its expected token contribution. The final $+1$ is the standard bonus token; deeper sub-branching only adds contributions, so Eq. 1 is a lower bound. Full derivation in Appendix E.1. $\square$

Since the spine tree includes both the pure PLD chain ($m=B$, all $w_i=0$) and a near-pure TR tree ($m=1$) as special cases, its yield never falls below either standalone strategy; $p_s \rightarrow 1$ recovers pure spine, $p_s \rightarrow p_t$ recovers isotropic. In practice, the spine ratio adapts via an exponential moving average of the acceptance rate (Section 4.3).

Proposition 2 (Optimal Branch Allocation). *The allocation $\{w_i^*\}$ maximizing the synergy term under total branch budget $B_t = B - m$ satisfies:*

$$w_i^* = w_0^* - \frac{|\ln p_s|}{|\ln(1-p_t)|} \cdot i \quad (2)$$

where w_0^* is set by the budget constraint $\sum w_i = B_t$ (interior solution; otherwise its water-filling truncation, Appendix E.6). The allocation decreases linearly with depth—deeper nodes are reached

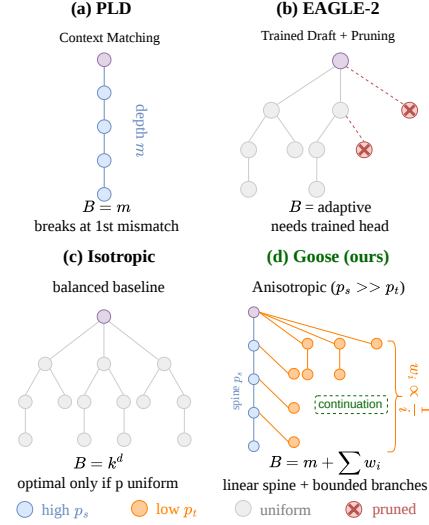


Figure 2: **Speculation-tree topologies** (B fixed). (a) PLD spine; (b) EAGLE-2 pruned tree; (c) isotropic tree, our equal-budget control; (d) GOOSE’s anisotropic spine tree (Section 4.1).

less often (p_s^i decays exponentially)—and the slope $|\ln p_s|/|\ln(1-p_t)|$ equalizes the marginal value of one extra branch across positions.

The proof follows from Lagrange multipliers; see Appendix E.2. In practice, we approximate the linear schedule with a $1/i$ harmonic rule (Algorithm 1) that preserves the monotone-decreasing shape without estimating acceptance rates. Empirically, τ is robust to the r/ρ split within this family (Figure 5b); a measured-acceptance allocator improves on the harmonic rule by a further $\sim 6\%$ (Appendix D.10).

Remark (i.i.d. as a modeling approximation). The i.i.d. acceptance model in Propositions 1–3 is a *modeling approximation*, and it enters only the theoretical analysis, not the measurements: the headline 12–33% anisotropic-vs-isotropic gain (Table 1) is a direct equal-budget measurement that assumes no acceptance law. Positive intra-chain correlation, if anything, makes the modeled allocation conservative; empirical gains in Table 1 confirm the model captures the dominant qualitative structure (Appendix E.4). Empirically, acceptance decay over depth is fit better by a geometric law ($R^2=0.77$ vs. 0.69; Qwen3-8B traces, $\approx 0.5M$ tree nodes per depth), whereas the all-reject frequency as a function of branch width is fit better by a power law (Llama-3 traces, Appendix E.5; with a selection-effect caveat). In Appendix E.6 we re-derive the optimal allocation under that empirically-fit tail ($\hat{\alpha}=0.261$) and show the optimum keeps the same root-concentrated anisotropic shape; our deployed harmonic schedule is more conservative than either optimum at the root while retaining residual depth-side breadth.

Proposition 3 (Spine Tree Dominance). *For acceptance rates $p_s > p_t > 0$ and any budget B large enough that the spine contains at least two nodes ($B \geq m + 2$, practically $B \geq 10$ for $p_s/p_t \geq 8$), the spine tree with optimally allocated branches (Eq. 2) achieves strictly higher expected yield than the best single-source isotropic tree of the same budget:*

$$\mathbb{E}[\tau_{\text{spine}}^*] > \mathbb{E}[\tau_{\text{iso}}^*] \quad \text{whenever } p_s > p_t. \quad (3)$$

The gap increases monotonically with p_s/p_t and vanishes as $p_s \rightarrow p_t$.

Proof sketch. Start from the optimal isotropic tree of budget B (the single-rate optimum of Sequoia-style DP (Chen et al., 2024) when every token carries rate p_t). Upgrade one token per depth from p_t to $p_s > p_t$, creating a spine. At each modified depth d , the probability of at least one accepted candidate increases by $(p_s - p_t)(1 - p_t)^{w_d - 1} > 0$, a strictly positive improvement that propagates through the yield. Optimizing the allocation (Proposition 2) adds further gains. When $p_s \rightarrow p_t$, the upgrade vanishes, recovering the isotropic baseline. Full proof in Appendix E.3. $\square$

4 Method: GOOSE

GOOSE follows the standard speculative-decoding loop (Cai et al., 2024)—generate candidates, verify them in one forward pass, accept the longest correct path—but reshapes the candidate tree into a single asymmetric structure that combines two complementary token sources (Figure 2d). Each decoding cycle begins from an *anchor token*, the last token accepted in the previous cycle, and proceeds in three stages (Figure 3). *Stage 1—Build spine tree.* GOOSE assembles a *candidate pool* from two sources. Context matching (PLD) retrieves a continuation from previously generated text, forming a deep chain called the *spine*. The *adjacency table*, our bigram extension of TR’s adjacency matrix (Section 4.1; implementation in Appendix C.1), supplies statistically likely alternatives that fork from spine nodes as *branches*. BUILDSPINETREE allocates the node budget B between spine and branches (Section 4.1), then packs all candidates into one sequence with a *tree attention mask*: a modified causal mask that lets each node attend only to its root-to-node ancestors, so all paths are verified in one forward pass. *Stage 2—Verify and harvest.* One LLM forward pass scores every candidate. All logits—including those at rejected branches—are harvested into the adjacency table, and the accepted prefix updates the PLD context index, enriching both sources for future cycles. *Stage 3—Select longest correct path.* A greedy walk identifies the longest path verified as correct by the target model (Section 4.2); the last accepted token becomes the anchor of the next cycle.

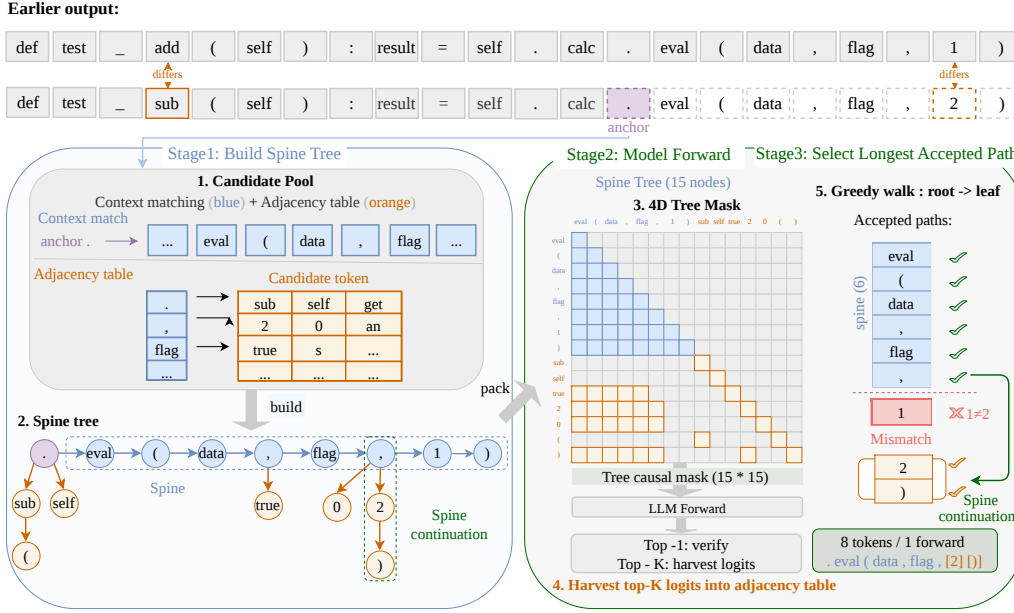


Figure 3: **GOOSE pipeline overview.** From anchor . , Stage 1 draws from a unified *candidate pool*: context matching produces the spine (blue), the adjacency table supplies branches (orange). Stage 2 verifies all candidates via one LLM forward pass. Stage 3 selects the longest accepted path via a greedy walk, discovering **spine continuation**: “2” → “)” extends beyond the spine mismatch, yielding 8 tokens per call.

The tree shape is not fixed. When context-match confidence is high, GOOSE skips tree construction and verifies the PLD chain linearly (*bypass mode*; Section 4.3); when no context match exists, the spine vanishes and the full budget forms a TR-only tree. The asymmetric topology enables a synergy we call *spine continuation*: when a mismatch occurs on the spine, a branch at that point can extend the accepted path, recovering tokens that neither source alone would have found.

Two refinements improve candidate quality (ablated in Table 2): agreement among n -gram lengths on the first token is treated as high confidence and extends the spine aggressively, and the adjacency table uses bigram rather than unigram conditioning to sharpen branch predictions. The complete decode loop is in Algorithm 2 (Appendix).

4.1 Spine Tree Construction

Tree construction. Given the anchor token and node budget B , one parameter controls the budget split: the *spine ratio* r , which sets what fraction of B goes to spine nodes (adapted online; Section 4.3). Construction proceeds in four steps (Algorithm 1, Appendix): (1) lay the spine—allocate up to $B \cdot r$ nodes as a linear chain of context-matched tokens; (2) add root branches—attach the top adjacency-table alternatives at the root, using a share $(1 - \rho)$ of the remaining budget (spine-branch ratio $\rho = 0.5$ by default); (3) add spine branches—distribute the remaining share ρ across spine nodes, with wider branches near the root and narrower ones deeper, approximating the optimal allocation derived in Proposition 2; (4) extend all branches recursively via the adjacency table up to a maximum depth of 6. As introduced above, the adjacency table conditions on the two most recent tokens rather than TR’s original single-token lookup (Luo et al., 2025), improving prediction accuracy.

4.2 Unified Verification and Non-Degradation

The spine tree implicitly contains all candidate paths; verification requires only a single greedy walk.

Greedy walk with source priority. Starting at the root, the walk advances to a child whose draft token equals the target model’s greedy argmax at the current node—pure token-identity matching, with *no per-token score gate*: draft-side scores (TR adjacency counts, PLD match length) are used only at construction time and are never consulted during verification, and no acceptance probability is carried from parent to child. At each node the walk checks the PLD (spine) child before the TR (branch) children; duplicates are excluded, so at most one child can match and the priority never changes the accepted path; once the walk leaves the spine for a branch it does not return (Appendix B.3). The walk terminates when no child matches, and the accepted path falls into one of three categories: (a) *Pure* PLD (all spine nodes, equivalent to standalone context-match), (b) *Spine continuation* (spine followed by a TR branch at the mismatch point—the synergy unique to the spine tree), or (c) *Pure* TR (root branch only, equivalent to standalone transition). No multi-path bookkeeping is needed; the greedy walk automatically discovers the longest accepted path. At termination, the target model samples one additional *bonus token* from its distribution at the last accepted position, following the standard speculative-decoding guarantee (Leviathan et al., 2023).

Proposition 4 (Non-Degradation Guarantee). *For any node budget B , the spine tree’s expected yield—the expected number of tokens accepted per cycle—is at least as large as that of standalone context matching or standalone transition-tree decoding, each given the full budget B .*

Proof sketch. (i) Standalone PLD yields a chain bounded by the match length, not B ; GOOSE preserves those tokens in full (short matches fit within $\lfloor Br \rfloor$; long matches trigger bypass, Section 4.3) and fills the unused budget with branches, so its accepted path is at least as long. (ii) Versus a standalone transition tree, the spine swaps breadth nodes for higher-acceptance context-matched depth, which can only maintain or increase expected path length (Proposition 3); with no context match, the spine vanishes and the transition tree is recovered exactly. $\square$

Remark. The guarantee holds for expected yield, not directly wall-clock, and parts (i)/(ii) rely on the runtime adaptations of Section 4.3 (minimum spine ratio, bypass, graceful fallback) for the boundary cases a fixed shape cannot cover; Table 2 quantifies their contributions.

The greedy walk is formalized in Algorithm 3; the full decoding loop, including tree construction and logit harvesting, is in Algorithm 2 (Appendix).

4.3 Confidence-Adaptive Topology

Not every cycle benefits equally from tree construction. GOOSE uses a continuous *PLD confidence* signal to modulate tree shape, ranging from a full spine tree down to a simple linear chain.

Confidence signal. GOOSE queries context matching with multiple n -gram lengths $\{3, 4, 5\}$. Confidence is a function of two indicators: (i) *consensus*—at least two n -gram lengths return continuations that agree on the first token, and (ii) *chain length*—long matches (8 or more tokens) indicate high PLD acceptance. When confidence is high, tree construction is skipped and the spine is verified as a single linear sequence (*bypass* mode), preserving the guarantee of Proposition 4. The spine ratio r is adjusted each cycle by an EMA of the PLD acceptance rate: high averages deepen the spine ($r \rightarrow 0.50$), low ones shift budget to branches ($r \rightarrow 0.15$), converging within 3–5 cycles (details in Appendix C.3).

5 Experiments

5.1 Setup

Models, hardware, and datasets. We evaluate five instruction-tuned models spanning 7B–33B parameters: Vicuna-7B/13B/33B (Zheng et al., 2023), Llama-3-8B-Instruct (Meta AI, 2024), and Qwen3-8B (Qwen Team, 2025), all in FP16. The 7B–13B models run on a single NVIDIA A40 (48 GB); Vicuna-33B runs on $2 \times$ A100-40GB. We evaluate on five benchmarks spanning code, math, and dialogue: HumanEval (Chen et al., 2021), MBPP (Austin et al., 2021), ClassEval (Du et al., 2023), GSM8K (Cobbe et al., 2021), and MT-Bench (Zheng et al.,

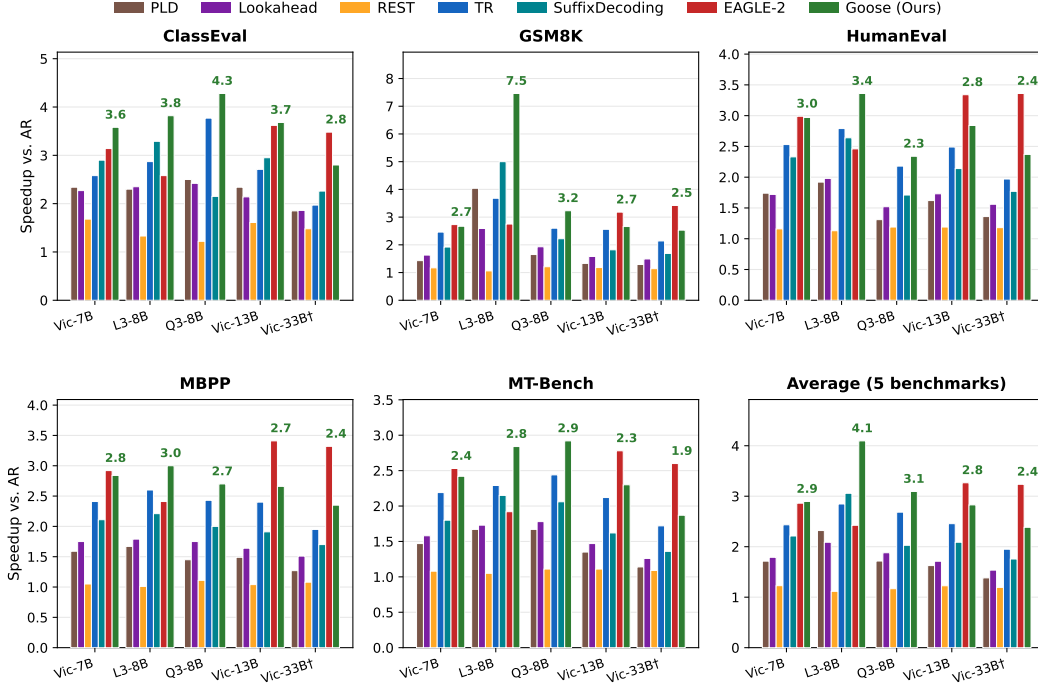


Figure 4: Wall-clock speedup over autoregressive (AR) decoding across five models and five benchmarks. All methods are lossless (greedy decoding, identical output); GOOSE values are annotated above each group and the sixth panel is the per-model mean. SuffixDecoding is the strongest training-free system reported in prior work, EAGLE-2 the strongest trained one shown (absent on Qwen3-8B, which has no released head). Per-benchmark τ and Sequoia: Table 3; EAGLE-3: Appendix D.4. †Vicuna-33B runs on $2 \times A100-40GB$; all others on a single A40.

2023)—ranging from high repetition (code templates) to low repetition (open dialogue). Dataset sizes and generation lengths are listed in Table 4. Batch size 1; seed 0.

Baselines and metrics. AR (standard autoregressive), standalone PLD (Saxena, 2023), standalone TR (Luo et al., 2025) (using its default imbalanced BFS tree with unigram adjacency), Lookahead Decoding (Fu et al., 2024), REST (He et al., 2024), and EAGLE-2 (Li et al., 2024b) (draft-head method; Vicuna-7B, Llama-3-8B, Vicuna-13B, and Vicuna-33B using official released heads). For the topology comparison (Section 5.3), we evaluate **Isotropic(k)**: a balanced k -ary tree with $k \in \{3, 5\}$ and the same node budget, using GOOSE’s draft source. This baseline is a *topology control* (uniform shape at equal budget), not a reproduction of Sequoia, whose DP builds *non-uniform* trees (Chen et al., 2024); we compare against its official implementation in Section 5.2. All methods use greedy decoding and produce identical output (lossless). We additionally compare with EAGLE-3 (Li et al., 2025) in Section 5.2; its drafter is a single decoder layer plus an FC fusion layer, comparable in scale to EAGLE-2’s head. We report τ (compression ratio: mean tokens accepted per verification call; cf. Section 3), tok/s (wall-clock throughput), and Speedup (tok/s vs. AR); all hyperparameters are fixed across datasets and models (node budget $B=60$, max branch depth $D=6$, top- $K=10$ adjacency candidates, spine-branch ratio $\rho=0.5$; full list in Appendix C.3). During the first few cycles, the adjacency table is empty; GOOSE gracefully degrades to pure PLD (or AR if no context match exists) and populates the table from the resulting logits.

5.2 Main Results

Figure 4 summarizes the main results. GOOSE achieves $1.9\text{--}4.3\times$ lossless speedup across all five models, consistently outperforming every training-free baseline. Speedup varies predictably with acceptance heterogeneity: ClassEval (repetitive templates) yields up to

Table 1: Compression ratio τ for spine vs. isotropic trees (budget $B=60$). Δ_3 : relative gain over Iso(3).

Model	Dataset	Spine (Ours)	Iso(3)	Iso(5)	Δ_3
L3-8B	ClassEval	4.89	3.67	3.60	+33%
	HumanEval	4.11	3.32	3.27	+24%
	MT-Bench	3.50	2.76	2.76	+27%
Q3-8B	ClassEval	5.19	3.97	3.92	+31%
	GSM8K	3.87	3.00	3.01	+29%
	HumanEval	2.78	2.49	2.51	+12%
	MBPP	3.20	2.73	2.77	+17%
	MT-Bench	3.51	2.81	2.81	+25%
Average		3.88	3.09	3.08	+25.4%

Table 2: Ablation on Qwen3-8B (τ). Each row removes one design choice; $\bar{\Delta}$: macro-averaged relative change.

Config.	HE	GS	MB	CE	MT	$\bar{\Delta}$
GOOSE (Ours)	2.78	3.87	3.20	5.19	3.51	—
<i>Idea 1: Anisotropic topology (§4.1)</i>						
w/o spine branches	2.71	3.70	3.10	5.03	3.46	-2.9%
w/o bigram adjacency	2.69	3.66	3.11	5.01	3.38	-3.7%
<i>Idea 2: Confidence-adaptive budget (§4.3)</i>						
w/o consensus bypass	2.77	3.83	3.05	4.60	3.22	-5.1%
w/o context-match spine	2.68	3.76	3.11	5.00	3.37	-3.4%

4.3 $\times$; MT-Bench (open dialogue) still reaches 1.9–2.9 $\times$, confirming that TR branches alone contribute meaningful acceleration. One outlier: on Llama-3-8B GSM8K, near-verbatim repetition inflates speedup to 7.5 $\times$ ($\tau=9.25$); all other 24 settings fall within the 1.9–4.3 $\times$ speedup range.

Comparison with EAGLE-2. EAGLE-2’s trained draft head yields higher τ (Table 3), but its per-step latency offsets the advantage: GOOSE outperforms EAGLE-2 on Llama-3-8B (4.1 $\times$ vs. 2.4 $\times$), matches on Vicuna-7B (2.9 $\times$ vs. 2.9 $\times$), while EAGLE-2 leads on Vicuna-13B (2.8 $\times$ vs. 3.3 $\times$) and Vicuna-33B (2.4 $\times$ vs. 3.2 $\times$). Two factors explain this pattern. *Model size sets the wall-clock winner*: at ≤ 8 B the base forward is cheap, so zero drafter cost outweighs EAGLE’s higher τ ; at ≥ 13 B the slower base forward lets that higher τ convert and win. *Task repetition sets the margin*: the spine fires most on repetitive workloads and least on open-ended dialogue. We read this as an empirical trend on these models, not a strict size law. As a training-free method, GOOSE also covers models for which no trained draft head exists (Qwen3-8B).

Comparison with stronger baselines. Three further systems bracket the strongest alternatives; SuffixDecoding and Sequoia also appear in Table 3. SuffixDecoding (Oliaro et al., 2025), the strongest training-free drafter reported in prior work, drafts from a suffix tree over the prompt and prior outputs—richer statistics than GOOSE’s bigram adjacency table. At full scale (6 models $\times$ 13 datasets), GOOSE leads τ in all 78 cells and wall-clock throughput in all 52 cells with full accounting (geometric mean +35–36%); per-cycle instrumentation attributes the gap mainly to depth allocation—GOOSE speculates 52–60 nodes per non-dry cycle at depths 5.7–22, against SuffixDecoding’s 3–8 at 2.9–7.3—with coverage (twice as many dry cycles, whose draft is empty or a single token) and cold start second, and merging its two trees into one does not close it (Appendix D.8). Conversely, swapping our PLD spine for that suffix tree raises τ in all 15 cells tested (+3–47%, Appendix D.6), so the topology is agnostic to which source feeds the spine. Sequoia (Chen et al., 2024) optimizes tree shape under a single acceptance vector: we reproduce its published τ within $\pm 6\%$ on its released Llama-2-7B stack, where its trained 68M drafter wins τ on all three of its datasets yet the two systems trade wall-clock (GOOSE +8% on C4, Sequoia +5% on CNN/DM, OpenWebText within noise). Its per-cycle drafter forward decides the outcome and costs more at modern scale: there Sequoia reaches τ of 6.2–9.2 yet never exceeds 1.60 $\times$ wall-clock, and on Qwen3-8B it drops below AR outright (Appendix D.9). EAGLE-3 (Li et al., 2025) is the strongest trained drafter. Its head is trained on chat-template data, so the chat-template protocol is its fair setting; there it wins 4 of 5 datasets (+53–79% tok/s, with GOOSE retaining ClassEval at +39%). Under our raw-prompt protocol the result inverts and GOOSE leads all five (+25–83% on the four non-truncated datasets; Appendix D.4). We do not claim to beat EAGLE-3 in its native setting; the two are cross-category (Section 6).

5.3 Anisotropic vs. Isotropic Topology and Ablation

Table 1 confirms the prediction of Section 3: the spine tree wins all 8 settings by 12–33% (we report Llama-3-8B and Qwen3-8B as architecturally distinct representatives; Vicuna variants show similar trends). Iso($k=5$) performs similarly to Iso($k=3$), showing that the isotropic regime itself—not k —is the bottleneck.

Ablation analysis. Table 2 isolates each design idea by removing one component at a time on Qwen3-8B. **Idea 1 (Anisotropic topology).** Removing spine branches (-2.9%) or bigram adjacency (-3.7%) each degrades τ , confirming that the advantage requires both branch placement and transition quality. **Idea 2 (Confidence-adaptive budget).** Consensus bypass is the largest contributor (-5.1% , up to -11.4% on ClassEval): when all n -gram lengths agree on a long continuation, GOOSE extends the spine directly without tree construction. Removing the spine (-3.4%) shows the PLD signal itself is essential; topology and adaptation are complementary. **Idea 3 (Cross-source spine continuation).** Disabling spine continuation—the greedy walk’s extension from a broken spine into a TR branch (Section 4.2)—while keeping the anisotropic shape costs -4.4% on average (-1.7% HumanEval to -6.7% MT-Bench), measured as a matched pair (both arms re-run on identical code and configuration). The tree-*shape* gain itself is measured separately and assumption-free in Table 1 ($+12$ – 33% at equal budget over the isotropic control). To complement the ablation, we note that moving from TR-only isotropic to full GOOSE on Qwen3-8B (MT-Bench) raises τ from 2.81 to 3.51 ($+25\%$); per-component contributions are the leave-one-out deltas of Table 2.

5.4 Analysis

Source decomposition and synergy. Figure 5a decomposes each model’s mean τ into the best standalone baseline $\max(\tau_{\text{PLD}}, \tau_{\text{TR}})$ and the synergy gain from the spine-tree topology. The synergy ratio ($\tau_{\text{GOOSE}} / \max(\tau_{\text{PLD}}, \tau_{\text{TR}})$) exceeds 1.0 on all 25 settings (average $1.24\times$; per-model breakdown in Appendix D.1). The source split varies by task; even on the weakest datasets, every A40 cell stays above $2.6\times \tau$ (Appendix D.2). A per-accept source attribution across four models confirms the sources are complementary: PLD’s share spans 33–85% by task, and where it is sparsest TR supplies the majority without τ collapsing (Appendix D.5). Tree construction adds negligible CPU overhead ($<1\%$ of wall-clock time). GOOSE has no neural drafter: the draft comes from table lookup, and the only model call per cycle is the verification forward. This makes the conversion from τ to wall-clock tight. On Llama-3-8B HumanEval, $\tau=4.11$ yields a measured $3.36\times$ speedup, a $\approx 82\%$ conversion; the shortfall is tree construction plus the slightly larger, weight-bound verification forward.

6 Discussion and Conclusion

We introduced the spine tree, an anisotropic draft-tree topology that exploits acceptance heterogeneity among training-free draft sources. Without any training, GOOSE achieves 1.9 – $4.3\times$ lossless speedup, outperforms isotropic baselines by 12–33% (Table 1), and matches or exceeds EAGLE-2 wall-clock speed on two of the four models with published heads.

Where training-free speculation is valuable. Trained-drafter and training-free methods are cross-category. A trained head buys raw τ where its training distribution matches; a training-free drafter buys zero-cost, day-one deployment and still wins outright at $\leq 8\text{B}$ (Llama-3-8B HumanEval: $3.36\times$ vs. EAGLE-2’s $2.46\times$). The trade-off flips at $\geq 13\text{B}$, where the slower base forward lets EAGLE-2’s higher τ convert ($3.24\times$ vs. GOOSE’s $2.38\times$ on Vicuna-33B; Table 3).

Limitations. Our main results use batch size 1, the latency-sensitive, memory-bandwidth-bound regime where speculation is strongest. A batch-size sweep (Appendix D.7) shows GOOSE leading at small batch and batched AR overtaking as batch size grows (crossover ≈ 4 – 16), the general speculation-vs-batching trade-off. The adjacency table stays small ($<7\text{MB}$), and our analysis assumes greedy decoding; production-grade batched serving and sampling-based verification (Leviathan et al., 2023) remain future work.

Acknowledgments

This work was supported by the Nakajima Foundation. The authors sincerely thank the chairs and reviewers of COLM 2026 for their thoughtful and insightful feedback; their reviews and the discussion period greatly improved this work.

References

- Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models, 2021. URL <https://arxiv.org/abs/2108.07732>.
- 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. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 5209–5235. PMLR, 21–27 Jul 2024. URL <https://proceedings.mlr.press/v235/cai24b.html>.
- Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebggen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021. URL <https://arxiv.org/abs/2107.03374>.
- Zhuoming Chen, Avner May, Ruslan Svirschevski, Yuhsun Huang, Max Ryabinin, Zhihao Jia, and Beidi Chen. Sequoia: Scalable and robust speculative decoding. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 129531–129563. Curran Associates, Inc., 2024. doi: 10.52202/079017-4116. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/ea1f5f0878d43ff4fb8bf64ef4a2326c-Paper-Conference.pdf.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL <https://arxiv.org/abs/2110.14168>.
- Xueying Du, Mingwei Liu, Kaixin Wang, Hanlin Wang, Junwei Liu, Yixuan Chen, Jiayi Feng, Chaofeng Sha, Xin Peng, and Yiling Lou. ClassEval: A manually-crafted benchmark for evaluating LLMs on class-level code generation, 2023. URL <https://arxiv.org/abs/2308.01861>.
- Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, Ahmed Aly, Beidi Chen, and Carole-Jean Wu. LayerSkip: Enabling early exit inference and self-speculative decoding. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*

- (Volume 1: Long Papers), pp. 12622–12642, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.681. URL <https://aclanthology.org/2024.acl-long.681/>.
- Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. Break the sequential dependency of LLM inference using lookahead decoding. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 14060–14079. PMLR, 21–27 Jul 2024. URL <https://proceedings.mlr.press/v235/fu24a.html>.
- Zhenyu He, Zexuan Zhong, Tianle Cai, Jason Lee, and Di He. REST: Retrieval-based speculative decoding. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 1582–1595, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.88. URL <https://aclanthology.org/2024.naacl-long.88/>.
- Sehoon Kim, Karttikeya Mangalam, Suhong Moon, Jitendra Malik, Michael Mahoney, Amir Gholami, and Kurt Keutzer. Speculative decoding with big little decoder. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 39236–39256. Curran Associates, Inc., 2023. doi: 10.52202/075280-1705. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/7b97adeafa1c51cf65263459ca9d0d7c-Paper-Conference.pdf.
- Nguyen-Khang Le, Truong Dinh Do, and Le-Minh Nguyen. SPECTRA: Faster large language model inference with optimized internal and external speculation. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14015–14034, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.685. URL <https://aclanthology.org/2025.acl-long.685/>.
- Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 19274–19286. PMLR, 23–29 Jul 2023. URL <https://proceedings.mlr.press/v202/leviathan23a.html>.
- Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE: Speculative sampling requires rethinking feature uncertainty. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 28935–28948. PMLR, 21–27 Jul 2024a. URL <https://proceedings.mlr.press/v235/li24bt.html>.
- Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE-2: Faster inference of language models with dynamic draft trees. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 7421–7432, Miami, Florida, USA, November 2024b. Association for Computational Linguistics. 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 *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL <https://openreview.net/forum?id=4exx1hUffq>.
- Xianzhen Luo, Yixuan Wang, Qingfu Zhu, Zhiming Zhang, Xuanyu Zhang, Qing Yang, and Dongliang Xu. Turning trash into treasure: Accelerating inference of large language models with token recycling. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and

- Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6816–6831, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.338. URL <https://aclanthology.org/2025.acl-long.338/>.
- Meta AI. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 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, Volume 3*, ASPLOS '24, pp. 932–949, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400703867. doi: 10.1145/3620666.3651335. URL <https://doi.org/10.1145/3620666.3651335>.
- Gabriele Oliaro, Zhihao Jia, Daniel F Campos, and Aurick Qiao. SuffixDecoding: Extreme speculative decoding for emerging AI applications. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL <https://openreview.net/forum?id=uwL0vbeEVn>.
- Qwen Team. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.
- Apoorv Saxena. Prompt lookup decoding. <https://github.com/apoorvumang/prompt-lookup-decoding>, 2023.
- Shwetha Somasundaram, Anirudh Phukan, and Apoorv Saxena. PLD+: Accelerating LLM inference by leveraging language model artifacts. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 6090–6104, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-195-7. doi: 10.18653/v1/2025.findings-naacl.338. URL <https://aclanthology.org/2025.findings-naacl.338/>.
- Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. Blockwise parallel decoding for deep autoregressive models. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/c4127b9194fe8562c64dc0f5bf2c93bc-Paper.pdf.
- 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 Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 7655–7671, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.456. URL <https://aclanthology.org/2024.findings-acl.456/>.
- 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, 2023. URL <https://arxiv.org/abs/2304.04487>.
- Jun Zhang, Jue Wang, Huan Li, Lidan Shou, Ke Chen, Gang Chen, and Sharad Mehrotra. Draft & verify: Lossless large language model acceleration via self-speculative decoding. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11263–11282, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.607. URL <https://aclanthology.org/2024.acl-long.607/>.
- Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph Gonzalez, and Ion Stoica. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural*

Information Processing Systems, volume 36, pp. 46595–46623. Curran Associates, Inc., 2023. doi: 10.52202/075280-2020. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/91f18a1287b398d378ef22505bf41832-Paper-Datasets_and_Benchmarks.pdf.

Yongchao Zhou, Kaifeng Lyu, Ankit Singh Rawat, Aditya Krishna Menon, Afshin Ros-tamizadeh, Sanjiv Kumar, Jean-François Kagy, and Rishabh Agarwal. DistillSpec: Improving speculative decoding via knowledge distillation. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=rsY6J3ZaTF>.

A Full Main Results

Table 3 reports compression ratio (τ) and wall-clock speedup (Spd) for all five models across five benchmarks. GOOSE achieves the highest average speedup on three of five models; EAGLE-2 leads on Vicuna-13B and Vicuna-33B thanks to its trained draft head (official EAGLE heads throughout) and has no published head for Qwen3-8B. For the Vicuna-33B EAGLE-2 GSM8K cell, speedup is computed against the AR baseline of its own run environment; τ is environment-independent. The per-benchmark τ values are the ground truth for the synergy analysis in Appendix D.1.

Table 3: Full main results. Speedup (Spd): wall-clock tok/s ratio vs. AR measured in the same run. τ : compression ratio (tokens per target-model forward; drafter forwards are not counted, which favors the trained rows). **Bold**: best per column. All methods produce identical output (lossless). Shaded rows denote methods requiring per-model training. Sequoia appears on the two targets with released modern drafters, at $N=60$ to match our budget B ; its τ follows Sequoia’s official convention (prefill call excluded), at most 1.7% above the table’s convention. EAGLE-2 is absent on Qwen3-8B, which has no released head. [†]Vicuna-33B evaluated on NVIDIA A100 (2×40GB); all other models on A40. The Vicuna-33B GSM8K EAGLE-2 cell comes from a separate run of that environment, so its Spd is taken against that run’s own AR; τ is unaffected.

Model	Method	ClassEval		GSM8K		HumanEval		MBPP		MT-Bench		Avg.
		Spd	τ	Spd	τ	Spd	τ	Spd	τ	Spd	τ	Spd
Vicuna-7B	AR	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	PLD	2.34	2.50	1.43	1.46	1.74	1.78	1.59	1.62	1.47	1.48	1.71
	TR	2.58	3.29	2.46	2.96	2.53	3.06	2.41	2.86	2.19	2.60	2.43
	Lookahead	2.27	2.41	1.63	1.67	1.72	1.77	1.75	1.78	1.58	1.60	1.79
	REST	1.68	1.73	1.17	1.19	1.16	1.17	1.05	1.05	1.08	1.08	1.23
	SuffixDecoding	2.90	3.34	1.92	1.98	2.33	2.44	2.11	2.16	1.80	1.85	2.21
	EAGLE-2	3.14	5.50	2.73	4.51	2.99	5.02	2.92	4.83	2.53	4.21	2.86
	GOOSE (Ours)	3.58	4.64	2.67	3.23	2.97	3.63	2.84	3.40	2.42	2.88	2.90
lama-3-8B	AR	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	PLD	2.30	2.49	4.04	4.18	1.92	2.01	1.67	1.72	1.67	1.72	2.32
	TR	2.87	3.64	3.68	4.52	2.79	3.40	2.60	3.11	2.29	2.82	2.85
	Lookahead	2.35	2.57	2.59	2.80	1.98	2.08	1.79	1.86	1.73	1.80	2.09
	REST	1.33	1.35	1.06	1.07	1.13	1.14	1.01	1.01	1.05	1.05	1.12
	SuffixDecoding	3.29	3.46	5.00	5.10	2.64	2.72	2.21	2.26	2.15	2.18	3.06
	Sequoia	1.52	8.63	1.56	8.89	1.53	8.56	1.48	8.42	1.36	7.57	1.49
	EAGLE-2	2.58	5.29	2.75	5.52	2.46	4.95	2.41	4.83	1.92	3.85	2.42
	GOOSE (Ours)	3.82	4.89	7.46	9.25	3.36	4.11	3.00	3.61	2.84	3.50	4.10
Qwen3-8B	AR	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	PLD	2.50	2.61	1.65	1.71	1.31	1.36	1.45	1.50	1.67	1.73	1.72
	TR	3.77	4.50	2.60	3.03	2.18	2.55	2.43	2.81	2.44	2.86	2.68
	Lookahead	2.42	2.42	1.93	1.88	1.52	1.47	1.75	1.68	1.78	1.74	1.88
	REST	1.22	1.14	1.21	1.16	1.19	1.12	1.11	1.04	1.11	1.03	1.17
	SuffixDecoding	2.15	2.21	2.22	2.38	1.71	1.68	2.00	1.94	2.06	2.18	2.03
	Sequoia	0.82	7.32	0.85	7.58	0.76	6.81	0.82	7.34	0.70	6.22	0.79
	GOOSE (Ours)	4.28	5.19	3.23	3.87	2.34	2.78	2.70	3.20	2.92	3.51	3.09
Vicuna-13B	AR	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	PLD	2.34	2.47	1.33	1.36	1.62	1.66	1.49	1.51	1.35	1.36	1.63
	TR	2.71	3.34	2.56	3.02	2.49	2.94	2.40	2.78	2.12	2.46	2.46
	Lookahead	2.14	2.26	1.58	1.62	1.73	1.78	1.64	1.67	1.47	1.49	1.71
	REST	1.61	1.66	1.18	1.21	1.19	1.20	1.04	1.04	1.11	1.11	1.23
	SuffixDecoding	2.95	3.36	1.82	1.90	2.14	2.26	1.91	1.97	1.62	1.68	2.09
	EAGLE-2	3.62	6.03	3.18	4.98	3.34	5.24	3.41	5.30	2.78	4.33	3.27
	GOOSE (Ours)	3.68	4.59	2.66	3.14	2.84	3.37	2.66	3.11	2.30	2.67	2.83
/icuna-33B [†]	AR	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	PLD	1.85	1.99	1.29	1.33	1.36	1.40	1.27	1.30	1.14	1.15	1.38
	TR	1.97	2.62	2.14	2.63	1.97	2.37	1.95	2.29	1.72	2.01	1.95
	Lookahead	1.86	2.02	1.49	1.55	1.56	1.56	1.51	1.49	1.26	1.23	1.54
	REST	1.48	1.54	1.14	1.18	1.18	1.16	1.08	1.05	1.09	1.05	1.19
	SuffixDecoding	2.26	2.59	1.69	1.81	1.77	1.82	1.70	1.72	1.36	1.37	1.76
	EAGLE-2	3.48	5.15	3.42	4.73	3.36	4.65	3.32	4.55	2.60	3.58	3.24
	GOOSE (Ours)	2.80	3.69	2.53	3.10	2.37	2.83	2.35	2.74	1.87	2.15	2.38

B Algorithm Details

B.1 Spine Tree Construction

Algorithm 1 BUILDSPINETREE

Require: Anchor token t_a ; PLD draft chain $\mathbf{d} = (d_1, \dots, d_m)$

Require: Adjacency table $\mathcal{A}$; node budget B ; max depth D

Require: Spine ratio $r \in (0, 1)$; spine-branch ratio $\rho \in (0, 1)$

Ensure: Spine tree $\mathcal{T}$ with $\leq B$ nodes and tree attention mask

```

1:  $b_s \leftarrow \min(m, \lfloor Br \rfloor)$                                 ▷ PLD spine node count
2:  $b_r \leftarrow \lfloor (B-1-b_s)(1-\rho) \rfloor$                         ▷ Root-level branch count
3:  $b_p \leftarrow B-1-b_s-b_r$                                     ▷ Spine-level branch count

4:  $d_0 \leftarrow t_a$                                             ▷ Anchor as spine root
5: for  $i = 1$  to  $b_s$  do                                       ▷ Step 1: lay PLD spine
6:   Add  $d_i$  as child of  $d_{i-1}$ 
7: end for

8: Attach top- $b_r$  successors from  $\mathcal{A}(t_a)$  to root           ▷ Step 2: root branches

9: for  $i = 1$  to  $b_s$  do                                       ▷ Step 3: spine branches (harmonic decay)
10:   $a_i \leftarrow \lfloor b_p \cdot \frac{1/i}{\sum_{j=1}^{b_s} 1/j} \rfloor$            ▷ Allocation for spine node  $d_i$ 
11:  Attach top- $a_i$  successors from  $\mathcal{A}(d_i)$  to  $d_i$ 
12: end for

13: BFS-extend all branch leaves through  $\mathcal{A}$  up to depth  $D$     ▷ Step 4
14: return  $\mathcal{T}$ 

```

Duplicate entries (e.g., spine continuation d_{i+1} already present in $\mathcal{A}(d_i)$) are excluded during attachment to avoid redundant nodes. The BFS extension in Step 4 is a standard breadth-first expansion: starting from each branch leaf, it greedily appends top-scoring successors from $\mathcal{A}$ up to depth D within the remaining budget.

B.2 Decoding Loop

GOOSE routes each cycle by source availability and confidence (Section 4.3): (1) high-confidence or long context match triggers linear bypass; (2) any draft source available triggers the spine tree with unified greedy walk verification; (3) no source available triggers AR fallback. When only transitions (no context match) are available, $\mathbf{d} = \emptyset$ and the tree degenerates to a TR-only tree; when only a context match is available, the tree reduces to a linear chain. Both are special cases that preserve the expected-yield guarantee of Proposition 4.

CONTEXTMATCH (called at line 5 of Algorithm 2) is a standard n -gram lookup: given the generated sequence and a set of query lengths $\{3, 4, 5\}$, it finds the longest matching prefix in the context, copies the continuation as the draft chain $\mathbf{d}$, and returns a consensus flag c (true iff ≥ 2 matches agree on the first continuation token).

B.3 Unified Greedy Walk Verification

At each tree node, the walk checks PLD (spine) children before TR (branch) children, giving priority to the high-acceptance source. Once the walk transitions from spine to branch, it cannot return, producing the “spine continuation” path described in Section 4.2. The walk naturally yields the longest accepted path and covers all three outcome categories: pure PLD (spine only), spine continuation (PLD prefix then TR suffix), or pure TR (root branch only).

Algorithm 2 GOOSE Decoding Loop**Require:** Target model $\mathcal{M}$; prompt $\mathbf{x}$; max tokens N **Require:** Budget B ; spine-branch ratio ρ ; max depth D ; bypass threshold ℓ_{byp} ; adjacency width K **Ensure:** Generated tokens $\mathbf{y}$, identical to autoregressive output

```

1: Build context index from  $\mathbf{x}$ ; init adjacency table  $\mathcal{A}$  with top- $K$  width
2:  $\ell \leftarrow \mathcal{M}(\mathbf{x})$ ; update  $\mathcal{A}$  with top- $K$  entries per position from  $\ell$ 
3:  $t_{\text{anc}} \leftarrow \arg \max \ell[-1]$ ;  $\hat{p}_s \leftarrow 0.3$  ▷ Running estimate of spine acceptance  $p_s$ 
4: while  $|\mathbf{y}| < N$  and not EOS do
5:    $(\mathbf{d}, c) \leftarrow \text{CONTEXTMATCH}(\mathbf{y})$  ▷ Draft chain  $\mathbf{d}$ ; consensus flag  $c$ 
6:   if  $\mathbf{d} \neq \emptyset$  and  $(|\mathbf{d}| \geq \ell_{\text{byp}} \text{ or } c)$  then
7:      $(\mathbf{y}_{\text{acc}}, b) \leftarrow \text{linear-verify } \mathbf{d} \text{ with } \mathcal{M}$  ▷ Bypass: high-confidence PLD
8:   else if  $\mathbf{d} \neq \emptyset$  or  $\mathcal{A}$  has successors for  $t_{\text{anc}}$  then
9:      $r \leftarrow \text{SPINERATIO TIER}(\hat{p}_s)$  ▷ Adaptive spine ratio (Section 4.3)
10:     $\mathcal{T} \leftarrow \text{BUILD SPINE TREE}(t_{\text{anc}}, \mathbf{d}, \mathcal{A}, r, B, \rho, D)$ 
11:     $(\mathbf{y}_{\text{acc}}, b) \leftarrow \text{UNIFIED GREEDY WALK}(\mathcal{M}, \mathcal{T})$ 
12:   else
13:      $b \leftarrow \arg \max \mathcal{M}(t_{\text{anc}})$ ;  $\mathbf{y}_{\text{acc}} \leftarrow \emptyset$  ▷ AR fallback
14:   end if
15:   Update  $\mathcal{A}$  from all logits of this cycle ▷ Including rejected branches
16:   Update  $\hat{p}_s$  via EMA;  $t_{\text{anc}} \leftarrow b$ ; append  $\mathbf{y}_{\text{acc}} \| b$  to  $\mathbf{y}$ 
17: end while

```

Algorithm 3 UNIFIEDGREEDYWALK: Verification with Source Priority**Require:** Model $\mathcal{M}$; spine tree $\mathcal{T}$ (nodes tagged PLD or TR)**Ensure:** Accepted token list; bonus token b

```

1:  $\ell \leftarrow \mathcal{M}(\mathcal{T})$  with tree attention mask ▷ Single forward pass
2:  $\hat{x}_i \leftarrow \arg \max \ell_i$  for each node  $i$  ▷ Greedy prediction per node

3:  $v \leftarrow \text{root}$ ; accepted  $\leftarrow []$ 
4: while  $v$  has children in  $\mathcal{T}$  do
5:   if  $\exists$  PLD-child  $v'$  of  $v$ :  $\mathcal{T}[v'] = \hat{x}_v$  then ▷ Priority 1: spine
6:     accepted.append( $v'$ );  $v \leftarrow v'$ ; continue
7:   end if
8:   if  $\exists$  TR-child  $v'$  of  $v$ :  $\mathcal{T}[v'] = \hat{x}_v$  then ▷ Priority 2: branch
9:     accepted.append( $v'$ );  $v \leftarrow v'$ ; continue
10:  end if
11:  break ▷ No child matches
12: end while

13:  $b \leftarrow \hat{x}_v$  ▷ Bonus token from last accepted position
14: return accepted,  $b$ 

```

C Implementation and Setup

C.1 Implementation Details

This section provides the full engineering details summarized in Section 4.3.

Dense GPU bigram adjacency table. The adjacency table $\mathcal{A}$ (Section 4.1) is implemented as a two-tier structure (dense GPU unigram tensors plus a bigram hash table). **Tier 1 (Unigram):** Two dense tensors of shape $|V| \times K$ on GPU store, respectively, the token IDs and logit-derived scores of the top- K successors for each vocabulary token; lookup is $O(1)$ with no CPU transfer. **Tier 2 (Bigram):** A hash table mapping $(t_{i-1}, t_i) \mapsto$ top- K successors, exploiting the fact that two-token context substantially sharpens predictions (e.g., successors of f after def are much more predictable than f alone).

Confidence-aware branch width. Branch width scales with the transition score in $\mathcal{A}$: each successor’s allocated children (from the $1/i$ harmonic schedule in Step 3 of Algorithm 1) are further modulated by its score relative to its siblings, and successors whose score falls below 0.01 are pruned entirely. This focuses the node budget on high-confidence successors while keeping the total tree size within B .

Aggressive logit harvesting. Every forward pass’s logits, including those at *rejected branches* and at prefill, are harvested into the adjacency table via a batched topk ($< 200 \mu\text{s}$ overhead). This populates $\mathcal{A}$ far faster than the verified-positions-only strategy used in prior work.

C.2 Datasets and Model Details

Table 4: Benchmark details.

Benchmark	Domain	Samples	Max tokens
HumanEval	Code	164	512
MBPP	Code	500	512
ClassEval	Code	100	512
GSM8K	Math	1,319	1,024
MT-Bench	Dialogue	80	1,024

C.3 Hyperparameters and Experimental Setup

Table 5 lists all hyperparameters and Table 6 the hardware and software configuration.

D Extended Experimental Analysis

D.1 Synergy Decomposition and Hyperparameter Sensitivity

The synergy ratio $\tau_{\text{GOOSE}} / \max(\tau_{\text{PLD}}, \tau_{\text{TR}})$ exceeds 1.0 on every model (range 1.16–1.45 $\times$, mean 1.24 $\times$), confirming that the gain is a consistent structural phenomenon across all 25 settings. The single highest cell is Llama-3-8B/GSM8K at 2.05, inflated by repetition (see Appendix D.2); excluding that cell, per-model synergy ranges from 1.16 to 1.24.

D.2 Cross-Task Robustness

Strong-affinity datasets (ClassEval, GSM8K) share repetitive structure, such as class templates or chain-of-thought boilerplate, which yields high PLD-path acceptance. Weak-affinity datasets (MBPP, MT-Bench) contain shorter, less repetitive outputs that shift budget toward branches (most clearly on MBPP), yet still achieve $\tau > 2.6\times$. The cross-model CV is below 14% for four of five datasets: GSM8K is the exception (CV=52.2%) because

Table 5: All hyperparameters (fixed across datasets and models).

Parameter	Value
Context match n-gram lengths	3, 4, 5
Max spine continuation	20
Transition top- K	10
Tree node budget B	60
Max branch (transition) depth D	6
Min score threshold	0.01
Spine-branch ratio ρ	0.5
EMA smoothing coefficient	0.3
Spine ratio tiers (r)	$\hat{p}_s < 0.2 \rightarrow 0.15, < 0.4 \rightarrow 0.30, \geq 0.4 \rightarrow 0.50$
Linear bypass threshold	8 tokens

Table 6: Hardware and software configuration.

Item	Specification
GPU (7B–13B)	NVIDIA A40 (48 GB GDDR6)
GPU (33B)	2 × A100-PCIE-40GB (80 GB total)
CUDA Toolkit	12.8
PyTorch	2.9.1+cu128
Transformers	4.57 (Hugging Face)
Precision	FP16 (float16)

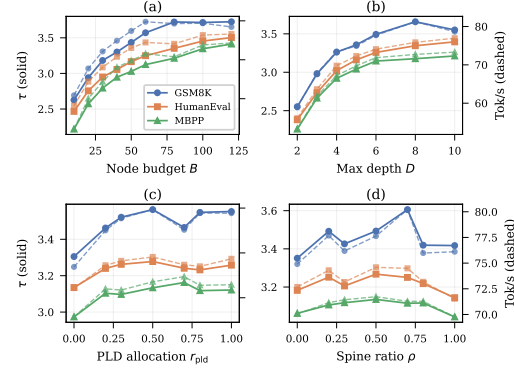
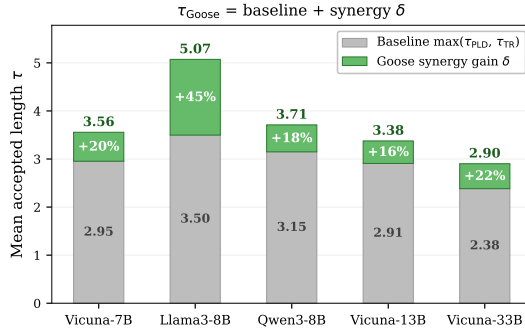


Figure 5: **(a)** GOOSE’s mean τ decomposed into the best standalone baseline $\max(\tau_{PLD}, \tau_{TR})$ (gray) and the synergy gain from the spine-tree topology (green); the gain ranges from +16% (Vicuna-13B) to +45% (Llama-3-8B). **(b)** Hyperparameter sensitivity (Qwen3-8B): each panel sweeps one parameter; τ plateaus at $B \geq 60$ and $D \geq 6$, and varies by fewer than 0.6 τ units across r and ρ . Panels (c)/(d) sweep the spine ratio r and the spine-branch ratio ρ of Section 4.1.

Llama-3-8B exhibits degenerate repetition loops on arithmetic tasks, inflating its τ to 9.25, an anomaly that disappears when that single model is excluded (CV drops to 9.5%). Overall, the strong/weak gap is moderate (mean 4.85 vs. 3.24, ratio 1.50 $\times$; the strong-affinity mean is lifted by the Llama-3-8B GSM8K outlier noted above), showing that GOOSE degrades gracefully rather than collapsing on unfavorable inputs; per-prompt variance within each cell is analyzed in Appendix D.3.

D.3 Prompt-Level Variance

Across all 25 model $\times$ dataset settings, median τ tracks the mean closely (Pearson $r > 0.98$) and the interquartile range (IQR) stays within $0.5\text{--}2.3\times$ for 23 of 25 cases. The two exceptions (Llama-3-8B/GSM8K and Qwen3-8B/ClassEval) are traced to degenerate repetition loops that inflate τ ; excluding the top 5% of prompts reduces their IQR to levels consistent with other settings. Overall, the tight IQR confirms that GOOSE’s gains are broadly distributed across prompts rather than driven by outliers.

D.4 EAGLE-2/3: Chat-Template and Raw-Prompt Protocols

On raw prompts (Table 8; chat protocol in Table 7) EAGLE’s chat-trained head loses τ , and its per-cycle draft forward erases the remaining lead: GOOSE wins wall-clock on all five datasets, even where EAGLE’s τ is comparable or higher (MT-Bench, MBPP). GSM8K is fixed-budget (96% truncated), so its τ and tok/s are inflated for the repetition-driven GOOSE and should be read qualitatively.

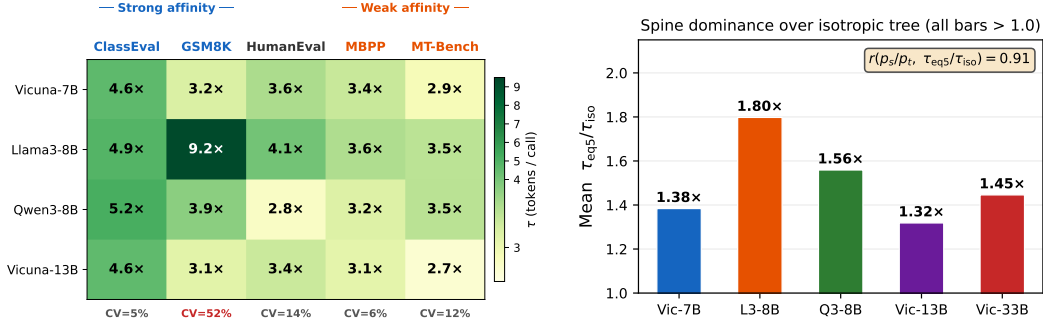


Figure 6: **(a)** τ across four A40 models and five datasets grouped by GOOSE’s affinity. CV below each column shows cross-model coefficient of variation. **(b)** Per-model mean τ_{eq5}/τ_{iso} across 24 settings; all bars exceed 1.0, confirming spine dominance (Pearson $r=0.91$).

Table 7: GOOSE vs. EAGLE-3 under EAGLE’s native chat-template protocol (Llama-3.1-8B, A40), the primary protocol for this baseline (Section 5.2). τ / tok/s.

Dataset	GOOSE		EAGLE-3	
	τ	tok/s	τ	tok/s
HumanEval	2.59	66.6	5.79	109.0
ClassEval	5.35	131.3	5.34	94.5
MT-Bench	2.22	57.0	5.36	102.0
MBPP	2.36	61.4	5.69	108.7
GSM8K	2.76	70.3	5.69	107.8

D.5 Per-Accept Source Attribution

A natural concern is that when input n -gram matches are rare the spine degenerates. We instrumented GOOSE to label every accepted token by its source (PLD spine vs. TR branch) on four models spanning 1.1B–8B. Table 9 reports PLD’s share of accepted tokens (TR is the remainder).

PLD’s share is generally highest on template-heavy ClassEval (61–71%) and on Llama-3-8B’s repetition-prone GSM8K (84%), and generally lowest on HumanEval/MBPP, where it falls to about a third. Even at its sparsest, TR supplies the remaining majority of accepts and τ does not collapse (e.g., $\tau=2.78$ on Qwen3-8B HumanEval, still +12% over the equal-budget isotropic tree, Table 1): TR needs no prompt match (it is built from the model’s own logits), so the tree stays full when the spine is empty. The spine therefore does not degenerate when n -gram matches are rare; the budget shifts to branches, which is precisely the adaptive behavior of Section 4.3.

D.6 Swapping the Spine Source: PLD $\rightarrow$ SuffixDecoding

PLD and TR are one instance of GOOSE’s two acceptance classes, not a limit of the method: any source emitting a high-acceptance chain can serve as the spine, and any source of plausible alternatives as branches. We show this by replacing the PLD spine with *SuffixDecoding’s suffix tree* (chain mode, official ArcticInference cache, not a re-implementation) while keeping the TR branches and the topology unchanged. The spine source is the only difference between the two arms; consensus bypass is off on both sides. τ rises in all 15 model $\times$ dataset cells (+3% to +47%, Table 10):

The suffix tree is a stronger chain source than PLD (empirical-frequency scoring vs. longest-exact match), yet the anisotropic topology and verification are unchanged: the gain composes with whichever spine source is strongest. On Llama-3.1-8B the τ gain carries to wall-clock (+7–42% tok/s). The swap also preserves losslessness: verification accepts

Table 8: Raw-prompt protocol (our paper’s protocol, applied to every baseline): Llama-3.1-8B, A40, full n . τ / tok/s.

Dataset	GOOSE		EAGLE-2		EAGLE-3	
	τ	tok/s	τ	tok/s	τ	tok/s
HumanEval	4.50	114.8	4.14	62.2	3.29	62.6
ClassEval	5.54	133.4	4.11	58.9	4.18	74.8
MT-Bench	3.63	90.7	3.51	52.2	3.87	72.6
MBPP	3.51	91.0	4.19	63.6	3.55	68.5
GSM8K	8.90	221.7	4.28	63.6	4.62	86.2

Table 9: PLD share of accepted tokens (%); TR supplies the remainder. Llama-3-8B and Qwen3-8B are paper models (full n); the two ~ 1 B models are robustness checks ($n=30$).

Model	ClassEval	MT-Bench	HumanEval	GSM8K	MBPP
Qwen3-8B	70.8	53.3	33.0	48.3	38.3
Llama-3-8B	62.1	53.5	53.4	83.8	45.2
Llama-3.2-1B	61.4	35.1	60.7	37.9	36.7
TinyLlama-1.1B	66.1	84.5	33.2	59.8	33.4

only the target model’s own argmax token regardless of draft source, and we confirmed token-identical output to greedy AR under fp32 for the swapped variant (Llama-3.2-1B HumanEval: 10/10 (method, sample) pairs across {GOOSE, GOOSE-Suffix-spine} $\times$ 5 prompts).

D.7 Batch-Size Sweep

We sweep batch size with cross-request batched verification for GOOSE against padded request-batched AR, on Llama-3-8B with the *same* HuggingFace stack on both sides; a self-test verifies the batched decoder is batch-lossless (batched and $bs=1$ decoding produce token-identical output, checked in fp32). The GOOSE prototype here is the bypass-off core decoder (budget 32), so its $bs=1$ speedup (~ 1.7 – $2.0\times$) understates the full system ($3.36\times$ on HumanEval); the object of interest is the *shape* of the curves in Table 11.

Speculative decoding and request batching draw on the same idle compute. At $bs=1$, decoding is bandwidth-bound and GOOSE’s tree verification accepts several tokens per weight load almost for free. Batching amortizes those same weight loads across requests, so AR scales roughly linearly; GOOSE, whose forward already processes batch $\times$ tree-size tokens, saturates compute earlier and plateaus (on the A40, aggregate throughput rises $\sim 2.6\times$ from $bs=1$ to the $bs=16$ plateau; on the 3090 it rises $\sim 1.7\times$ and saturates by $bs=4$ – 8). The crossover is dataset- and hardware-dependent but always present and always ordered the same way: GSM8K (longer generations, higher τ) crosses later than HumanEval on both GPUs, and the A40 crosses later than the 3090. GPU-utilization profiling during these runs is consistent with compute saturation rather than host overhead. The prototype retains finished requests as padding rather than evicting them, wasting some large-batch compute; a production-grade batched implementation is left as future work.

D.8 SuffixDecoding Mechanism Decomposition and Merged-Tree Variant

Table 12 reports the head-to-head of Section 5.2 on SuffixDecoding’s own evaluation model. Cell accounting for Section 5.2: the grid is 6 models (our five plus Llama-3.1-8B) $\times$ 13 datasets (our five plus SuffixDecoding’s eight) = 78 cells, all with τ ; “full accounting” means the 52 cells (four main-grid A40 models $\times$ 13 datasets; the models are Vicuna-7B, Llama-3-8B, Qwen3-8B, and Vicuna-13B) where wall-clock was logged under the paper harness, and the 32 own-dataset cells (those four models $\times$ its eight datasets) are a subset of the 52.

Table 10: τ with the PLD spine swapped for SuffixDecoding’s suffix tree (TR branches and topology unchanged, bypass off on both sides).

τ : PLD→Suffix spine	HumanEval	MBPP	GSM8K	MT-Bench	ClassEval
Llama-3.1-8B (full n)	4.21→ 4.80	3.30→ 3.63	6.28→ 9.26	3.22→ 3.91	5.00→ 5.89
Llama-3.2-1B ($n=30$)	3.75→ 4.33	3.04→ 3.27	3.38→ 3.60	2.55→ 2.63	4.29→ 5.00
TinyLlama-1.1B ($n=30$)	2.84→ 3.08	2.88→ 3.09	4.56→ 5.41	6.19→ 8.92	5.03→ 5.74

Table 11: Aggregate tok/s (total tokens / wall-clock) vs. batch size, Llama-3-8B HumanEval ($n=164$) and GSM8K ($n=256$), greedy, fp16.

GPU / task	Method	bs=1	bs=4	bs=8	bs=16	bs=32
A40, HumanEval	GOOSE	59.0	129.8	141.5	149.4	143.0
	AR	31.0	96.5	168.2	265.0	366.1
A40, GSM8K	GOOSE	62.4	138.5	155.7	168.7	156.4
	AR	31.1	79.0	142.1	244.9	382.2
RTX 3090, HumanEval	GOOSE	73.6	132.0	131.1	124.1	—
	AR	42.4	136.1	228.3	339.0	—
RTX 3090, GSM8K	GOOSE	78.6	138.4	139.1	133.2	—
	AR	42.2	111.7	197.9	327.3	—

On SuffixDecoding’s own evaluation datasets (Llama-3.1-8B, τ): SWE-Bench-Lite 4.04 vs. 2.78 (+45%), Magicoder 2.78 vs. 1.67 (+67%), WildChat 3.01 vs. 1.90 (+58%), and +20–64% across the five Spec-Bench tasks.

Table 13 backs the mechanism claims of Section 5.2 with the full 4-model $\times$ 5-dataset grid (τ ; $n=100$ ClassEval, 200 GSM8K/MBPP, 164 HumanEval, 80 MT-Bench; max 512 new tokens; A40). Arms: GOOSE (cold, per-request, budget 60); SuffixDecoding official two-tree cache, cold and warm (cache seeded with the 8 preceding requests of the same dataset, sequential order); and our merged single-tree variant, warm. The merged tree is *our* construction (the published method keeps two trees and selects per cycle); it is built by per-sequence insertion and its semantics were machine-checked against the official code (35/35 assertions, including 300-trial fuzz bit-exactness). In the main comparison (Section 5.2), all baselines run cold and per-request: SuffixDecoding’s larger published agentic speedups use a warmed cache, a serving setting outside this benchmark rather than a capability we disabled, and its Hybrid modes are excluded as not training-free (they add a trained EAGLE-3 drafter). Suffix arms use a max-spec factor of 2 (SuffixDecoding’s per-match cap on draft length, $2 \times$ the match length), which outperformed its default factor of 1 in all 20 cells, so the baseline is tuned generously. Because of fp16 argmax near-ties, arms can diverge on a minority of samples (per-arm total token counts differ by up to 2.7%). We therefore compare τ (tokens per model call), computed on each arm’s own trace.

Three readings. (i) GOOSE cold beats every suffix arm, including warm, in all 20 cells ($1.35\text{--}1.60 \times$ the best suffix τ), despite the protocol asymmetry favoring SuffixDecoding. (ii) The merged tree trails the official two-tree method in 18 of 20 cells (typically -0.5 to -4.5% ; the two exceptions are $\leq 0.6\%$ wins): pooling prompt-local and global statistics dilutes locally-certain continuations below the drafting threshold, the dilution mechanism we anticipated when constructing the variant. (iii) The per-cycle instrumentation (same runs) locates GOOSE’s advantage: dry-cycle fraction (cycles whose draft is empty or a single token) 0.11–0.22 vs. 0.22–0.45; nodes drafted per usable (non-dry) cycle 52–60 at depth 5.7–22 vs. 3.4–7.9 at depth 2.9–7.3; and first-quartile-of-generation accepted tokens per cycle 0.92–2.90 vs. 0.25–1.65.

D.9 Sequoia: Official Head-to-Head and Modern-Stack Full Grid

Table 14 gives the head-to-head against Sequoia’s official released stack. Both systems run under their own released protocols: Sequoia prefills 128 tokens and decodes to a fixed 256-

Table 12: GOOSE vs. SuffixDecoding (tree mode, the stronger of its two training-free modes) on Llama-3.1-8B (A40, greedy, full n). τ / tok/s. *GSM8K fixed-budget.

	HumanEval	MBPP	MT-Bench	ClassEval	GSM8K*
GOOSE	4.50/114.8	3.51/91.0	3.63/90.7	5.54/133.4	8.90/221.7
SuffixDecoding (tree)	2.98/86.6	2.14/63.5	2.33/68.0	3.97/108.9	5.59/162.0

Table 13: τ on the full grid: GOOSE (cold) vs. SuffixDecoding official (cold/warm) and the merged-tree variant (warm). **Bold** = column best per row.

Model	Dataset	GOOSE	Suf. (cold)	Suf. (warm)	Merged (warm)
Llama-3-8B	ClassEval	4.92	3.53	3.66	3.55
	GSM8K	5.90	3.38	3.79	3.68
	HumanEval	4.13	2.76	2.94	2.88
	MBPP	3.58	2.25	2.54	2.54
	MT-Bench	2.71	1.63	1.73	1.71
Llama-3.1-8B	ClassEval	5.50	3.97	4.06	4.05
	GSM8K	4.99	3.13	3.36	3.24
	HumanEval	4.53	2.98	3.20	3.15
	MBPP	3.44	2.12	2.41	2.39
	MT-Bench	2.69	1.67	1.73	1.71
Llama-3.2-1B	ClassEval	4.38	3.03	3.13	3.09
	GSM8K	2.97	1.77	2.06	2.03
	HumanEval	4.05	2.68	2.99	2.95
	MBPP	3.25	1.99	2.29	2.25
	MT-Bench	2.39	1.48	1.55	1.54
Qwen3-8B	ClassEval	3.50	2.22	2.55	2.55
	GSM8K	3.15	1.91	2.07	1.98
	HumanEval	2.74	1.69	1.90	1.88
	MBPP	3.13	1.96	2.11	2.08
	MT-Bench	2.79	1.71	1.73	1.74

token total (its official setting), while GOOSE generates longer (220–256 tokens per sample). Prefill is charged once per sample, so it is amortized over $\approx 2\times$ more cycles for GOOSE, inflating its tok/s by roughly two points relative to a length-matched run; τ is essentially unaffected (GOOSE’s τ counts the prefill forward while Sequoia’s official counters exclude it, which only understates GOOSE).

Table 14: GOOSE vs. official Sequoia on Llama-2-7B (A40, greedy, $n=50$; Section 5.2). Sequoia uses its released trained 68M drafter; GOOSE is training-free. Wall-clock margins carry a ~ 2 -point protocol uncertainty from the generation-length asymmetry described above.

Dataset	GOOSE		Sequoia (official)		Sequoia paper τ
	τ	tok/s	τ	tok/s	
C4	4.16	116.9	4.79	107.8	5.08
CNN/DM	3.04	86.6	3.91	91.1	4.05
OpenWebText	3.28	93.4	3.95	91.9	3.86

Table 15 reports our positional-DP reimplementation of Sequoia on modern targets (Section 5.2), with the drafter forward charged into wall-clock; a reimplementation is required because the released stack hard-codes Llama-2-era rotary embeddings and vocabulary sizes and supports neither Llama-3.1’s RoPE scaling nor Qwen3’s QK-norm. τ follows Sequoia’s counting convention (generated tokens per drafting–verification cycle, prefill excluded). Sequoia’s acceptance vector is calibrated on 8 held-back samples per dataset; on the three full-pool datasets this slice overlaps the evaluation set, an overlap that can only favor Sequoia. Validation proceeds in two steps. First, re-running Sequoia’s released tree-search on its released Llama-2-7B + 68M acceptance-rate vector reproduces our DP’s

tree level-for-level, with identical level sizes and expected accepted length. Second, the measured τ of our runner is nevertheless 8.6–24.2% *above* the official code on the same data, a residual we did not trace to a single cause, as the stacks differ beyond the tree itself. Since the gap only inflates Sequoia’s numbers, the wall-clock conclusions below are conservative. Lossless-rate accounting on the modern-stack cells (most severely Qwen3) is confounded by fp16 argmax near-ties (as in Appendix D.8), so we report τ and wall-clock only.

Table 15: Modern-stack Sequoia: τ and wall-clock speedup vs. AR at tree sizes $N=60/128$ (N = node count in Sequoia’s notation; $N=60$ matches our budget B ; A40, greedy, full n). Drafters: Llama-3.2-1B for Llama targets, Qwen3-0.6B for Qwen3-8B.

Target	Dataset	N=60		N=128	
		τ	Spd	τ	Spd
Llama-3-8B	ClassEval	8.63	1.52	8.95	1.52
	GSM8K	8.89	1.56	9.12	1.54
	HumanEval	8.56	1.53	8.88	1.51
	MBPP	8.42	1.48	8.75	1.48
	MT-Bench	7.57	1.36	8.00	1.38
Llama-3.1-8B	ClassEval	8.98	1.58	9.24	1.57
	GSM8K	8.97	1.60	9.15	1.56
	HumanEval	8.72	1.54	8.95	1.52
	MBPP	8.35	1.48	8.66	1.47
	MT-Bench	7.81	1.37	8.30	1.40
Qwen3-8B	ClassEval	7.32	0.82	7.81	0.85
	GSM8K	7.58	0.85	8.17	0.89
	HumanEval	6.81	0.76	7.47	0.82
	MBPP	7.34	0.82	7.88	0.86
	MT-Bench	6.22	0.70	6.71	0.73

The high- τ /low-speedup pattern is structural: Sequoia’s per-cycle drafter forwards (a 1B/0.6B model on a 60–128-node tree) cost a large fraction of what verification saves, and on Qwen3-8B they cost *more*: every cell is below $1\times$. GOOSE needs no drafter forward, which is exactly the regime difference Section 6 describes.

D.10 Principled Dynamic-Tree Allocator: Full Grid

Proposition 2 prescribes a linear branch schedule, which GOOSE approximates with a hand-tuned harmonic rule (Section 4.1). To measure what that approximation costs, we replace it with the greedy dynamic-tree expansion used by trained-drafter methods (Li et al., 2024b), parameterized by *measured* acceptance rather than drafter confidence: expand the global top- B candidates under $\text{Score}(\text{child}) = \text{Score}(\text{parent}) \cdot \hat{p}[\text{source}][\text{rank}]$, where $\hat{p}$ is the empirical acceptance frequency of a source’s rank- k candidate (rank = position among a node’s candidates by draft score), calibrated on a held-out prompt set disjoint from all evaluation benchmarks; multiplied along a path, the per-rank table induces the per-depth acceptance profile. Under independence, $\text{Score}(v)$ is the probability that v ’s entire root path is accepted, so expected accepted tokens = summed scores; no child outscore its parent, so the top- B set forms a tree and greedy expansion is exactly optimal for this objective. At strict equal budget ($B=60$) its static arm beats the deployed heuristic on 17 of 19 full-size cells (geometric mean +5.9%; Table 16). The two losing cells, GSM8K on the 8B Llamas, share one cause: long verbatim repetitions push the realized spine acceptance far above its trace average, so the static $\hat{p}$ under-builds the deep spine. The deployed heuristic can thus be read as a hand-tuned approximation that the principled allocator subsumes.

Table 16 reports the full equal-budget grid (τ ; full n except GSM8K at $n=500$; $B=60$). All arms spend an identical budget: the deployed heuristic runs with its confidence budget-extension disabled, the mechanism that would otherwise let high-confidence cycles slightly exceed B . “Inst.” scores each candidate by a calibrated map from its own instance features (adjacency count, match length) to acceptance, the closest training-free analogue of a trained drafter’s confidence.

Table 16: Equal-budget allocation comparison, τ (full n except GSM8K at $n=500$). **Bold** = row best. The two GSM8K/8B-Llama regressions are diagnosed above.

Model	Dataset	Heuristic	Allocator	Allocator (inst.)
Llama-3-8B	ClassEval	4.80	5.39	5.24
	GSM8K	8.84	7.48	7.25
	HumanEval	4.03	4.47	4.50
	MBPP	3.58	3.92	3.98
	MT-Bench	3.41	3.56	3.47
Llama-3.1-8B	ClassEval	5.46	5.95	5.81
	GSM8K	8.15	7.38	7.18
	HumanEval	4.47	4.86	4.96
	MBPP	3.48	3.78	3.85
	MT-Bench	3.62	3.69	3.71
Llama-3.2-1B	ClassEval	4.27	4.59	4.78
	GSM8K	3.17	3.48	3.57
	HumanEval	3.98	4.28	4.40
	MBPP	3.18	3.50	3.58
	MT-Bench	2.68	2.85	2.89
Qwen3-8B	ClassEval	3.44	3.94	3.92
	HumanEval	2.73	2.96	2.96
	MBPP	3.14	3.45	3.47
	MT-Bench	3.43	3.53	3.54

Qwen3-8B GSM8K is omitted because its instance-calibrated arm did not complete; on the two arms that did run ($n=500$), the allocator leads the heuristic (allocator 4.10 vs. heuristic 3.86).

E Theoretical Proofs and Verification

E.1 Proof of Proposition 1 (Spine Tree Expected Yield)

Proof. We use the indicator random variable (IRV) decomposition introduced for speculative decoding by Leviathan et al. (2023). For each position in the speculation tree, let $I = \mathbf{1}[\text{the accepted path reaches and accepts this position}]$. Since $\tau = \sum_{\text{positions}} I$ (plus the bonus token), linearity of expectation gives $\mathbb{E}[\tau] = \sum \Pr[I = 1] + 1$.

Spine positions ($d = 1, \dots, m$). Spine position d is reached and accepted iff all d spine tokens along the chain are accepted. Under the i.i.d. acceptance model, $\Pr[I = 1] = p_s^d$. Summing: $\mathbb{E}[\text{spine}] = \sum_{d=1}^m p_s^d$.

Branch positions at node i ($i = 0, \dots, m-1$). A branch subtree at node i is entered iff (a) all i preceding spine tokens are accepted (probability p_s^i) and (b) the $(i+1)$ -th spine token is rejected (probability $1-p_s$). Given entry, the branch token (depth offset $j = 0$) is accepted with probability $\phi_i = 1 - (1-p_t)^{w_i}$. For $j = 1, \dots, D-1$, each subsequent chain token is accepted independently with probability p_t , so depth offset j is *reached and accepted* with probability $\phi_i \cdot p_t^j$. Summing over all offsets within the branch subtree:

$$\mathbb{E}[\text{branch at node } i \mid \text{entry}] = \sum_{j=0}^{D-1} \phi_i p_t^j = \phi_i \left(1 + \sum_{j=1}^{D-1} p_t^j \right) = \phi_i (1 + \bar{\ell}),$$

where $\bar{\ell} = \sum_{j=1}^{D-1} p_t^j$ as defined in the main text. Here the “1” accounts for the branch token itself, and $\bar{\ell}$ for the expected subsequent chain extension. Multiplying by the entry probability and summing over all spine nodes:

$$\mathbb{E}[\text{synergy}] = \sum_{i=0}^{m-1} p_s^i (1-p_s) \phi_i (1 + \bar{\ell}).$$

Combining. Adding the spine contribution, the synergy contribution, and the +1 bonus token:

$$\mathbb{E}[\tau] \geq \sum_{i=1}^m p_s^i + \sum_{i=0}^{m-1} p_s^i (1-p_s) \phi_i (1 + \bar{\ell}) + 1.$$

The inequality holds because branches are modeled as independent linear chains; any deeper sub-branching within a branch (absent in this setup) would only add further contributions, confirming the lower-bound direction. The bound is tight when each branch subtree is an independent linear chain with no further sub-branching. $\square$

E.2 Proof of Optimal Branch Allocation

Proof. Let $\Delta = \sum_{i=0}^{m-1} p_s^i (1-p_s) \phi_i (1 + \bar{\ell})$ denote the synergy term in Equation (1), with $\phi_i = 1 - (1-p_t)^{w_i}$. The marginal synergy of adding one branch at position i is:

$$\frac{\partial \Delta}{\partial w_i} = p_s^i (1-p_s) (1-p_t)^{w_i} \ln \frac{1}{1-p_t} (1 + \bar{\ell}).$$

At an interior optimum under the constraint $\sum w_i = B_t$, all marginals must be equal (zero-width coordinates instead obey the Karush–Kuhn–Tucker (KKT) inequality, their marginal not exceeding the common value): $p_s^i (1-p_t)^{w_i} = \mu$ for a Lagrange constant μ . Taking logarithms: $i \ln p_s + w_i \ln(1-p_t) = \ln \mu$, which gives $w_i = w_0 - \frac{|\ln p_s|}{|\ln(1-p_t)|} \cdot i$, i.e., linearly decreasing with depth. The intercept w_0 is set by $\sum_{i=0}^{m-1} w_i = B_t$. $\square$

E.3 Proof of Anisotropic Dominance

Proof. Let the total node budget be B and define $p_s > p_t > 0$.

Step 1 (Replacement lemma). Consider any tree $\mathcal{T}$ of budget B where every candidate token has acceptance probability p_t . At depth d , let w_d denote the number of candidates; the probability of advancing past depth d is $q_d = 1 - (1 - p_t)^{w_d}$. Replace one of the w_d candidates with a spine token of acceptance p_s :

$$q'_d = 1 - (1 - p_s)(1 - p_t)^{w_d-1} = q_d + (p_s - p_t)(1 - p_t)^{w_d-1}.$$

Since $p_s > p_t$ and $(1 - p_t)^{w_d-1} > 0$, we have $q'_d > q_d$. The expected yield of any tree is $Y = \sum_{d=1}^D \prod_{j=1}^d q_j + 1$, which is strictly increasing in each q_j . Hence $Y' > Y$: replacing one branch token with a spine token at any single depth strictly improves yield.

Step 2 (Constructing a dominating spine tree). Start from the optimal isotropic tree $\mathcal{T}_{\text{iso}}^*$ of budget B —the optimum of Sequoia-style DP (Chen et al., 2024) restricted to a single source at rate p_t —which maximizes Y over all such trees. Let D be its maximum depth. For $d = 1, \dots, \min(m, D)$, apply Step 1 along a single root-to-leaf path, replacing one p_t -token per depth with a p_s -spine token. The budget is unchanged (B), but each replaced depth has $q'_d > q_d$, so:

$$Y_{\text{spine}} > Y_{\text{iso}}^*.$$

This spine tree is a feasible (not necessarily optimal) instantiation of Proposition 1.

Step 3 (Optimization improves further). Reallocating branch budget according to Proposition 2 can only increase the yield: $Y_{\text{spine}}^* \geq Y_{\text{spine}} > Y_{\text{iso}}^*$.

Step 4 (Gap monotonicity). The per-depth gain $(p_s - p_t)(1 - p_t)^{w_d-1}$ increases linearly in $(p_s - p_t)$, so $Y_{\text{spine}}^* - Y_{\text{iso}}^*$ is monotonically increasing in p_s/p_t (with p_t fixed) and vanishes as $p_s \rightarrow p_t$. $\square$

E.4 Numerical Verification of the Yield Bound

We verify the yield lower bound of Proposition 1 against measured compression ratios across all five target models and five benchmarks (24 settings; the Vicuna-33B GSM8K cell is excluded because its diagnostics come from a separate run of that environment; cf. Table 3). For each setting we extract $\hat{p}_s$ and $\hat{p}_t$ from the logged spine and branch statistics, estimate the average tree shape (m, B) from the per-cycle tree-shape logs, and compute three quantities: (i) τ_{eq5} , the lower bound from Eq. 1 with the optimal linear allocation of Proposition 2; (ii) τ_{iso} , the yield of an isotropic tree with fan-out 3 and the same budget using only p_t ; (iii) τ_{meas} , the measured per-prompt mean of accepted tokens per cycle (a per-prompt convention that differs from the pooled τ of Table 3).

The bound is valid ($\tau_{\text{eq5}} \leq \tau_{\text{meas}}$) in all 24 model $\times$ dataset settings, confirming that Proposition 1 provides a correct lower bound. The bound is conservative (median $\tau_{\text{eq5}}/\tau_{\text{meas}} \approx 38\%$), which is expected: the theory does not account for the consensus bypass mechanism (Section 4.3) or multi-cycle warm-start effects, both of which raise the effective acceptance rate in practice.

More importantly, $\tau_{\text{eq5}} > \tau_{\text{iso}}$ in all 24 settings (median ratio $\approx 1.52\times$), empirically confirming the dominance result of Proposition 3. The gain correlates strongly with the heterogeneity ratio $\hat{p}_s/\hat{p}_t$ (Pearson $r = 0.91$), consistent with the monotonicity predicted by Step 4 of the proof (Appendix E.3). Figure 6 visualizes the per-model dominance ratio.

E.5 Empirical Tail of the All-Reject Probability

For every node of the speculation tree we log the branching factor k and whether all k candidate children were rejected, over 10 trace files (Llama-3-8B and Llama-3.2-1B across the five benchmarks; 93,791 nodes). Within each k -bin we estimate the all-reject frequency $r_k = \Pr[\text{all } k \text{ candidates rejected}]$ and fit two families by least squares in log space (R^2 reported on $\log r_k$): geometric, $r_k = cq^k$, and power law, $r_k = ck^b$. Because Goose’s anisotropic tree induces a structural correlation between k and depth (wide forks sit near the root), we fit both on the aggregate data (k -bins with ≥ 200 observations) and within fixed-depth bins (≥ 100 observations per cell):

Bin	k -bins	obs.	Geometric R^2	Power-law R^2
aggregate	11	93,791	0.387	0.606
depth 0	11	28,787	0.466	0.587
depth 1	10	19,843	0.692	0.722

Deeper bins are sparse (3–7 populated k -cells) and neither family fits well there (at depths 2–5, $R^2 \leq 0.27$ for both, with the geometric nominally ahead; depth 6 has only 3 cells). On the well-populated bins the power law fits better, with exponent magnitude 0.261 (aggregate) and 0.189 (depth 0). The raw fitted slopes are *positive* (r_k grows with k), a selection effect of the adaptive builder discussed in Appendix E.6; the magnitudes serve only as tail-heaviness scales in the causal, decreasing orientation.

E.6 Optimal Branch Allocation under a Power-Law Tail

Setup. Proposition 2 derives the optimal branch widths $\{w_i^*\}$ under the i.i.d. model $\Pr[\text{all reject at depth } i \mid w_i] = (1 - p_t)^{w_i}$, i.e. a *geometric* tail in w_i . The i.i.d. assumption is a modeling approximation; here we ask how the optimal allocation changes when it is replaced by the tail family that the trace data actually prefer. Empirically (Appendix E.5), the all-reject frequency r_k as a function of the number of branch candidates k is better described by a power law than by a geometric: the aggregate fit over 11 k -bins gives $R^2 = 0.606$ (power law) versus 0.387 (geometric) with fitted exponent magnitude $\hat{\alpha} = 0.261$, and the same ordering holds within the most populous fixed-depth bin, which controls for the structural $k \leftrightarrow$ depth correlation of the anisotropic tree (depth 0, $n=28,787$: $R^2 = 0.587$ versus 0.466, exponent magnitude 0.189). We use the aggregate $\hat{\alpha} = 0.261$ below and report robustness to the depth-0 value. One caveat is stated up front: the *raw* fitted slope is positive (r_k grows with k) because the tree builder adaptively assigns more candidates at harder positions (a selection effect, not a causal effect of k), so we use the fitted magnitude only as a measure of tail heaviness, in the causal (decreasing) orientation of (4).

Model. At spine depth i , replace the geometric per-depth synergy factor with a power-law objective term $\tilde{\phi}_i$ (a weight absorbing the reach probability, not itself a probability; q_i here is distinct from q_d in Appendix E.3):

$$\tilde{\phi}_i(w_i) = 1 - q_i w_i^{-\alpha}, \quad q_i = (1 - p_t) p_s^i, \quad (4)$$

where p_s^i carries the probability that the spine reaches depth i (mirroring the synergy weights of Proposition 1; the geometric benchmark below uses the same reach weights, so the comparison is like-for-like) and the factor $(1 - p_t)$ calibrates the tail to the i.i.d. model at $w_i = 1$. For large w this tail is heavier than geometric (at the fitted parameters the two curves cross near $w \approx 26$); the geometric tail is recovered in the $\alpha \rightarrow \infty$ limit of the shifted power-law (Lomax) family, $(1 + w/(\alpha\sigma))^{-\alpha} \rightarrow e^{-w/\sigma}$, of which (4) is the heavy-tailed end.

Lagrangian. Maximising $\sum_i \tilde{\phi}_i(w_i)$ under the branch-budget constraint $\sum_i w_i = B_t$ gives

$$\mathcal{L} = \sum_i [1 - q_i w_i^{-\alpha}] - \lambda (\sum_i w_i - B_t).$$

Stationarity $\partial \mathcal{L} / \partial w_i = 0$ yields $\alpha q_i w_i^{-\alpha-1} = \lambda$, hence

$$w_i^* = C \cdot p_s^{i/(\alpha+1)}, \quad C \text{ set by } \sum_{i=0}^{m-1} w_i^* = B_t. \quad (5)$$

Each summand is strictly concave on $w_i > 0$ and the marginal value $\alpha q_i w_i^{-\alpha-1}$ diverges as $w_i \rightarrow 0^+$, so (5) is the unique constrained optimum and, unlike the geometric case below, every depth receives strictly positive width. Widths below one candidate lie outside the fitted region ($k \geq 1$), so positivity there is an artifact of the continuous relaxation; the substantive contrast is the graded shallow-depth profile versus the corner.

Effect of the tail model on the optimum. Under the geometric model, the stationarity condition gives the linear profile of Proposition 2, $w_i = w_0 - i |\ln p_s| / |\ln(1-p_t)|$; making the non-negativity constraint explicit, the exact optimum (by the KKT conditions) is its water-filling truncation $w_i^* = \max(0, w_0 - i |\ln p_s| / |\ln(1-p_t)|)$. At the measured medians ($p_s=0.21$, $p_t=0.033$) the slope is $|\ln 0.21| / |\ln 0.967| \approx 46.5$ (larger than the entire branch budget $B_t=30$), so the geometric optimum degenerates to the *corner* solution: the whole budget at the root, $w^* = (B_t, 0, \dots, 0)$. The corner obtains for any $B_t \leq 46.5$; under Proposition 2’s accounting $B_t=B-m=48$, a second depth becomes active with width 0.75, leaving the contrast unchanged. Under the power-law tail, the optimum (5) instead decays *geometrically* in i at rate $p_s^{1/(\alpha+1)} = 0.21^{1/1.261} \approx 0.29$ (widths shrink $\approx 3.4\times$ per depth step) and keeps a strictly positive width at every depth. Replacing the geometric assumption with the empirically fitted tail thus changes the optimum from “all branches at the root” to a graded, root-heavy schedule with a long shallow tail; the qualitative anisotropy of Proposition 2 (concentrate branches near the root) is preserved under both models, and the heavier-than-geometric empirical tail is exactly what rewards retaining residual breadth at deeper nodes. (With the depth-0 exponent 0.189 the decay rate is $0.21^{1/1.189} \approx 0.27$; the schedule is essentially unchanged.)

Position of the deployed schedule. With branch budget $B_t=30$ (the branch share at the spine-ratio ceiling $r=0.5$; the mid tier $r=0.30$ gives $B_t \approx 42$, also below the corner threshold), $p_s=0.21$, $p_t=0.033$, and the schedule truncated at $m=12$ depths (a representative spine length; the comparison is qualitatively insensitive to m), the three schedules are

i	0	1	2	3	4	5	6	≥ 7
w_i^* geometric (KKT)	30.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
w_i^* power-law	21.3	6.2	1.8	0.5	0.2	0.0	0.0	0.0
$1/(i+1)$ (deployed rule, idealized)	9.7	4.8	3.2	2.4	1.9	1.6	1.4	1.2–0.8

The deployed harmonic schedule is *more conservative* (less root-concentrated) than both optima at the root ($w_0=9.7$ vs. power-law 21.3 and geometric 30) and retains breadth at deep nodes to which neither optimum assigns appreciable width. Measured in ℓ_1 over normalised schedules (each scaled to unit sum), the deployed heuristic is closer to the power-law optimum ($\ell_1=0.87$) than to the geometric one ($\ell_1=1.36$): moving from the geometric assumption to the empirically fitted tail moves the predicted optimum *toward* the deployed schedule. Goose’s hand-tuned $1/(i+1)$ rule is thus a deliberately conservative compromise (we idealize it here by its harmonic schedule; Algorithm 1 additionally gives the root a separate share and indexes the spine harmonic from depth 1, which together raise the deployed root share to ≈ 0.49 of the branch budget without changing the ordering): it implements the monotone, root-heavy shape on which both models agree, while hedging against model misspecification with residual breadth at depths where the power-law (but not the geometric) model still assigns positive width. We expect a non-degradation guarantee analogous to Proposition 3 to carry over under either model, since the largest width satisfies $w_0 \geq B_t/m \geq 1$ whenever $B_t \geq m$.

Scope of the analysis. The allocation analysis governs only *how* a fixed branch budget is spread, not *whether* heterogeneous allocation helps: the 12–33% equal-budget gains of Table 1 are measured end-to-end and hold regardless of which tail model is assumed. Both the geometric and the power-law optima prescribe the same operative shape, monotone root-heavy widths, which is what the deployed $1/(i+1)$ rule implements; the empirical power-law fit additionally explains why the heuristic’s residual depth-side breadth is not wasteful.