WHERE DRAFT TREES LOSE TARGET MASS: EXIT-GUIDED SPECULATIVE DECODING

Anonymous authors
Paper under double-blind review

ABSTRACT

Tree-based speculative decoding organizes draft tokens into a bounded tree so that multiple continuations can be verified in one target-model pass. However, the tree is constructed from draft-side scores while its usefulness is ultimately determined by the target model, creating a fundamental draft–target mismatch under a finite tree budget. We study two questions: once a draft tree is fixed, can a better exact verifier accept more draft tokens, and if not, how can target feedback improve the tree itself? We answer both through a target-flow view of the fixed draft tree. We identify a canonical *exit law* that describes where target continuations leave the tree, and prove that one plus the resulting target coverage is a sharp upper bound on the expected output-block length, including the bonus token, of any exact path verifier. Moreover, every verifier attaining this ceiling must realize the same exit and bonus-token law; representative predraw-and-follow and sequential residual verifiers already attain it. This theory directly yields *Tree Exit Verification* (TEV), an exact verifier that realizes the canonical law through one exit-node decision and one bonus-token decision, exposing a regular, level-parallel verification procedure. The same exit law also localizes where target probability is missing from the current tree. We use it as node-level feedback to train the drafter on its inference-time draft trees, reallocating finite tree budget toward target-relevant regions. Experiments across dialogue, code, and mathematical reasoning validate the fixed-tree equivalence, show that the *Exit-Guided Draft-Tree Training* (ExitTrain) improves average output-block length by 13%, and that TEV reduces verifier-stage latency by 15%, yielding 14% end-to-end speedup over DDTree. Together, our results separate the two remaining opportunities in bounded tree speculative decoding: *better draft trees for higher acceptance, and more direct verification for lower latency*. Code is available at https://anonymous.4open.science/r/TEV-ICLR-348C/.

1 INTRODUCTION

Large language models (LLMs) decode autoregressively, requiring an expensive target-model forward pass per token. Speculative decoding reduces this cost by using a lightweight draft model to propose future tokens that the target verifies in parallel (Leviathan et al., 2023; Chen et al., 2023). While early methods draft a single sequence, recent systems organize many draft tokens into a tree, allowing multiple possible continuations to share one target-model verification pass (Miao et al., 2024; Chen et al., 2024; Li et al., 2024b; 2025). Tree speculative decoding can therefore be viewed as three stages: the *drafter* proposes draft tokens and their scores; the *tree builder* selects and organizes a limited number of them into a draft tree; and the *verifier* uses the target model to determine how far generation can safely follow a path in that tree while preserving the target distribution. Existing systems implement verifier differently. DDTree's predraw-and-follow verifier samples a target token at each tree node and follows a matching drafted child (Ringel & Romano, 2026), whereas EAGLE-style sequential residual verifier tests drafted children sequentially while updating the remaining target probability (Li et al., 2025). DFlash with DDTree provides one concrete realization of this drafter–builder–verifier pipeline (Chen et al., 2026; Ringel & Romano, 2026).

Under a fixed tree budget, the draft tree determines which future prefixes can be verified in a single target forward pass. Among exponentially many possible prefixes, only a small number can appear in the tree. Since the tree must be built *before* the target model evaluates it, practical builders rely on draft-model scores (e.g., confidence) to decide which branches to keep (Li et al., 2024b; 2025;

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

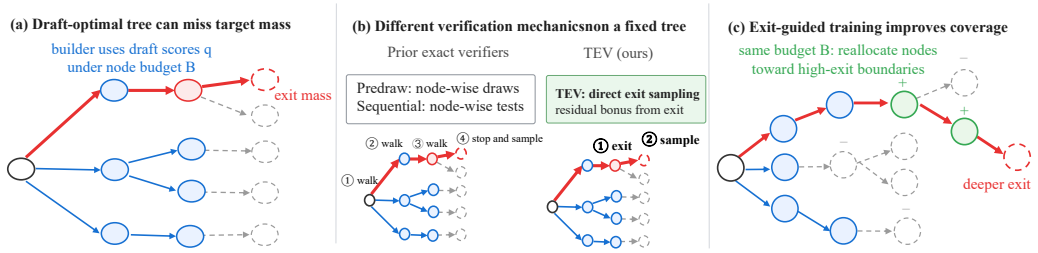


Figure 1: **Target-flow view of tree speculative decoding.** Blue denotes selected draft prefixes, gray dashed branches omitted candidates, red target-probability flow, and green reallocated tree budget. **(a)** Under a finite budget, draft-score-based tree construction can miss target-important branches, causing target mass to exit the tree. **(b)** On a fixed tree, prior exact verifiers realize the same optimal acceptance law through node-wise draws or tests, whereas TEV realizes it by direct exit-node sampling followed by bonus-token sampling. **(c)** Exit-guided training reallocates budget toward high-exit boundaries, improving target-mass coverage and output-block length.

Ringel & Romano, 2026). Yet the usefulness of those branches is ultimately determined by the target model. When the draft and target prefer different continuations, limited tree budget may be spent on branches that look promising to the drafter but are rarely followed by the target, while target-likely branches remain insufficiently expanded in the tree. Fig. 1(a) illustrates this *draft–target tree-allocation gap*. Although we use DDTree as our main instantiation, the same mismatch arises whenever draft-side information is used to construct a bounded draft tree before target verification.

This mismatch raises two fundamental questions. *First*, once a draft tree is fixed, how much freedom remains in exact verification? Recent work like HSD (Zhou et al., 2026) shows that improved verification can increase acceptance for a sampled draft chain. So can another exact verifier similarly accept more tokens from the *same tree*, or have existing exact verifiers already reached the tree's maximum achievable acceptance, leaving only execution efficiency to optimize? *Second*, if existing verifiers already saturate the fixed-tree ceiling, how can target-model feedback improve future draft trees? Can we identify where target probability leaves the tree and use this signal to allocate the limited tree budget more effectively? These questions reveal two opportunities in tree speculative decoding: *how much target probability the tree covers, and how efficiently that support is verified*.

Our approach and contributions. We answer the both questions through a unified target-flow view of the realized draft tree. First, we develop a verifier-independent theory showing that the maximum expected output-block length on a fixed tree is one plus the target probability mass covered by the selected support. This theory exposes a canonical target-exit law, from which we derive *Tree Exit Verification* (TEV) as a direct and implementation-friendly realization of optimal fixed-tree verification. Finally, we use the same exit law as node-resolved training feedback to improve target-mass coverage under a finite tree budget, thereby increasing draft acceptance. Fig. 1 summarizes these three contributions.

First, we characterize the acceptance limit of exact path verifiers on a fixed tree: the maximum expected output-block length equals one plus its target-mass coverage, and every verifier attaining this bound induces the same canonical exit-node and bonus-token law. DDTree's predraw-and-follow (Ringel & Romano, 2026) and EAGLE-style exact sequential residual verification (Li et al., 2025) realize this law and thus have identical accepted-path distributions on the same tree. Once verification is saturated, higher acceptance requires better tree coverage, not a different verifier.

Second, we derive *Tree Exit Verification* (TEV), an exact, direct realization of this optimal law. Rather than using auxiliary per-node samples or sequential residual decisions, TEV samples an exit node directly and then a bonus token from its target residual (Fig. 1(b)). It preserves the fixed-tree acceptance ceiling while enabling sibling-order-invariant, level-parallel, fixed-shape execution: *statistical saturation does not imply computational saturation*.

Third, we introduce *Exit-guided Draft-Tree Training* (ExitTrain) to improve finite-budget tree coverage. ExitTrain evaluates the draft tree with the frozen target under actual tree-parent contexts and uses exit mass to reweight draft likelihoods, guiding budget allocation toward high-impact exit boundaries (Fig. 1(c)). Unlike token- or feature-level alignment (Zhang et al., 2025; Hu et al., 2025;

Li et al., 2025), position-wise distillation (Lei et al., 2026), or scalar tree-reward optimization (Hu et al., 2026), this supervision localizes uncovered target mass directly on the draft tree.

Finally, we evaluate ExitTrain + TEV across dialogue, code, and mathematical reasoning benchmarks and multiple target-model families, against DFlash (Chen et al., 2026), DDTree (Ringel & Romano, 2026), and Draft-OPD (Lei et al., 2026). Relative to DDTree with the same tree builder and budget, our system improves average output-block length (including the bonus token) by up to **13%** and decode throughput by up to **14%**. Separately, replacing predraw verification with TEV reduces verifier-stage latency by **15%** without changing saturated expected output-block length on a fixed tree. These results support two complementary gains: *better trees for higher acceptance and direct verification for lower latency*.

2 Background and Fixed-Tree Setup

Tree speculative decoding. A *drafter* proposes candidates, a budget-B *builder* selects a prefix-closed tree, and one target pass lets a *verifier* return an accepted path plus a bonus token. Systems combine multi-head, feature-autoregressive, recurrent, or block-diffusion drafting with parallel tree verification (Cai et al., 2024; Li et al., 2024a; Cheng et al., 2024; Chen et al., 2026; Miao et al., 2024; Chen et al., 2024; Li et al., 2024b; 2025; Svirschevski et al., 2024). Builders optimize this support using draft-side expected acceptance (OPT-Tree), block-diffusion marginals (DDTree), or autoregressive correction (DARTree) (Wang et al., 2025; Ringel & Romano, 2026; Li et al., 2026). We instantiate DFlash with DDTree's cumulative-score best-first builder (Chen et al., 2026; Ringel & Romano, 2026). These construction methods improve which tree is built, whereas our theory asks what any non-anticipating exact verifier can achieve once a prefix tree has been realized.

Two representative exact verifiers. DDTree's *predraw-and-follow* independently samples one target token at each node (Ringel & Romano, 2026). Starting at the root, it follows a matching drafted child; otherwise, it stops and returns the sample as the bonus token. EAGLE-style *exact sequential residual verification* instead tests children in order (Li et al., 2025). After each rejection, it removes that child's target mass and renormalizes the residual, following the first accepted child or sampling a bonus token if all are rejected. Sec. 3 shows that despite these different computations, both attain the same optimal law on a fixed tree. HSD (Zhou et al., 2026) instead verifies a sampled draft chain.

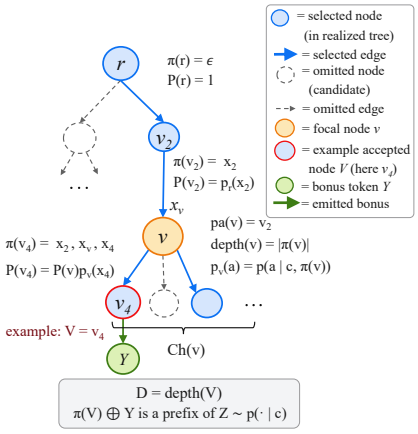


Figure 2: **Visualization of a fixed-tree.**

Related draft-model training. Prior work aligns drafters through on-policy or online distillation (Zhou et al., 2024; Liu et al., 2024), feature-, token-, or multi-step objectives (Zhang et al., 2025; Hu et al., 2025; Li et al., 2025), tree rewards (Hu et al., 2026), replayed trajectories (Lei et al., 2026), or acceptable-path modeling (Zou et al., 2026). Closest to ours, GTO optimizes a scalar tree reward (Hu et al., 2026), Learning to Draft learns throughput-driven depth and verification-size policies (Zhang et al., 2026), and VAT reweights simulated first-rejection positions (Gu et al., 2026). ExitTrain instead derives normalized node-level supervision from the canonical exit law of each realized finite tree, directly reallocating a fixed tree budget toward uncovered target mass.

3 The Canonical Exit Law and the Fixed-Tree Ceiling

We condition on a decoding context and a realized draft tree. By following target probability through this fixed tree, we characterize both the largest expected output-block length any exact verifier can achieve and the unique law attained at that ceiling.

Fixed-tree setup. Let $\mathcal{X}$ be the token vocabulary and let $p(\cdot \mid c)$ denote the target continuation law given context c. A drafter and budget-B builder produce a finite, rooted, prefix-closed tree $\mathcal{T} = (\mathcal{V}, \mathcal{E}, r)$ (Fig. 2), where $\mathcal{V}$ is the node set, $\mathcal{E}$ is the directed edge set, and r is the root. Set

$\pi(r) = \epsilon$, the empty sequence. Let $\mathcal{X}^*$ denote the set of finite token sequences. Each node v represents a speculative continuation prefix $\pi(v) \in \mathcal{X}^*$; for $v \neq r$, let $\mathrm{pa}(v)$, x_v, $\mathrm{d}(v) = |\pi(v)|$, and $\mathrm{Ch}(v)$ denote its parent, incoming token, depth, and children, so that $\pi(v) = \pi(\mathrm{pa}(v)) \,\|\, x_v$. Here $\|$ denotes sequence–token concatenation, $\alpha \preceq \beta$ means that sequence α is a prefix of β, and $\mathbf{1}[\cdot]$ denotes an indicator. Sibling tokens are distinct. Set $P(r) = 1$; one target-model tree forward pass gives

$$p_v(a) := p(a \mid c, \pi(v)), \quad a \in \mathcal{X}, \tag{1}$$

$$P(v) := P(\mathrm{pa}(v)) p_{\mathrm{pa}(v)}(x_v) = p(\pi(v) \mid c), \tag{2}$$

where $P(v)$ is the target prefix probability. A verifier returns a path ending at V and a bonus token Y. Let $D := \mathrm{d}(V)$ be the number of accepted draft tokens, and let $O := \pi(V) \,\|\, Y$ and $L := |O| = D + 1$ be the output block and its length, including the bonus token. Let U denote auxiliary control randomness used, for example, by randomized early-stopping decisions and available before target tokens are revealed; randomness represented by the coupled target continuation Z is not included in U. We call the verifier an *exact path verifier* if there exists a coupling with a target continuation $Z = (Z_1, Z_2, \ldots)$ such that

$$Z \mid (c, \mathcal{T}, U) \sim p(\cdot \mid c), \qquad O = Z_{1:L} \text{ a.s.}, \tag{3}$$

and L is a stopping time with respect to $\mathcal{H}_t := \sigma(\mathcal{T}, U, Z_1, \ldots, Z_t)$. Thus, the realized tree and independent control randomness are available at time zero, but neither may encode future target tokens. The target-prefix identity in Eq. (3) is sufficient for the fixed-tree ceiling and equality characterization below; the stopping-time condition ensures that variable-length blocks compose safely across decoding rounds. Appendix A.1 formalizes random construction and round-wise composition.

3.1 Target flow and the canonical exit law

We first describe how the target model interacts with a *fixed* draft tree, without referring to any particular verifier. Consider a continuation $Z \sim p(\cdot \mid c)$ from the target model. Starting from the root, Z follows matching drafted children for as long as its next token is represented in the tree. When the required next token is absent—or when a leaf is reached—the continuation leaves the draft tree. We call its deepest matched node the *exit node*, denoted by $V^\star(Z)$ and illustrated in Fig. 2. Intuitively, the exit node marks *where the draft tree stops covering a target continuation*.

For a node v, let

$$\mathcal{A}(v) := \{x_u : u \in \mathrm{Ch}(v)\}, \quad \rho(v) := 1 - \sum\nolimits_{u \in \mathrm{Ch}(v)} p_v(x_u), \quad w(v) := P(v)\rho(v). \tag{4}$$

Here, $\mathcal{A}(v)$ contains the draft tokens available after v. $P(v)$ is the probability that the target continuation reaches v. Conditioned on reaching v, the *local exit probability* $\rho(v)$ is the fraction of target next-token probability not covered by its drafted children. So $w(v)$ is the overall probability that the target reaches v and exits the draft tree there. For a leaf, no drafted child is available, so $\rho(v) = 1$.

Lemma 1 (Canonical exit law)**.** *The target-induced exit node satisfies*

$$\Pr(V^\star(Z) = v \mid c, \mathcal{T}) = w(v), \qquad \sum\nolimits_{v \in \mathcal{V}} w(v) = 1. \tag{5}$$

The first identity in Lemma 1 says that $w(v)$ is exactly the probability that a target continuation remains inside the draft tree up to v and leaves there. The second follows because every target continuation has exactly one exit node in a finite tree. Thus, $\{w(v)\}_{v \in \mathcal{V}}$ is a normalized distribution over where the target leaves the draft tree, which we call the *canonical exit law*. Importantly, this law is determined only by the target model and the fixed draft tree—not by a verifier. See the complete proof and probability-conservation argument in Appendix A.2.

3.2 From the exit law to the fixed-tree acceptance ceiling

Let D_{A} and $L_{\mathsf{A}} = D_{\mathsf{A}} + 1$ denote, respectively, the accepted draft-token count and output-block length of an exact path verifier A. Accepting node u requires the target continuation to contain prefix $\pi(u)$, so its acceptance probability is at most $P(u)$. Summing these node-wise limits motivates

$$C_p(\mathcal{T}) := \sum\nolimits_{u \in \mathcal{V} \setminus \{r\}} P(u) = \sum\nolimits_{v \in \mathcal{V}} w(v)\mathrm{d}(v), \tag{6}$$

which we call the *target coverage* of the draft tree.

The first form sums the target probability of every represented non-root prefix. The second, which follows from the exit law, identifies the same quantity as the expected exit depth, i.e., the expected number of accepted draft tokens. Because every round also emits one bonus token, the corresponding expected output-block length at saturation is $1 + C_p(\mathcal{T})$.

For $\rho(v) > 0$, define the residual target distribution

$$p_v^{\mathrm{res}}(a) := p_v(a)\mathbf{1}[a \notin \mathcal{A}(v)] \,/\, \rho(v). \tag{7}$$

which removes drafted-child tokens and renormalizes; at a leaf, $p_v^{\mathrm{res}} = p_v$. We call a verifier *saturated* when $\mathbb{E}[D \mid c, \mathcal{T}] = C_p(\mathcal{T})$, equivalently when $\mathbb{E}[L \mid c, \mathcal{T}] = 1 + C_p(\mathcal{T})$. The following theorem gives this sharp ceiling and the unique output law at equality; Appendix A.3 provides the proof.

Theorem 1 (Fixed-tree ceiling and canonical saturation)**.** *For any exact path verifier* A *and any non-root node* u*,*

$$\Pr_{\mathsf{A}}(u \textit{ is accepted} \mid c, \mathcal{T}) \le P(u). \tag{8}$$

Consequently,

$$\mathbb{E}[D_{\mathsf{A}} \mid c, \mathcal{T}] \le C_p(\mathcal{T}), \qquad \mathbb{E}[L_{\mathsf{A}} \mid c, \mathcal{T}] \le 1 + C_p(\mathcal{T}). \tag{9}$$

If either length bound is tight, every node-wise bound is tight and, for every v *with* $w(v) > 0$*,*

$$\Pr_{\mathsf{A}}(V = v \mid c, \mathcal{T}) = w(v), \qquad Y \mid (V = v, c, \mathcal{T}) \sim p_v^{\mathrm{res}}. \tag{10}$$

Hence, all saturated exact path verifiers induce the same accepted-path and bonus-token law.

The theorem has two implications. First, $C_p(\mathcal{T})$ is the maximum expected number of accepted draft tokens, and $1 + C_p(\mathcal{T})$ is the maximum expected output-block length reported by our experimental metric: no exact verifier can recover a drafted token absent from the tree. Second, saturation makes every node-wise bound tight, forcing the exit law w and residual bonus-token law p_v^{res}. Saturated verifiers may therefore differ computationally but not in their accepted-path distribution. The representative verifiers in Sec. 2 attain this ceiling (Appendix A.4).

Corollary 1 (Representative verifiers are saturated)**.** *Predraw-and-follow in DDTree (Ringel & Romano, 2026) and exact sequential residual verification in EAGLE-style tree decoding (Li et al., 2025) both attain the fixed-tree ceiling. Therefore, on the same draft tree and target distributions, they share the canonical exit law, accepted-path distribution, expected accepted draft-token count, and expected output-block length.*

Whenever node v is reached, both representative verifiers follow each drafted child u with probability $p_v(x_u)$ and exit with the remaining probability $\rho(v)$. They therefore realize the same canonical law despite different internal computations.

3.3 What remains after fixed-tree saturation

The fixed-tree result leaves two opportunities. The same law can be executed more efficiently, motivating TEV in Sec. 4: *statistical saturation does not imply computational saturation*. Higher output-block length, however, requires increasing $C_p(\mathcal{T})$ itself. A large $w(v)$ localizes target mass that reaches v but leaves the current tree, motivating the training method in Sec. 5. This result complements HSD (Zhou et al., 2026), which exploits target-to-draft ratios while verifying a random sampled chain rather than conditioning on a realized tree (Appendix A.5).

4 TEV: Direct Verification from the Canonical Exit Law

Sec. 3 resolves the statistical question: saturated exact verifiers share the same canonical exit law and output-block length on a fixed tree. The remaining freedom is computational—*how can this common law be realized efficiently?* Existing verifiers recover it through per-node draws or sequential child decisions. Tree Exit Verification (TEV) instead realizes it directly.

Theorem 1 specifies the required output: exit at v with probability $w(v)$ and then draw the bonus token from p_v^{res}. TEV therefore computes the exit distribution, draws one exit node, and samples one residual bonus token. Algorithm 1 in Appendix B gives the complete procedure after the shared

target-model tree forward pass produces $\{p_v\}$. The drafter need not be queried again: it has already determined what appears in the realized tree, while the target distributions determine how probability flows through that support.

Because Algorithm 1 implements the canonical law, its correctness follows from Sec. 3.2; Appendix B.1 gives the proof.

Corollary 2 (TEV is exact and saturated)**.** *For any context and realized prefix-closed draft tree,* TEV *is an exact path verifier and attains the fixed-tree ceiling. In particular, we have*

$$\Pr_{\mathrm{TEV}}(u \textit{ is accepted} \mid c, \mathcal{T}) = P(u) \quad (\forall u \in \mathcal{V} \setminus \{r\}), \qquad \mathbb{E}[L_{\mathrm{TEV}} \mid c, \mathcal{T}] = 1 + C_p(\mathcal{T}). \tag{11}$$

By Lemma 1, a target continuation exits according to w and produces its next token according to p_v^{res}. TEV reproduces these distributions; node u is accepted exactly when the sampled exit lies in its subtree, whose total exit mass is $P(u)$.

Efficiency. TEV shares the accepted-path law of predraw and sequential residual verification, but removes their per-node samples or data-dependent child tests. It uses node-local reductions, depth-wise propagation, and one exit-node and bonus-token draw, yielding a sibling-order-invariant, fixed-shape, level-parallel procedure. Sec. 6.2 evaluates this latency benefit separately from output-block length.

5 ExitTrain: Exit-Guided Draft-Tree Training

Sec. 4 addresses the remaining computational freedom: TEV changes how the canonical law is executed, not the fixed-tree acceptance ceiling. We now turn to the second question from Sec. 1: *how can target feedback build a better draft tree?* Theorem 1 provides the key: once verification is saturated, higher output-block length requires greater target coverage, and the exit law reveals where that coverage is missing.

Let H be the maximum draft depth. For an expandable node v with $\mathrm{d}(v) < H$ and a missing child token $a \notin \mathcal{A}(v)$, write $\mathcal{T} \oplus (v, a)$ for the tree obtained by adding child (v, a). Then

$$C_p(\mathcal{T} \oplus (v, a)) - C_p(\mathcal{T}) = P(v)p_v(a), \tag{12}$$

$$\sum\nolimits_{a \notin \mathcal{A}(v)} P(v)p_v(a) = P(v)\rho(v) = w(v). \tag{13}$$

Hence, for an expandable node v, $w(v)$ measures the total one-step target coverage currently missing below v. A large $w(v)$ means that the target frequently reaches this prefix but substantial next-token probability is not represented by its drafted children. These are precisely the regions where allocating more tree capacity can keep target continuations inside the tree for longer. This expansion interpretation does not apply to a hard-horizon leaf. If H is the maximum draft depth and $\mathrm{d}(v) = H$, then $\rho(v) = 1$ and $w(v) = P(v)$; weighting its incoming token rewards a path that already carries target mass to the full horizon rather than missing one-step coverage.

This observation directly suggests our training strategy. Let q_θ be the current drafter with parameters θ. For each training context c, q_θ and the same tree builder used at inference construct a bounded tree $\widehat{\mathcal{T}}_\theta(c)$. We keep its structure fixed within the update, evaluate every node with the frozen target model under its actual tree-parent context, and compute the exit probabilities $w(v)$. Training therefore receives node-level feedback about where the *current inference-time draft tree* fails to cover target probability, rather than supervision from only a single draft path.

For a minibatch $\mathcal{B}$ of contexts $\{c_i\}_{i \in \mathcal{B}}$, let $\widehat{\mathcal{T}}_i = (\widehat{\mathcal{V}}_i, \widehat{\mathcal{E}}_i, r)$ be the realized tree for example i, let $w_i(v)$ be its detached target-derived exit weight, and let $q_{\theta,d}(\cdot \mid c_i)$ be DFlash's position-wise marginal distribution at future draft position d. We reweight the likelihood of the incoming token at each non-root node:

$$\mathcal{L}_{\mathrm{exit}}(\theta) := -\frac{\sum_{i \in \mathcal{B}} \sum_{v \in \widehat{\mathcal{V}}_i \setminus \{r\}} w_i(v) \log q_{\theta,\mathrm{d}(v)}(x_v \mid c_i)}{\sum_{i \in \mathcal{B}} (|\widehat{\mathcal{V}}_i| - 1)}. \tag{14}$$

Consequently, nodes at the same depth share the same draft row even when they have different tree-parent prefixes; in contrast, p_v is evaluated under the actual prefix $(c_i, \pi(v))$. All target-derived quantities and the realized tree structure are detached; gradients flow only through the drafter.

Why reinforce the position-wise marginal probability of the incoming draft token of an exit node? In DDTree, a prefix is ranked by cumulative draft scores along its path. Increasing the marginal probability assigned to x_v at position $\mathrm{d}(v)$ therefore raises the future priority of v and its descendants, making the builder more likely to retain and further expand that region. Weighting this update by $w(v)$ directs future tree budget toward prefixes that the target frequently reaches but the current tree insufficiently expands. Because the top-B builder is discrete, this is a *tree-allocation surrogate*, not the exact gradient of target coverage; its role is to improve *where* the bounded tree spends its nodes.

Finally, let $\mathcal{L}_{\mathrm{tok}}$ denote the standard dense token-prediction loss and let $\lambda_{\mathrm{exit}} \geq 0$ control the exit-guided term. We optimize

$$\mathcal{L}(\theta) = \mathcal{L}_{\mathrm{tok}}(\theta) + \lambda_{\mathrm{exit}} \mathcal{L}_{\mathrm{exit}}(\theta). \tag{15}$$

The two terms are complementary: the dense token loss teaches the drafter *what to predict*, while the exit-guided loss teaches a bounded tree *where to allocate its capacity*. Our default loss includes hard-horizon leaves, so $w(v)$ has two roles: missing-coverage credit at expandable nodes and successful-deep-reach credit at depth H. The root requires separate treatment: $w(r)$ measures target mass missed by every first-position draft token, but r has no incoming draft edge and is omitted from $\mathcal{L}_{\mathrm{exit}}$. Consequently, the included exit weights sum to $1 - w(r)$ rather than one, and the root-exit mass receives no direct credit from $\mathcal{L}_{\mathrm{exit}}$; coverage of missing first-position tokens is addressed through the dense token objective. In this way, the same canonical exit law serves two distinct roles: TEV uses it to realize fixed-tree verification directly, while exit-guided training uses it to improve the tree itself. Sec. 6.2 studies alternative weights, and Appendix C.2 gives an efficient implementation.

6 Experiments

Setup. We evaluate Qwen3-4B/8B (Yang et al., 2025), LLaMA-3.1-8B (Grattafiori et al., 2024), Gemma-4-12B-IT (Gemma Team, 2026), and Qwen3-Coder-30B-A3B (Yang et al., 2025) on MT-Bench (Zheng et al., 2023), HumanEval (Chen et al., 2021), GSM8K (Cobbe et al., 2021), and MATH-500 (Hendrycks et al., 2021). Drafter training uses `open-perfectblend` (Xu et al., 2024) prompts and online responses rolled out by each corresponding target model (Appendix D).

Methods. DDTree uses the released DFlash drafter, cumulative-score best-first builder, and predraw verifier (Chen et al., 2026; Ringel & Romano, 2026); Draft-OPD replaces only the drafter with its position-wise distilled checkpoint (Lei et al., 2026). **ExitTrain + TEV** denotes the complete system combining our exit-guided drafter with TEV. Appendix D gives full details.

Metrics. Because all methods preserve the target distribution, we report speedup over autoregressive decoding (SR) and average output-block length per draft–verify cycle ($\tau = \mathbb{E}[L]$), including the bonus token, at $T \in \{0, 1\}$.

6.1 Main Results

Table 1 shows that ExitTrain + TEV achieves the highest τ and SR in every reported setting. The Qwen3-4B/8B comparisons provide the most controlled view because they use the same builder and budget and include both DDTree and Draft-OPD. Across the two model sizes and temperatures, ExitTrain + TEV improves average τ over DDTree by 8.8%–13.8% and average SR by 11.5%–13.8%. It also exceeds Draft-OPD by 4.2%–6.9% in τ and 5.5%–7.3% in SR, showing that the gains persist over both the released and position-wise-distilled drafters under greedy and stochastic decoding.

The improvement also transfers across target-model families. On LLaMA-3.1-8B, ExitTrain + TEV improves average τ by 17.7%/10.4% and SR by 16.1%/10.1% at T=0/1. The corresponding τ and SR gains are 5.6%/4.8% and 5.7%/4.8% on Gemma-4-12B-IT, and 5.2%/4.3% and 5.2%/4.5% on Qwen3-Coder-30B-A3B. Thus, the advantage remains positive for dense, instruction-tuned, and MoE targets, although its magnitude varies across model families.

The gains also exhibit meaningful task variation. On Qwen3, GSM8K yields the largest increases in τ: 20.5%–22.4% at T=0 and 16.3%–17.8% at T=1, while HumanEval shows smaller but consistent improvements. More broadly, increases in τ translate consistently into higher SR. Because the builder and budget ($B = 64$) are fixed, the additional accepted length cannot come from verifying more candidates; it is the empirical signature of better target-prefix coverage, which raises the saturated output-block length $1 + C_p(\mathcal{T})$.

Table 1: Speedup ratio (SR) and average output-block length including the bonus token (τ) on MT-Bench, HumanEval, GSM8K, and MATH-500. All tree-based systems share the same builder and tree budget $B = 64$. ExitTrain + TEV combines our exit-guided drafter with TEV.

		MT-Bench		HumanEval		GSM8K		MATH-500		Average	
Model	**Method**	$SR\uparrow$	$\tau\uparrow$	$SR\uparrow$	$\tau\uparrow$	$SR\uparrow$	$\tau\uparrow$	$SR\uparrow$	$\tau\uparrow$	$SR\uparrow$	$\tau\uparrow$
Temperature = 0											
Qwen3-4B	DDTree	3.81	5.41	5.71	8.13	5.65	8.48	6.83	9.16	5.50	7.80
	Draft-OPD	4.16	5.73	5.96	8.35	6.11	9.30	7.02	10.14	5.81	8.38
	ExitTrain + TEV	**4.28**	**6.17**	**6.09**	**8.74**	**6.91**	**10.22**	**7.23**	**10.26**	**6.13**	**8.85**
Qwen3-8B	DDTree	3.75	5.54	5.83	8.01	5.88	8.49	6.23	9.76	5.43	7.95
	Draft-OPD	3.90	5.92	6.02	8.46	6.27	9.33	6.84	10.17	5.76	8.47
	ExitTrain + TEV	**4.22**	**6.12**	**6.15**	**8.73**	**7.20**	**10.39**	**7.15**	**10.69**	**6.18**	**8.98**
LLaMA-3.1-8B	DDTree	3.55	5.53	4.33	6.53	3.79	5.89	3.70	5.71	3.84	5.92
	ExitTrain + TEV	**3.69**	**5.82**	**4.72**	**7.24**	**4.75**	**7.48**	**4.70**	**7.34**	**4.46**	**6.97**
Gemma-4-12B-IT	DDTree	4.17	5.44	4.83	5.66	4.39	5.29	4.87	5.86	4.57	5.56
	ExitTrain + TEV	**4.35**	**5.57**	**5.09**	**5.92**	**4.70**	**5.66**	**5.17**	**6.33**	**4.83**	**5.87**
Qwen3-Coder-30B-A3B	DDTree	3.70	4.87	7.57	9.95	5.26	6.81	5.61	7.38	5.53	7.25
	ExitTrain + TEV	**3.83**	**5.03**	**7.83**	**10.32**	**5.61**	**7.43**	**6.00**	**7.74**	**5.82**	**7.63**
Temperature = 1											
Qwen3-4B	DDTree	3.44	5.04	5.22	7.53	5.48	7.85	5.65	7.88	4.95	7.08
	Draft-OPD	3.74	5.26	5.49	7.80	5.88	8.58	6.08	8.50	5.30	7.54
	ExitTrain + TEV	**3.98**	**5.86**	**5.64**	**8.06**	**6.47**	**9.25**	**6.26**	**9.08**	**5.59**	**8.06**
Qwen3-8B	DDTree	3.42	5.03	4.83	7.06	5.23	7.84	5.50	8.17	4.74	7.03
	Draft-OPD	3.51	5.21	4.95	7.29	5.79	8.57	5.66	8.28	4.98	7.34
	ExitTrain + TEV	**3.74**	**5.40**	**5.11**	**7.47**	**6.35**	**9.12**	**5.96**	**8.59**	**5.29**	**7.65**
LLaMA-3.1-8B	DDTree	2.43	3.97	3.71	5.67	3.07	4.81	2.26	3.71	2.87	4.54
	ExitTrain + TEV	**2.52**	**4.08**	**3.84**	**5.93**	**3.67**	**5.77**	**2.63**	**4.25**	**3.16**	**5.01**
Gemma-4-12B-IT	DDTree	3.65	4.71	4.13	5.06	4.48	5.41	4.57	5.78	4.21	5.24
	ExitTrain + TEV	**3.83**	**4.83**	**4.39**	**5.28**	**4.70**	**5.83**	**4.74**	**6.02**	**4.41**	**5.49**
Qwen3-Coder-30B-A3B	DDTree	3.48	4.53	7.26	9.61	5.22	6.75	5.43	7.13	5.35	7.01
	ExitTrain + TEV	**3.61**	**4.72**	**7.39**	**9.92**	**5.57**	**7.17**	**5.78**	**7.42**	**5.59**	**7.31**

Crossing drafter training and verification. The crossed evaluation in Appendix D.2 confirms this attribution on Qwen3-4B at $T{=}1$. Holding Predraw fixed, exit training raises average τ from 7.08 to 8.05 and SR from $4.95\times$ to $5.55\times$; with TEV, the corresponding changes are 7.09 to 8.06 and $4.97\times$ to $5.59\times$. In contrast, for a fixed drafter, Predraw and TEV differ by at most 0.03 tokens per benchmark, consistent with equal expected output-block length under Theorem 1; the small empirical discrepancies arise from separate finite-sample decoding runs. TEV nevertheless improves average end-to-end SR by 0.4% with the original drafter and 0.7% with the exit-trained drafter. Together, these results separate the two contributions: ExitTrain supplies most of the acceptance gain by improving the tree, while TEV provides orthogonal execution savings, reducing verifier-decision and cache-commit latency by 12.0%–24.3% across the tested budgets (Appendix D.1).

6.2 Ablation Study

Exit-guided training. We compare the released DFlash checkpoint with three variants trained under matched initialization, data, and compute: token-only finetuning, prefix-weighted training, and exit-guided training. All use the same builder and $B = 64$. The prefix-weighted control replaces $w(v)$ with $P(v)$, rescaled to match the total non-root exit weight while retaining the same loss denominator. The two controls therefore separate the benefit of additional training from the benefit of assigning credit specifically to exit boundaries.

Table 2 first shows that additional training is itself useful: token-only finetuning increases average τ from 7.80/7.08 to 8.36/7.47 at $T{=}0/1$. Exit-guided training then improves over this matched control by a further 5.9%/7.9% in τ and 6.4%/6.3% in SR, demonstrating that its gain is not explained by additional training on `open-perfectblend` alone. The SR numbers compare the complete systems; the crossed evaluation in Sec. 6.1 shows that TEV contributes only 0.7% end-to-end with the exit-trained drafter, so the dominant improvement comes from training.

Table 2: Effect of the draft-training objective on Qwen3-4B. All three trained drafters share the initialization, the `open-perfectblend` mixture, the compute budget, the tree builder, and the budget $B = 64$

	MT-Bench		HumanEval		GSM8K		MATH-500		Average	
Draft training objective	$SR\uparrow$	$\tau\uparrow$	$SR\uparrow$	$\tau\uparrow$	$SR\uparrow$	$\tau\uparrow$	$SR\uparrow$	$\tau\uparrow$	$SR\uparrow$	$\tau\uparrow$
Temperature = 0										
Original DFlash drafter	3.81	5.41	5.71	8.13	5.65	8.48	6.83	9.16	5.50	7.80
Finetuned DFlash drafter	3.97	5.71	5.83	8.36	6.33	9.54	6.91	9.83	5.76	8.36
Prefix weight $P(v)$	4.05	5.82	5.99	8.43	6.07	9.25	6.71	9.33	5.71	8.21
Exit weight $w(v)$ (ours)	**4.28**	**6.17**	**6.09**	**8.74**	**6.91**	**10.22**	**7.23**	**10.26**	**6.13**	**8.85**
Temperature = 1										
Original DFlash drafter	3.44	5.04	5.22	7.53	5.48	7.85	5.65	7.88	4.95	7.08
Finetuned DFlash drafter	3.58	5.20	5.43	7.63	6.18	8.58	5.86	8.46	5.26	7.47
Prefix weight $P(v)$	3.69	5.45	5.48	7.88	6.26	8.85	5.99	8.68	5.35	7.72
Exit weight $w(v)$ (ours)	**3.98**	**5.86**	**5.64**	**8.06**	**6.47**	**9.25**	**6.26**	**9.08**	**5.59**	**8.06**

The prefix-weighted control tests whether target reachability alone provides sufficient tree supervision. It reaches $\tau = 8.21/7.72$, whereas exit weighting reaches 8.85/8.06, improvements of 7.8%/4.4%; the corresponding SR gains are 7.4%/4.5%. Moreover, prefix weighting trails token-only finetuning at T=0, despite using target prefix probabilities. The distinction is that $P(v)$ measures how often a node is reached, whereas $w(v) = P(v)\rho(v)$ additionally identifies reached boundaries whose selected children leave substantial target mass uncovered (Eq. (13)). The results therefore support localized exit feedback rather than reachability weighting alone.

Verifier-stage efficiency. Appendix D.1 and Table 3 isolate verifier execution by holding the drafter, builder, and realized tree fixed and excluding the shared target-model tree forward pass. TEV reduces verifier-decision and cache-commit latency at every tested budget: 12.9%/12.0% at $B = 32$, 15.2%/14.0% at $B = 64$, and 24.3%/17.1% at $B = 128$ for $T = 0.6/1.0$. The increasing advantage at wider trees is consistent with replacing Predraw's per-node sampling with level-wise probability propagation and one exit-node draw. Because this stage occupies only a small fraction of each decoding cycle, these reductions translate into the smaller end-to-end gains observed in the crossed evaluation.

Tree-budget robustness. Appendix D.3 and Table 5 evaluate the $B = 64$-trained drafter at $B \in \{32, 64, 128\}$ without budget-specific retraining. ExitTrain + TEV remains best in all six budget–temperature settings: relative to DDTree, it improves τ by 11.6%–13.8% and SR by 11.4%–12.9%; relative to Draft-OPD, the gains are 5.6%–6.9% and 4.5%–6.2%. Its advantage therefore persists for both narrower and wider inference trees, indicating that the learned exit-guided allocation is not tied to the training-time tree size.

7 Conclusion

We study tree speculative decoding through target probability flow and show that the maximum expected output-block length on a fixed tree is one plus its target-mass coverage. The canonical exit law gives a verifier-independent ceiling: saturating exact verifiers share the same exit and bonus-token law, so missing support cannot be recovered by verification alone. This suggests two complementary directions. TEV realizes the optimal law with a regular, level-parallel procedure to reduce verification cost, while exit-guided training uses node-level exit signals to improve finite-budget coverage and output-block length. Overall, our framework separates *what tree to build* from *how to verify it efficiently*.

Limitations. ExitTrain incurs one-time training overhead from target rollouts and tree evaluation, but fine-tunes released DFlash checkpoints without architectural or verifier changes and can be amortized during deployment. Our systems results are limited to batch-one decoding with DDTree; alternative builders and high-throughput serving remain future work.

AI USE STATEMENT

Generative AI tools were used solely for language polishing and improving the clarity and readability of the manuscript. The authors reviewed all AI-assisted edits and take full responsibility for the final content of this work.

ETHICS STATEMENT

This work studies the efficiency of lossless large language model inference. It does not involve human subjects, private user data, or the collection of new personal information. Our experiments use publicly available models and benchmarks under their respective terms of use. By construction, TEV preserves the output distribution of the target model and does not intentionally modify its capabilities, safety behavior, or intended use. More efficient inference may reduce computational and energy costs, but may also lower the cost of deploying language models at scale. Consequently, downstream uses of our method inherit the potential safety, bias, privacy, and misuse risks of the underlying target models. Users remain responsible for evaluating these risks and complying with the licenses and usage policies of the deployed models and datasets.

REPRODUCIBILITY STATEMENT

Sections 3–5 provide the definitions, assumptions, fixed-tree results, TEV construction, and exit-guided training objective used in this work. Complete proofs of the fixed-tree ceiling, verifier equivalence, and TEV exactness are provided in Appendices A and B. Additional derivations and implementation details for the exit-guided objective are given in Appendix C, while the model, dataset, training, hardware, and evaluation configurations are documented in Appendix D. An anonymized repository at https://anonymous.4open.science/r/TEV-ICLR-348C/ will provide the training and inference code, configuration files, evaluation scripts, and instructions required to reproduce the reported experiments.

REFERENCES

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. Medusa: Simple LLM inference acceleration framework with multiple decoding heads. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 5209–5235. PMLR, 2024. 3

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. URL https://arxiv.org/abs/2302.01318. 1

Jian Chen, Yesheng Liang, and Zhijian Liu. DFlash: Block diffusion for flash speculative decoding. In *International Conference on Machine Learning*, 2026. URL https://icml.cc/virtual/2026/poster/64301. 1, 3, 7

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. URL https://arxiv.org/abs/2107.03374. 7

Zhuoming Chen, Avner May, Ruslan Svirschevski, Yuhsun Huang, Max Ryabinin, Zhihao Jia, and Beidi Chen. Sequoia: Scalable and robust speculative decoding. In *Advances in Neural Information Processing Systems*, volume 37, pp. 129531–129563, 2024. 1, 3

Yunfei Cheng, Aonan Zhang, Xuanyu Zhang, Chong Wang, and Yi Wang. Recurrent drafter for fast speculative decoding in large language models. *arXiv preprint arXiv:2403.09919*, 2024. URL https://arxiv.org/abs/2403.09919. 3

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. *arXiv preprint arXiv:2110.14168*, 2021. URL https://arxiv.org/abs/2110.14168. 7

Gemma Team. Gemma 4 technical report, 2026. URL https://arxiv.org/abs/2607.02770. 7

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. URL https://arxiv.org/abs/2407.21783. 7

Geonmo Gu, Byeongho Heo, HeeJae Jun, Yoohoon Kang, Sangmin Lee, Sangdoo Yun, and Dongyoon Han. Verification-aware training for speculative decoding. *arXiv preprint arXiv:2608.30135*, 2026. URL https://arxiv.org/abs/2608.30135. 3

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021. 7

Shijing Hu, Jingyang Li, Xingyu Xie, Zhihui Lu, Kim-Chuan Toh, and Pan Zhou. GRIFFIN: Effective token alignment for faster speculative decoding. In *Advances in Neural Information Processing Systems*, volume 38, pp. 125648–125673, 2025. 2, 3

Shijing Hu, Jingyang Li, Zhihui Lu, and Pan Zhou. Bridging draft policy misalignment: Group tree optimization for speculative decoding. In *International Conference on Learning Representations*, 2026. URL https://iclr.cc/virtual/2026/poster/10008369. 3

Haodi Lei, Yafu Li, Haoran Zhang, Shunkai Zhang, Qianjia Cheng, Xiaoye Qu, Ganqu Cui, Bowen Zhou, Ning Ding, Yun Luo, and Yu Cheng. Draft-opd: On-policy distillation for speculative draft models. *arXiv preprint arXiv:2605.29343*, 2026. URL https://arxiv.org/abs/2605.29343. 3, 7

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 19274–19286. PMLR, 2023. 1

Tianyi Li, Yaxin Luo, Xinyi Shang, and Zhiqiang Shen. DARTree: Speculative diffusion decoding with autoregressive draft trees. *arXiv preprint arXiv:2608.13524*, 2026. URL https://arxiv.org/abs/2608.13524. 3

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE: Speculative sampling requires rethinking feature uncertainty. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 28935–28948. PMLR, 2024a. 3

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE-2: Faster inference of language models with dynamic draft trees. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 7421–7432. Association for Computational Linguistics, 2024b. doi: 10.18653/v1/2024.emnlp-main.422. 1, 3

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE-3: Scaling up inference acceleration of large language models via training-time test. In *Advances in Neural Information Processing Systems*, volume 38, pp. 136737–136756, 2025. 1, 2, 3, 5

Xiaoxuan Liu, Lanxiang Hu, Peter Bailis, Alvin Cheung, Zhijie Deng, Ion Stoica, and Hao Zhang. Online speculative decoding. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 31131–31146. PMLR, 2024. 3

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*, pp. 932–949, 2024. doi: 10.1145/3620666.3651335. 1, 3

Liran Ringel and Yaniv Romano. Accelerating speculative decoding with block diffusion draft trees. *arXiv preprint arXiv:2604.12989*, 2026. URL https://arxiv.org/abs/2604.12989. 1, 2, 3, 5, 7

Ruslan Svirschevski, Avner May, Zhuoming Chen, Beidi Chen, Zhihao Jia, and Max Ryabinin. Specexec: Massively parallel speculative decoding for interactive LLM inference on consumer devices. In *Advances in Neural Information Processing Systems*, volume 37, pp. 16342–16368, 2024. 3

Jikai Wang, Yi Su, Juntao Li, Qingrong Xia, Zi Ye, Xinyu Duan, Zhefeng Wang, and Min Zhang. Opt-tree: Speculative decoding with adaptive draft tree structure. *Transactions of the Association for Computational Linguistics*, 13:188–199, 2025. 3

Tengyu Xu, Eryk Helenowski, Karthik Abinav Sankararaman, Di Jin, Kaiyan Peng, Eric Han, Shaoliang Nie, Chen Zhu, Hejia Zhang, Wenxuan Zhou, Zhouhao Zeng, Yun He, Karishma Mandyam, Arya Talabzadeh, Madian Khabsa, Gabriel Cohen, Yuandong Tian, Hao Ma, Sinong Wang, and Han Fang. The perfect blend: Redefining RLHF with mixture of judges, 2024. URL https://arxiv.org/abs/2409.20370. 7, 21

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388. 7

Jiebin Zhang, Zhenghan Yu, Liang Wang, Nan Yang, Eugene Yu, Zheng Li, Yifan Song, Dawei Zhu, Xingxing Zhang, Furu Wei, et al. Learning to draft: Adaptive speculative decoding with reinforcement learning. In *International Conference on Learning Representations*, volume 2026, pp. 30798–30819, 2026. 3

Lefan Zhang, Xiaodan Wang, Yanhua Huang, and Ruiwen Xu. Learning harmonized representations for speculative sampling. In *International Conference on Learning Representations*, 2025. URL https://iclr.cc/virtual/2025/poster/29560. 2, 3

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-Bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36, pp. 46595–46623, 2023. 7

Yongchao Zhou, Kaifeng Lyu, Ankit Singh Rawat, Aditya Krishna Menon, Afshin Rostamizadeh, Sanjiv Kumar, Jean-François Kagy, and Rishabh Agarwal. Distillspec: Improving speculative decoding via knowledge distillation. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=rsY6J3ZaTF. 3

Yuxuan Zhou, Fei Huang, Heng Li, Fengyi Wu, Tianyu Wang, Jianwei Zhang, Junyang Lin, and Zhi-Qi Cheng. Overcoming joint intractability with lossless hierarchical speculative decoding. In *International Conference on Learning Representations*, 2026. URL https://www.iclr.cc/virtual/2026/poster/10010023. 2, 3, 5, 17

Xiandong Zou, Jianshu Li, Jing Huang, and Pan Zhou. Variational speculative decoding: Rethinking draft training from token likelihood to sequence acceptance. In *International Conference on Machine Learning*, 2026. URL https://icml.cc/virtual/2026/poster/66012. 3

A Detailed Theory for Fixed-Tree Verification

This appendix provides the technical details for Section 3. Unless stated otherwise, we condition on a decoding context c and a fixed, finite, rooted, prefix-closed draft tree $\mathcal{T} = (\mathcal{V}, \mathcal{E}, r)$. Sibling edges carry distinct draft tokens. For each node v, p_v is its target next-token distribution, $P(v)$ is its target prefix probability, $\rho(v)$ is its local exit probability, and $w(v) = P(v)\rho(v)$ is its exit probability.

A.1 Random trees, auxiliary randomness, and round-wise composition

The results in Section 3 are stated conditional on a realized draft tree, so the tree builder may itself be deterministic or stochastic. Let $Q(\cdot \mid c)$ denote the conditional law of $\mathcal{T}$, which is constructed before verification, and let $R(\cdot \mid c, \mathcal{T})$ denote the conditional law of auxiliary control randomness U available at time zero. This U may randomize stopping or control decisions but excludes randomness represented by the coupled target continuation. For each realized pair $(\mathcal{T}, U)$, suppose there exists a measurable conditional coupling $\Gamma_{\mathcal{T},U}$ of the output block O, a target continuation Z, and its length $L = |O|$ such that

$$Z \mid (c, \mathcal{T}, U) \sim p(\cdot \mid c), \qquad O = Z_{1:L} \quad \Gamma_{\mathcal{T},U}\text{-almost surely}, \tag{16}$$

and L is a stopping time with respect to $\mathcal{H}_t = \sigma(\mathcal{T}, U, Z_1, \ldots, Z_t)$. Construct the joint coupling by first drawing $\mathcal{T} \sim Q(\cdot \mid c)$, then $U \sim R(\cdot \mid c, \mathcal{T})$, and finally (O, Z, L) from $\Gamma_{\mathcal{T},U}$. For any measurable set A of target continuations,

$$\Pr(Z \in A \mid c) = \mathbb{E}_{\mathcal{T},U|c}[\Pr(Z \in A \mid c, \mathcal{T}, U)] \tag{17}$$

$$= p(A \mid c). \tag{18}$$

Moreover, $O = Z_{1:L}$ almost surely, and $\{L \le t\}$ remains $\mathcal{H}_t$-measurable. Thus, conditional non-anticipating exactness is preserved after marginalizing over random tree construction and auxiliary randomness. In our setting, neither the drafter-constructed tree nor U has access to future target randomness.

Lemma 2 (Round-wise composition)**.** *At round k, let c_k be the current context, $\mathcal{T}_k$ the realized tree, U_k the auxiliary control randomness, $Z^{(k)}$ the coupled target continuation, and L_k the output-block length, so $O_k = Z^{(k)}_{1:L_k}$. Suppose each round satisfies Eq. (16) with these round-indexed quantities and L_k is a stopping time with respect to $\mathcal{H}_{k,t} := \sigma(\mathcal{T}_k, U_k, Z^{(k)}_1, \ldots, Z^{(k)}_t)$. Then repeatedly appending the returned blocks preserves the autoregressive distribution induced by p.*

Proof. For a realized round-k output length $L_k = \ell$, the event $\{L_k = \ell\}$ is measurable with respect to $\mathcal{H}_{k,\ell}$ and therefore depends only on $(\mathcal{T}_k, U_k, Z^{(k)}_{1:\ell})$, not on $Z^{(k)}_{\ell+1:}$. Because $Z^{(k)} \mid (c_k, \mathcal{T}_k, U_k) \sim p(\cdot \mid c_k)$, conditioning on the emitted block $O_k = Z^{(k)}_{1:\ell}$ and on the stopping event does not change the target law of the unrevealed continuation beyond conditioning on the emitted prefix itself. Restarting round $k+1$ from the updated context therefore uses exactly the same conditional law as ordinary target decoding. Induction over k proves the claim. □

A.2 Exit partition, probability conservation, and target coverage

Proof of Lemma 1. Let $Z = (Z_1, Z_2, \ldots) \sim p(\cdot \mid c)$. Because $\mathcal{T}$ is finite and prefix-closed, there is a unique deepest node $V^\star(Z)$ such that $\pi(V^\star(Z)) \preceq Z$. If $V^\star(Z)$ is internal, the next target token cannot equal the draft token of any child; otherwise the continuation would match a deeper node. If $V^\star(Z)$ is a leaf, it is automatically the deepest matching node. Hence the events

$$E_v := \{V^\star(Z) = v\}, \qquad v \in \mathcal{V}, \tag{19}$$

are disjoint and exhaustive.

The continuation reaches v with probability

$$\Pr(\pi(v) \preceq Z \mid c, \mathcal{T}) = p(\pi(v) \mid c) = P(v). \tag{20}$$

Conditioned on reaching v, its next token is outside the drafted child-token set $\mathcal{A}(v)$ with probability

$$1 - \sum_{u \in \mathrm{Ch}(v)} p_v(x_u) = \rho(v). \tag{21}$$

Therefore,

$$\Pr(E_v \mid c, \mathcal{T}) = P(v)\rho(v) = w(v). \tag{22}$$

Since the events $\{E_v\}_{v\in\mathcal{V}}$ form a partition,

$$\sum_{v\in\mathcal{V}} w(v) = 1. \tag{23}$$

□

The exit probabilities also satisfy a local probability-conservation identity:

$$w(v) + \sum_{u\in\mathrm{Ch}(v)} P(u) = P(v)\rho(v) + \sum_{u\in\mathrm{Ch}(v)} P(v)p_v(x_u) \tag{24}$$

$$= P(v). \tag{25}$$

Thus, target probability reaching v either exits at v or continues through one drafted child. Summing Equation 25 over the subtree rooted at a node u cancels every descendant prefix probability and gives

$$\sum_{v\succeq u} w(v) = P(u), \tag{26}$$

where $v \succeq u$ means that v lies in the subtree rooted at u.

Finally, $\mathrm{d}(v)$ equals the number of non-root nodes on the path from the root to v. Using Equation 26 and exchanging the order of summation,

$$\sum_{v\in\mathcal{V}} w(v)\mathrm{d}(v) = \sum_{v\in\mathcal{V}} w(v) \sum_{u\in\mathcal{V}\setminus\{r\}} \mathbf{1}[v \succeq u] \tag{27}$$

$$= \sum_{u\in\mathcal{V}\setminus\{r\}} \sum_{v\succeq u} w(v) \tag{28}$$

$$= \sum_{u\in\mathcal{V}\setminus\{r\}} P(u). \tag{29}$$

This proves the two equivalent forms of target coverage in Equation 6.

A.3 Proof of the fixed-tree ceiling and canonical saturation

Proof of Theorem 1. Let A be an exact path verifier. Under the coupling defined in Sec. 3, let $Z \sim p(\cdot \mid c)$ be the target continuation and let the verifier output be $O = Z_{1:L}$. The ceiling argument uses only this target-prefix identity, not the stopping-time property. For every non-root node u, define

$$A_u := \{u \text{ is included in the accepted draft path}\}, \tag{30}$$

$$F_u := \{\pi(u) \preceq Z\}. \tag{31}$$

Whenever A_u occurs, the verifier output begins with $\pi(u)$ and is a prefix of Z; therefore $A_u \subseteq F_u$ almost surely. Consequently,

$$\Pr_{\mathsf{A}}(A_u \mid c, \mathcal{T}) \le \Pr(F_u \mid c, \mathcal{T}) = P(u), \tag{32}$$

which proves Equation 8.

An accepted root-to-node path contains exactly one non-root node for every accepted draft token, so

$$D_{\mathsf{A}} = \sum_{u\in\mathcal{V}\setminus\{r\}} \mathbf{1}[A_u]. \tag{33}$$

Taking expectations and applying Equation 32 gives

$$\mathbb{E}[D_{\mathsf{A}} \mid c, \mathcal{T}] = \sum_{u\neq r} \Pr_{\mathsf{A}}(A_u \mid c, \mathcal{T}) \tag{34}$$

$$\le \sum_{u\neq r} P(u) = C_p(\mathcal{T}). \tag{35}$$

Since $L_{\mathsf{A}} = D_{\mathsf{A}} + 1$, we also have $\mathbb{E}[L_{\mathsf{A}} \mid c, \mathcal{T}] \le 1 + C_p(\mathcal{T})$, proving Equation 9.

Now suppose that A is saturated. For each non-root node, let

$$\delta_u := P(u) - \Pr_{\mathsf{A}}(A_u \mid c, \mathcal{T}) \ge 0. \tag{36}$$

Saturation implies $\sum_{u \neq r} \delta_u = 0$, and hence every $\delta_u = 0$. Thus

$$\Pr_{\mathsf{A}}(A_u \mid c, \mathcal{T}) = P(u) \qquad \text{for every } u \neq r. \tag{37}$$

Since $A_u \subseteq F_u$ and both events have the same probability, $A_u = F_u$ almost surely. Because the draft tree is finite, these equalities hold simultaneously for all nodes outside a single null event.

Let $A_r = F_r$ be the full sample space. The accepted path terminates at v exactly when v is accepted and none of its children is accepted. The child acceptance events are disjoint, so

$$\Pr_{\mathsf{A}}(V = v \mid c, \mathcal{T}) = \Pr_{\mathsf{A}}(A_v \mid c, \mathcal{T}) - \sum_{u \in \mathrm{Ch}(v)} \Pr_{\mathsf{A}}(A_u \mid c, \mathcal{T}) \tag{38}$$

$$= P(v) - \sum_{u \in \mathrm{Ch}(v)} P(u) \tag{39}$$

$$= P(v)\rho(v) = w(v). \tag{40}$$

Moreover, the simultaneous identities $A_u = F_u$ imply that V equals the target-induced exit node $V^\star(Z)$ almost surely. Since the verifier output satisfies $O = \pi(V) \,\|\, Y = Z_{1:L}$ with $L = \mathrm{d}(V) + 1$, Y is the target token immediately after $\pi(V)$. Therefore, for every internal node v with $w(v) > 0$,

$$\Pr(Y = a \mid V = v, c, \mathcal{T}) = \frac{p_v(a)\mathbf{1}[a \notin \mathcal{A}(v)]}{\rho(v)} = p_v^{\mathrm{res}}(a). \tag{41}$$

At a leaf, $\mathcal{A}(v) = \emptyset$ and $\rho(v) = 1$, so the conditional law is p_v. This proves Equation 10 and the uniqueness of the accepted-path and bonus-token law at saturation. □

A.4 Proof that the representative verifiers are saturated

We consider the two target-local procedures defined in Section 2. Both operate on the same fixed draft tree and the same target distributions $\{p_v\}$. Sibling draft tokens are distinct. For the target-induced flow, define $L^\star := \mathrm{d}(V^\star(Z)) + 1$. This is the first index at which the matched node is a leaf or $Z_{L^\star}$ is not a drafted child token, so $\{L^\star \le t\}$ is determined by $Z_{1:t}$ and $L^\star$ is a stopping time. We show that each procedure realizes the joint target-induced exit and bonus-token law; coupling its output to $Z_{1:L^\star} = \pi(V^\star(Z)) \,\|\, Y^\star(Z)$ therefore establishes non-anticipating exactness as well as saturation.

A.4.1 Predraw-and-follow

At every node v, predraw-and-follow independently draws a token $R_v \sim p_v$. Starting from the root, it follows child $u \in \mathrm{Ch}(v)$ if $R_v = x_u$; otherwise it exits at v and returns R_v as the bonus token.

Fix a node v and write its root-to-node path as $r = v_0, v_1, \ldots, v_d = v$. The procedure reaches v exactly when $D_{v_{i-1}} = x_{v_i}$ for all $i = 1, \ldots, d$. Hence

$$\Pr(\text{reach } v \mid c, \mathcal{T}) = \prod_{i=1}^{d} p_{v_{i-1}}(x_{v_i}) = P(v). \tag{42}$$

Conditioned on reaching v, the independent draw R_v is outside the drafted child-token set with probability

$$1 - \sum_{u \in \mathrm{Ch}(v)} p_v(x_u) = \rho(v). \tag{43}$$

Therefore,

$$\Pr(\text{exit at } v \mid c, \mathcal{T}) = P(v)\rho(v) = w(v). \tag{44}$$

Conditioned on an internal exit, the bonus token follows p_v^{res}; at a leaf it follows p_v. Thus the accepted-path and bonus-token law matches the target-induced law, so predraw-and-follow is exact. Its expected accepted draft-token count is $\sum_v w(v)\mathrm{d}(v) = C_p(\mathcal{T})$, so its expected output-block length is $1 + C_p(\mathcal{T})$ and it is saturated.

A.4.2 EXACT SEQUENTIAL RESIDUAL VERIFICATION

Fix an internal node v reached by the verifier. Let its drafted children be tested in an arbitrary order $u_1, \dots, u_k$, with draft tokens $x_1, \dots, x_k$. Define

$$p_j := p_v(x_j), \qquad M_j := \sum_{\ell=1}^{j} p_\ell, \qquad M_0 := 0. \tag{45}$$

Let $g^{(0)} := p_v$. If child u_j is rejected and the rejection event has positive probability, exact residual renormalization removes token x_j:

$$g^{(j)}(a) := \frac{g^{(j-1)}(a)\mathbf{1}[a \neq x_j]}{1 - g^{(j-1)}(x_j)}. \tag{46}$$

At step j, the verifier follows u_j with conditional probability $g^{(j-1)}(x_j)$. If all children are rejected, it draws the bonus token from $g^{(k)}$. A leaf has no child tests and returns a bonus token from p_v.

For every j with $M_j < 1$, induction gives

$$g^{(j)}(a) = \frac{p_v(a)}{1 - M_j}, \qquad a \notin \{x_1, \dots, x_j\}. \tag{47}$$

Indeed, assuming the identity at $j-1$,

$$g^{(j)}(a) = \frac{p_v(a)/(1 - M_{j-1})}{1 - p_j/(1 - M_{j-1})} = \frac{p_v(a)}{1 - M_j}. \tag{48}$$

If $M_j = 1$, rejecting the first j children has probability zero, so no later residual state affects the output.

Before child u_j is tested,

$$g^{(j-1)}(x_j) = \frac{p_j}{1 - M_{j-1}}. \tag{49}$$

The probability of rejecting the first $j-1$ children telescopes to

$$\prod_{i=1}^{j-1} \left(1 - g^{(i-1)}(x_i)\right) = \prod_{i=1}^{j-1} \frac{1 - M_i}{1 - M_{i-1}} = 1 - M_{j-1}. \tag{50}$$

Thus, the unconditional probability of following child u_j is

$$(1 - M_{j-1}) \frac{p_j}{1 - M_{j-1}} = p_j = p_v(x_j). \tag{51}$$

Similarly, the probability of rejecting every child is

$$\prod_{i=1}^{k} \frac{1 - M_i}{1 - M_{i-1}} = 1 - M_k = \rho(v), \tag{52}$$

and, whenever $\rho(v) > 0$, the final residual target distribution is

$$g^{(k)}(a) = \frac{p_v(a)\mathbf{1}[a \notin \mathcal{A}(v)]}{\rho(v)} = p_v^{\mathrm{res}}(a). \tag{53}$$

Hence the local transition kernel is independent of the order of the drafted children: each child u is followed with probability $p_v(x_u)$, while the remaining probability $\rho(v)$ exits with bonus-token law p_v^{res}. Multiplying the child-transition probabilities along a root-to-v path gives reach probability $P(v)$ and exit probability $w(v)$. Thus the accepted-path and bonus-token law matches the target-induced law, so the verifier is exact. Its expected accepted draft-token count is $C_p(\mathcal{T})$, its expected output-block length is $1 + C_p(\mathcal{T})$, and hence it is saturated.

Algorithm 1 Tree Exit Verification (TEV) for speculative verification

Require: Fixed tree $\mathcal{T} = (\mathcal{V}, \mathcal{E}, r)$ and target distributions $\{p_v\}_{v \in \mathcal{V}}$
Ensure: Accepted path followed by one bonus token
$P(r) \leftarrow 1$
for each depth $d = 0, 1, \ldots$ **do**
  for all v at depth d **in parallel do**
    $\rho(v) \leftarrow 1 - \sum_{u \in \mathrm{Ch}(v)} p_v(x_u)$; $w(v) \leftarrow P(v)\rho(v)$
    $P(u) \leftarrow P(v)p_v(x_u)$ for all $u \in \mathrm{Ch}(v)$
Draw $V \sim \mathrm{Categorical}(\{w(v)\}_{v \in \mathcal{V}})$
Draw $Y \sim p_V$ if V is a leaf, and otherwise $Y \sim p_V^{\mathrm{res}}$
return $\pi(V) \,\|\, Y$

A.5 Why the fixed-tree result does not contradict HSD

The fixed-tree ceiling applies after a draft tree has already been realized. HSD (Zhou et al., 2026) studies a different probability structure: a draft chain $x_{1:k}$ is sampled from draft distributions $q_1, \ldots, q_k$, and verification retains the target-to-draft ratios

$$r_t := \frac{p_t(x_t)}{q_t(x_t)}. \tag{54}$$

For a realized chain, position-wise speculative verification accepts the full chain with probability

$$\prod_{t=1}^{k} \min\{1, r_t\} \le \min\left\{1, \prod_{t=1}^{k} r_t\right\}. \tag{55}$$

When ratios above and below one occur along the same chain, position-wise clipping prevents surplus at one position from compensating for a deficit at another; hierarchical chain verification can recover part of this joint compensation.

Our analysis instead conditions on a realized draft tree. At that point, the draft model affects verification through which draft tokens are present in the tree, while the fixed-tree ceiling follows from target prefix events. Even if a verifier also consults draft scores, no exact path verifier can accept a draft token absent from the realized tree. The two results therefore address different stages of speculative decoding and are complementary rather than contradictory.

B Correctness of Tree Exit Verification

B.1 Proof of exactness and saturation of TEV

Proof of Corollary 2. Fix a context c and a realized finite, prefix-closed draft tree $\mathcal{T}$. Let $Z \sim p(\cdot \mid c)$, let $V^\star(Z)$ be its target-induced exit node, and let $Y^\star(Z)$ be the target token immediately following $\pi(V^\star(Z))$.

By Lemma 1, $\Pr(V^\star(Z) = v \mid c, \mathcal{T}) = w(v)$. Conditioned on an internal exit at v with $w(v) > 0$, the next target token lies outside $\mathcal{A}(v)$ and therefore follows p_v^{res}; conditioned on a leaf exit, it follows p_v. These are exactly the two distributions used by Algorithm 1. Hence

$$(V, Y)_{\mathrm{TEV}} \overset{d}{=} \big(V^\star(Z), Y^\star(Z)\big). \tag{56}$$

We may therefore couple the two pairs to be equal almost surely. Under this coupling, set $L := \mathrm{d}(V) + 1 = \mathrm{d}(V^\star(Z)) + 1$. This is the first index at which the matched node is a leaf or the next target token leaves the drafted child set. Hence $\{L \le t\}$ is determined by $Z_{1:t}$ and is therefore measurable with respect to $\mathcal{H}_t = \sigma(\mathcal{T}, U, Z_{1:t})$. Moreover,

$$O = \pi(V) \,\|\, Y = Z_{1:L} \qquad \text{almost surely.} \tag{57}$$

Thus TEV satisfies the non-anticipating exactness definition (without using any additional early-stopping randomness U).

For any non-root node u, TEV accepts u exactly when its exit node lies in the subtree rooted at u. Using Equation 26,

$$\Pr_{\text{TEV}}(u \text{ is accepted} \mid c, \mathcal{T}) = \sum_{v \succeq u} w(v) = P(u). \tag{58}$$

Thus TEV attains every node-wise bound in Theorem 1. Finally,

$$\mathbb{E}[D_{\text{TEV}} \mid c, \mathcal{T}] = \sum_{u \in \mathcal{V} \setminus \{r\}} \Pr_{\text{TEV}}(u \text{ is accepted} \mid c, \mathcal{T}) \tag{59}$$

$$= \sum_{u \in \mathcal{V} \setminus \{r\}} P(u) = C_p(\mathcal{T}), \tag{60}$$

$$\mathbb{E}[L_{\text{TEV}} \mid c, \mathcal{T}] = 1 + C_p(\mathcal{T}). \tag{61}$$

Therefore TEV attains the fixed-tree ceiling and is saturated. If the draft tree is random, the same conclusion follows by conditioning on the realized tree and applying Appendix A.1. □

C Additional Details for Exit-Guided Training

C.1 One-step target-coverage gain

Let v be an expandable node and let $a \notin \mathcal{A}(v)$ be a missing draft token. Write $u = (v, a)$ for the new child and $\mathcal{T}^+ = \mathcal{T} \oplus (v, a)$ for the tree obtained by adding only this node. The target prefix probabilities of all existing nodes are unchanged, and the new node has

$$P(u) = P(v)p_v(a). \tag{62}$$

Therefore, by the definition of target coverage,

$$C_p(\mathcal{T}^+) - C_p(\mathcal{T}) = P(u) = P(v)p_v(a), \tag{63}$$

which proves Equation 12. Summing over every missing child token gives

$$\sum_{a \notin \mathcal{A}(v)} P(v)p_v(a) = P(v)\left(1 - \sum_{u \in \text{Ch}(v)} p_v(x_u)\right) \tag{64}$$

$$= P(v)\rho(v) = w(v), \tag{65}$$

which proves Equation 13. This is a gross one-step gain before enforcing the fixed tree budget; a budget-preserving update may also need to remove another node.

C.2 Efficient evaluation of the exit-guided loss

Let $\mathcal{N}$ denote the multiset of non-root draft-tree nodes in a minibatch, let $c(v)$ denote the training context associated with node v, and let the detached weight be $\omega_v = w(v)$. Suppose that the incoming draft token x_v is scored by the DFlash position-wise draft-logit row $z_{\text{row}(v)} \in \mathbb{R}^{|\mathcal{X}|}$, where $\text{row}(v)$ indexes the pair consisting of v's minibatch example and future position $\text{d}(v)$. Within one training example, this row is shared by all tree nodes at the same depth and is not conditioned on $\pi(\text{pa}(v))$. Then

$$\sum_{v \in \mathcal{N}} \omega_v \left[-\log q_{\theta, \text{row}(v)}(x_v \mid c(v))\right]$$

$$= \sum_{v \in \mathcal{N}} \omega_v \left[\text{logsumexp}\big(z_{\text{row}(v)}\big) - z_{\text{row}(v), x_v}\right] \tag{66}$$

$$= \sum_{m} \left(\sum_{v \in \mathcal{N}:\, \text{row}(v) = m} \omega_v\right) \text{logsumexp}(z_m) - \sum_{v \in \mathcal{N}} \omega_v z_{\text{row}(v), x_v}. \tag{67}$$

Thus, each distinct draft-logit row requires only one log-normalizer, and the second term uses a gather over the selected draft-token logits. The minibatch normalization in Equation 14 is applied after this numerator is computed. This decomposition reuses the batched draft-model language head and avoids materializing a separate normalized vocabulary distribution for every selected tree node.

Table 3: Verifier-stage latency per decoding cycle on Qwen3-4B across tree budgets. The measured interval covers verifier decision and cache commit and excludes the shared target-model tree forward pass. For each B, both verifiers use the same drafter, builder, and realized trees.

		MT-Bench		HumanEval		GSM8K		MATH-500		Average	
B	**Verifier**	ms↓	red.	ms↓	red.	ms↓	red.	ms↓	red.	ms↓	red.
Temperature = 0.6											
32	Predraw	1.371	—	1.435	—	1.428	—	1.393	—	1.407	—
	TEV	**1.204**	**12.2%**	**1.245**	**13.2%**	**1.239**	**13.2%**	**1.211**	**13.0%**	**1.225**	**12.9%**
64	Predraw	1.392	—	1.487	—	1.443	—	1.488	—	1.453	—
	TEV	**1.212**	**12.9%**	**1.268**	**14.7%**	**1.193**	**17.3%**	**1.257**	**15.5%**	**1.232**	**15.2%**
128	Predraw	1.620	—	1.719	—	1.681	—	1.616	—	1.659	—
	TEV	**1.246**	**23.1%**	**1.265**	**26.4%**	**1.259**	**25.1%**	**1.248**	**22.8%**	**1.254**	**24.3%**
Temperature = 1.0											
32	Predraw	1.337	—	1.368	—	1.348	—	1.411	—	1.366	—
	TEV	**1.187**	**11.2%**	**1.235**	**9.7%**	**1.184**	**12.1%**	**1.197**	**15.2%**	**1.201**	**12.0%**
64	Predraw	1.416	—	1.428	—	1.398	—	1.442	—	1.421	—
	TEV	**1.200**	**15.3%**	**1.240**	**13.2%**	**1.207**	**13.7%**	**1.240**	**14.0%**	**1.222**	**14.0%**
128	Predraw	1.549	—	1.545	—	1.535	—	1.509	—	1.534	—
	TEV	**1.266**	**18.3%**	**1.276**	**17.3%**	**1.269**	**17.3%**	**1.277**	**15.4%**	**1.272**	**17.1%**

D Additional Experimental Details

D.1 Verifier-stage efficiency

For each budget $B \in \{32, 64, 128\}$, we hold the drafter, builder, and realized tree fixed across verifiers and time only verifier decision and cache commit, excluding the shared target-model tree forward pass. Corollary 1 guarantees identical fixed-tree expected acceptance for Predraw and TEV at every budget.

TEV reduces verifier-stage latency on every benchmark and budget. Averaged over tasks, the reduction grows from 12.9% to 15.2% to 24.3% as B increases from 32 to 64 to 128 at $T = 0.6$; the corresponding reductions at $T = 1.0$ are 12.0%, 14.0%, and 17.1%. Because the drafter, realized tree, and target forward pass are fixed within each comparison, these measurements isolate execution of the common canonical law. The widening gap with tree size is consistent with Predraw's per-node sampling cost, whereas TEV uses level-wise probability propagation followed by one exit-node and one bonus-token draw. Even so, the stage-level reduction is larger than the end-to-end improvement in Table 4 because verifier decision and cache commit constitute only a small fraction of total decoding time.

D.2 Crossed drafter–verifier evaluation

Table 4 crosses the drafter checkpoint with the verification procedure while holding the DDTree builder, $B = 64$ budget, target model, decoding distribution, and runtime configuration fixed. The comparison reports end-to-end speedup and output-block length at $T = 1$; verifier-stage latency is isolated separately in Table 3. We omit $T = 0$ because its degenerate target distribution reduces verification to greedy decoding rather than the sampling regime for which TEV is designed.

For a fixed drafter, the empirical τ differences between Predraw and TEV are at most 0.03 tokens, as expected for separate finite-sample runs of two saturated verifiers with the same expected output-block length. TEV nevertheless improves average SR from $4.95\times$ to $4.97\times$ with the original drafter and from $5.55\times$ to $5.59\times$ with the exit-trained drafter. In contrast, changing the drafter has the dominant effect: under Predraw, exit training increases average τ by 13.7% and SR by 12.1%;

Table 4: Crossed drafter–verifier evaluation on Qwen3-4B at $T = 1$. Each cell reports end-to-end speedup ratio (SR) and average output-block length including the bonus token (τ). The builder and tree budget $B = 64$ are fixed across all four combinations.

		MT-Bench		HumanEval		GSM8K		MATH-500		Average	
Drafter	**Verifier**	$SR\uparrow$	$\tau\uparrow$	$SR\uparrow$	$\tau\uparrow$	$SR\uparrow$	$\tau\uparrow$	$SR\uparrow$	$\tau\uparrow$	$SR\uparrow$	$\tau\uparrow$
Original DFlash	Predraw	3.44	5.04	5.22	7.53	5.48	7.85	5.65	7.88	4.95	7.08
	TEV	3.47	5.07	5.22	7.52	5.51	7.88	5.67	7.90	4.97	7.09
Exit-trained DFlash	Predraw	3.95	5.86	5.61	8.05	6.44	9.23	6.22	9.06	5.55	8.05
	TEV	**3.98**	**5.86**	**5.64**	**8.06**	**6.47**	**9.25**	**6.26**	**9.08**	**5.59**	**8.06**

Table 5: Qwen3-4B performance across tree budgets on MT-Bench, HumanEval, GSM8K, and MATH-500. ExitTrain is trained at $B = 64$ and evaluated at every budget without retraining.

		MT-Bench		HumanEval		GSM8K		MATH-500		Average	
B	**Method**	$SR\uparrow$	$\tau\uparrow$	$SR\uparrow$	$\tau\uparrow$	$SR\uparrow$	$\tau\uparrow$	$SR\uparrow$	$\tau\uparrow$	$SR\uparrow$	$\tau\uparrow$
Temperature = 0											
32	DDTree	3.69	5.19	5.47	7.74	5.40	8.03	6.50	8.74	5.26	7.43
	Draft-OPD	3.91	5.32	5.60	7.82	5.87	8.92	6.84	9.70	5.56	7.94
	ExitTrain + TEV	**4.11**	**5.87**	**5.72**	**8.22**	**6.68**	**9.78**	**6.94**	**9.77**	**5.86**	**8.41**
64	DDTree	3.81	5.41	5.71	8.13	5.65	8.48	6.83	9.16	5.50	7.80
	Draft-OPD	4.16	5.73	5.96	8.35	6.11	9.30	7.02	10.14	5.81	8.38
	ExitTrain + TEV	**4.28**	**6.17**	**6.09**	**8.74**	**6.91**	**10.22**	**7.23**	**10.26**	**6.13**	**8.85**
128	DDTree	3.95	5.77	5.93	8.57	5.86	8.82	6.89	9.60	5.66	8.19
	Draft-OPD	4.25	6.10	6.10	8.64	6.28	9.63	7.14	10.40	5.94	8.69
	ExitTrain + TEV	**4.40**	**6.50**	**6.28**	**9.10**	**7.15**	**10.60**	**7.43**	**10.72**	**6.31**	**9.23**
Temperature = 1											
32	DDTree	3.31	4.82	5.00	7.15	5.24	7.50	5.39	7.60	4.73	6.77
	Draft-OPD	3.56	4.97	5.34	7.49	5.58	8.12	5.86	8.14	5.09	7.18
	ExitTrain + TEV	**3.79**	**5.50**	**5.38**	**7.75**	**6.09**	**8.78**	**6.02**	**8.52**	**5.32**	**7.64**
64	DDTree	3.44	5.04	5.22	7.53	5.48	7.85	5.65	7.88	4.95	7.08
	Draft-OPD	3.74	5.26	5.49	7.80	5.88	8.58	6.08	8.50	5.30	7.54
	ExitTrain + TEV	**3.98**	**5.86**	**5.64**	**8.06**	**6.47**	**9.25**	**6.26**	**9.08**	**5.59**	**8.06**
128	DDTree	3.50	5.32	5.42	7.93	5.74	8.35	5.75	8.38	5.10	7.50
	Draft-OPD	3.80	5.54	5.81	8.31	5.99	8.89	6.17	8.78	5.44	7.88
	ExitTrain + TEV	**4.06**	**6.08**	**5.77**	**8.45**	**6.60**	**9.66**	**6.36**	**9.30**	**5.70**	**8.37**

under TEV, the gains are 13.7% and 12.5%. This separates improved tree allocation from the smaller verifier-execution gain.

D.3 Tree-budget robustness

Table 5 varies only the inference-time DDTree budget. All drafter checkpoints are fixed; in particular, ExitTrain is trained with $B = 64$ and evaluated without retraining at $B \in \{32, 64, 128\}$. We report each benchmark separately together with the four-benchmark average.

Increasing the budget raises absolute output-block length for every method. ExitTrain + TEV improves average τ from 8.41 to 8.85 to 9.23 at $T = 0$ and from 7.64 to 8.06 to 8.37 at $T = 1$ as B grows from 32 to 64 to 128; average SR increases from $5.86\times$ to $6.13\times$ to $6.31\times$ and from $5.32\times$ to $5.59\times$ to $5.70\times$, respectively. More importantly, its advantage persists at every budget. Relative to DDTree, the gains span 11.6%–13.8% in τ and 11.4%–12.9% in SR; relative to Draft-OPD, they

span 5.6%–6.9% and 4.5%–6.2%. Thus, the drafter trained with $B = 64$ transfers to both smaller and larger inference trees without budget-specific retraining.

Unless otherwise stated, the main and crossed experiments use a tree budget of $B = 64$ non-root nodes, BF16 model weights, SDPA for target-model attention, FlashAttention-2 for the drafter, batch size one, and at most 2,048 generated tokens. Every evaluation configuration uses one NVIDIA H20-96G GPU. We evaluate 128 GSM8K examples, all 164 HumanEval problems, 128 MATH-500 examples, and 80 MT-Bench questions; GSM8K, HumanEval, and MATH-500 are repeated three times and MT-Bench twice. Qwen3 models use their chat template with thinking disabled. Speedup ratios are computed against autoregressive decoding measured under the same runtime configuration as the corresponding speculative run.

Training configuration. All runs initialize from the official DFlash checkpoints released at `https://huggingface.co/collections/z-lab/dflash`. We train online with SpecForge using prompts from the `open-perfectblend` mixture (Xu et al., 2024): the frozen target model generates rollout responses during training, and the drafter is updated on these target-generated trajectories. Training uses one 8×H20-96G server, with one GPU assigned to target-model rollout and the remaining seven GPUs assigned to draft-model training. All models use learning rate 10^{-4} and batch size 4. The measured wall-clock training times on this server are approximately 35 hours for Qwen3-4B, 55 hours for Qwen3-8B, 100 hours for Gemma-4-12B-IT, and 80 hours for Qwen3-Coder-30B-A3B. Every training-objective comparison in Sec. 6.2 uses the same initialization, online data pipeline, and compute budget.