ResiSpec: Enhancing Multi-Candidate Speculative Sampling via Residual Distribution Shaping

Zhi-Kai Chen[1,2], **Jun-Jie Tao**[3], **Wei-Xiang Mao**[3], **De-Chuan Zhan**[1,2], **Han-Jia Ye**[1,2*]

[1]School of Artificial Intelligence, Nanjing University, China
[2]National Key Laboratory for Novel Software Technology, Nanjing University, China
[3]Nanjing University, China

Abstract

The efficiency of Large Language Model (LLM) serving is fundamentally limited by the sequential nature of autoregressive decoding. Speculative Decoding (SD) mitigates this by using a lightweight draft model to speculate future tokens, which are then validated by the LLM in a single parallel forward pass. To further boost efficiency, multi-candidate schemes propose diverse candidate sets to increase the likelihood of token acceptance. However, we show that these schemes are bottlenecked by Residual Drift: a phenomenon where the rejection of initial candidates causes the residual target distribution to diverge from the draft model's predictions. This shift renders subsequent candidates ineffective and forces the system into expensive resampling. To resolve this, we propose ResiSpec, a framework that strategically reforms the proposal distribution during verification to anchor the residual target mass within the draft model's high-confidence regions. By mathematically re-aligning the verification process without compromising output exactness, ResiSpec prevents candidate obsolescence and achieves up to 1.92× speedup over state-of-the-art multi-candidate methods. Code is available at https://github.com/Czzzk/Resispec.

Introduction

The widespread adoption of Large Language Models (LLMs) (Achiam et al. 2023; Touvron et al. 2023b) has made efficient inference a critical priority for reducing both user latency and operational costs (Pope et al. 2023). However, standard auto-regressive (AR) decoding is fundamentally limited by its sequential nature, generating tokens one-by-one (Chen et al. 2023). This process creates a severe I/O bottleneck, as the system must repeatedly access the expanding Key-Value (KV) cache to compute attention for each new output (Yu et al. 2022). This sequential dependency prevents the system from fully leveraging the massive parallel processing power of modern GPUs, leaving compute cores significantly underutilized while waiting for memory traffic (Dao et al. 2022; Fu et al. 2024; Leviathan, Kalman, and Matias 2023). Consequently, the efficiency of LLM serving remains low, as most execution time is consumed by the overhead of accessing context rather than performing computation.

To overcome these sequential constraints, Speculative Decoding (SD) (Leviathan, Kalman, and Matias 2023; Chen et al. 2023; Xia et al. 2024) introduces a "speculate-then-verify" paradigm. The core idea is to decouple the generation process: a lightweight draft model rapidly predicts a sequence of candidate tokens at a fraction of the cost, which the larger target model then validates in a single, parallel forward pass (Spector and Re 2023; He et al. 2024; Huang, Guo, and Wang 2024). This approach directly addresses the I/O bottleneck by exploiting the Transformer's inherent ability to process multiple tokens concurrently with nearly the same memory latency as a single-token step (Vaswani et al. 2017; Narayanan et al. 2021; Pope et al. 2022). By shifting from one-by-one generation to bulk verification, SD effectively increases the number of generated tokens per memory-access cycle. Consequently, performance is contingent upon acceptance length, where longer validated sequences translate into a substantial reduction in total inference steps.

To fully exploit this parallel capacity and maximize the "tokens-per-load" efficiency, recent advancements have transitioned from linear, single-sequence speculation to multi-candidate schemes, such as tree-based or batch-sampling verification (Miao et al. 2024; Cai et al. 2024; Li et al. 2024b). The core idea is to expand the search space by pre-sampling multiple candidate tokens for the same or branching positions. By presenting the target model with a diverse set of "choices" in a single forward pass, these schemes aim to increase the statistical probability that at least one high-quality continuation is accepted. This strategy effectively seeks to maximize the step size—the number of confirmed tokens—of each inference iteration.

While multi-candidate speculative decoding schemes—such as tree-based or batch verification—have significantly advanced LLM inference efficiency by expanding the search space beyond linear sequences, they encounter a critical scaling bottleneck. Theoretically, presenting more candidates should increase the probability of finding a high-quality match; however, in practice, the marginal utility of additional candidates diminishes rapidly as the batch size or tree width grows. We observe that as the verification chain lengthens, the acceptance probability of each subsequent candidate drops progressively, often leading to performance plateaus or even efficiency regressions. This raises a fundamental question: why do existing multi-candidate frameworks fail to fully leverage a larger search space to achieve proportional speedups?

*Corresponding author.

We formalize this scalability limit as Residual Drift. This phenomenon stems from the mathematical necessity of maintaining "exactness" during verification: when the target model rejects a draft candidate, it must compensate by shifting its sampling distribution to a residual distribution. Crucially, this residual is naturally concentrated in the draft model's "blind spots"—semantic regions where the lightweight draft model has low predictive confidence or fails to capture complex dependencies. Because all candidates in a multi-candidate scheme are pre-sampled from the original draft distribution (before any rejections occur), they are inherently ill-suited for this shifted residual. Consequently, an early rejection often triggers a "cascade of obsolescence": the target model's updated criteria move so far away from the draft's knowledge base that the remaining pre-sampled candidates become statistically irrelevant, forcing expensive resampling and negating the parallelism benefits of the multi-candidate design.

To resolve this, we propose ResiSpec (Figure 2). Our key insight is that the mathematical path to exactness is not unique; the drift observed in prior work is a byproduct of suboptimal verification criteria rather than an inevitable cost of zero-bias sampling. ResiSpec reformulates the verification criteria by integrating an auxiliary proxy of draft density, which steers the rejection residuals back toward the draft model's high-confidence zones. By reshaping the transition logic, ResiSpec ensures that pre-sampled candidates remain viable throughout the verification chain without compromising the statistical integrity of the target model.

- Characterization of Residual Drift: We formalize *Residual Drift* as a fundamental bottleneck in multi-candidate speculative decoding, where sequential rejections shift the target distribution into the draft model's blind spots, invalidating parallel candidates.
- The ResiSpec Framework: We propose ResiSpec, which reshapes verification criteria by introducing an auxiliary proxy to keep rejection residuals proximal to the draft model's high-density regions. This ensures the viability of pre-sampled candidates even after preceding rejections.
- Theoretical Rigor and Speedup: We prove ResiSpec's mathematical exactness and demonstrate up to 1.92× speedup over multi-candidate methods while maintaining zero-bias sampling across benchmarks.

Related Work

Auto-regressive Acceleration and Speculative Decoding. Large Language Models (LLMs) generate text auto-regressively (Radford et al. 2019; Mann et al. 2020; Achiam et al. 2023; Touvron et al. 2023a), a process constrained by memory bandwidth. Speculative Decoding (SD) (Leviathan, Kalman, and Matias 2023; Chen et al. 2023) emerged as a pivotal lossless acceleration paradigm, using a small draft model to generate candidate tokens that the target model verifies in parallel. While SD guarantees mathematical equivalence to the original distribution, its performance is capped by the draft model's hit rate on a single linear sequence (Xia et al. 2024; Hu et al. 2025a; Yin et al. 2024).

Multi-Candidate and Tree-Based Speculation. To overcome the limited acceptance rate of linear speculative decoding, recent work explores multiple candidate paths in parallel. *SpecInfer* (Miao et al. 2024) introduced tree-based attention, enabling the target model to verify branching token sequences in a single forward pass. Subsequent methods improve this paradigm along two directions: candidate generation and tree construction. For candidate generation, draft-model-free methods such as *EAGLE* (Li et al. 2024b,a), *Medusa* (Cai et al. 2024), *Hydra* (Ankner et al. 2024), and *KOALA* (Zhang, Zhao, and Chen 2024) use specialized heads or lightweight modules to generate candidates from the target model's hidden states. For tree construction, *Sequoia* (Chen et al. 2024) uses dynamic programming to optimize tree shape under a budget, while prior work (Hu et al. 2025b) studies theoretical criteria for selecting tokens in the verification tree.

Scalability Limits of Standard Verification Protocols Despite the success of multi-candidate speculation, standard sampling protocols fail to scale efficiently with candidate counts, leading to a bottleneck in acceptance length. We bridge this gap with ResiSpec, a new sampling paradigm tailored for multi-candidate verification. By optimizing the sampling logic for large-scale branching trees, ResiSpec significantly enhances the marginal utility of additional candidates and achieves a higher expected token yield.

Preliminary

Speculative Decoding. The key idea behind speculative decoding is to accelerate autoregressive sampling by leveraging a draft model to propose candidate tokens, which are then verified by the target model. Let the target model define a conditional distribution $P_{\mathtt{tgt}}(x \mid x_{<t})$ and the draft model define $P_{\mathtt{dft}}(x \mid x_{<t})$, where t is the current step. The algorithm operates in two stages: drafting and verification.

During the drafting stage, the draft model generates $\gamma \in \mathbb{Z}^+$ candidate tokens from its own sampling distribution $P_{\mathtt{dft}}(x_t \mid x_{<t})$. Since these tokens are only approximate, a verification step is performed. In this stage, the target model evaluates the speculative tokens $t+1 : t+n$ in parallel using a single forward pass. Each token is accepted with probability $\min(1, p_{\mathtt{tgt}}(x)/p_{\mathtt{dft}}(x))$. If none of the speculative tokens are accepted, the target model resamples according to the adjusted distribution

$$p'_{\mathtt{tgt}}(x) = \mathrm{norm}\Big(\max(0,\, p_{\mathtt{tgt}}(x) - p_{\mathtt{dft}}(x)) \Big) \qquad (1)$$

ensuring correctness while still leveraging the draft model for potential speedup.

Multi-candidate Tree-structured Verification. To explore multiple potential future paths, recent schemes organize N speculative tokens into a tree-structured proposal $\mathcal{T}$. The tree $\mathcal{T}$ is characterized by its depth d, which specifies the number of tokens predicted sequentially forward along each path, and its branching factor b, which denotes the number of parallel candidates proposed at each position.

Specifically, at any given node in the tree, the draft model proposes b independent tokens $\{x_1, \ldots, x_b\}$ for the same speculative position. To preserve the exactness of the target

distribution, these b candidates are verified sequentially. Let $p^{(1)}_{\texttt{tgt}}(x)$ be the initial target distribution for the first candidate x_1. For each $i \in \{1, \ldots, b\}$, candidate x_i is accepted with probability: $\min(1, p^{(i)}_{\texttt{tgt}}(x_i)/q_{\texttt{dft}}(x_i))$ where $q_{\texttt{dft}}$ is the draft distribution. If x_i is rejected, the target distribution for the next candidate x_{i+1} is updated to the residual distribution $p^{(i+1)}_{\texttt{tgt}}$:

$$p^{(i+1)}_{\texttt{tgt}}(x) = \frac{\max\left(0, p^{(i)}_{\texttt{tgt}}(x) - q_{\texttt{dft}}(x)\right)}{\sum_{x' \in \mathcal{X}} \max\left(0, p^{(i)}_{\texttt{tgt}}(x') - q_{\texttt{dft}}(x')\right)}. \quad (2)$$

In a tree structure, this verification process is applied at each level. If a candidate x_i is accepted, the process continues to its children in the next level (depth); if all b candidates at a certain position are rejected, the branch is truncated, and a final token is resampled from the cumulative residual distribution $p^{(b+1)}_{\texttt{tgt}}(x)$. By using a tree-attention mask, the target model can verify all paths within $\mathcal{T}$ in a single forward pass, significantly increasing the potential number of accepted tokens per step.

Limitation of Multi-candidate Speculative Decoding

Multi-candidate speculative decoding aims to maximize accepted tokens by verifying a batch of N candidates $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ for one position. Although a broader search space should improve acceptance, its efficiency is throttled by Residual Drift. In standard verification (Equation 2), candidates are checked sequentially to preserve exactness. Once $\mathbf{x}_1$ is rejected, $\mathbf{x}_2$ must be verified against the residual distribution $P_{\text{res}}(x) = \max(0, p(x) - q(x)) / \sum_{x'} \max(0, p(x') - q(x'))$. By construction, $P_{\text{res}}(x)$ assigns zero probability where $q(x)$ already meets or exceeds $p(x)$, concentrating mass in the draft model's "blind spots." This creates a severe support mismatch: all N candidates are sampled i.i.d. from the original draft distribution $q(x)$, while later verification criteria shift away from $q(x)$'s high-density regions.

Our empirical results confirm this precipitous decline in marginal utility. As illustrated in Figure 1, the first candidate $\mathbf{x}_1$ maintains a meaningful acceptance rate, while the probability for subsequent candidates $\mathbf{x}_{i \geq 2}$ rapidly approaches zero. This pattern indicates a cascade of obsolescence: because candidates in the same batch are sampled from the high-density support of $q(x)$, an early rejection shifts the verification criteria toward a residual distribution where the remaining candidates have little statistical visibility. To quantify the resulting scaling inefficiency, we vary the number of candidates N under a fixed speculation depth and report the efficiency score η, defined as the marginal EAL gain per additional candidate. Appendix Table 5 shows that increasing N yields only modest EAL improvements while η drops sharply, indicating that wider speculation spends increasing parallel verification compute on candidates with diminishing acceptance utility.

These observations reveal a scaling wall: expanding tree width incurs linear verification cost but yields sub-linear acceptance gains because the residual distribution diverges

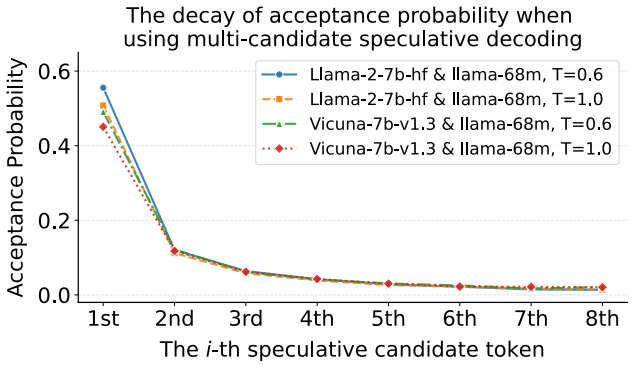

Figure 1: **The Decay of Acceptance Probability.** Sequential candidate acceptance rates on CNN/DM using Llama-2-7B and Vicuna-7B (Peng et al. 2023) as targets with Llama-68M as the draft. Across temperatures ($T \in \{0.6, 1.0\}$), acceptance drops sharply after the first candidate, leaving later candidates with near-zero marginal utility.

from the draft distribution. This raises a central question: *Can we break this bottleneck by mitigating Residual Drift?* If rejection residuals can remain aligned with high-density regions of $q(x)$, subsequent candidates may stay useful even after earlier failures. By re-aligning the verification criteria with the draft model's support, we can transform multi-candidate decoding from a game of diminishing returns into a scalable acceleration paradigm. Next, we present ResiSpec.

Method: Residual Distribution Shaping

In this section, we present ResiSpec, a framework that ensures high acceptance rates in multi-candidate speculative decoding by actively shaping the residual distribution. The innovation is the introduction of an auxiliary proxy distribution $k(x)$, which replaces the original draft distribution $q(x)$ during the verification step. By strategically constructing $k(x)$, we ensure that if a candidate is rejected, the resulting residual distribution $p'(x)$ remains proximal to the draft distribution $q(x)$, thereby maximizing the acceptance probability for all subsequent candidates in the batch.

Problem Formulation

In standard speculative decoding, a candidate x is accepted with probability $\min(1, p(x)/q(x))$. If rejected, the algorithm samples from a residual distribution. To gain control over the failure state, we propose verifying candidates using a proxy distribution $k(x)$ in place of $q(x)$.

Definition 1 (Proxy Mechanism)**.** Given a target distribution $p(x)$ and an unnormalized residual mass $r(x)$, the proxy distribution $k(x)$ is defined as:

$$k(x) = p(x) - r(x) + s(x) \quad (3)$$

where $r(x)$ represents the mass reserved for the residual distribution $p'(x) = r(x)/Z$, and $s(x) \geq 0$ is a slack function.

The slack function $s(x)$ serves as a compensatory mass to restore normalization of $k(x)$. To prevent this injected mass

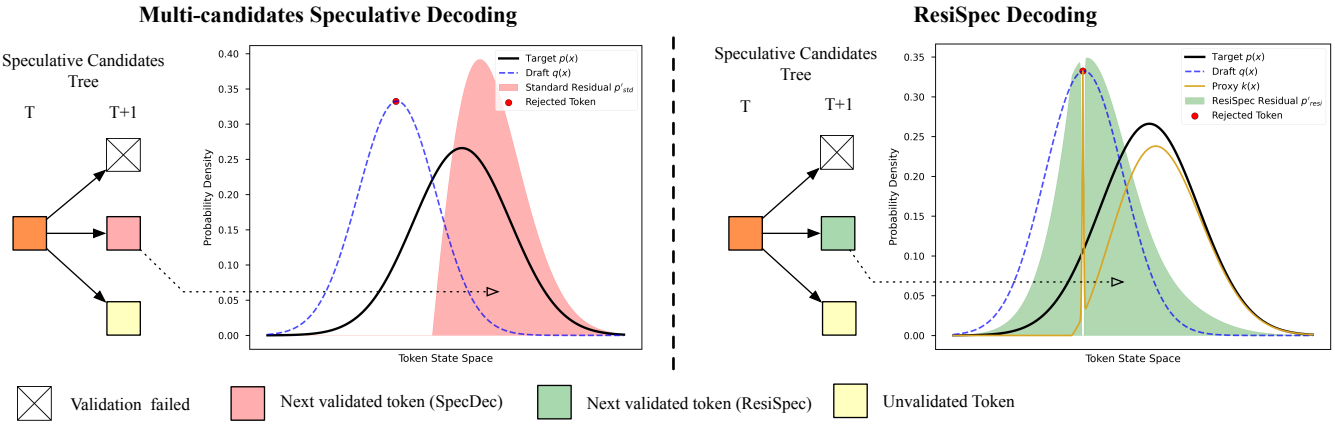

Figure 2: Comparison of ResiSpec vs. standard multi-candidate verification. In traditional schemes, sequential rejections trigger Residual Drift, shifting the target distribution into regions mathematically incompatible with the draft model (red areas), causing candidates to become obsolete. ResiSpec mitigates this drift by employing an auxiliary proxy to reshape the residual distribution (green areas), re-aligning it with the draft model's high-density support. This proximity ensures that pre-sampled candidates remain viable, effectively sustaining high acceptance rates across the entire verification batch.

from interfering with the reserved distribution, $s(x)$ is only permitted to be non-zero where the residual mass is absent ($r(x) = 0$). This leads to the disjoint support constraint:

$$\text{supp}(s) \cap \text{supp}(r) = \emptyset, \quad \text{i.e., } s(x) \cdot r(x) = 0, \forall x \in \mathcal{X} \quad (4)$$

For the sampling process to remain exact (i.e., the marginal distribution is $p(x)$), we impose the constraints on $k(x)$:

1. **Probability Constraint**: $\sum_x k(x) = 1$ and $k(x) \geq 0$.
2. **Local Alignment**: At the sampled point x_i, $k(x_i) = q(x_i)$, ensuring consistency with the draft model's local probability.

Proposition 1 (Global Mass Identity)**.** *For the proxy $k(x)$ to be a valid probability distribution, the total mass of the slack function $s(x)$ must be exactly equal to the total reserved residual mass $Z = \sum_x r(x)$, provided $r(x) \leq p(x)$ for all $x \in \mathcal{X}$.*

Proof. To satisfy the normalization axiom $\sum_x k(x) = 1$, we expand the definition:

$$\begin{aligned} \sum_x k(x) &= \sum_x (p(x) - r(x) + s(x)) = 1 \\ &\Rightarrow \sum_x p(x) - \sum_x r(x) + \sum_x s(x) = 1 \\ &\Rightarrow 1 - Z + \sum_x s(x) = 1 \Rightarrow \sum_x s(x) = Z. \end{aligned} \quad (5)$$

This identity reveals a fundamental conservation law in our framework: any probability mass "reserved" from the target to form the residual must be precisely balanced by the slack mass "injected" to construct the proxy. □

Definition 2 (The Construction Objective)**.** Since subsequent candidates are pre-sampled from $q(x)$, the probability of future acceptance is maximized when the overlap between the residual $p'(x)$ and the draft $q(x)$ is maximized. This is equivalent to minimizing the L_1 distance:

$$\min_k \sum_x |p'(x) - q(x)|, \quad \text{where } p'(x) = \frac{r(x)}{Z}, \quad Z = \sum_x r(x). \quad (6)$$

Theoretical Boundaries: Ideal Shaping and Limits

To derive the optimal residual shaping strategy, we seek a formulation that satisfies the rigorous exactness constraints established in Definition 1 while attaining the global minimum of the L_1 objective defined in Definition 2.

Ideal Alignment and the Capacity Bound. We begin by identifying the ideal shape of the residual mass $r(x)$ and then determine the maximum budget Z that the system can sustain without violating probability axioms.

Lemma 1 (Global Optimality of Residual Matching)**.** *To achieve the global minimum of the L_1 objective $\mathcal{J} = \sum_{x \in \mathcal{X}} |r(x)/Z - q(x)|$, the residual mass must satisfy the ideal scaling relationship:*

$$r(x) = Z \cdot q(x), \quad \forall x \in \mathcal{X} \quad (7)$$

Proof. By the properties of the L_1 norm, $\mathcal{J} \geq 0$, with equality if and only if the normalized residual $p'(x) = r(x)/Z$ is identical to $q(x)$. This directly implies $r(x) = Zq(x)$. □

Proposition 2 (The Capacity Bound of Ideal Shaping)**.** *For the ideal residual $r(x) = Zq(x)$ to be part of a valid proxy distribution $k(x) = p(x) - r(x) + s(x)$ while adhering to the disjoint support constraint ($s \cdot r = 0$), the total reserved mass Z is bounded by the minimum density ratio:*

$$Z \leq \min_{x: q(x) > 0} \frac{p(x)}{q(x)} \quad (8)$$

Proof. Substituting the ideal shape $r(x) = Zq(x)$ into the non-negativity constraint $k(x) \geq 0$ from Definition 1, we have:

$$p(x) - Zq(x) + s(x) \geq 0 \tag{9}$$

For any x where $q(x) > 0$, the ideal residual $r(x) = Zq(x)$ is strictly positive. According to the disjoint support constraint in Definition 1, we must have $s(x) = 0$ for these tokens. The inequality thus simplifies to:

$$p(x) - Zq(x) \geq 0 \implies Z \leq \frac{p(x)}{q(x)} \tag{10}$$

Taking the minimum over all x in the support of q ensures the proxy remains non-negative, yielding the stated bound. □

The Degeneracy and Support Paradox. In real-world LLM inference, the theoretical feasible region defined by $Z \leq \min_{x:q(x)>0} \frac{p(x)}{q(x)}$ is often degenerate or operationally unsustainable. This stems from two fundamental conflicts:

1. The Vanishing Margin: Due to the "long-tail" (Baevski and Auli 2018) nature of language models, there frequently exist tokens where the target model $p(x) \to 0$ while the draft model $q(x) > 0$. In such cases, the upper bound $\min \frac{p(x)}{q(x)}$ collapses to near-zero, effectively shrinking the feasible region to $\{0\}$. A budget of $Z \approx 0$ would deplete the residual mass $r(x)$, stripping the system of the probability budget needed to support subsequent candidates.

2. The Alignment-Identity Contradiction: To achieve the global minimum of the L_1 objective ($L_1 = 0$), we require $r(x) = Z \cdot q(x)$. If the draft model has full support ($q(x) > 0, \forall x$), then $r(x) > 0$ must also hold. Under the *disjoint support constraint* ($s \cdot r = 0$), this forces the slack function $s(x)$ to be zero everywhere, which, by Proposition 1, implies $Z = \sum s(x) = 0$.

Crucially, accepting the degenerate solution $Z = 0$ causes the proxy distribution to revert to the target ($k(x) = p(x)$). This directly violates the Local Alignment constraint $k(x_i) = q(x_i)$, as the rejected candidate x_i is typically a point of over-estimation where $q(x_i) > p(x_i)$. We therefore reformulate the optimal shaping of the residual as a constrained approximation problem, relaxing the ideal shape to maintain a functional residual budget.

Practical Implementation: The ResiSpec Algorithm

Given that the theoretical ideal is unattainable, we propose a minimal-budget approximation strategy. According to Theorem 2, the L_1 distance between the shaped residual $p'(x)$ and the draft distribution $q(x)$ is strictly monotonically increasing with respect to Z whenever $Z > Z^*$. Therefore, to maintain the fidelity of the draft model's trajectory and maximize the acceptance utility for subsequent candidates, we should seek the smallest possible Z that satisfies the physical constraints of the verification step.

Since x_0 is a point where the draft model over-estimates the target ($q(x_0) > p(x_0)$), and no residual mass can be reserved at a point where the target density is already exhausted (i.e., $r(x_0) = 0$), the definition of $k(x)$ at x_0 simplifies to $q(x_0) = p(x_0) + s(x_0)$. This necessitates a local mass injection of:

$$s(x_0) = q(x_0) - p(x_0) \tag{11}$$

Given the Global Mass Identity $Z = \sum_x s(x)$ and the non-negativity requirement $s(x) \geq 0$, the total budget Z is strictly lower-bounded by this local discrepancy:

$$Z = s(x_0) + \sum_{x \neq x_0} s(x) \geq |q(x_0) - p(x_0)| \tag{12}$$

This bound represents the minimum "price" the system must pay in terms of residual mass to bridge the gap between the draft's prediction and the target's reality at the failure point.

Algorithm 1 ResiSpec Sampling and Verification

(*The underlined step marks ResiSpec's proxy computation, distinguishing it from standard multi-candidate protocols.*)

Input: Prefix $x_{<n}$, target model $\mathcal{P}$, draft model $\mathcal{Q}$, and candidates k.
Output: A verified token x sampled via the ResiSpec protocol.
Initialize residual $R \leftarrow \mathcal{P}$, draft $D \leftarrow \mathcal{Q}$
for $i = 1 \to k$ **do**
  Sample $x_i \sim D, r_i \sim \text{Uniform}(0, 1)$
  if $r_i < \frac{R(x_i)}{D(x_i)}$ **then**
    Return x_i ▷ Accept x_i
  else
    ▷ Construct Proxy Distribution
    $\underline{K \leftarrow \text{GetProxy}(R, Q, x_i)}$ ▷ *ResiSpec proxy step*
    ▷ Use Proxy Distribution to Calculate Residual Distribution instead of Draft distribution
    $R \leftarrow \text{norm}(\max(R - K, 0))$
  end if
end for
▷ Resampling From the Residual Distribution
Return $x \sim R$

Function GETPROXY(p, q, x_0)
$Z \leftarrow q(x_0) - p(x_0)$ ▷ Total residual budget
$r_i \leftarrow \min(Z \cdot q_i, p_i)$ for all $i \neq x_0$, and $r_{x_0} \leftarrow 0$
$\Delta \leftarrow Z - \sum_i r_i$ ▷ Mass lost due to clipping at p_i
Update r by distributing Δ into the remaining capacity $(p - r)$ ▷ Ensures $\sum r = Z$ and $r \leq p$
$k \leftarrow p - r + s$, then set $k(x_0) \leftarrow q(x_0)$ ▷ Align with q at x_0
Return k

Guided by the monotonicity principle (Theorem 2)—which suggests that any Z larger than necessary will further deviate the residual from the draft's manifold—we select the minimal Z that accommodates this local gap. In practice, we employ the following heuristic:

$$Z_{applied} = \max(Z^*, |q(x_0) - p(x_0)|) \tag{13}$$

This adjustment ensures that $k(x)$ remains a valid probability distribution while preventing the residual mass from collapsing due to long-tail noise. The complete procedure for dynamically adjusting $r(x)$ and Z to balance theoretical exactness and inference speed is detailed in Algorithm 1.

In the event of a verification failure, GETPROXY intervenes to adjust the residual distribution R prior to the next verification step. Its primary role is to prevent the target distribution

Target LLM	Draft Model	T	Dataset	ResiSpec		SpecInfer		ResiSpec vs SpecInfer	
				Acc.↑	Thru.↑	Acc.↑	Thru.↑	Acc. Gain	Speedup
Llama-2-7B	JF68M	1	CNN/DM	4.68	96.43	2.51	57.29	1.86×	1.68×
Llama-2-7B	JF68M	1	OpenWebText	4.69	96.06	2.51	56.76	1.86×	1.69×
Llama-2-7B	JF68M	1	C4	4.67	96.90	2.64	61.02	1.77×	1.59×
Llama-2-7B	JF68M	0.6	CNN/DM	4.14	85.03	2.59	59.41	1.60×	1.43×
Llama-2-7B	JF68M	0.6	OpenWebText	4.16	85.62	2.49	56.48	1.67×	1.52×
Llama-2-7B	JF68M	0.6	C4	4.13	85.91	2.66	61.69	1.55×	1.39×

Table 1: Performance comparison between ResiSpec and the baseline SpecInfer. Regarding the configuration of the tree, we use a 3-branch complete tree with a depth of 5. The results in the table are averaged over 200 runs to reduce variance and ensure statistical reliability. Experimental results demonstrate that ResiSpec significantly increased the accepted length (by up to 1.86×) and throughput (by up to 1.68×), validating its effectiveness in accelerating large language model inference.

Method	**Acc. Len↑**	**Thru. (tok/s)↑**	**Acc. Len Gain**	**Speedup**
Sequoia	3.07	88.14	–	–
Sequoia+ResiSpec	4.32	115.32	1.41×	1.31×
EAGLE	3.12	61.82	–	–
EAGLE+ResiSpec	3.85	71.17	1.23×	1.15×

Table 2: Compatibility of ResiSpec with existing multi-candidate speculative decoding methods at $T = 1.0$. ResiSpec is evaluated as an add-on to Sequoia (Chen et al. 2024) and EAGLE (Li et al. 2024b) under matched decoding settings, improving both metrics.

from drifting into regions of low draft density. By aligning the mass of R with the draft model Q, the system ensures that the remaining candidates $\{x_{i+1}, \ldots, x_k\}$ encounter verification criteria compatible with the draft model's knowledge. The function's logic is as follows:

Step 1: Budgeting and Initial Projection. It first calculates the mass budget $Z = |q(x_0) - p(x_0)|$. It then performs an initial projection by setting $r_i = \min(Zq_i, p_i)$, effectively "clipping" the desired draft shape into the target model's available capacity.

Step 2: Capacity-Aware Redistribution. Any mass deficit Δ caused by clipping is redistributed to tokens where $p_i > r_i$. This "water-filling" approach ensures that the total residual mass strictly meets the budget Z without violating the physical upper bound $p(x)$.

Step 3: Proxy Formulation. Finally, it constructs $k = p - r$, with a local adjustment at x_0 to ensure $k(x_0) = q(x_0)$. This ensures that the acceptance ratio at the failed point remains mathematically consistent with the original draft.

Theoretical Verification

A prerequisite for any speculative sampling framework is the preservation of the target model's distribution, ensuring that acceleration is achieved without compromising output quality. We formally prove that ResiSpec satisfies this exactness property; specifically, our iterative residual-shaping process, mediated by the proxy distribution $k(x)$, is shown to be provably unbiased, maintaining a marginal distribution identical to the target model $p(x)$ across all verification steps. A rigorous theoretical justification is provided in Appendix A.1. This theoretical guarantee is further corroborated by empirical validation, where we demonstrate zero distributional shift compared to standard auto-regressive decoding.

Experiment

Experiment Setup

Model and Baseline. We evaluate ResiSpec in comparison with SpecInfer (Miao et al. 2024) using Llama-2-7B (Touvron et al. 2023b) as the target model and JackFram/Llama-68M (JF68M) (Miao et al. 2024) as the draft model. To ensure evaluation coverage, we conduct experiments on three widely-used text corpora: CNN/DM (See, Liu, and Manning 2017), OpenWebText (Gokaslan and Cohen 2019), and C4 (Raffel et al. 2020). All evaluations are performed under two temperature settings, $T = 1.0$ and $T = 0.6$, to assess performance across different sampling regimes.

Metrics. We assess the efficiency of speculative decoding using two metrics. The first metric is *Accepted Length* (Acc. Len), which measures the average number of tokens successfully accepted per speculative decoding step; higher values indicate more effective draft utilization. The second metric is *Throughput* (Thru.), defined as the number of generated tokens per second, which directly reflects end-to-end inference efficiency. All experiments are conducted on a single NVIDIA RTX 3090 GPU and averaged over multiple runs.

Main Results

We perform an extensive evaluation comparing ResiSpec with SpecInfer, with detailed results presented in Table 1. Considering that both ResiSpec and SpecInfer employ a tree-based speculative decoding method, we use a 3-branch complete tree with a depth of 5 for all experiments. The findings demonstrate that ResiSpec achieves statistically significant improvements over the baseline method. Specifically, we observe consistent enhancements in accepted length (up to **1.86**×) and throughput (up to **1.68**×) across all tested configurations. Notably, these performance gains are sustained across diverse data domains and varying sampling temperatures, highlighting the robustness and generalizability of our proposed distribution shaping mechanism.

Compatibility with Existing Methods. To examine whether residual shaping is tied to a specific baseline, we

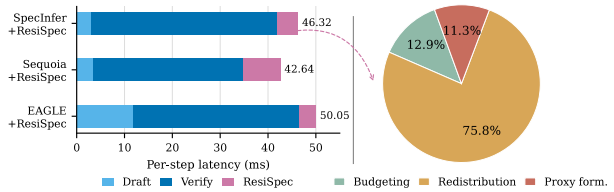

Figure 3: Runtime analysis of ResiSpec. Left: per-step latency across SpecInfer, Sequoia, and EAGLE when augmented with ResiSpec, decomposed into drafting, target verification, and ResiSpec shaping. Right: ResiSpec's internal runtime breakdown across budgeting, redistribution, and proxy formulation, shown by percentage.

integrate ResiSpec with two different multi-candidate speculative decoding methods. EAGLE improves the drafter side by accelerating candidate generation, while Sequoia optimizes the verification tree used by the target model. As shown in Table 2, ResiSpec is evaluated as an add-on, reporting accepted-length and throughput improvements under matched decoding settings.

Computational Overhead. We further profile the runtime of ResiSpec's proxy construction and residual update relative to a single target-model verification step. Most of this overhead comes from memory-bound GPU operations over token distributions, rather than heavy arithmetic; fused kernels could therefore further reduce ResiSpec's latency. Importantly, ResiSpec does not introduce an additional model forward pass, and its cost comes from distribution-level operations after verification. Figure 3 reports per-step latency broken into drafting, verification, and ResiSpec shaping, and further decomposes ResiSpec's time into budgeting, redistribution, and proxy formulation.

Ablation Study

To further dissect the efficacy of ResiSpec, we analyze both tree topology and target-draft model pairing. For topology, we compare tree structures under an iso-compute constraint by selecting configurations with similar verification costs, ensuring that performance differences reflect distribution alignment rather than extra parallel computation. Table 3 shows that ResiSpec consistently outperforms SpecInfer, with larger gains on deeper and narrower trees, indicating stronger mitigation of accumulated Residual Drift over longer speculation horizons. We also evaluate additional target-draft model pairs and datasets in Appendix D, where ResiSpec consistently improves accepted length and throughput over SpecInfer, suggesting that residual shaping is not tied to a single model family or capacity gap.

Empirical Validation of Distributional Exactness

To empirically verify the theoretical exactness proven in Theorem 1 in Appendix A.1, we measure the Kullback-Leibler (KL) divergence (Kullback and Leibler 1951) between the original target distribution $p(x)$ and the effective marginal distribution produced by ResiSpec. The effective distribution is reconstructed by exhaustively marginalizing the probability mass across all possible verification trajectories within a candidate batch, including all sequential acceptance and rejection paths (see Appendix B for the full derivation).

As shown in Table 4, we compare ResiSpec against standard multi-candidate speculative decoding. Our results demonstrate that ResiSpec maintains the target distribution with high fidelity. Specifically, the KL divergence between the ResiSpec-generated distribution and the target model remains on the order of 10^{-10}, a level of discrepancy that is indistinguishable from standard multi-candidate methods. This negligible error is consistent with numerical floating-point noise rather than any systematic algorithmic bias. These findings confirm that ResiSpec is stochastically equivalent to the target model, ensuring that our distribution shaping does not sacrifice generation quality for speed.

Tree	Verification Costs	ResiSpec Thru.↑	SpecInfer Thru.↑	SpeedUp
d7b2	127	101.01	52.69	1.92×
d5b3	121	94.25	56.92	1.66×
d4b5	156	60.83	42.46	1.43×
d3b11	133	44.21	37.89	1.17×

Table 3: Impact of tree topologies under comparable verification budgets on Llama-2-7B with JF68M. Verification cost is the number of candidate positions verified concurrently. ResiSpec gains more over SpecInfer on deeper, narrower trees.

Method	**Data**	**KL** ($T = 1.0$)	**KL** ($T = 0.6$)
ResiSpec	CNN/DM	2.64e-10	4.38e-10
	OpenWebText	7.15e-10	4.37e-10
	C4	1.02e-9	1.39e-10
SpecInfer	CNN/DM	1.04e-9	3.55e-10
	OpenWebText	1.63e-9	3.89e-10
	C4	9.10e-10	1.09e-10

Table 4: KL divergence between the target distribution $p(x)$ and the sampling distributions of ResiSpec and SpecInfer on Llama-2-7B with JF68M. Near-zero values across datasets and temperatures confirm distributional preservation.

Conclusion

We address the "residual drift" bottleneck in multi-candidate speculative decoding by proposing ResiSpec, a distribution-shaping framework. By aligning rejection residuals with the draft model through a proxy distribution, ResiSpec prevents the collapse of acceptance rates during sequential verification. As a verification-time add-on, ResiSpec can also complement existing candidate-generation and tree-construction methods. Our method maintains exact sampling while achieving up to 1.92× speedup, providing a robust and efficient solution for high-throughput LLM acceleration.

References

Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F. L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Anadkat, S.; et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Ankner, Z.; Parthasarathy, R.; Nrusimha, A.; Rinard, C.; Ragan-Kelley, J.; and Brandon, W. 2024. Hydra: Sequentially-dependent draft heads for medusa decoding. *arXiv preprint arXiv:2402.05109*.

Baevski, A.; and Auli, M. 2018. Adaptive input representations for neural language modeling. *arXiv preprint arXiv:1809.10853*.

Bapna, A.; Arivazhagan, N.; and Firat, O. 2020. Controlling computation versus quality for neural sequence models. *arXiv preprint arXiv:2002.07106*.

Bishop, C. M.; and Nasrabadi, N. M. 2006. *Pattern recognition and machine learning*. Springer.

Burton, F. W. 2012. Speculative computation, parallelism, and functional programming. *IEEE Transactions on Computers*, 100(12): 1190–1193.

Cai, T.; Li, Y.; Geng, Z.; Peng, H.; Lee, J. D.; Chen, D.; and Dao, T. 2024. Medusa: Simple llm inference acceleration framework with multiple decoding heads. *arXiv preprint arXiv:2401.10774*.

Chen, C.; Borgeaud, S.; Irving, G.; Lespiau, J.-B.; Sifre, L.; and Jumper, J. 2023. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*.

Chen, Z.; May, A.; Svirschevski, R.; Huang, Y.; Ryabinin, M.; Jia, Z.; and Chen, B. 2024. Sequoia: Scalable, robust, and hardware-aware speculative decoding. *arXiv preprint arXiv:2402.12374*.

Dao, T.; Fu, D.; Ermon, S.; Rudra, A.; and Ré, C. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35: 16344–16359.

Fu, Y.; Bailis, P.; Stoica, I.; and Zhang, H. 2024. Break the sequential dependency of llm inference using lookahead decoding. *arXiv preprint arXiv:2402.02057*.

Gokaslan, A.; and Cohen, V. 2019. OpenWebText Corpus.

He, Z.; Zhong, Z.; Cai, T.; Lee, J.; and He, D. 2024. Rest: Retrieval-based speculative decoding. In *Proceedings of the 2024 conference of the North American chapter of the association for computational linguistics: Human language technologies (volume 1: long papers)*, 1582–1595.

Hu, Y.; Liu, Z.; Dong, Z.; Peng, T.; McDanel, B.; and Zhang, S. Q. 2025a. Speculative decoding and beyond: An in-depth survey of techniques. *arXiv preprint arXiv:2502.19732*.

Hu, Z.; Zheng, T.; Viswanathan, V.; Chen, Z.; Rossi, R. A.; Wu, Y.; Manocha, D.; and Huang, H. 2025b. Towards optimal multi-draft speculative decoding. *arXiv preprint arXiv:2502.18779*.

Huang, K.; Guo, X.; and Wang, M. 2024. Specdec++: Boosting speculative decoding via adaptive candidate lengths. *arXiv preprint arXiv:2405.19715*.

Kullback, S.; and Leibler, R. A. 1951. On information and sufficiency. *The annals of mathematical statistics*, 22(1): 79–86.

Leviathan, Y.; Kalman, M.; and Matias, Y. 2023. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Li, Y.; Wei, F.; Zhang, C.; and Zhang, H. 2024a. Eagle-2: Faster inference of language models with dynamic draft trees. *arXiv preprint arXiv:2406.16858*.

Li, Y.; Wei, F.; Zhang, C.; and Zhang, H. 2024b. EAGLE: speculative sampling requires rethinking feature uncertainty. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Luo, X.; Wang, Y.; Zhu, Q.; Zhang, Z.; Zhang, X.; Yang, Q.; and Xu, D. 2025. Turning trash into treasure: Accelerating inference of large language models with token recycling. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 6816–6831.

Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; Agarwal, S.; et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 1(3): 3.

Miao, X.; Oliaro, G.; Zhang, Z.; Cheng, X.; Wang, Z.; Zhang, Z.; Wong, R. Y. Y.; Zhu, A.; Yang, L.; Shi, X.; Shi, C.; Chen, Z.; Arfeen, D.; Abhyankar, R.; and Jia, Z. 2024. 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, 932–949. ACM.

Narayanan, D.; Shoeybi, M.; Casper, J.; LeGresley, P.; Patwary, M.; Korthikanti, V. A.; Vainbrand, D.; Kashinkunti, P.; Bernauer, J.; Catanzaro, B.; Phanishayee, A.; and Zaharia, M. 2021. Efficient Large-Scale Language Model Training on GPU Clusters Using Megatron-LM. arXiv:2104.04473.

Peng, B.; Li, C.; He, P.; Galley, M.; and Gao, J. 2023. Instruction tuning with gpt-4. *arXiv preprint arXiv:2304.03277*.

Pope, R.; Douglas, S.; Chowdhery, A.; Devlin, J.; Bradbury, J.; Heek, J.; Xiao, K.; Agrawal, S.; and Dean, J. 2023. Efficiently scaling transformer inference. *Proceedings of machine learning and systems*, 5: 606–624.

Pope, R.; Douglas, S.; Chowdhery, A.; Devlin, J.; Bradbury, J.; Levskaya, A.; Heek, J.; Xiao, K.; Agrawal, S.; and Dean, J. 2022. Efficiently Scaling Transformer Inference. arXiv:2211.05102.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I.; et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8): 9.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 21(140): 1–67.

Robert, C. P.; Casella, G.; and Casella, G. 1999. *Monte Carlo statistical methods*, volume 2. Springer.

See, A.; Liu, P. J.; and Manning, C. D. 2017. Get To The Point: Summarization with Pointer-Generator Networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, 1073–1083.

Spector, B.; and Re, C. 2023. Accelerating llm inference with staged speculative decoding. *arXiv preprint arXiv:2308.04623*.

Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.-A.; Lacroix, T.; Rozière, B.; Goyal, N.; Hambro, E.; Azhar, F.; et al. 2023a. LLaMA: open and efficient foundation language models. arXiv. *arXiv preprint arXiv:2302.13971*.

Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Xia, H.; Yang, Z.; Dong, Q.; Wang, P.; Li, Y.; Ge, T.; Liu, T.; Li, W.; and Sui, Z. 2024. Unlocking efficiency in large language model inference: A comprehensive survey of speculative decoding. *arXiv preprint arXiv:2401.07851*.

Xia, M.; Gao, T.; Zeng, Z.; and Chen, D. 2023. Sheared llama: Accelerating language model pre-training via structured pruning. *arXiv preprint arXiv:2310.06694*.

Yin, M.; Chen, M.; Huang, K.; and Wang, M. 2024. A theoretical perspective for speculative decoding algorithm. *Advances in Neural Information Processing Systems*, 37: 128082–128117.

Yu, G.-I.; Jeong, J. S.; Kim, G.-W.; Kim, S.; and Chun, B.-G. 2022. Orca: A distributed serving system for {Transformer-Based} generative models. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*, 521–538.

Zhang, K.; Zhao, J.; and Chen, R. 2024. KOALA: Enhancing Speculative Decoding for LLM via Multi-Layer Draft Heads with Adversarial Learning. *arXiv preprint arXiv:2408.08146*.

Zhang, P.; Zeng, G.; Wang, T.; and Lu, W. 2024. Tinyllama: An open-source small language model. *arXiv preprint arXiv:2401.02385*.

A Theoretical Proof for ResiSpec Exactness

A.1 Correctness and Exactness

Theorem 1 (Universal Exactness of ResiSpec)**.** *Let $p(x)$ be the target distribution and $q(x)$ be the draft distribution. For a candidate token $x \sim q(x)$, let $k(x)$ be a proxy distribution that satisfies the **Local Alignment** condition: $k(x) = q(x)$ at the specific value of the sampled candidate. If we define the acceptance probability as:*

$$\alpha(x) = \min\left(1, \frac{p(x)}{k(x)}\right) \tag{14}$$

and the residual distribution as:

$$p_{res}(x) = \frac{p(x) - \min(p(x), k(x))}{1 - \beta} \tag{15}$$

where $\beta = \sum_{x \in \mathcal{X}} \min(p(x), k(x))$, then the marginal distribution of the output token X is exactly $p(x)$.

Proof. The core intuition of ResiSpec is that while the proxy distribution $k(x)$ is used to shape the residual, the sampling exactness is preserved as long as $k(x)$ "mimics" the draft distribution $q(x)$ at the point of verification. To prove this, we consider the marginal probability $P(X = x')$ for any token $x' \in \mathcal{X}$ by summing the probabilities of two mutually exclusive events: (1) x' is proposed and accepted, or (2) a candidate is proposed and rejected, and the system subsequently samples x' from the residual distribution.

Step 1: Analysis of the Acceptance Path. For the output X to be x' via acceptance, the draft model must first propose $x \sim q(x)$ where $x = x'$, and the verification step must subsequently accept it. Crucially, at this specific point of verification, the Local Alignment condition $k(x') = q(x')$ is active. The probability mass contributed by this path is:

$$\begin{aligned} P(\text{Accept}, X = x') &= q(x') \cdot \alpha(x') \\ &= q(x') \cdot \min\left(1, \frac{p(x')}{k(x')}\right) \end{aligned} \tag{16}$$

By substituting the Local Alignment condition $k(x') = q(x')$, we can rewrite the acceptance probability in terms of k:

$$P(\text{Accept}, X = x') = k(x') \cdot \min\left(1, \frac{p(x')}{k(x')}\right) = \min(k(x'), p(x')) \tag{17}$$

This step is vital: it shows that by aligning k with q locally, the acceptance path effectively samples from the intersection of p and our proxy k.

Step 2: Analysis of the Rejection Path. The rejection path occurs if any proposed candidate x is rejected. The total probability of rejection, $P(\text{Reject})$, is the complement of the total acceptance mass:

$$P(\text{Reject}) = 1 - \sum_{x \in \mathcal{X}} P(\text{Accept}, X = x) = 1 - \sum_{x \in \mathcal{X}} \min(p(x), k(x)) = 1 - \beta \tag{18}$$

Once a rejection occurs, the system samples x' from the residual distribution $p_{\text{res}}(x')$. The probability mass contributed by this path is the product of the total rejection probability and the conditional probability of sampling x':

$$\begin{aligned} P(\text{Reject}, X = x') &= P(\text{Reject}) \cdot p_{\text{res}}(x') \\ &= (1 - \beta) \cdot \frac{p(x') - \min(p(x'), k(x'))}{1 - \beta} \\ &= p(x') - \min(p(x'), k(x')) \end{aligned} \tag{19}$$

Note that the normalization constant $1 - \beta$ cancels out, leaving a residual mass that perfectly complements the acceptance mass defined in Eq. 17.

Step 3: Summation of Marginal Probabilities. According to the law of total probability, the final marginal distribution $P(X = x')$ is the sum of the results from Eq. 17 and Eq. 19:

$$\begin{aligned} P(X = x') &= P(\text{Accept}, X = x') + P(\text{Reject}, X = x') \\ &= \min(p(x'), k(x')) + [p(x') - \min(p(x'), k(x'))] \\ &= p(x') \end{aligned} \tag{20}$$

Since this identity holds for all $x' \in \mathcal{X}$, the output distribution is identically the target distribution $p(x)$.

Conclusion. The proof demonstrates that ResiSpec remains an exact sampling algorithm. By coupling the residual distribution to the same proxy $k(x)$ used in the acceptance test, any deviation of $k(x)$ from the original draft distribution $q(x)$ is mathematically compensated for in the failure state, provided that k and q are aligned at the point of verification. □

A.2 Monotonicity of Residual Shaping Loss with Redistribution

Theorem 2 (Monotonicity with Redistribution)**.** *Let $q(x)$ and $p(x)$ be the draft and target distributions, respectively. Define the unnormalized residual mass $r(x)$ subject to the capacity constraint $0 \le r(x) \le p(x)$ and the global mass constraint $\sum_x r(x) = Z$. Let $p'(x) = r(x)/Z$ be the normalized residual distribution. The minimum L_1 distance $\mathcal{D}_{L1}(Z) = \sum_x |p'(x) - q(x)|$ is:*

1. *Identically zero for $Z \in (0, Z^*]$, where $Z^* = \min_{x:q(x)>0} \frac{p(x)}{q(x)}$.*
2. *Strictly monotonically increasing for $Z \in (Z^*, 1]$.*

Proof. To minimize $\mathcal{D}_{L1}(Z) = \frac{1}{Z}\sum_x |r(x) - Zq(x)|$, we define the local deviation as $E(x) = r(x) - Zq(x)$. Since $\sum_x r(x) = Z$ and $\sum_x Zq(x) = Z$, the total deviation sums to zero: $\sum_x E(x) = 0$. This implies that the sum of positive deviations must exactly balance the sum of negative deviations:

$$\sum_{x:E(x)>0} |E(x)| = \sum_{x:E(x)<0} |E(x)| \tag{21}$$

Consequently, the L_1 distance can be expressed solely in terms of the negative deviations:

$$\mathcal{D}_{L1}(Z) = \frac{1}{Z}\left(\sum_{E(x)>0} |E(x)| + \sum_{E(x)<0} |E(x)|\right) = \frac{2}{Z}\sum_{x:E(x)<0} |E(x)| \tag{22}$$

Under the capacity constraint $r(x) \le p(x)$, a negative deviation ($E(x) < 0$) is mandatory if and only if $p(x) < Zq(x)$. To minimize the total deviation, we must set $r(x) = p(x)$ in these "bottleneck" regions and $r(x) \ge Zq(x)$ elsewhere (Redistribution). Thus, the minimal negative deviation at any point x is $|E(x)| = \max(0, Zq(x) - p(x))$. Substituting this into the L_1 objective:

$$\mathcal{D}_{L1}(Z) = \frac{2}{Z}\sum_x \max(0, Zq(x) - p(x)) = 2\sum_x \max\left(0, q(x) - \frac{p(x)}{Z}\right) \tag{23}$$

We now analyze the behavior of $\mathcal{D}_{L1}(Z)$ across the two regimes of Z:

Case 1: $Z \in (0, Z^*]$**.** By the definition of $Z^* = \min \frac{p(x)}{q(x)}$, for any $Z \le Z^*$, we have $\frac{p(x)}{Z} \ge \frac{p(x)}{Z^*} \ge q(x)$ for all x. Thus, $q(x) - \frac{p(x)}{Z} \le 0$ for all x, and $\mathcal{D}_{L1}(Z) = 0$. In this regime, the target model has sufficient capacity to perfectly mirror the draft distribution's shape.

Case 2: $Z \in (Z^*, 1]$**.** In this regime, the set of bottleneck tokens $\mathcal{S}_Z = \{x \mid q(x) > \frac{p(x)}{Z}\}$ is non-empty. Let $f(Z) = 2\sum_{x\in\mathcal{S}_Z}(q(x) - \frac{p(x)}{Z})$. Taking the derivative with respect to Z:

$$\frac{d\mathcal{D}_{L1}}{dZ} = 2\sum_{x\in\mathcal{S}_Z} \frac{p(x)}{Z^2} \tag{24}$$

Since $p(x) > 0$ for $x \in \mathcal{S}_Z$ and $Z^2 > 0$, we have $\frac{d\mathcal{D}_{L1}}{dZ} > 0$. This proves that the L_1 distance is strictly monotonically increasing with Z once the residual mass exceeds the critical bottleneck capacity Z^*.

Conclusion: Any mass Z allocated beyond Z^* necessitates a "clipping" of the draft shape at bottleneck points and a corresponding "redistribution" of mass to non-bottleneck points. Both actions contribute equally to the increase in L_1 distance, justifying the use of a minimal mass budget to preserve distribution fidelity. □

B Calculation of Empirical Validation of Distributional Exactness

To empirically verify that ResiSpec maintains zero-bias sampling in a multi-candidate setting (e.g., batch or tree-based speculation), we measure the discrepancy between the theoretical target distribution $p(x)$ and the aggregate marginal distribution produced by the ResiSpec verification chain. In a multi-candidate scenario with n candidates $\{x_1, x_2, \ldots, x_n\}$, the final output token X is determined by a sequential search for the first accepted candidate, or a fallback to a residual distribution if all candidates are rejected.

B.1 Path-based Marginal Probability Calculation

The empirical marginal distribution $P(X = x')$ is calculated by summing the probability masses of all mutually exclusive execution paths. Let α_i denote the acceptance probability of the i-th candidate and $1 - \beta_i$ denote the probability of rejection at step i. The possible paths are:

1. **Path 1: Accepted at the first candidate.** The probability that the first candidate x_1 is x' and is accepted is:

$$P_1(x') = \min(k_1(x'), p(x')) \tag{25}$$

2. **Path i ($2 \leq i \leq n$): Accepted at the i-th candidate.** For the system to accept the i-th candidate, all preceding $i-1$ candidates must have been rejected. Due to the exactness property of the ResiSpec residual (see Appendix A.1), the target distribution effectively shifts to $p_{\text{res},i-1}(x)$ after $i-1$ rejections. The probability mass is:

$$P_i(x') = \left(\prod_{j=1}^{i-1} (1 - \beta_j) \right) \cdot \min(k_i(x'), p_{\text{res},i-1}(x')) \tag{26}$$

where $p_{\text{res},i-1}$ is the residual distribution reshaped by ResiSpec after the $(i-1)$-th rejection.

3. **Path $n+1$: All candidates rejected.** If all n candidates fail verification, the system samples x' directly from the final residual distribution $p_{\text{res},n}(x')$. The probability mass is:

$$P_{\text{fail}}(x') = \left(\prod_{j=1}^{n} (1 - \beta_j) \right) \cdot p_{\text{res},n}(x') \tag{27}$$

B.2 Verification via KL Divergence

According to Theorem 1, the total marginal probability $\hat{p}(x') = \sum_{i=1}^{n} P_i(x') + P_{\text{fail}}(x')$ must identically equal the target distribution $p(x')$. We evaluate this identity by calculating the **Kullback-Leibler (KL) Divergence** between the reconstructed distribution $\hat{p}$ and the ground-truth target distribution p:

$$D_{KL}(\hat{p} \parallel p) = \sum_{x \in \mathcal{X}} \hat{p}(x) \log \frac{\hat{p}(x)}{p(x)} \tag{28}$$

In our implementation, we recorded the full logits for both the draft and target models across various benchmarks. Across all test cases, the measured D_{KL} consistently remained near zero (typically $< 10^{-8}$), which is within the expected range of floating-point numerical error. This empirical result confirms that ResiSpec introduces no distributional bias, ensuring that the accelerated inference remains mathematically equivalent to standard auto-regressive decoding.

C Reproducibility Details

Execution environment. All throughput experiments reported in the revised evaluation are run inside the same Docker environment to avoid differences caused by host-side Python, CUDA, or library versions. For each matched comparison, the baseline and the corresponding ResiSpec variant use the same container image, the same model checkpoints, and the same dataset order. We bind each run to a single GPU with `CUDA_VISIBLE_DEVICES=0`. Baseline and ResiSpec runs are executed sequentially, with one independent Python process per dataset-method pair, so that the GPU memory allocator state and kernel scheduling of one method do not interfere with the other. We record raw logs and parsed CSV summaries for every run, and compute throughput from the number of generated tokens divided by measured wall-clock decoding time after warmup.

Models and datasets. The main fixed-tree experiments use Llama-2-7B as the target model and JackFram/Llama-68M as the draft model. To address experimental-scope concerns, we additionally evaluate Vicuna-7B-v1.3 with Llama-68M, Sheared-LLaMA-2.7B with TinyLlama-1.1B, and Sheared-LLaMA-2.7B with Sheared-LLaMA-1.3B under the same fixed-tree SpecInfer-style setting. The compatibility experiments use Llama-2-7B with Llama-68M for Sequoia, and Llama-2-7B-Chat with the corresponding EAGLE Llama-2-Chat drafter for EAGLE. Unless otherwise stated, all model checkpoints are loaded from the same local HuggingFace cache used by the Docker container. The dataset suite is CNN/DM, OpenWebText, and C4. For the 200-example runs, we use a fixed dataset order and evaluate the same prompt indices for the baseline and ResiSpec under each configuration. EAGLE prompt-generation runs use two warmup prompts followed by 200 measured prompts.

Decoding configurations. For the SpecInfer-style fixed-tree experiments, both the baseline and ResiSpec use the same complete 3-branch tree with depth 5 and a maximum sequence length budget of 512. We evaluate both $T = 0.6$ and $T = 1.0$ with nucleus probability $p = 1.0$. For dynamic-tree Sequoia, we use the original Sequoia tree-selection logic with the same L40 grow maps for the baseline and ResiSpec, and set $M = 384$, $T = 1.0$, and $p = 1.0$ for the 200-example comparison. For EAGLE, we use `total_token=60`, `depth=5`, and `max_new_tokens=128`; the main compatibility table uses `top_k=10`, and the sensitivity rerun with `top_k=15` follows the same Docker, dataset, and sequential-execution protocol.

ResiSpec integration. In fixed-tree SpecInfer-style decoding, ResiSpec reshapes the residual distribution at each sequential verification position in the candidate tree. In Sequoia, ResiSpec is added after Sequoia has selected its dynamic tree, so the tree budget and topology are controlled by the original Sequoia policy and only the verification-time sampling distribution is changed. In EAGLE, ResiSpec is applied only to consecutive candidate siblings that share the same parent node. After each rejection, the implementation updates the current target distribution p and draft/proxy distribution q exactly as in the ResiSpec recurrence before verifying the next sibling.

Randomness control. All sampling-based experiments use fixed pseudo-random seeds. Before each benchmark run, we initialize Python's `random` module, NumPy, PyTorch CPU random state, and PyTorch CUDA random states with the same base seed,

and set cuDNN to deterministic mode. To make per-example sampling reproducible while avoiding identical random streams across prompts, the i-th evaluated prompt uses seed $s + i$, where s is the base seed for that run. The same seed schedule is applied to ResiSpec and the corresponding baseline under each matched model, dataset, temperature, and tree configuration.

Timing and diagnostics. Throughput runs disable auxiliary KL diagnostics and other distribution-checking instrumentation, because these checks require additional logit materialization and are not part of the decoding algorithm. Distribution preservation is evaluated separately using the procedure in Appendix B. For the runtime breakdown in Figure 3, we add CUDA synchronization around the measured regions and report per-speculation-step latency for draft generation, target-model verification, and ResiSpec proxy computation. These profiling runs are used only for component attribution; the synchronized timings are not mixed with the end-to-end throughput measurements.

D Ablation Experiment Results

Candidates (N)	**EAL↑**	**Marginal Gain ($\triangle$EAL)**	**Efficiency (η)**
1	1.89	-	-
2	2.17	0.28	0.280
4	2.48	0.31	0.155
8	2.76	0.28	0.070

Table 5: **Scaling Inefficiency of Multi-Candidate Schemes.** With speculation depth fixed at 4, increasing candidates N yields only modest EAL gains while the efficiency score η drops sharply. Results are averaged over CNN/DM using Llama-2-7B.

Target	Draft	T	Data	Tree	Resi. Acc.	Resi. Thru.	Spec. Acc.	Spec. Thru.
Sheared-2.7B	Sheared-1.3B	1	CNN/DM	d5b3	4.88	78.309	3.88	66.313
Sheared-2.7B	Sheared-1.3B	1	CNN/DM	d4b4	3.94	82.988	3.42	76.695
Sheared-2.7B	Sheared-1.3B	1	OpenWebText	d5b3	4.86	78.447	3.92	67.660
Sheared-2.7B	Sheared-1.3B	1	OpenWebText	d4b4	3.95	83.060	3.45	76.864
Sheared-2.7B	Sheared-1.3B	1	C4	d5b3	4.84	78.362	3.98	68.679
Sheared-2.7B	Sheared-1.3B	1	C4	d4b4	3.95	82.850	3.44	77.340
Sheared-2.7B	Sheared-1.3B	0.6	CNN/DM	d5b3	4.88	78.927	4.12	71.174
Sheared-2.7B	Sheared-1.3B	0.6	CNN/DM	d4b4	3.96	83.110	3.55	79.681
Sheared-2.7B	Sheared-1.3B	0.6	OpenWebText	d5b3	4.89	78.998	4.10	70.922
Sheared-2.7B	Sheared-1.3B	0.6	OpenWebText	d4b4	3.97	81.833	3.54	78.424
Sheared-2.7B	Sheared-1.3B	0.6	C4	d5b3	4.88	77.939	4.13	70.685
Sheared-2.7B	Sheared-1.3B	0.6	C4	d4b4	3.97	82.559	3.51	78.083
Sheared-2.7B	TinyLlama-1.1B	1	CNN/DM	d5b3	4.79	82.170	3.71	68.587
Sheared-2.7B	TinyLlama-1.1B	1	CNN/DM	d4b4	3.92	86.656	3.30	78.432
Sheared-2.7B	TinyLlama-1.1B	1	OpenWebText	d5b3	4.83	82.798	3.69	68.074
Sheared-2.7B	TinyLlama-1.1B	1	OpenWebText	d4b4	3.92	86.496	3.30	78.125
Sheared-2.7B	TinyLlama-1.1B	1	C4	d5b3	4.81	82.645	3.71	68.493
Sheared-2.7B	TinyLlama-1.1B	1	C4	d4b4	3.93	86.656	3.27	77.580
Sheared-2.7B	TinyLlama-1.1B	0.6	CNN/DM	d5b3	4.84	82.508	3.87	71.609
Sheared-2.7B	TinyLlama-1.1B	0.6	CNN/DM	d4b4	3.95	86.957	3.43	81.699
Sheared-2.7B	TinyLlama-1.1B	0.6	OpenWebText	d5b3	4.79	82.441	3.80	70.423
Sheared-2.7B	TinyLlama-1.1B	0.6	OpenWebText	d4b4	3.95	87.395	3.32	79.005
Sheared-2.7B	TinyLlama-1.1B	0.6	C4	d5b3	4.81	82.508	3.79	70.077
Sheared-2.7B	TinyLlama-1.1B	0.6	C4	d4b4	3.93	87.036	3.32	78.858
Sheared-2.7B	Tiny-Vicuna-1B	1	CNN/DM	d5b3	4.81	82.508	3.53	65.274
Sheared-2.7B	Tiny-Vicuna-1B	1	CNN/DM	d4b4	3.93	86.496	3.18	75.429
Sheared-2.7B	Tiny-Vicuna-1B	1	OpenWebText	d5b3	4.81	82.170	3.51	65.050
Sheared-2.7B	Tiny-Vicuna-1B	1	OpenWebText	d4b4	3.93	86.656	3.17	75.302
Sheared-2.7B	Tiny-Vicuna-1B	1	C4	d5b3	4.81	82.441	3.58	66.327
Sheared-2.7B	Tiny-Vicuna-1B	1	C4	d4b4	3.93	86.957	3.21	76.278
Sheared-2.7B	Tiny-Vicuna-1B	0.6	CNN/DM	d5b3	4.78	81.967	3.71	68.823
Sheared-2.7B	Tiny-Vicuna-1B	0.6	CNN/DM	d4b4	3.95	87.103	3.25	77.519
Sheared-2.7B	Tiny-Vicuna-1B	0.6	OpenWebText	d5b3	4.79	82.051	3.64	67.510
Sheared-2.7B	Tiny-Vicuna-1B	0.6	OpenWebText	d4b4	3.95	86.957	3.21	76.588
Sheared-2.7B	Tiny-Vicuna-1B	0.6	C4	d5b3	4.79	81.882	3.65	67.660
Sheared-2.7B	Tiny-Vicuna-1B	0.6	C4	d4b4	3.95	87.191	3.24	77.058
Llama-2-7B	JF68M	1	CNN/DM	d5b3	4.68	95.057	2.51	56.851
Llama-2-7B	JF68M	1	CNN/DM	d4b4	3.88	84.317	2.47	58.962
Llama-2-7B	JF68M	1	OpenWebText	d5b3	4.69	94.004	2.51	56.980
Llama-2-7B	JF68M	1	OpenWebText	d4b4	3.87	84.246	2.47	59.172
Llama-2-7B	JF68M	1	C4	d5b3	4.67	94.429	2.64	59.952
Llama-2-7B	JF68M	1	C4	d4b4	3.89	84.674	2.55	60.976
Llama-2-7B	JF68M	0.6	CNN/DM	d5b3	4.14	84.603	2.59	58.997
Llama-2-7B	JF68M	0.6	CNN/DM	d4b4	3.77	82.035	2.47	59.172
Llama-2-7B	JF68M	0.6	OpenWebText	d5b3	4.16	84.531	2.49	56.690
Llama-2-7B	JF68M	0.6	OpenWebText	d4b4	3.75	81.566	2.42	58.072
Llama-2-7B	JF68M	0.6	C4	d5b3	4.13	84.388	2.66	60.570
Llama-2-7B	JF68M	0.6	C4	d4b4	3.74	81.499	2.57	61.652

Notes:
T: temperature; Tree: prediction tree shape; Acc.: accepted length; Thru.: tokens per second.

Table 6: Ablation Experiment Results of Different Model Pair Combinations. Here, we use Llama-2-7B and Sheared-LLaMA-2.7B (Xia et al. 2023) as the target model and Sheared-LLaMA-1.3B, TinyLlama-1.1B (Zhang et al. 2024), Tiny-Vicuna-1B and JF68M as the draft model. The experimental results show that ResiSpec significantly outperforms SpecInfer in terms of accepted length and throughput under different configurations.