DiffuSpec: Unlocking Diffusion Language Models for Speculative Decoding

Guanghao Li[1,2*], Zhihui Fu[3], Min Fang[3], Qibin Zhao[3],
Ming Tang[2†], Chun Yuan[1†], Jun Wang[3]
[1]Tsinghua Shenzhen International Graduate School, Tsinghua University
[2]Southern University of Science and Technology [3]OPPO Research Institute

Abstract

Autoregressive (AR) decoding in large language models (LLMs) is latency-bounded by strictly sequential token generation. Speculative decoding mitigates this bottleneck by letting a fast drafter propose multi-token candidates that are then verified in parallel by the target model; yet most existing systems still rely on AR drafters, limiting wall-clock gains. We present **DiffuSpec**, which repurposes a *diffusion language model* (DLM) as a *parallel* drafter to generate multi-token proposals in a single forward pass while remaining compatible with standard AR verifiers. However, DLM drafting presents unique challenges: **1)** bidirectional conditioning produces a token lattice where locally optimal tokens may fail to form a valid causal sequence; **2)** the mechanism requires tuning the draft length, which induces a speed–quality trade-off. To address these issues, we introduce (i) *Causal-consistency Path Search* (CPS) to extract verifier-aligned causal paths from the lattice, and (ii) an *Adaptive Draft-Length* (ADL) controller that adjusts proposal lengths using online acceptance feedback. Across benchmarks, DiffuSpec achieves up to $3\times$ wall-clock speedup and consistently outperforms strong baselines, demonstrating diffusion-based drafting as a competitive alternative to AR drafters for speculative decoding.

1 Introduction

Large language models (LLMs) continue to improve with scale, yet autoregressive (AR) decoding remains a latency bottleneck because generating K tokens requires K serial forward passes (Leviathan et al., 2023; Hoffmann et al., 2022). A common line of work accelerates inference via pruning and sparsity, quantization, or knowledge distillation, but these techniques often introduce accuracy trade-offs or additional engineering complexity (Frantar et al., 2022; Frantar and Alistarh, 2023; Xu et al., 2024). Speculative decoding offers a nearly lossless alternative: a fast drafter first proposes multi-token candidates, and then the target model verifies them in parallel, which preserves the target distribution while reducing wall-clock time (Xia et al., 2024). However, the speedup hinges on two factors: the drafter's per-step drafting throughput and the verification acceptance rate.

In practice, prior speculative decoding improves one of these factors by drafting faster with smaller autoregressive drafters or retrieval-based proposals (Leviathan et al., 2023; Chen et al., 2023; He et al., 2023; Saxena, 2023), which can limit acceptance, or by increasing acceptance via training or calibrating the drafter to better match the target model (e.g., EAGLE-style) (Li et al., 2024b,a, 2025). Yet most drafters remain autoregressive and still require one forward pass per drafted token, so draft generation can become a limiting factor for end-to-end speedups.

Recent advances in diffusion language models (DLMs) (Li et al., 2022) offer a promising path toward *parallel* speculative drafting: a DLM can propose a block of per-position token candidates in a single forward pass, enabling highly parallel draft generation. In addition, several DLMs (Fig. 1b), such as Dream-style models, are obtained by fine-tuning autoregressive LMs and thus tend to be well aligned with same-family AR targets, often yielding high acceptance in practice (Ye et al., 2025). Together, these properties match the drafter desiderata—high drafting throughput and strong proposal quality—making DLMs a compelling fit for efficient AR verification.

However, DLM proposals are generated under bidirectional conditioning rather than strict left-to-right causality. This induces a diffusion token lattice over per-position candidates, where the locally highest-probability token at each position need not form a causal left-to-right path. In addition, DLM

*E-mail: ligh24@mails.tsinghua.edu.cn
†Corresponding authors (tangm3@sustech.edu.cn, yuanc@sz.tsinghua.edu.cn)

Findings of the Association for Computational Linguistics: ACL 2026, pages 20896–20910

drafting requires specifying a draft length in advance. Consequently, we study two practical questions: (i) **causal alignment**: how to select, from this lattice, a left-to-right path aligned with AR verification to maximize acceptance; and (ii) **draft length**: how to choose the block size to balance drafting cost against verification acceptance, since longer drafts increase proposal cost without guaranteeing higher acceptance. While prior work such as SpecDiff (Christopher et al., 2024) has begun to explore diffusion-based drafters, a systematic treatment of causal alignment and draft-length selection in this setting remains under-explored.

To address these two issues, we present **DiffuSpec**, which reuses a DLM as the drafter in place of the usual autoregressive model and wraps it with two lightweight components: (i) a *causal-consistency path search* (CPS) over the diffusion token lattice that selects a left-to-right path aligned with AR verification to improve acceptance; and (ii) an *adaptive draft-length* (ADL) controller that sets the next draft length based on recent acceptance statistics and the realized generated length. DiffuSpec is implemented as a plug-in drafter module on top of an SPS-style speculative decoding interface (Leviathan et al., 2023): it only replaces the drafter side of the standard drafter–verifier interface, requires no architectural changes to the target model, and integrates into existing serving stacks with minimal modification. Across diverse generation tasks, DiffuSpec delivers up to $3\times$ wall-clock speedup over strong baselines.

In summary, our main contributions are:

- We study diffusion language models (DLMs) as drafters for speculative decoding and identify two key challenges caused by bidirectional conditioning and a preset draft length.
- We propose **DiffuSpec** with two lightweight components: CPS to extract verifier-aligned causal paths from the diffusion lattice, and an ADL controller to adapt proposal lengths from recent acceptance feedback.
- We achieve up to $3\times$ wall-clock speedup across tasks, consistently outperforming strong speculative decoding baselines.

2 Related Work

Speculative decoding. Speculative decoding accelerates autoregressive (AR) generation by letting a fast *drafter* propose multiple tokens for parallel verification by a target LLM, while preserving the target distribution (Xia et al., 2024; Sun et al., 2025). Existing methods mainly differ in the drafter and verification design. One line uses smaller pretrained AR drafters (Leviathan et al., 2023; Chen et al., 2023) or *retrieval/cache*-based proposals (He et al., 2023; Saxena, 2023), often paired with verifier-side optimizations such as block verification and parallel cache-tree validation (Sun et al., 2024; Miao et al., 2024; Svirschevski et al., 2024). Another line performs *lookahead* updates without an auxiliary drafter (Fu et al., 2024). A third line predicts multiple tokens per step via multi-token prediction (MTP) heads (Cai et al., 2024; Ankner et al., 2024) or trains feature/token-level drafters (Li et al., 2024b,a, 2025). Notably, MTP schemes attach auxiliary heads to predict several future tokens from the current AR state, but decoding still advances one MTP step at a time; lookahead is bounded by head number/depth, so end-to-end speedups remain constrained by the AR backbone. Overall, the first two lines can suffer from low acceptance under mismatch or weak retrieval, while the third requires additional training/interface changes and remains tied to autoregressive computation with non-negligible drafting latency.

Diffusion language models. Discrete/latent diffusion for text ranges from early D3PMs (Austin et al., 2021) and Diffusion-LM (Li et al., 2022) to hybrids with pretrained LMs (Zhou et al., 2023; He et al., 2022) and recent scaling/adaptation frameworks (Gong et al., 2024). Large pretrained DLMs have been reported to be competitive with similarly sized AR baselines while retaining diffusion-style parallel refinement (Nie et al., 2025; Ye et al., 2025). In particular, Dream-like models (Ye et al., 2025) are finetuned from strong AR LMs in the same family, so their token distributions remain well aligned with larger AR targets, making them especially suitable as drafters for speculative decoding. At inference time, DLMs natively support parallel multi-token updates with iterative refinement but pay for bidirectional attention and multiple denoising steps; this has motivated deployment-time accelerators such as adaptive KV caching, dynamic cache eviction, and suffix-dropout pruning (Liu et al., 2025; Song et al., 2025; Chen et al., 2025). These traits—single- or few-pass proposal of token blocks together with strong proposal quality—make DLMs promising candidates as drafters for speculative decoding.

Diffusion as a drafter for speculative decoding. Christopher et al. (2024) first showed that a dis-

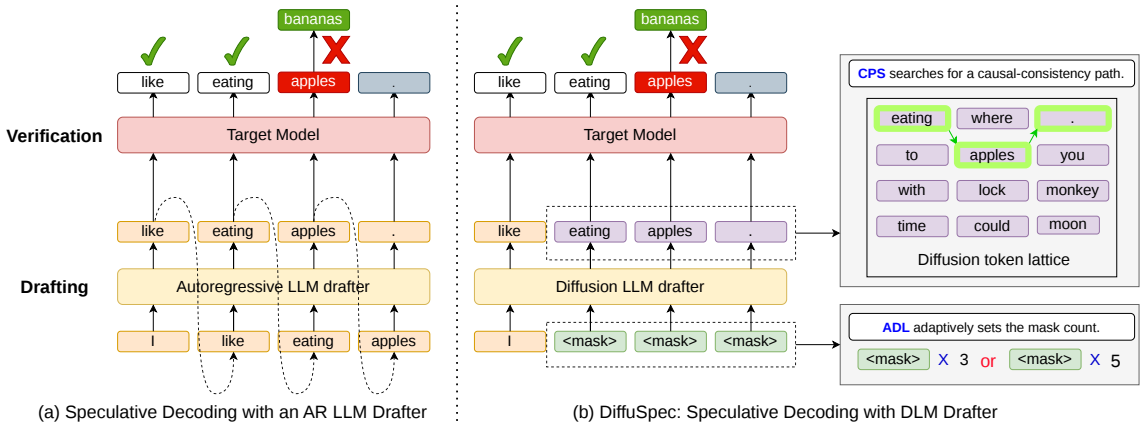


Figure 1: **Speculative decoding: AR vs. DiffuSpec.** (a) **AR drafter:** drafts are produced sequentially and then block-verified by the target AR model. (b) **DiffuSpec (DLM drafter):** a single forward pass proposes a block for one-shot parallel verification; within DiffuSpec, causal-consistency path search (CPS) selects a left-to-right path from the diffusion token lattice, and the adaptive draft-length (ADL) controller sets the next draft length by selecting how many masked positions to fill.

crete diffusion model can draft sequences for AR verification, validating the feasibility of diffusion-based drafting. However, prior work lacks a systematic analysis of how draft length and the diffusion-induced token lattice with relaxed causality interact with AR verification. In contrast, **DiffuSpec** reuses existing DLMs as drafters and introduces (a) a *causal-consistency path search* (CPS) over the diffusion-induced token lattice and (b) an *adaptive draft-length* (ADL) controller, which together improve accepted prefixes and wall-clock speedups under AR block verification.

3 Preliminaries—Speculative Decoding

Let p_θ be the target autoregressive (AR) language model and $\mathbf{x}_{1:j}$ the current prefix. Speculative decoding (Leviathan et al., 2023; Chen et al., 2023; Xia et al., 2024) accelerates generation under a *drafter–verifier* interface: a fast drafter proposes a short continuation, and the target AR model verifies it in parallel while preserving the p_θ distribution.

Drafting. Given $\mathbf{x}_{1:j}$, a drafter q_ϕ proposes a length-k_t block $\hat{\mathbf{y}}_{j+1:j+k_t} = (\hat{y}_{j+1}, \ldots, \hat{y}_{j+k_t})$ conditioned on $\mathbf{x}_{1:j}$, and records per-position conditional probabilities $\{q_\phi(\hat{y}_{j+i} \mid \mathbf{x}_{1:j+i-1})\}_{i=1}^{k_t}$. Here $t = 1, 2, \ldots$ indexes speculative steps.

Parallel verification. The target model evaluates the drafted tokens in a single parallel pass, producing $\{p_\theta(\hat{y}_{j+i} \mid \mathbf{x}_{1:j+i-1})\}_{i=1}^{k_t}$, and then processes them left-to-right. For each position $i \in \{1, \ldots, k_t\}$, the standard acceptance rule is

$$\alpha_{t,i} = \min\left(1, \frac{p_\theta(\hat{y}_{j+i} \mid \mathbf{x}_{1:j+i-1})}{q_\phi(\hat{y}_{j+i} \mid \mathbf{x}_{1:j+i-1})}\right). \quad (1)$$

If $\hat{y}_{j+i}$ is rejected, a replacement is sampled from the residual distribution proportional to $\big[p_\theta(\cdot \mid \mathbf{x}_{1:j+i-1}) - q_\phi(\cdot \mid \mathbf{x}_{1:j+i-1})\big]_+$, where $[u]_+ = \max(u, 0)$, followed by normalization; all remaining drafted tokens are discarded before continuing. This procedure is unbiased with respect to p_θ (Leviathan et al., 2023, App. A.1) and admits verifier-side engineering such as block or tree-based parallel verification to further reduce latency (Sun et al., 2024; Miao et al., 2024).

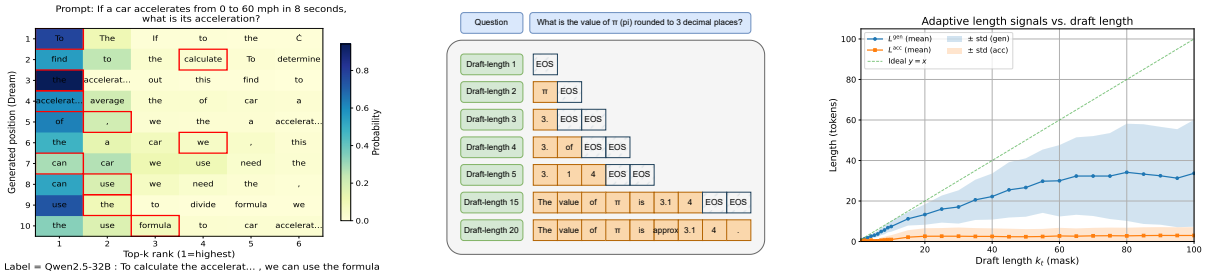
Accepted prefix length. At speculative step t with proposal length k_t, let $A_{t,i} \in \{0, 1\}$ indicate whether the i-th drafted token is accepted sequentially by the verifier. Define $I_{t,i} = \mathbf{1}\{A_{t,1} = \cdots = A_{t,i} = 1\}$ with $I_{t,0} = 1$. Then the number of tokens actually committed is

$$L_t^{\mathrm{acc}} = \sum_{i=1}^{k_t} I_{t,i} = \sum_{i=1}^{k_t} \prod_{r=1}^{i} A_{t,r}. \quad (2)$$

The verifier appends the accepted prefix $\hat{\mathbf{y}}_{j+1:j+L_t^{\mathrm{acc}}}$ and discards the remainder, yielding the updated prefix $\mathbf{x}_{1:j+L_t^{\mathrm{acc}}}$. Decoding terminates early if an EOS token is accepted. We use L_t^{acc} as a per-step measure of useful progress; holding latency fixed, larger values imply higher speedup.

4 DiffuSpec

As shown in Fig. 1b, **DiffuSpec** departs from conventional speculative decoding by replacing the AR drafter with a diffusion language model (DLM) that proposes a length-k_t draft in a single forward pass, and by augmenting drafting with *causal-consistency path search* (CPS) and an *adaptive*



(a) **Token-mass diffusion.** Joint refinement spreads probability across positions; top-1 per position may break causality.

(b) **Effect of draft length.** Larger k_t yields more complete drafts, until an early EOS truncates further content.

(c) **Signals vs. k_t.** Mean±std of L^{gen} and L^{acc}; $y{=}x$ marks the ideal should-generate line.

Figure 2: **Key observations motivating CPS and ADL.**

draft-length (ADL) controller. We next describe these three components in turn.

4.1 DLM as a drafter

Unlike autoregressive models with fixed left-to-right factorization, diffusion language models learn a non-autoregressive denoising mapping that reconstructs clean text from corrupted text (Austin et al., 2021; Gong et al., 2024; Nie et al., 2025; Ye et al., 2025; Chen et al., 2025).

Training. Let $\mathbf{x}^{(0)}$ be a clean sequence and $\mathbf{x}^{(\eta)}$ its corrupted counterpart at noise level $\eta \in [0,1]$. We define a forward corruption kernel r with a user-specified discrete noise prior π_{noise}; specifically, $r\left(x_i^{(\eta)} \mid x_i^{(0)}\right) = (1-\eta)\,\mathbf{1}\{x_i^{(\eta)} = x_i^{(0)}\} + \eta\,\pi_{\text{noise}}\left(x_i^{(\eta)}\right)$, where $\sum_v \pi_{\text{noise}}(v) = 1$ (e.g., all mass on [MASK] or a mixture over noise symbols). A parameterized denoiser q_ϕ is trained with token-wise cross-entropy to predict originals at corrupted positions. We sample $\eta \sim \mathcal{U}[0,1]$, $\mathbf{x}^{(0)} \sim \mathcal{D}$, and $\mathbf{x}^{(\eta)} \sim r(\cdot \mid \mathbf{x}^{(0)}, \eta)$. Let $\mathcal{I}(\eta) = \{i : x_i^{(\eta)} \neq x_i^{(0)}\}$ denote the corrupted positions.

$$\mathcal{L}(\phi) = -\,\mathbb{E}\left[\sum_{i\in\mathcal{I}(\eta)} \log q_\phi\big(x_i^{(0)} \mid \mathbf{x}^{(\eta)}\big)\right]. \quad (3)$$

Here q_ϕ is a bidirectional attention transformer.

Inference (iterative refinement). Given a prefix $\mathbf{p} = \mathbf{x}_{1:j}$ and target length k_t, we initialize $\mathbf{y}^{(0)} = \mathbf{p} \circ (\texttt{[MASK]})^{k_t}$ and set $\Omega_0 = \{j{+}1, \ldots, j{+}k_t\}$, where $\circ$ denotes concatenation. For refinement steps $s = 1, \ldots, S$, we compute per-position distributions $\{q_\phi(y_i \mid \mathbf{y}^{(s-1)})\}_{i\in\Omega_{s-1}}$, choose an update subset $U_s \subseteq \Omega_{s-1}$ (e.g., top-K by confidence), and update

$$y_i^{(s)} = \begin{cases} \arg\max\limits_{v\in\mathcal{V}} q_\phi\big(y_i{=}v \mid \mathbf{y}^{(s-1)}\big), & \text{if } i \in U_s, \\ y_i^{(s-1)}, & \text{otherwise.} \end{cases} \quad (4)$$

We then remove updated positions by setting $\Omega_s \leftarrow \Omega_{s-1} \setminus U_s$, and repeat until $\Omega_s = \varnothing$. By default, we use a single refinement pass ($S{=}1$) to isolate drafting cost; we ablate $S{>}1$ in Sec. 5.

Integration with speculative decoding. At speculative step t with prefix $\mathbf{x}_{1:j}$, a DLM proposes a length-k_t block $\hat{\mathbf{y}}_{j+1:j+k_t}$ in essentially one refinement pass, and exposes per-position top-m candidate sets with log-scores under the current draft context $\mathbf{y}^{(S)}$. For verifier-side acceptance with a DLM drafter, we approximate left-to-right conditionals by masking all in-block positions except the token at $j{+}i$; namely, we mask the first $i{-}1$ (past) and the remaining $k_t{-}i{+}1$ (future) draft slots.

$$q_\phi^{\text{L2R}}(v \mid \mathbf{x}_{1:j+i-1}) := q_\phi\Big(v \,\Big|\, \mathbf{x}_{1:j} \circ (\texttt{[MASK]})^{i-1} \circ (\texttt{[MASK]})^{k_t-i+1}\Big). \quad (5)$$

We use q_ϕ^{L2R} in the standard acceptance ratio. Accordingly, Sec. 4.2 introduces **CPS** to align proposals causally with the AR verifier, and Sec. 4.3 presents **ADL** to set k_t near the speed–quality sweet spot.

4.2 Causal-Consistency Path Search (CPS)

Phenomenon and motivation. Under relaxed causality, the DLM refines tokens jointly within a block. As a result, token probability mass spreads across positions and the per-position top-1 chosen by the DLM is not necessarily the best left-to-right choice for the AR verifier p_θ (Fig. 2a). To mitigate this mismatch, we explicitly search before verification—for a left-to-right path that is both

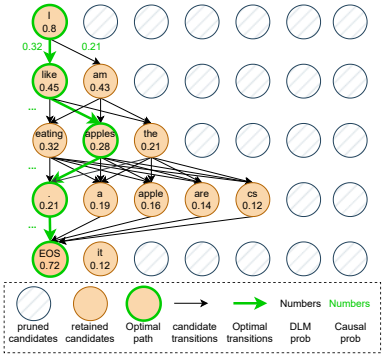


Figure 3: **Pruned candidate lattice and CPS.** We keep tokens via a cumulative-mass threshold τ (e.g., 0.8), always retain EOS, early-stop after the first EOS, and select the best path using a DLM score plus a causal (n-gram) proxy.

high-confidence under the DLM and fluent under a causal proxy (Fig. 3).

Lattice and pruning. We first specify the search space. From the final DLM pass, for each position $i = 1{:}k_t$ we extract a candidate set $\mathcal{C}_{j+i}$ (top-M) with log-scores $\ell^{\text{dlm}}_{j+i}(v) = \log q_\phi\big(v \mid \mathbf{x}_{1:j}, \mathbf{y}^{(S)}_{\setminus(j+i)}\big)$, i.e., conditioning on the current draft context except the target position. The naive Cartesian product over $\{\mathcal{C}_{j+i}\}_{i=1}^{k_t}$ is exponential, so we apply a training-free, mass-adaptive pruning rule that respects local uncertainty. Let $p_{j+i}(v) = \exp(\ell^{\text{dlm}}_{j+i}(v))$. We retain the smallest number of candidates whose cumulative probability mass exceeds a threshold τ, capped by $M_{\max}$. Let $\text{Top}{-}m$ denote the *set* of the m highest-probability tokens within $\mathcal{C}_{j+i}$ ranked by $p_{j+i}(\cdot)$. Here $M_{\max}$ is a global cap on the per-position candidate size after pruning, i.e., $|\mathcal{C}_{j+i}| \le M_{\max}$ for all i.

$$M_i = \min\Big\{m \ge 1 : \sum_{v \in \text{Top}{-}m} p_{j+i}(v) \ge \tau\Big\}. \quad (6)$$

We set $M_i \leftarrow \min(M_i, M_{\max})$ and prune $\mathcal{C}_{j+i} \leftarrow \text{Top} - M_i$. This makes $|\mathcal{C}_{j+i}|$ entropy-adaptive—peaky positions keep few candidates; flatter ones keep more, capped by $M_{\max}$. In addition, we stop expanding once the first EOS is placed: diffusion proposals tend to pad with EOS after the content is "complete" (qualitative trend in Fig. 2b), so exploring beyond the first EOS rarely yields causal gains.

Scoring and search. Let $m_{\max}$ denote the depth up to (and including) the first EOS encountered during expansion. Given the pruned lattice, we score $\pi = (\pi_1, \ldots, \pi_m)$ by combining DLM confidence with a small causal proxy (e.g., an n-gram or a tiny causal LM):

$$\mathcal{S}(\pi) = \sum_{i=1}^{m} \Big[\lambda\, \ell^{\text{dlm}}_{j+i}(\pi_i) + (1-\lambda)\, \ell^{\text{ng}}_{j+i}(\pi_{1:i})\Big]. \quad (7)$$

where ℓ^{ng}_{j+i} is the causal proxy log-score of $\mathbf{x}_{1:j} \circ \pi_{1:i}$ and $\lambda \in [0,1]$ trades off between DLM confidence and causal fluency. We then run left-to-right beam search (beam B) on the pruned lattice until EOS is placed. If $\bar{C}$ denotes the average branching factor after pruning, the per-step complexity is $O(B\,\bar{C}\,m_{\max})$. As $\tau \to 1$ and B increases, the result approaches the unpruned optimum.

Effect. By entropy-adaptive pruning, early stopping at the first EOS, and the causal–denoising score in (7), CPS pushes the first p_θ–q_ϕ mismatch farther to the right, thereby increasing the expected accepted length L^{acc}_t and improving end-to-end speed.

4.3 Adaptive Draft Length (ADL)

Phenomenon and motivation. Draft length k_t jointly determines drafting cost, proposal quality, and verifier acceptance. Short drafts often yield terse fragments; moderate drafts capture more complete reasoning; very long drafts saturate content and trigger early EOS while also accumulating off-path tokens that the verifier rejects (Fig. 2b). Empirically, the EOS-aware generation length L^{gen} increases with k_t and then saturates, and the accepted length L^{acc} tracks it (Fig. 2c). The saturation point, however, is instance dependent and varies across prompts and along the trajectory, leading to large variance as shown in Fig. 2c. A fixed k_t therefore either wastes compute when too long or throttles progress when too short, which motivates an adaptive controller.

Signals. Given the drafted block $\hat{\mathbf{y}}_{j+1:j+k_t}$, let s_t be the index of the first EOS (or $+\infty$ if none) and define the EOS-aware generation signal

$$L^{\text{gen}}_t = \min(s_t - 1,\ k_t). \quad (8)$$

We compute s_t from the raw DLM draft before CPS; since CPS also early-stops at the first EOS, both signals are aligned. After parallel verification we obtain the accepted prefix length L^{acc}_t as defined in Sec. 3. To reduce volatility from occasional early EOS or transient rejections, we use exponential moving averages with $\rho \in (0,1]$:

$$\begin{aligned} \tilde{L}^{\text{gen}}_t &= (1-\rho)\tilde{L}^{\text{gen}}_{t-1} + \rho L^{\text{gen}}_t, \\ \tilde{L}^{\text{acc}}_t &= (1-\rho)\tilde{L}^{\text{acc}}_{t-1} + \rho L^{\text{acc}}_t. \end{aligned} \quad (9)$$

These smoothed signals provide stable feedback for adjusting the proposal length k_t.

Controller. With guardrails $k_{\min} \le k_{t+1} \le k_{\max}$, we adopt a one-line $O(1)$ policy:

$$\hat{k}_{t+1} = \left\lceil \tilde{L}_t^{\text{gen}} + \delta \, \mathbf{1}\{\tilde{L}_t^{\text{acc}} \ge \tilde{L}_t^{\text{gen}}\} \right\rceil, \quad k_{t+1} = \text{clip}(\hat{k}_{t+1}, k_{\min}, k_{\max}). \tag{10}$$

where $\text{clip}(z, a, b) = \min\{\max\{z, a\}, b\}$ and $\delta > 0$ is a small growth increment that activates when the verifier keeps up, namely when the accepted length matches the generated length on average. Intuitively, $\tilde{L}_t^{\text{gen}}$ estimates how much content the DLM is likely to produce before EOS, and $\tilde{L}_t^{\text{acc}}$ reflects whether those tokens are reliably accepted; therefore, we increase k_t only when both signals consistently agree.

Effect. ADL tracks the instance-specific speed–quality sweet spot in real time. As k_t grows into the saturation regime, L_t^{gen} plateaus and the controller stabilizes; when acceptance lags, the policy avoids oversizing drafts; when acceptance catches up, it expands gently via δ.

4.4 Serving-compatible framework

As summarized in Fig. 1b and Alg. 1, each speculative step in DiffuSpec follows a fixed four-stage pipeline with no changes to the target model and only minimal serving-stack adjustments: *(i) Drafting* with a DLM to produce a length-k_t block and per-position candidates; *(ii) CPS* on a pruned candidate lattice to select a left-to-right path aligned with AR causality; *(iii) Parallel verification* by the target p_θ (using q_ϕ^{L2R} in the acceptance ratio) to return the accepted prefix length L_t^{acc} and advance the prefix; *(iv) ADL* to update the next draft length k_{t+1} from the signal L_t^{gen} and verifier feedback L_t^{acc}, within guardrails $[k_{\min}, k_{\max}]$. By improving the acceptance profile via CPS and right-sizing proposals via ADL, DiffuSpec increases L_t^{acc} per step while keeping drafting cost near the speed–quality sweet spot. In particular, DiffuSpec can be plugged into existing speculative decoding stacks by swapping the AR drafter with a DLM drafter and attaching CPS/ADL as lightweight pre-/post-processing modules at the standard drafter–verifier interface. Moreover, our main experiments are conducted under greedy verification (temperature $= 0$). In this quality-locked regime, DiffuSpec is lossless with respect to plain AR-greedy decoding of the target model, as formalized below.

Proposition 4.1 (Lossless greedy equivalence under greedy verification)**.** *Let $p_\theta(\cdot \mid x_{<t})$ be the target autoregressive verifier, and let*

$$g(x_{<t}) := \arg\max_{v \in \mathcal{V}} p_\theta(v \mid x_{<t})$$

denote its greedy next-token rule. Under greedy verification (temperature $= 0$), for any draft block proposed by DiffuSpec—including CPS with beam/MAP-style search—the resulting decoded sequence is identical, token-by-token, to plain AR-greedy decoding of p_θ.

The full proof and the discussion of stochastic decoding are provided in Appendix H.

5 Experiments

Datasets. We evaluate DiffuSpec on Spec-Bench (Xia et al., 2024), which spans six task families: *Multi-turn Conversation* (MT; Zheng et al., 2023), *Machine Translation* (Trans), *Summarization* (Sum; Nallapati et al., 2016), *Open-domain QA* (QA; Kwiatkowski et al., 2019), *Mathematical Reasoning* (Math; Cobbe et al., 2021), and *Retrieval-Augmented Generation* (RAG; Karpukhin et al., 2020). See Appendix A for dataset details.

Speed metrics. We report (i) *Mean Accepted Tokens (MAT)*, the expected length of consecutively accepted prefixes per speculative step, averaged over all steps and examples; and (ii) *Speedup*, defined as end-to-end throughput relative to the AR-greedy baseline on the same target model and hardware. All timings are wall-clock and account for DLM drafting, CPS, ADL, and parallel verification. To ensure comparable quality (quality-locked setting), verification is performed with greedy decoding (temperature $= 0$), yielding task metrics statistically indistinguishable from AR-greedy.

Baselines. We compare DiffuSpec against a range of speculative decoding methods, including SPS (Leviathan et al., 2023), Lookahead (Fu et al., 2024), PLD (Saxena, 2023), Recycling (Luo et al., 2024), SAMD (Hu et al., 2024), EAGLE2 (Li et al., 2024b,a), EAGLE3 (Li et al., 2025), and SpecDiff (Christopher et al., 2024). To isolate the effect of our ADL controller, we additionally evaluate a Minions-style variant that is identical to DiffuSpec, except that ADL is replaced with the length controller proposed in Minions (Wang et al., 2024). We refer to this variant as *Minions*. All methods are run in our unified evaluation stack under the same hardware and decoding configuration.

Method	Speedup (× vs. AR, ↑)						Mean (MAT / Speedup)
	MT	Trans	Sum	QA	Math	RAG	
Lookahead	1.37×	1.16×	1.15×	1.33×	1.52×	1.21×	1.82 / 1.30×
PLD	1.83×	1.29×	2.76×	1.87×	1.55×	2.37×	2.11 / 1.93×
Recycling	2.15×	1.85×	2.03×	2.06×	2.45×	1.83×	3.13 / 2.07×
SAMD	1.99×	1.54×	**3.38**×	2.44×	1.63×	**3.27**×	2.18 / 2.35×
EAGLE2	2.47×	1.56×	1.64×	1.91×	3.18×	1.64×	3.47 / 2.09×
EAGLE3	3.01×	2.35×	3.03×	2.41×	3.12×	2.86×	4.28 / 2.80×
SPS	1.69×	1.64×	1.74×	1.50×	1.86×	1.62×	6.18 / 1.67×
SpecDiff	2.65×	2.61×	1.96×	2.41×	2.95×	2.02×	6.05 / 2.69×
Minions	3.02×	3.18×	2.37×	2.93×	3.91×	2.29×	6.44 / 2.97×
DiffuSpec	**3.09**×	**3.38**×	2.41×	**3.03**×	**4.02**×	2.38×	**6.99 / 3.08**×

Table 1: **Main results on Spec-Bench.** Per-task columns report *Speedup* only; the rightmost column reports the macro *Mean (MAT / Speedup)*.

CPS	ADL	Mean-MAT	Mean-Speedup
✓	✓	**6.99**	**3.08**×
✓	✗	6.95	2.98×
✗	✓	6.43	2.73×
✗	✗	6.05	2.69×

Table 2: **Ablation on components.**

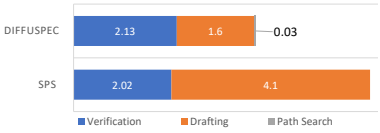


Figure 4: **Per-step wall-clock time (s).** Mean seconds per round in drafting, verification, and CPS.

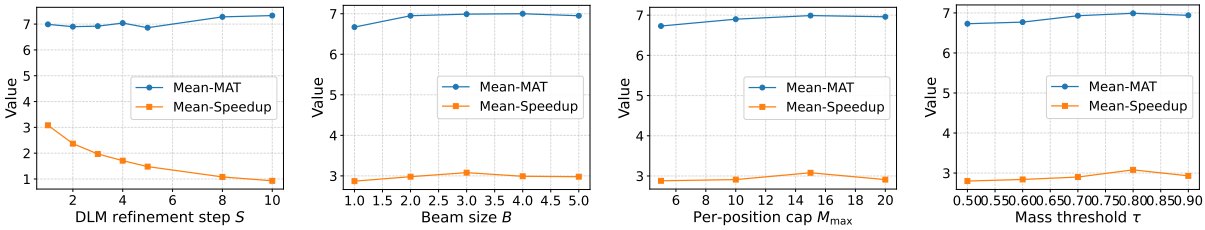
Targets and drafters. Unless otherwise stated, we use publicly available checkpoints throughout to ensure reproducibility and fair comparisons. The target AR model p_θ is Qwen2.5-32B (Xu et al., 2025) for all methods, including ours. DiffuSpec uses Dream-7B (Ye et al., 2025) as the diffusion drafter; SpecDiff and Minions also use Dream-7B for a fair comparison among diffusion-based approaches. For SPS, we follow the standard setup with Qwen2.5-7B as the autoregressive drafter. Lookahead, PLD, Recycling, and SAMD do not use a separate drafter, operating on the target model or its cache/retrieval stack. EAGLE2 and EAGLE3 augment Qwen2.5-32B with trained drafter heads; we follow their official repositories and use publicly released community checkpoints. Overall, this setup fixes the verifier across methods while using the standard drafter for each baseline.

Implementation details. Experiments run on a single NVIDIA A100 (80GB) with 11 CPU cores and 100GB RAM, PyTorch 2.6.0. Following prior work (Kou et al., 2024; Luo et al., 2024), verification uses greedy decoding (temperature $= 0$) with batch size $= 1$ and KV cache enabled. Unless otherwise stated, all reported results are obtained under this quality-locked setting; thus, by Proposition 4.1, DiffuSpec is lossless with respect to plain AR-greedy decoding of the target model, and our comparisons measure acceleration at matched output quality. Unless stated, DiffuSpec uses a single diffusion refinement step (S=1) to isolate drafting cost. Controller and search hyperparameters are fixed across tasks: $k_{\min}$=20, $k_{\max}$=30, beam size B=3, mass threshold τ=0.8, per-position cap $M_{\max}$=15, mixing weight λ=0.5, controller increment δ=10 tokens, and EMA smoothing ρ=0.5. The causal proxy is a 3-gram KenLM trained offline on a text corpus and reused across all tasks.

5.1 Effectiveness

Overall comparison. Tab. 1 summarizes wall-clock speedups on Spec-Bench. DiffuSpec achieves the best overall performance, with a Mean-MAT of 6.99 and a Mean-Speedup of 3.08×, outperforming all speculative decoding baselines under the same Qwen2.5-32B target, hardware, and decoding configuration. Compared to strong autoregressive competitors, DiffuSpec improves both quality and efficiency (e.g., vs. SPS: +0.81 MAT and +1.41× speedup; vs. EAGLE3: +2.71 MAT and +0.28× speedup), even though EAGLE3 requires additional training and architectural modifications. At the task level, DiffuSpec attains the highest speedups on *MT/Trans/QA/Math*, with 3.09 × /3.38 × /3.03 × /4.02×, indicating consistently longer accepted prefixes and faster end-to-end progress at matched quality.

Comparison to diffusion-based and adaptive-length baselines. Among diffusion-based methods, SpecDiff already demonstrates that a DLM can serve as an effective drafter, but DiffuSpec consistently further improves both metrics under an identical Qwen2.5-32B + Dream-7B configuration (Mean-MAT: 6.99 vs. 6.05; Mean-Speedup: 3.08× vs. 2.69×). This gain aligns well with our design goals: CPS explicitly extracts a causal left-to-right path from the diffusion-induced token lattice, while ADL adaptively steers the draft length toward the speed–quality sweet spot. Replacing ADL with a Minions-style acceptance-only controller yields the "Minions" row in Tab. 1; DiffuSpec still achieves notably higher Mean-MAT (6.99 vs. 6.44) and Mean-Speedup (3.08× vs. 2.97×), with uniformly consistent improvements across all task families (e.g., 3.09× vs. 3.02× on MT and 4.02× vs. 3.91× on Math).



(a) S refinement steps (b) B beam size (c) $M_{\max}$ per-position cap (d) τ mass threshold

Figure 5: **Sensitivity to decoding/search hyperparameters.** Each panel plots *Mean-MAT* and *Mean-Speedup* versus a single knob under the quality-locked setting.

Where the speedup comes from. Fig. 4 decomposes wall-clock time into *drafting*, *verification*, and *CPS search*. DiffuSpec reduces drafting cost relative to SPS by using a single DLM forward pass to propose multiple tokens, while CPS adds only minor overhead (averaging 1.1% across tasks). In our setup, SPS employs a 7B AR drafter close to the target's capacity; the resulting sequential passes frequently dominate wall-clock and partially blunt the benefits of verifier parallelism—MAT remains relatively high, yet end-to-end speedup is still modest in practice. By contrast, DiffuSpec with Dream-7B achieves consistently larger speedups at comparable or higher MAT by combining two levers: (i) higher per-step drafting throughput (a non-AR DLM pass) and (ii) higher acceptance via *CPS*, with *ADL* further right-sizing proposals. Crucially, latency is governed by the *number of drafter and target passes per accepted token*, not the drafter's parameter count alone: although a single Dream-7B pass is heavier than a single 1-layer drafter pass, its higher MAT and parallel drafting allow DiffuSpec to commit a similar number of tokens with far fewer total passes, resulting in lower end-to-end wall-clock time under the same Qwen2.5-32B target. Together, these mechanisms translate acceptance gains into practical wall-clock acceleration.

5.2 Ablation

Tab. 2 quantifies the contributions of CPS and ADL. In the *no-CPS* variant, we skip causal-consistency path search and directly form the draft by taking, at each position, the highest-probability token from the DLM marginal (argmax) for a given draft length, which is then passed to the standard SPS-style block verifier. In the *no-ADL* variant, we disable the controller and fix the draft length for all steps, so that CPS operates on a token lattice of fixed width. Enabling either module improves both *Mean-MAT* and *Mean-Speedup* over the plain variant, while enabling both yields the best overall performance (6.99 MAT, $3.08\times$). Compared to the plain system (6.05 / $2.69\times$), *CPS-only* raises MAT by $+0.90$ and speedup by $+0.29\times$, whereas *ADL-only* adds $+0.38$ MAT and $+0.04\times$, respectively. Thus, CPS accounts for most acceptance gains—consistent with its role in aligning diffusion proposals with AR causality—while ADL primarily translates these gains into wall-clock speedup by adaptively setting k_t. When combined, they deliver a total improvement of $+0.39\times$ over the plain system. Additional analysis of draft-length choices is provided in Appendix D, and task-wise ablations with full results appear in Appendix C (Tab. 4).

5.3 Hyperparameter sensitivity

Across decoding/search knobs (Fig. 5a–5d), we observe consistent speed–quality trade-offs under the quality lock. Increasing the number of DLM refinement steps S improves proposal quality and acceptance (Mean-MAT $6.99 \rightarrow 7.33$ from $S{=}1$ to 10; Fig. 5a), but substantially reduces throughput (Mean-Speedup $3.08\times \rightarrow 0.93\times$), so we fix $S{=}1$. Enlarging the CPS beam B improves causal paths and modestly raises Mean-MAT, peaking around $B{=}3{\sim}4$ (Fig. 5b); however, overhead causes speedup to plateau or regress beyond $B{=}3$, so we set $B{=}3$. Increasing the per-position cap $M_{\max}$ relaxes pruning and helps until $M_{\max}{\approx}15$ (Fig. 5c); further branching yields negligible gains and slightly hurts speed, motivating our choice of $M_{\max}{=}15$. Raising the mass threshold τ retains more local probability and improves acceptance/speed up to $\tau{\approx}0.8$ (Fig. 5d); higher values add compute with little benefit, so we use $\tau{=}0.8$. Overall, CPS-related knobs $(B, M_{\max}, \tau)$ are robust over a range, while multi-step refinement S trades acceptance for latency. Orthogonally, ADL controls proposal size, helping convert CPS-driven acceptance gains into wall-clock acceleration.

6 Conclusion

We present **DiffuSpec**, a speculative decoding framework that leverages a *diffusion language model* (DLM) as a *parallel* drafter for standard autoregressive (AR) verification. DiffuSpec comprises two lightweight components: *causal-consistency path search* (CPS), which extracts verifier-aligned left-to-right paths from the bidirectionally conditioned draft lattice, and an *adaptive draft-length* (ADL) controller, which manages the speed–quality trade-off by adjusting proposal lengths using simple online acceptance feedback signals. Across a broad set of benchmarks and decoding settings, DiffuSpec consistently delivers strong wall-clock speedups over speculative decoding baselines under the quality-locked setting. Finally, DiffuSpec fits naturally into standard drafter–verifier serving pipelines with minimal changes, making diffusion-based drafting a practical option for accelerating AR decoding.

Limitations

For reproducibility, our experiments use pretrained diffusion language model (DLM) drafters. At present, the ecosystem of off-the-shelf DLMs is still evolving, and publicly available checkpoints do not yet match the breadth of small autoregressive (AR) drafters across model sizes and domains. This is largely an ecosystem and engineering consideration rather than a limitation of the decoding algorithm. When needed, a suitable DLM drafter can be obtained via distillation or lightweight adaptation (e.g., modest fine-tuning) and then reused across tasks to amortize the one-time cost. We expect this consideration to gradually become less pronounced as more DLM variants are released and supporting tools mature.

Acknowledgments

This work is supported by the National Key R&D Program of China (2022YFB4701400/4701402), SSTIC Grant (KJZD20230923115106012, KJZD20230923114916032, GJHZ20240218113604008).

References

Zachary Ankner, Rishab Parthasarathy, Aniruddha Nrusimha, Christopher Rinard, Jonathan Ragan-Kelley, and William Brandon. 2024. Hydra: Sequentially-dependent draft heads for medusa decoding. *arXiv preprint arXiv:2402.05109*.

Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. 2021. Structured denoising diffusion models in discrete state-spaces. *Advances in neural information processing systems*, 34:17981–17993.

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

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. 2023. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*.

Xinhua Chen, Sitao Huang, Cong Guo, Chiyue Wei, Yintao He, Jianyi Zhang, Hai Li, Yiran Chen, and 1 others. 2025. Dpad: Efficient diffusion language models with suffix dropout. *arXiv preprint arXiv:2508.14148*.

Jacob K Christopher, Brian R Bartoldson, Tal Ben-Nun, Michael Cardei, Bhavya Kailkhura, and Ferdinando Fioretto. 2024. Speculative diffusion decoding: Accelerating language generation through diffusion. *arXiv preprint arXiv:2408.05636*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Elias Frantar and Dan Alistarh. 2023. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International conference on machine learning*, pages 10323–10337. PMLR.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2022. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*.

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

Shansan Gong, Shivam Agarwal, Yizhe Zhang, Jiacheng Ye, Lin Zheng, Mukai Li, Chenxin An, Peilin Zhao, Wei Bi, Jiawei Han, and 1 others. 2024. Scaling diffusion language models via adaptation from autoregressive models. *arXiv preprint arXiv:2410.17891*.

Zhengfu He, Tianxiang Sun, Kuanning Wang, Xuanjing Huang, and Xipeng Qiu. 2022. Diffusionbert: Improving generative masked language models with diffusion models. *arXiv preprint arXiv:2211.15029*.

Zhenyu He, Zexuan Zhong, Tianle Cai, Jason D Lee, and Di He. 2023. Rest: Retrieval-based speculative decoding. *arXiv preprint arXiv:2311.08252*.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, and 1 others. 2022. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*.

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

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick SH Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *EMNLP (1)*, pages 6769–6781.

Siqi Kou, Lanxiang Hu, Zhezhi He, Zhijie Deng, and Hao Zhang. 2024. Cllms: Consistency large language models. In *Forty-first International Conference on Machine Learning*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, and 1 others. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR.

Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. 2022. Diffusion-lm improves controllable text generation. *Advances in neural information processing systems*, 35:4328–4343.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2024a. EAGLE-2: Faster inference of language models with dynamic draft trees. In *Empirical Methods in Natural Language Processing*.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2024b. EAGLE: Speculative sampling requires rethinking feature uncertainty. In *International Conference on Machine Learning*.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2025. EAGLE-3: Scaling up inference acceleration of large language models via training-time test. In *Annual Conference on Neural Information Processing Systems*.

Zhiyuan Liu, Yicun Yang, Yaojie Zhang, Junjie Chen, Chang Zou, Qingyuan Wei, Shaobo Wang, and Linfeng Zhang. 2025. dllm-cache: Accelerating diffusion large language models with adaptive caching. *arXiv preprint arXiv:2506.06295*.

Xianzhen Luo, Yixuan Wang, Qingfu Zhu, Zhiming Zhang, Xuanyu Zhang, Qing Yang, and Dongliang Xu. 2024. Turning trash into treasure: Accelerating inference of large language models with token recycling. *arXiv preprint arXiv:2408.08696*.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, and 1 others. 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*, pages 932–949.

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, and 1 others. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. 2025. Large language diffusion models. *arXiv preprint arXiv:2502.09992*.

Apoorv Saxena. 2023. Prompt lookup decoding.

Yuerong Song, Xiaoran Liu, Ruixiao Li, Zhigeng Liu, Zengfeng Huang, Qipeng Guo, Ziwei He, and Xipeng Qiu. 2025. Sparse-dllm: Accelerating diffusion llms with dynamic cache eviction. *arXiv preprint arXiv:2508.02558*.

Shengyin Sun, Yiming Li, Xing Li, Yingzhao Lian, Weizhe Lin, Hui-Ling Zhen, Zhiyuan Yang, Chen Chen, Xianzhi Yu, Mingxuan Yuan, and 1 others. 2025. Scaling up, speeding up: A benchmark of speculative decoding for efficient llm test-time scaling. *arXiv preprint arXiv:2509.04474*.

Ziteng Sun, Uri Mendlovic, Yaniv Leviathan, Asaf Aharoni, Jae Hun Ro, Ahmad Beirami, and Ananda Theertha Suresh. 2024. Block verification accelerates speculative decoding. *arXiv preprint arXiv:2403.10444*.

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

Nadav Timor, Jonathan Mamou, Daniel Korat, Moshe Berchansky, Gaurav Jain, Oren Pereg, Moshe Wasserblat, and David Harel. 2025. Accelerating LLM inference with lossless speculative decoding algorithms for heterogeneous vocabularies. In *Forty-second International Conference on Machine Learning*.

Siqi Wang, Hailong Yang, Xuezhu Wang, Tongxuan Liu, Pengbo Wang, Xuning Liang, Kejie Ma, Tianyu Feng, Xin You, Yongjun Bao, and 1 others. 2024.

Minions: Accelerating large language model inference with aggregated speculative execution. *arXiv preprint arXiv:2402.15678*.

Heming Xia, Zhe Yang, Qingxiu Dong, Peiyi Wang, Yongqi Li, Tao Ge, Tianyu Liu, Wenjie Li, and Zhifang Sui. 2024. Unlocking efficiency in large language model inference: A comprehensive survey of speculative decoding. *arXiv preprint arXiv:2401.07851*.

Jin Xu, Zhifang Guo, Jinzheng He, Hangrui Hu, Ting He, Shuai Bai, Keqin Chen, Jialin Wang, Yang Fan, Kai Dang, and 1 others. 2025. Qwen2. 5-omni technical report. *arXiv preprint arXiv:2503.20215*.

Xiaohan Xu, Ming Li, Chongyang Tao, Tao Shen, Reynold Cheng, Jinyang Li, Can Xu, Dacheng Tao, and Tianyi Zhou. 2024. A survey on knowledge distillation of large language models. *arXiv preprint arXiv:2402.13116*.

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. 2025. Dream 7b: Diffusion large language models. *arXiv preprint arXiv:2508.15487*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623.

Kun Zhou, Yifan Li, Wayne Xin Zhao, and Ji-Rong Wen. 2023. Diffusion-nat: Self-prompting discrete diffusion for non-autoregressive text generation. *arXiv preprint arXiv:2305.04044*.

A Dataset and Implementation Details

Datasets. We follow SPEC-BENCH (Xia et al., 2024) across six task families, using the official splits and preprocessing; prompts match §5. For *Multi-turn Conversation* (MT) we use **MT-Bench** with pairwise judging (Zheng et al., 2023). *Machine Translation* (Trans) follows Spec-Bench's public WMT-style news configuration. *Summarization* (Sum) is **CNN/DailyMail** (Nallapati et al., 2016). *Open-domain QA* (QA) is **Natural Questions** (Kwiatkowski et al., 2019). *Mathematical Reasoning* (Math) uses **GSM8K** (Cobbe et al., 2021). *Retrieval-Augmented Generation* (RAG) follows the **DPR** pipeline over Wikipedia (Karpukhin et al., 2020).

Task	Dataset(s)	Metric(s)
MT	MT-Bench (Zheng et al., 2023)	Win rate (pairwise)
Trans	Spec-Bench translation (WMT-style)	BLEU
Sum	CNN/DailyMail (Nallapati et al., 2016)	ROUGE-L
QA	Natural Questions (Kwiatkowski et al., 2019)	EM / F1
Math	GSM8K (Cobbe et al., 2021)	Accuracy
RAG	DPR over Wikipedia (Karpukhin et al., 2020)	Accuracy

Table 3: Spec-Bench datasets and evaluation metrics used in our experiments.

Implementation details. We build our evaluation harness on top of SPEC-BENCH (Xia et al., 2024), reusing its official data loaders, prompt templates and stop criteria. All systems share the same hardware/software stack as §5 (single NVIDIA A100 80GB, 11 CPU cores, 100GB RAM, PyTorch 2.6.0). Verification uses greedy decoding (temperature $= 0$) with batch size $= 1$ and KV cache enabled; we report tokens/s averaged over the full evaluation set, excluding model-loading and first-batch warmup. Wall-clock timing includes tokenization, drafter forward(s), path search, verifier forward, and residual sampling.

Unless otherwise stated, DiffuSpec adopts a single diffusion refinement step (S=1). Controller and search hyperparameters are fixed across tasks: $k_{\min}$=20, $k_{\max}$=30, beam size B=3, mass threshold τ=0.8, per-position cap $M_{\max}$=15, mixing weight λ=0.5, controller increment δ=10, and EMA smoothing ρ=0.5. The causal proxy is a 3-gram KenLM fitted *only* on the training split of each dataset (no test leakage). Speedup is defined as a unitless ratio: throughput(method) / throughput(AR-greedy) under identical runtime settings; MAT follows Sec. 3. CUDA events are synchronized at measurement points to ensure consistent timing.

B Full Algorithm of DiffuSpec

We provide the full pseudocode of DiffuSpec for completeness. Algorithm 1 instantiates the four-stage pipeline described in Sec. 4.4. All symbols (e.g., CPS score $\mathcal{S}$ in (7) and ADL rule in (10)) follow the main text.

C Full Ablation Results per Task

Table 4 expands Table 2 by reporting task-wise MAT and Speedup under different combinations of causal-consistency path search (CPS) and adaptive draft-length (ADL).

The task-wise breakdown confirms the complementary roles of CPS and ADL. CPS consistently yields larger gains, especially on QA and Math where alignment with AR verification is critical. ADL offers steady improvements by preventing over/under-drafting, with a visible effect on Summarization. Combining both mechanisms produces the best overall results, robust across all tasks.

D Fixed Draft Length Study

Fixed-k protocol. We fix $k_t \equiv k \in 10, 20, 30, 50, 100$ at every decoding step, truncate the diffusion token lattice to the first k positions, and run CPS to extract a proposal path. The AR verifier then applies standard block verification, yielding an accepted prefix of length $\leq k$ at each step. Table 5 summarizes the resulting speed–quality trade-off.

As k increases from 10 to 100, Mean-MAT generally rises (peaking at k=100 with 6.69), but Mean-Speedup peaks earlier at k=20/30 (both 2.98×) and then declines due to higher drafting and rejection costs. The adaptive controller (ADL) balances this trade-off online, attaining both the highest Mean-MAT (6.99) and the strongest Mean-Speedup (3.08×). This confirms the benefit of dynamic proposal sizing over fixed-k policies.

E Additional Results under Tokenizer Mismatch

In our main experiments, the drafter and target share the same tokenizer. As an additional robustness check, we further explore a *heterogeneous* vocabulary setting where the drafter and target use different tokenizers, and drafted probabilities must be mapped into the target vocabulary. This scenario has been studied in SPS, and heterogeneous-vocabulary support is available in standard implementations (e.g., `transformers`) (Timor et al.,

Algorithm 1: DIFFUSPEC (4-stage): DLM drafting + CPS + parallel verification + ADL

Input: prefix $\mathbf{x}_{1:j}$; target LM p_θ; DLM q_ϕ; ADL params $(k_{\min}, k_{\max}, \delta, \rho)$; CPS params $(M_{\max}, \tau, B, \lambda)$.

Init: $\tilde{L}_0^{\text{gen}} \leftarrow 0$, $\tilde{L}_0^{\text{acc}} \leftarrow 0$; set $k_1 \leftarrow k_{\max}$.

for $t = 1, 2, \ldots$ *until termination* **do**

(1) Draft: run DLM to produce a length-k_t block and per-position candidate sets $\{\mathcal{C}_{j+i}\}_{i=1}^{k_t}$ (top-$M_{\max}$) with scores $\ell_{j+i}^{\text{dlm}}(\cdot)$.

(2) CPS: on a pruned candidate lattice (cumulative-mass τ, always keep EOS, early-stop after the first EOS), run left-to-right beam search (beam B) using score $\mathcal{S}(\cdot)$ in (7) to obtain a left-to-right path $\hat{\mathbf{y}}_{j+1:j+m_t}$ *(path length m_t).*

(3) Parallel verification: block verification of $\hat{\mathbf{y}}_{j+1:j+m_t}$ with p_θ; compute acceptance using q_ϕ^{L2R}; obtain L_t^{acc}; append the accepted prefix and update $j \leftarrow j + L_t^{\text{acc}}$; if an EOS is accepted, **terminate**.

(4) ADL: compute L_t^{gen} from the proposal's first-EOS index s_t; update EMAs $\tilde{L}_t^{\text{gen}}, \tilde{L}_t^{\text{acc}}$; set k_{t+1} via (10).

CPS	ADL	MT		Trans		Sum		QA		Math		RAG	
		MAT	Spd	MAT	Spd	MAT	Spd	MAT	Spd	MAT	Spd	MAT	Spd
✓	✓	**7.02**	**3.12**×	**7.35**	**3.40**×	**6.25**	2.45×	**7.49**	**3.05**×	**7.61**	**4.05**×	**8.04**	2.40×
✓	✗	6.98	3.01×	7.29	3.28×	6.18	2.37×	7.40	2.92×	7.55	3.88×	7.96	2.34×
✗	✓	6.41	2.70×	6.72	2.79×	5.88	2.11×	6.92	2.55×	7.00	3.11×	7.25	2.14×
✗	✗	6.03	2.65×	6.48	2.61×	5.72	1.96×	6.75	2.41×	6.82	2.95×	7.08	2.02×

Table 4: **Task-wise ablation of DiffuSpec components.** CPS = causal-consistency path search; ADL = adaptive draft-length. Both components improve MAT and speedup (Spd) across tasks; **Spd denotes Speedup** (× vs. AR, ↑).

Policy	k=10	k=20	k=30	k=50	k=100	**ADL**
Mean-MAT	5.53	6.56	6.49	6.51	6.69	**6.99**
Mean-Speedup	2.74×	2.98×	2.98×	2.91×	2.78×	**3.08**×

Table 5: **Fixed-k vs. adaptive proposal length (quality-locked).** Means are computed across all tasks. ADL achieves the best speedup while also reaching the highest MAT, indicating a better speed–acceptance trade-off than fixed-k policies.

2025). DiffuSpec, like SPS, interacts with the target only through probability interfaces (the drafter's output probabilities and the target's next-token probabilities), and thus can directly reuse the same heterogeneous-vocabulary adapter without changing the core algorithm.

To illustrate this, we evaluate a heterogeneous setting where the target is Qwen3-32B and the drafter is Qwen2.5-7B (SPS) or Dream-7B (DiffuSpec), using the same adapter to bridge tokenizers. Table 6 reports results on Spec-Bench. DiffuSpec yields higher speedup than SPS (Mean-Speedup 1.42× vs. 1.03×) with comparable Mean-MAT (3.02 vs. 2.91), indicating that diffusion-based drafting remains effective under tokenizer mismatch.

F Additional Evidence on Model/Task Variants

To show that DiffuSpec is not limited to a single chat-model pair, we additionally evaluate a code-generation setting where DREAM-CODER-7B serves as the diffusion drafter and QWEN2.5-CODER-32B as the AR target. Table 7 reports results on the HumanEval benchmark under the same evaluation protocol as in the main experiments. DiffuSpec again achieves the highest wall-clock speedup among the considered baselines, indicating that the "DLM + CPS + ADL" pattern remains effective on a different task setting and code-oriented model variants within the same ecosystem.

G Output Visualizations

We provide qualitative k-sweeps showing how draft length shapes proposal style: short drafts tend to be terse; moderate drafts begin to exhibit step-by-step reasoning; very long drafts may drift or repeat. (All runs use the same prompt; only k varies. Visualization samples are raw drafts before CPS/verification and are not correctness-checked.)

Method / Target = Qwen3-32B	Speedup (× vs. AR, ↑)						Mean (MAT / Speedup)
	MT	Trans	Sum	QA	Math	RAG	
SPS (Qwen2.5-7B drafter)	1.05×	1.11×	0.89×	0.96×	1.16×	0.99×	2.91 / 1.03×
DiffuSpec (Dream-7B drafter)	1.23×	1.66×	0.96×	1.40×	2.17×	1.05×	3.02 / 1.42×

Table 6: **Heterogeneous-vocabulary setting.** Target is Qwen3-32B; SPS uses Qwen2.5-7B as the drafter, and DiffuSpec uses Dream-7B. Both methods employ the same heterogeneous-vocabulary adapter at the probability interface. Speedup is measured relative to Qwen3-32B greedy decoding with its own tokenizer.

Metric	Lookahead	PLD	Recycling	SAMD	SPS	DiffuSpec
MAT ↑	2.63	3.81	3.85	5.24	11.33	**10.94**
Speedup ↑	2.07×	4.17×	2.83×	5.29×	2.14×	**5.35**×

Table 7: HumanEval results using DREAM-CODER-7B as the diffusion drafter and QWEN2.5-CODER-32B as the target AR verifier. We report Mean Accepted Tokens (MAT) and mean wall-clock speedup.

Algebra: *Solve for x: $3x + 5 = 20$*

k=6: `Subtract 5 and divide by 3.`
k=60: `To solve for` x`, subtract 5 from both sides and then divide by 3 to isolate` x`.`
k=200: `To solve for` x`, we first isolate the variable by removing constants and normalizing the coefficient of` x `. . .`

Word problem: *A train travels at 60 km/h for 2.5 hours. How far?*

k=6: `150 km.`
k=20: `Distance = speed ×time =` $60 \times 2.5 = 150$ `km.`
k=60: `To find the distance, multiply the constant speed by the elapsed time; units remain in km.`

Calculus: *What is the derivative of $f(x) = 3x^2 + 2x + 1$?*

k=6: $6x + 1$
k=30: `The derivative of` $f(x) = 3x^2 + 2x + 1$ `is` $6x + 2$.
k=40: `The derivative of f(x) = 3x^2 + 2xx + 1 is ' ( xxx`

As k increases, drafts shift from terse answers to step-by-step reasoning (often with emerging chain-of-thought), which *initially* raises the verifier's accepted length: MAT grows for small-to-moderate k. Beyond a task-dependent sweet spot, however, we observe a clear *plateau*: very long drafts tend not to yield longer accepted prefixes—diffusion proposals begin to drift, repeat, or include partial phrases, so the AR verifier rejects earlier. Consequently, end-to-end speedup drops due to extra drafting and residual resampling, even though the draft itself is longer. This motivates *adaptive* proposal sizing (ADL) to stay near the knee of the MAT/speed trade-off, and *causal-consistency* path search (CPS) to keep proposals informative yet easy for the verifier to accept.

H Correctness and Scope

H.1 Proof of Proposition 4.1

For convenience, we restate the claim proved in the main text. Under greedy verification (temperature $= 0$), let

$$g(x_{<t}) := \arg\max_{v \in \mathcal{V}} p_\theta(v \mid x_{<t})$$

denote the target model's greedy next-token rule. At a speculative step with current prefix $x_{1:j}$, suppose the drafter proposes a block $\hat{y}_{j+1:j+k}$ by any mechanism, including CPS with beam/MAP-style search. The verifier processes the block left-to-right, accepting $\hat{y}_{j+i}$ iff $\hat{y}_{j+i} = g(x_{1:j+i-1})$; otherwise, at the first mismatch position it outputs $g(x_{1:j+i-1})$ and stops. The claim is that the resulting decoded sequence is identical, token-by-token, to plain AR-greedy decoding of p_θ.

Proof. We prove the claim by induction on the generated prefix length.

Initially, both procedures start from the same prefix, so the claim holds trivially. Assume that at some point the current prefix equals the prefix that would be produced by plain AR-greedy decoding of p_θ. Consider the next speculative step.

At position $j + 1$, there are two cases. If $\hat{y}_{j+1} \neq g(x_{1:j})$, then the verifier immediately outputs $g(x_{1:j})$ and stops, which is exactly the next token produced by plain AR-greedy decoding. If instead $\hat{y}_{j+1} = g(x_{1:j})$, then the verifier accepts

this token, so the prefix remains consistent with AR-greedy decoding.

Applying the same argument sequentially to positions $j+2, j+3, \ldots, j+k$, either: (i) a first mismatch occurs, at which point the verifier inserts the greedy token and terminates the step; or (ii) all drafted tokens match the greedy choices and are accepted. In both cases, the prefix after the speculative step is exactly the same as the prefix that plain AR-greedy decoding would produce.

Therefore, by induction, after every speculative step the current prefix remains identical to that of plain AR-greedy decoding. Hence the final decoded sequence produced by DiffuSpec is identical, token-by-token, to plain AR-greedy decoding of p_θ. □

H.2 When standard speculative-sampling unbiasedness holds

For completeness, we restate the classical unbiasedness guarantee of standard speculative sampling.

Proposition H.1. *At a single position t, let $p(\cdot) = p_\theta(\cdot \mid x_{<t})$ be the target distribution and $q(\cdot) = q_\phi(\cdot \mid x_{<t})$ be a proposal distribution. Standard speculative sampling draws a draft token $x \sim q$, accepts it with probability*

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

and otherwise samples from the residual distribution

$$r(x) \propto [p(x) - q(x)]_+.$$

Then the final output token is distributed exactly as p.

Proof. For any token x_0,

$$\begin{aligned}\Pr(\text{accept and output } x_0) &= q(x_0)\alpha(x_0) \\ &= \min\{p(x_0), q(x_0)\}.\end{aligned}$$

Let

$$\beta = \sum_x \min\{p(x), q(x)\}.$$

Then the residual distribution is

$$r(x) = \frac{p(x) - \min\{p(x), q(x)\}}{1-\beta},$$

and thus

$$\begin{aligned}\Pr(\text{reject and output } x_0) &= (1-\beta)\, r(x_0) \\ &= p(x_0) - \min\{p(x_0), q(x_0)\}.\end{aligned}$$

Summing the two cases yields $\Pr(\text{output } x_0) = p(x_0)$. □

The key condition behind Proposition H.1 is that the drafted token must be sampled from the same proposal distribution q that is used in the acceptance ratio.

H.3 Relation to stochastic decoding

In DiffuSpec, the acceptance ratio interfaces with an induced left-to-right proxy distribution q_ϕ^{L2R}. Under CPS, however, draft tokens are generated through a search procedure (e.g., beam/MAP-style path selection) rather than by sampling from the same proposal distribution used in the acceptance rule. Therefore, under stochastic decoding (temperature > 0), the classical unbiasedness guarantee in Proposition H.1 does not directly apply to DiffuSpec with CPS.

Accordingly, DiffuSpec with CPS in this regime should be viewed as an approximate/heuristic extension of speculative sampling, whose primary benefit is empirical acceleration. Our main experiments do not rely on this regime: they are conducted under greedy verification (temperature $= 0$), where Proposition 4.1 provides an exact lossless guarantee.

I Impact Statement

We propose DiffuSpec, a new speculative decoding approach that accelerates language-model inference while preserving the target model's verification behavior. By improving throughput and reducing latency, DiffuSpec can make high-quality generation more accessible and scalable. As with any efficiency improvement, faster generation may lower the barrier to large-scale content production and could amplify biases already present in the target model or its data. We recommend deploying DiffuSpec with standard safeguards (e.g., content filtering, rate limiting, and auditing) in high-stakes settings. Overall, we expect DiffuSpec to advance efficient inference while encouraging responsible AI practices to maximize societal benefits.

J The Use of Large Language Models

We used off-the-shelf LLMs solely for minor language polishing (grammar and wording). No datasets, results, or analyses in this paper were generated by LLMs. All technical content, experiments, and conclusions were produced and carefully verified by the authors.