MISTLETOE: STEALTHY ACCELERATION-COLLAPSE ATTACKS ON SPECULATIVE DECODING

Anonymous authors
Paper under double-blind review

ABSTRACT

Speculative decoding is widely used to accelerate large language model (LLM) inference by drafting multiple candidate tokens and verifying them with a target model in parallel. Its efficiency critically depends on the average accepted length τ, i.e., how many draft tokens survive each verification step. In this work, we identify a mechanism-level vulnerability in model-based speculative decoding: the drafter approximates the target model distribution, but this approximation is inevitably imperfect. Such a drafter–target mismatch creates an attack surface where small perturbations can preserve the target model's visible behavior while substantially reducing draft-token acceptability. We propose MISTLETOE, a stealthy acceleration-collapse attack against speculative decoding. It jointly optimizes a degradation objective that decreases drafter–target agreement and a semantic-preservation objective that constrains the target model's output distribution. To resolve the conflict between these objectives, we introduce a null-space projection mechanism that projects degradation gradients away from the local semantic-preserving direction, suppressing draft acceptance while minimizing semantic drift. Experiments across multiple target models and speculative decoding systems show that MISTLETOE substantially reduces average accepted length τ, collapses speedup, and lowers token throughput, while largely preserving task-level performance. Our work highlights a mechanism-level attack surface in speculative decoding beyond existing output robustness, motivating more robust LLM acceleration systems.

1 INTRODUCTION

Large language models (LLMs) have achieved remarkable capabilities across open-ended generation, reasoning, and interactive assistance (Grattafiori et al., 2024; Liu et al., 2024; Yang et al., 2025). However, autoregressive decoding remains inherently sequential, as each generated token requires a separate target-model invocation conditioned on the preceding context. Speculative decoding mitigates this bottleneck through a draft-then-verify paradigm: a lightweight drafter proposes candidate continuations, and the target model verifies them in parallel (Leviathan et al., 2023; Chen et al., 2023). By accepting multiple draft tokens in one target-model forward pass, speculative decoding can accelerate generation while preserving the target model's output distribution under standard verification rules. Thus, the practical efficiency of speculative decoding depends not merely on how many tokens are drafted, but on how many of them are accepted by the target verifier. The average accepted length τ therefore captures a central mechanism behind speculative acceleration.

Recent speculative decoding systems explicitly optimize this mechanism by improving drafter–target agreement. They introduce auxiliary prediction heads, target-model features, dynamic draft trees, fused intermediate representations, or shared computation to make draft proposals more acceptable to the target verifier (Cai et al., 2024; Li et al., 2024a;b; Ankner et al., 2024; Li et al., 2025). Alignment-oriented work further shows that mitigating token and feature misalignment improves draft-token acceptance, accepted length, and speedup (Hu et al., 2025a). These advances reveal that acceptance is not a secondary implementation detail, but a critical foundation that underpins the efficiency of speculative decoding. While residual drafter–target mismatch is usually treated as an efficiency bottleneck to be reduced, we show that it can also become an attack surface.

Existing safety-oriented studies on speculative decoding mainly examine privacy leakage or output safety (Wei et al., 2024; Wang et al., 2025b;a). In contrast, we ask whether the draft-verification

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

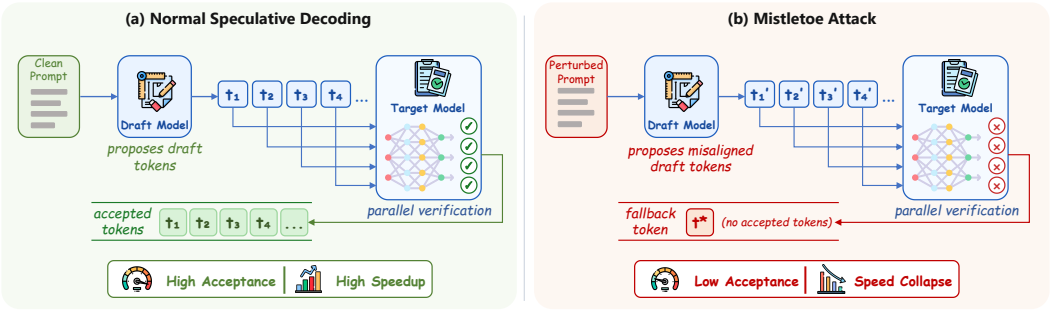


Figure 1: Illustration of acceptance collapse under MISTLETOE. In normal speculative decoding, the target model accepts many draft tokens per verification step, yielding high acceptance and high speedup. When speculative decoding is attacked by MISTLETOE, the final response semantics are largely preserved, but misaligned draft tokens are rejected by the target verifier, forcing fallback generation from target logits and collapsing the average accepted length τ and speedup.

pathway itself can be adversarially degraded while final responses remain largely preserved. We note that if a small perturbation preserves the target model's response distribution while causing the drafter proposals to diverge from the target verifier, the drafted tokens will be repeatedly rejected during verification. Consequently, τ collapses, the intended speedup disappears, and averaged token throughput decreases. We define this failure mode an *acceleration-collapse attack*, which disables the mechanism that makes generation fast while preserving the quality of generated content.

Based on these insights, we propose MISTLETOE, a stealthy acceleration-collapse attack against speculative decoding. As illustrated in Figure 1, normal speculative decoding commits multiple accepted draft tokens per verification step, whereas decoding under MISTLETOE frequently falls back to target-generated tokens. This reflects the parasitic nature of the attack: it remains unobtrusive at the output level while draining the efficiency benefit of the host decoding pipeline. Unlike content-level attacks that aim to alter the generated response, MISTLETOE targets the core verification-and-acceptance mechanism that makes speculative decoding fast. It increases target-side surprisal of drafter-proposed tokens to reduce their acceptability, while constraining the target model's output distribution to preserve response quality.

To reach the attack goal, a key challenge is the optimization conflict of drafter–target alignment. Draft tokens are designed to approximate the target model's high-probability continuations; therefore, perturbations that reduce draft-token acceptability can also disturb the target model's own output distribution. To address this conflict, MISTLETOE restricts the rejection direction to the local null space of the semantic-preservation constraint (Fang et al., 2024). This projection encourages updates that increase rejection pressure while limiting semantic drift. A KL-threshold filter further vetoes natural-language suffix candidates whose target-distribution drift exceeds a preset bound.

Experiments across multiple target models and speculative decoding systems show that MISTLETOE substantially reduces τ, speedup, and token throughput, while largely preserving task-level performance. These results demonstrate that speculative decoding can be vulnerable even when user-facing textual outputs appear normal, highlighting the need for robust and security-aware acceleration mechanisms.

In summary, our contributions are threefold:

- We identify *acceptance collapse* as a mechanism-level threat to speculative decoding, where adversarially amplified drafter–target mismatch reduces the average accepted length τ while leaving user-facing outputs largely preserved.
- We propose MISTLETOE, a stealthy acceleration-collapse attack that degrades the verification-and-acceptance pathway rather than directly corrupting generated content.
- We develop a null-space projected optimization method with KL-threshold filtering to reduce draft-token acceptability while suppressing semantic drift, and empirically demonstrate substantial degradation of speculative decoding efficiency across representative systems.

2 Related Work

2.1 Speculative Decoding for Efficient Inference

Speculative decoding accelerates autoregressive generation through a draft-then-verify paradigm, where a drafter proposes candidate tokens and the target model verifies them in parallel (Leviathan et al., 2023; Chen et al., 2023). Subsequent work improves drafting and verification through distillation, tree-based verification, retrieval- or phrase-based drafting, prediction heads, dynamic draft trees, and target-feature reuse (Zhou et al., 2024; Miao et al., 2024; He et al., 2024; Zhao et al., 2024; Cai et al., 2024; Li et al., 2024a;b; Ankner et al., 2024; Du et al., 2024; Li et al., 2025). Recent analyses and surveys further characterize the factors and design space governing speculative speedup (Yan et al., 2025; Hu et al., 2025b). Despite architectural differences, these methods generally seek to improve proposal quality or verification efficiency. GRIFFIN further identifies token and feature misalignment as a bottleneck for draft-token acceptance (Hu et al., 2025a), highlighting drafter–target agreement as a key determinant of speculative decoding efficiency.

2.2 Safety and Robustness of Speculative Decoding

Recent studies examine privacy and output-safety risks in speculative decoding, including side-channel leakage and safety-aware decoding (Wei et al., 2024; Wang et al., 2025b;a). More broadly, adversarial prompts can degrade LLM inference efficiency by increasing generation, latency, or energy costs (Feng et al., 2024; Dong et al., 2025; Li et al., 2026). These works expose privacy, safety, and system-level efficiency risks, but do not directly target the draft-verification acceptance mechanism.

In contrast, MISTLETOE exploits drafter–target mismatch to collapse draft-token acceptance while largely preserving final response behavior, exposing a mechanism-level performance-robustness threat specific to speculative acceleration.

3 Preliminaries

3.1 Speculative Decoding

We formalize the draft-then-verify process underlying speculative decoding. Let M_θ denote the target language model, or verifier, and D_ϕ denote the drafter. Given a prompt x, D_ϕ proposes draft tokens and M_θ verifies them in parallel. Let t index draft-then-verify cycles, and let $Y^{(t)}$ denote the accepted output prefix before the t-th cycle. Within each cycle, the drafter proposes $\hat{y}_1^{(t)}, \ldots, \hat{y}_K^{(t)}$, where K is the draft budget.

We denote the drafter and target verifier distributions for the i-th draft token by

$$\rho_\phi(\cdot \mid x, Y^{(t)}, \hat{y}_{<i}^{(t)}) \quad \text{and} \quad \pi_\theta(\cdot \mid x, Y^{(t)}, \hat{y}_{<i}^{(t)}),$$

respectively, where $\hat{y}_{<i}^{(t)}$ denotes previously drafted tokens in the same chain or tree branch. The drafter may be implemented as an independent model, auxiliary draft heads, or feature-reuse modules. Despite these architectural differences, acceleration is governed by drafter–target agreement.

A draft token is likely to be accepted only when the target assigns it probability comparable to the drafter. Under the standard speculative verification rule (Leviathan et al., 2023; Chen et al., 2023), the acceptance probability of $\hat{y}_i^{(t)}$ is

$$\alpha_i^{(t)} = \min\Big(1,\, \pi_\theta\Big(\hat{y}_i^{(t)} \mid x, Y^{(t)}, \hat{y}_{<i}^{(t)}\Big) \,/\, \rho_\phi\Big(\hat{y}_i^{(t)} \mid x, Y^{(t)}, \hat{y}_{<i}^{(t)}\Big)\Big). \tag{1}$$

If accepted, the draft token is committed to the output prefix; otherwise, the verifier falls back to the target logits and discards the remaining draft tokens along that path. Thus, $\{\alpha_i^{(t)}\}_{i=1}^K$ determine how many draft tokens survive verification. Let $a^{(t)}$ denote the number of tokens committed in cycle t, including accepted draft tokens and the target-generated fallback token. The average accepted length is

$$\tau = \mathbb{E}_t\left[a^{(t)}\right]. \tag{2}$$

In practice, τ is measured as the number of generated tokens divided by the number of target-model forward passes. A large τ indicates effective amortization of target computation, whereas $\tau \approx 1$ approaches vanilla autoregressive decoding. For tree-based speculative decoding, multiple branches may be verified with tree attention, but the same acceptance principle applies.

3.2 Threat Model and Objective

Attack scenario. We consider a malicious client interacting with an LLM service that uses speculative decoding for inference acceleration. The attacker does not intercept or modify prompts from benign users. Instead, it constructs its own natural-language prompt x and appends a short adversarial suffix δ, yielding

$$x_\delta = x \oplus \delta, \tag{3}$$

where $\oplus$ denotes concatenation. The perturbed prompt x_δ is then submitted through the normal inference interface. We restrict δ to a natural-language search space $\mathcal{N}_m \subseteq \mathcal{V}^m$, where m is the suffix length.

Attacker's capabilities. During attack construction, we assume white-box access to the target model M_θ and drafter D_ϕ, enabling gradient-based suffix optimization. The attacker does not modify either model, the speculative decoding algorithm, server-side configurations, or generated output tokens. At deployment time, the attacker only submits the optimized prompt x_δ. Moreover, a suffix optimized on a source configuration can be transferred to other target models or speculative decoding methods without access to their gradients or internals, enabling a black-box transfer setting; see Section 5.5 and Appendix B.

Attacker's goals. The attacker aims to reduce the acceptability of drafter-proposed tokens, thereby lowering the average accepted length τ and speculative speed-up, while preserving the target model's task-relevant output behavior. Thus, the attack targets the draft-verification pathway rather than directly manipulating the final response.

We formulate this objective as

$$\max_{\delta \in \mathcal{N}_m} \quad \mathcal{L}_{\text{rej}}(x, \delta) \qquad \text{s.t.} \qquad \mathcal{L}_{\text{sem}}(x, \delta) \leq \epsilon. \tag{4}$$

Here, $\mathcal{L}_{\text{rej}}$ measures rejection pressure on drafter-proposed tokens, while $\mathcal{L}_{\text{sem}}$ bounds target-model distributional drift induced by the suffix.

4 Method

4.1 Overview

To collapse speculative acceleration while preserving final-response behavior, MISTLETOE optimizes a short natural-language suffix appended to the clean prompt, as illustrated in Figure 2. It combines two information-theoretic objectives: Target-side Draft-Token Surprisal increases the mismatch between drafter proposals and the target verifier, reducing draft-token acceptability; KL-bounded Target Preservation constrains the adversarial target distribution to remain close to the clean one. Because these objectives can induce entangled gradients, MISTLETOE restricts the rejection direction to the local null space of the semantic-preservation constraint and uses the resulting feasible direction to guide natural-language suffix search, followed by a KL-bound veto. Together, these components reduce τ and speculative speedup while largely preserving user-facing response behavior.

4.2 On Acceptance Collapse: Surprisal under KL Constraints

Speculative decoding derives its speedup from the draft-then-verify mechanism, where drafter-proposed tokens are committed only when they remain sufficiently likely under the target verifier. MISTLETOE therefore aims to reduce the acceptability of draft tokens rather than directly corrupting the final response. Lowering the target probability assigned to drafter-proposed tokens decreases their acceptance probability, reduces the committed length $a^{(t)}$, and ultimately collapses the average accepted length τ. However, because the drafter is designed to approximate the target verifier, aggressively suppressing draft tokens can also distort the target distribution, causing semantic drift or abnormal outputs.

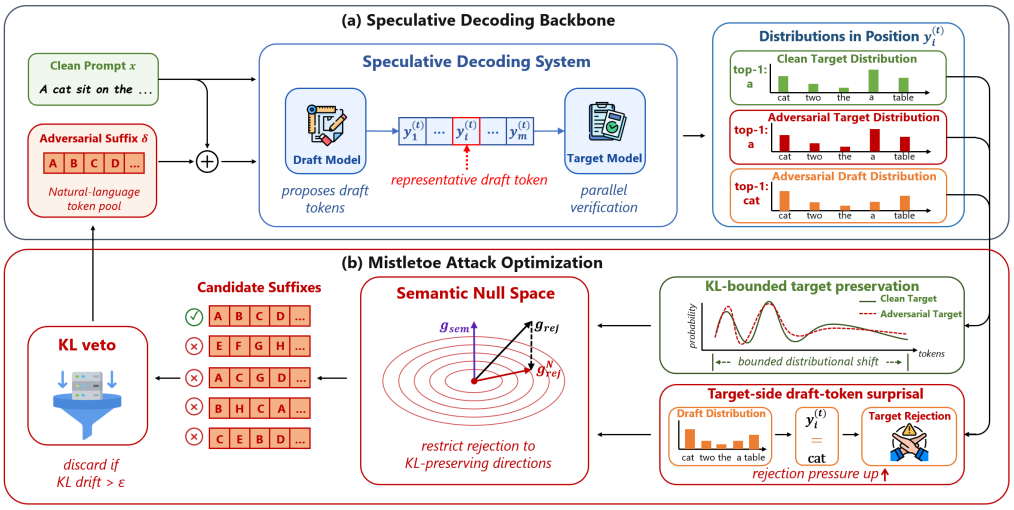


Figure 2: Pipeline of MISTLETOE. The adversarial suffix δ_k is appended to the clean prompt x and passed through the speculative decoding system. We visualize one representative draft token $\hat{y}_i^{(t)}$; in practice, the objectives aggregate over multiple positions. Target-side Draft-Token Surprisal increases rejection pressure by reducing the target verifier's confidence in drafter-proposed tokens, while KL-bounded Target Preservation constrains the adversarial target distribution to remain close to the clean one. The rejection direction is restricted to the semantic null space, producing a feasible update direction for natural-language suffix search. A KL-bound veto filters infeasible candidates, and the selected high-surprisal suffix becomes δ_{k+1}.

To instantiate the constrained objective defined in Section 3.2, we use two complementary quantities: one increases the target-side surprisal of draft tokens to create rejection pressure, while the other bounds target-verifier distributional drift to preserve clean response behavior.

Target-side draft-token surprisal. For a drafter-proposed token $\hat{y}_i^{(t)}$, its surprisal under the adversarial prompt x_δ is

$$s_\theta\left(\hat{y}_i^{(t)}; x_\delta\right) = -\log \pi_\theta\left(\hat{y}_i^{(t)} \mid x_\delta, Y^{(t)}, \hat{y}_{<i}^{(t)}\right). \tag{5}$$

Maximizing this quantity lowers the target verifier's confidence in drafter-proposed tokens, hence increasing rejection pressure. Given attacked draft-token positions $\mathcal{I}$, we define the rejection objective:

$$\mathcal{L}_{\text{rej}}(x, \delta) = \frac{1}{|\mathcal{I}|} \sum_{(t,i)\in\mathcal{I}} s_\theta\left(\hat{y}_i^{(t)}; x_\delta\right). \tag{6}$$

KL-bounded target preservation. While $\mathcal{L}_{\text{rej}}$ targets the acceleration pathway, the semantic constraint requires the target model's output behavior to remain close to its clean behavior. We therefore use the clean target distribution as an information-theoretic reference for the adversarial target distribution. Let $\mathcal{S}$ denote the positions used to estimate distributional drift:

$$\mathcal{L}_{\text{sem}}(x, \delta) = \frac{1}{|\mathcal{S}|} \sum_{t\in\mathcal{S}} D_{\text{KL}}\left(\pi_\theta(\cdot \mid x, Y^{(t)}) \,\middle\|\, \pi_\theta(\cdot \mid x_\delta, Y^{(t)})\right). \tag{7}$$

This KL term penalizes shifts from the clean target distribution to the adversarial one, preventing rejection from being achieved by broadly corrupting the target model's next-token preferences.

Together, Eq. (6) and Eq. (7) instantiate the constrained objective in Eq. (4). $\mathcal{L}_{\text{rej}}$ enlarges the mismatch between drafter proposals and the target verifier, while $\mathcal{L}_{\text{sem}}$ limits target-model distributional drift. Because the drafter approximates the target verifier, their gradients can conflict: rejection updates may increase semantic drift, while preservation updates may weaken the attack signal. This geometric entanglement motivates the null-space projected optimization introduced next.

4.3 NULL-SPACE PROJECTED OPTIMIZATION

To solve the constrained problem in Eq. (4), we seek update directions that increase rejection pressure while remaining locally feasible under the semantic-preservation constraint. Since the suffix δ is discrete, we compute gradients in a continuous relaxation of the suffix, such as the embedding or one-hot token space, and use them only to score token substitutions. Let $\mathbf{z}$ denote this relaxed suffix representation, and let $g_{\text{rej}} = \nabla_{\mathbf{z}} \mathcal{L}_{\text{rej}}(x, \delta)$ and $g_{\text{sem}} = \nabla_{\mathbf{z}} \mathcal{L}_{\text{sem}}(x, \delta)$ denote the rejection and semantic-drift gradients.

Local semantic null space. The key idea is to optimize rejection only within directions that locally preserve the target distribution. Under a first-order approximation, the semantic-preservation constraint defines a local feasible subspace around the current suffix. Let $J_{\text{sem}}(\mathbf{z}) = \nabla_{\mathbf{z}} \mathcal{L}_{\text{sem}}(x, \delta)^\top$ denote the Jacobian of the semantic-preservation objective. The local semantic null space is:

$$\mathcal{N}_{\text{sem}}(\mathbf{z}) = \{\Delta \mid J_{\text{sem}}(\mathbf{z})\Delta = 0\}. \tag{8}$$

Directions in $\mathcal{N}_{\text{sem}}(\mathbf{z})$ keep $\mathcal{L}_{\text{sem}}$ unchanged to first order, providing a local feasible subspace for rejection optimization.

Null-space projection. We construct the orthogonal projector onto the local semantic null space:

$$\mathbf{P}_{\mathcal{N}} = \mathbf{I} - J_{\text{sem}}^\top \left(J_{\text{sem}} J_{\text{sem}}^\top + \xi \mathbf{I}\right)^{-1} J_{\text{sem}}, \tag{9}$$

where ξ is a small damping term for numerical stability. The null-space rejection direction is $g_{\text{rej}}^{\mathcal{N}} = \mathbf{P}_{\mathcal{N}} g_{\text{rej}}$. In our scalar KL-constraint setting, this projection reduces to an efficient rank-one operation that removes the component of g_{rej} along g_{sem}. Unlike direct weighted optimization, this explicitly restricts rejection optimization to the local null space of the semantic-preservation constraint.

The final scoring direction combines feasibility restoration with null-space rejection:

$$g_{\text{final}} = -g_{\text{sem}} + \lambda g_{\text{rej}}^{\mathcal{N}}, \tag{10}$$

where λ controls the strength of acceleration degradation. The first term pulls the adversarial target distribution toward the clean target distribution, while the second increases rejection pressure within the local semantic null space.

Natural-language suffix update with KL-bound veto. The null-space direction provides a local continuous search signal, while the suffix remains a discrete natural-language sequence. We use g_{final} to score token substitutions drawn from a natural-language candidate vocabulary and construct the candidate set $\mathcal{C}(\delta)$. Since a discrete update may deviate from the local first-order approximation, each candidate δ' is re-evaluated by a forward pass. To enforce the semantic-preservation constraint in Eq. (4), we apply a KL-bound veto and select

$$\delta^\star = \arg\max_{\delta' \in \mathcal{C}(\delta)} \mathcal{L}_{\text{rej}}(x, \delta') \quad \text{s.t.} \quad \mathcal{L}_{\text{sem}}(x, \delta') \le \epsilon. \tag{11}$$

Candidates violating the KL bound are discarded, and among the remaining feasible candidates we choose the one with the largest rejection objective. The suffix is then updated as $\delta \leftarrow \delta^\star$. Overall, the null-space projection guides the search direction, while the KL-bound veto enforces the semantic constraint on actual candidates.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

Models and speculative decoding systems. We evaluate MISTLETOE on Vicuna-7B and Vicuna-13B (Chiang et al., 2023), Qwen3-8B (Yang et al., 2025), LLaMA-3.1-8B-Instruct (Grattafiori et al., 2024), and DeepSeek-R1-Distill-LLaMA-8B (DeepSeek-AI, 2025). For speculative decoding, we consider Medusa (Cai et al., 2024), Hydra (Ankner et al., 2024), EAGLE (Li et al., 2024a), EAGLE-2 (Li et al., 2024b), and EAGLE-3 (Li et al., 2025). We evaluate multiple decoding methods on the Vicuna backbones and use EAGLE-3 for the additional 8B target models. Among the Vicuna backbones, EAGLE-3 is evaluated only on Vicuna-13B because a corresponding Vicuna-7B checkpoint is unavailable. The target model and drafter remain fixed in all settings; the adversary only optimizes the appended natural-language suffix.

Table 1: Attack results of MISTLETOE. Entries are clean/attacked (attacked in red); the final row reports average absolute reductions with relative reductions in parentheses.

Model	Method	MT-Bench		HumanEval		GSM8K	
		Speed-up ↓	τ ↓	Speed-up ↓	τ ↓	Speed-up ↓	τ ↓
V 13B	Medusa	3.26×/**1.48**×	2.48/**2.09**	3.42×/**1.42**×	2.83/**2.16**	3.39×/**1.46**×	2.51/**2.32**
	Hydra	4.09×/**2.04**×	3.49/**3.06**	4.20×/**2.03**×	3.80/**3.10**	4.29×/**1.72**×	3.58/**3.07**
	EAGLE	3.16×/**1.67**×	3.37/**2.48**	3.86×/**1.79**×	4.32/**3.95**	3.66×/**1.69**×	3.79/**2.96**
	EAGLE-2	3.95×/**1.60**×	4.16/**2.92**	5.06×/**2.41**×	5.42/**3.55**	4.84×/**2.62**×	4.91/**3.25**
	EAGLE-3	5.18×/**2.38**×	5.73/**3.75**	6.17×/**2.77**×	7.08/**3.99**	5.47×/**1.83**×	5.95/**2.79**
V 7B	Medusa	3.44×/**2.38**×	2.41/**1.93**	3.75×/**2.77**×	2.69/**2.16**	3.46×/**2.50**×	2.47/**1.82**
	Hydra	4.73×/**1.28**×	3.54/**2.66**	5.01×/**2.08**×	3.52/**2.70**	4.37×/**2.24**×	3.84/**3.07**
	EAGLE	2.56×/**1.67**×	3.18/**2.33**	3.96×/**2.10**×	3.92/**3.19**	4.11×/**2.02**×	3.62/**3.06**
	EAGLE-2	2.54×/**1.38**×	4.28/**2.54**	4.15×/**3.16**×	5.17/**3.03**	4.99×/**2.71**×	4.68/**2.84**
Q3 8B	EAGLE-3	3.60×/**2.19**×	4.80/**3.91**	4.67×/**3.22**×	4.98/**4.16**	4.81×/**3.24**×	5.65/**4.57**
L3.1 8B	EAGLE-3	4.07×/**3.04**×	5.65/**4.74**	4.73×/**3.67**×	6.49/**5.10**	4.32×/**3.42**×	5.90/**5.07**
DS 8B	EAGLE-3	4.13×/**2.52**×	5.84/**4.28**	4.91×/**3.68**×	6.53/**5.65**	5.49×/**3.21**×	7.02/**4.62**
Avg. drop	–	↓**1.76**× (46.1%)	↓**1.02** (24.4%)	↓**1.90**× (42.1%)	↓**1.17** (23.6%)	↓**2.04**× (45.9%)	↓**1.21** (24.9%)

Datasets. We evaluate on three representative benchmarks covering open-ended dialogue, code generation, and mathematical reasoning. Specifically, we use all 80 questions from MT-Bench (Zheng et al., 2023), randomly sample 100 examples from HumanEval (Chen et al., 2021), and randomly sample 100 examples from GSM8K (Cobbe et al., 2021). These datasets allow us to test MISTLETOE across diverse generation scenarios, including instruction following, functional program synthesis, and multi-step mathematical reasoning.

Evaluation metrics. We report average accepted length τ and speedup over vanilla autoregressive decoding as the primary efficiency metrics. Lower τ indicates that fewer tokens are committed per target-model forward pass, while lower speedup indicates a weaker realized acceleration benefit. Together, they directly measure whether MISTLETOE collapses the draft-then-verify acceleration pathway. To evaluate semantic preservation, we additionally measure task performance before and after attack using GPT-4o scores on MT-Bench, Pass@1 on HumanEval, and answer accuracy on GSM8K. Higher task performance under attack indicates better preservation of the target model's original task-solving behavior.

Implementation details. All experiments are conducted on NVIDIA H20 GPUs. MISTLETOE optimizes a natural-language suffix $\delta \in \mathcal{N}_m$ with $m = 20$, while all model parameters remain frozen. Clean and attacked inputs are evaluated under the same speculative decoding configuration for fair comparison. Additional implementation details, including optimization hyperparameters and KL bounds, are provided in Appendix A.

5.2 MAIN RESULTS

Table 1 presents the main effectiveness of MISTLETOE across multiple target-model families, speculative decoding methods, and three datasets. MISTLETOE consistently reduces both speed-up and average accepted length τ across all evaluated settings. On MT-Bench, the average speed-up drops by $1.76\times$ (46.1%), while τ decreases by 1.02 on average. Similar reductions are observed on HumanEval and GSM8K, where speed-up drops by $1.90\times$ and $2.04\times$ (42.1% and 45.9%), while τ decreases by 1.17 and 1.21, respectively. These results show that MISTLETOE consistently degrades speculative acceleration across open-ended dialogue, code generation, and mathematical reasoning.

The attack remains effective across both strong and relatively modest clean-acceleration settings. For example, Vicuna-13B with EAGLE-3 drops from $6.17\times$ to $2.77\times$ on HumanEval and from $5.47\times$ to $1.83\times$ on GSM8K, with corresponding τ reductions from 7.08 to 3.99 and 5.95 to 2.79. Overall, the consistent decrease in τ confirms that the speed-up loss is driven by acceptance collapse: fewer tokens are committed per target-model forward pass as fewer drafter-proposed tokens survive verification. These results support our central claim that MISTLETOE disables speculative acceleration by degrading draft-token acceptability across models, methods, and tasks.

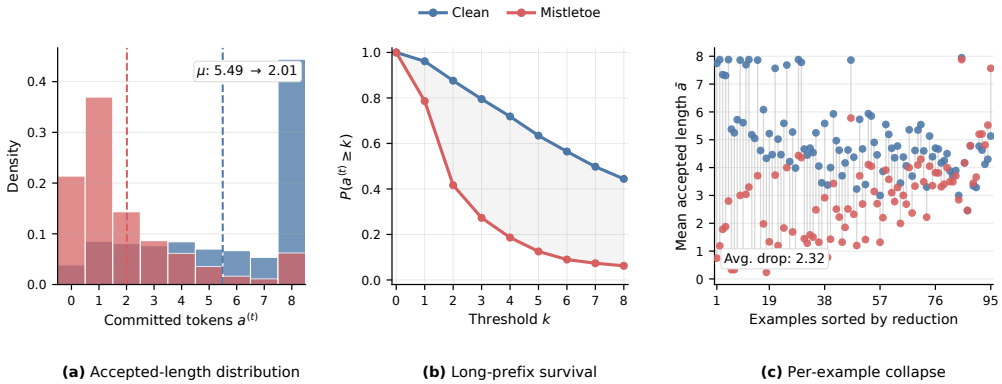


Figure 3: Mechanism visualization of acceptance collapse. The figure compares clean speculative decoding and MISTLETOE using verification-cycle-level accepted lengths. **(a) Accepted-length distribution:** MISTLETOE shifts the distribution of committed tokens per target-model forward pass $a^{(t)}$ toward smaller values. **(b) Long-prefix survival:** the survival probability $P(a^{(t)} \geq k)$ drops sharply under attack, showing that long draft prefixes rarely survive verification. **(c) Per-example collapse:** the mean accepted length decreases across most prompts, indicating that the collapse is broadly observed rather than driven by isolated outliers.

Table 2: Component ablation on V-13B/EAGLE-3/GSM8K.

Configuration	Component			Acceleration	
	$\mathcal{L}_{\text{rej}}$	$\mathcal{L}_{\text{sem}}$	Proj. Opt.	Speed-up ↓	τ ↓
Clean	×	×	×	5.47×	5.95
$\mathcal{L}_{\text{rej}}$ only	✓	×	×	3.30×	3.41
$\mathcal{L}_{\text{sem}}$ only	×	✓	×	4.47×	4.73
Naive joint	✓	✓	×	3.73×	4.13
Full MISTLETOE	✓	✓	✓	**1.83×**	**2.79**

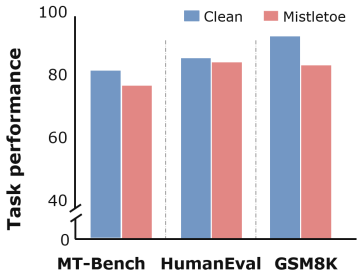


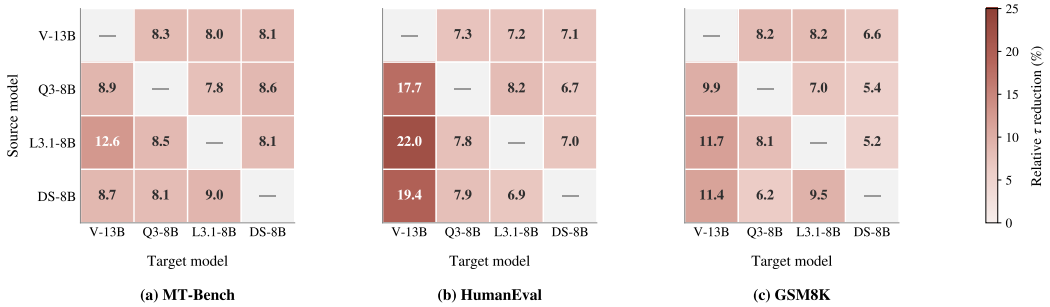
Figure 4: Task performance before and after MISTLETOE.

Figure 3 further explains the source of speed-up degradation. Compared with clean speculative decoding, MISTLETOE shifts the distribution of committed tokens per target-model forward pass $a^{(t)}$ toward smaller values and sharply reduces the survival probability $P(a^{(t)} \geq k)$ for long accepted prefixes. The per-example view shows that this reduction is broadly observed across prompts rather than caused by isolated outliers. These observations align with the τ reductions in Table 1, confirming that the attack disables speculative acceleration by collapsing draft-token acceptance.

5.3 ABLATION STUDY

Table 2 evaluates the contribution of each component of MISTLETOE on Vicuna-13B with EAGLE-3 on GSM8K. Using only $\mathcal{L}_{\text{rej}}$ already causes clear acceleration collapse, reducing speed-up from $5.47\times$ to $3.30\times$ and τ from 5.95 to 3.41, whereas $\mathcal{L}_{\text{sem}}$ alone yields weaker degradation because it does not directly encourage draft-token rejection. This confirms that rejection pressure primarily drives acceleration collapse, while the preservation objective constrains target-model drift.

Naive joint optimization remains weaker than the full method, reaching $3.73\times$ speed-up and $\tau = 4.13$, suggesting that simply combining rejection and preservation losses does not fully resolve their conflict. Stronger preservation pressure can suppress useful rejection updates, while unconstrained rejection may increase semantic drift. The full MISTLETOE objective achieves the strongest degradation, reducing speed-up to $1.83\times$ and τ to 2.79. These results support the role of null-space projection in suppressing draft-token acceptance while constraining semantic drift.



(a) MT-Bench **(b) HumanEval** **(c) GSM8K**

Figure 5: Cross-model transferability on MT-Bench, HumanEval, and GSM8K. Rows/columns denote source/target models, and each cell reports the relative reduction in τ. Diagonal entries are omitted.

5.4 Semantic Preservation

Figure 4 illustrates semantic preservation on Qwen3-8B across MT-Bench, HumanEval, and GSM8K. Performance remains close to the clean baseline on MT-Bench and HumanEval, decreasing from 8.11 to 7.66 and from 85% to 84%, respectively, while GSM8K shows a larger drop from 92% to 83%. Across all three target models, the corresponding averages decrease from 7.30 to 6.68 on MT-Bench, 83.7% to 82.3% on HumanEval, and 90.3% to 83.7% on GSM8K, showing some task-dependent variation in preservation.

Overall, MISTLETOE retains much of the target model's task-solving behavior despite substantial reductions in speculative acceleration. This separation between efficiency collapse and task-performance degradation supports that the attack primarily disrupts the draft-verification pathway rather than directly corrupting the target model's outputs.

5.5 Transferability Analysis

We first evaluate black-box cross-model transfer, where adversarial suffixes optimized on one source model are directly applied to other target models without further gradient access or re-optimization. Figure 5 summarizes transfer among Vicuna-13B, Qwen3-8B, LLaMA-3.1-8B, and DeepSeek-R1-Distill-LLaMA-8B with EAGLE-3. Transferred suffixes consistently reduce τ across model pairs and datasets, although transfer strength is asymmetric. On HumanEval, transferring from Qwen3-8B, LLaMA-3.1-8B, and DeepSeek-R1-Distill-LLaMA-8B to Vicuna-13B reduces τ by 17.7%, 22.0%, and 19.4%, respectively. These results show that the optimized suffixes are not specific to a single white-box source model and can remain effective under black-box transfer.

We further evaluate black-box transfer across speculative decoding methods. Suffixes optimized on EAGLE-3 are directly transferred to Medusa, Hydra, EAGLE, and EAGLE-2 without re-optimization, consistently reducing both speed-up and τ. This indicates that MISTLETOE can transfer across different speculative decoding architectures without requiring white-box access to the deployment system. Complete cross-method transfer results are provided in Appendix B.

6 Conclusion

We introduced MISTLETOE, an acceleration-collapse attack against speculative decoding. Instead of corrupting final responses, it reduces drafter-token acceptability under the target verifier through target-side surprisal, KL-bounded preservation, and null-space projected optimization. Experiments across multiple target-model families, speculative decoding methods, and benchmarks show consistent reductions in speed-up and τ, while task-level evaluations indicate that the target model's semantic behavior is largely preserved. Cross-model and cross-method transferability further show that the vulnerability is not confined to a single target model or speculative decoding architecture, motivating more robust acceleration mechanisms.

AI USE STATEMENT

We used OpenAI GPT-5.6 as an assistant for writing, editing, and presentation. In particular, it was used to refine grammar and phrasing, improve textual flow and clarity, suggest revisions to figure and table captions and layouts, assist with plotting and visualization code, and perform limited numerical consistency checks during manuscript preparation. We did not use generative AI to generate experimental results, make autonomous scientific decisions, or replace author verification of technical claims. All AI-assisted text, code, calculations, and visual materials were reviewed, verified, and edited by the authors. The authors take full responsibility for the final content of this work, including all text, claims, results, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies a performance-robustness vulnerability in speculative decoding with the goal of improving the security and reliability of LLM inference systems. Because the proposed attack could potentially be misused to increase serving costs or degrade the efficiency of deployed LLM services, we conduct our study in controlled experimental settings and present the method for research, auditing, and defense development rather than for disrupting real-world systems. Potential mitigation strategies include monitoring abnormal changes in accepted-length distributions, detecting unusual drafter–target mismatch, and designing verification mechanisms that are more robust to adversarial prompt perturbations. We hope that identifying this vulnerability will encourage the development of more robust speculative decoding systems.

REPRODUCIBILITY STATEMENT

We provide the methodological details, optimization objectives, and evaluation metrics required to reproduce MISTLETOE in the main paper. The appendix further documents the speculative decoding systems, target–speculative model pairings, experimental configurations, transfer settings, ablation studies, and additional qualitative results. All experiments are conducted using publicly available target models, speculative decoding frameworks, and benchmark datasets. We report the key implementation settings and model configurations used in our experiments, and will release the code and configuration files needed to reproduce the reported results as supplementary material.

REFERENCES

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

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

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

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

Wei-Lin Chiang, Zhuohan Li, Ziqing Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See https://vicuna. lmsys. org (accessed 14 April 2023)*, 2(3):6, 2023.

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

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025.

Jianshuo Dong, Ziyuan Zhang, Qingjie Zhang, Tianwei Zhang, Hao Wang, Hewu Li, Qi Li, Chao Zhang, Ke Xu, and Han Qiu. An engorgio prompt makes large language model babble on. In *International Conference on Learning Representations*, 2025.

Cunxiao Du, Jing Jiang, Yuanchen Xu, Jiawei Wu, Sicheng Yu, Yongqi Li, Shenggui Li, Kai Xu, Liqiang Nie, Zhaopeng Tu, and Yang You. GliDe with a CaPE: A low-hassle method to accelerate speculative decoding. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 11704–11720. PMLR, 2024.

Junfeng Fang, Houcheng Jiang, Kun Wang, Yunshan Ma, Shi Jie, Xiang Wang, Xiangnan He, and Tat-Seng Chua. Alphaedit: Null-space constrained knowledge editing for language models. *arXiv preprint arXiv:2410.02355*, 2024.

Xiaoning Feng, Xiaohong Han, Simin Chen, and Wei Yang. LLMEffiChecker: Understanding and testing efficiency degradation of large language models. *ACM Transactions on Software Engineering and Methodology*, 33(7):1–38, 2024. doi: 10.1145/3664812.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Zhenyu He, Zexuan Zhong, Tianle Cai, Jason Lee, and Di He. 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*, pp. 1582–1595. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.naacl-long.88.

Shijing Hu, Jingyang Li, Xingyu Xie, Zhihui Lu, Kim-Chuan Toh, and Pan Zhou. Griffin: Effective token alignment for faster speculative decoding. *arXiv preprint arXiv:2502.11018*, 2025a.

Yunhai Hu, Zining Liu, Zhenyuan Dong, Tianfan Peng, Bradley McDanel, and Sai Qian Zhang. Speculative decoding and beyond: An in-depth survey of techniques. *arXiv preprint arXiv:2502.19732*, 2025b.

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

Xingyu Li, Xiaolei Liu, Cheng Liu, Yixiao Xu, Kangyi Ding, Bangzhou Xin, and Jia-Li Yin. Loopllm: Transferable energy-latency attacks in llms via repetitive generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 31770–31777, 2026.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle: Speculative sampling requires rethinking feature uncertainty. *arXiv preprint arXiv:2401.15077*, 2024a.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle-2: Faster inference of language models with dynamic draft trees. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pp. 7421–7432, 2024b.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle-3: Scaling up inference acceleration of large language models via training-time test. *arXiv preprint arXiv:2503.01840*, 2025.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, Chunan Shi, Zhuoming Chen, Daiyaan Arfeen, Reyna Abhyankar, and Zhihao Jia. SpecInfer: Accelerating large language model serving with tree-based speculative inference and verification. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, volume 3, pp. 932–949. ACM, 2024. doi: 10.1145/3620666.3651335.

Jiayou Wang, Rundong Liu, Yue Hu, Huijia Wu, and Zhaofeng He. Secdecoding: Steerable decoding for safer llm generation. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 20504–20521, 2025a.

Xuekang Wang, Shengyu Zhu, and Xueqi Cheng. Speculative safety-aware decoding. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 12827–12841, 2025b.

Jiankun Wei, Abdulrahman Abdulrazzag, Tianchen Zhang, Adel Muursepp, and Gururaj Saileshwar. When speculation spills secrets: Side channels via speculative decoding in llms. *arXiv preprint arXiv:2411.01076*, 2024.

Minghao Yan, Saurabh Agarwal, and Shivaram Venkataraman. Decoding speculative decoding. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 6460–6473. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.naacl-long.328.

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

Weilin Zhao, Yuxiang Huang, Xu Han, Wang Xu, Chaojun Xiao, Xinrong Zhang, Yewei Fang, Kaihuo Zhang, Zhiyuan Liu, and Maosong Sun. Ouroboros: Generating longer drafts phrase by phrase for faster speculative decoding. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 13378–13393. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-main.742.

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

Yongchao Zhou, Kaifeng Lyu, Ankit Singh Rawat, Aditya Krishna Menon, Afshin Rostamizadeh, Sanjiv Kumar, Jean-François Kagy, and Rishabh Agarwal. DistillSpec: Improving speculative decoding via knowledge distillation. In *International Conference on Learning Representations*, 2024.

A MORE EXPERIMENTAL CONFIGURATION

We generate adversarial suffixes for text-based prompts to disrupt the efficiency of speculative decoding systems. We evaluate MISTLETOE across several widely used speculative decoding frameworks and target models. Unless otherwise specified, all systems follow their standard decoding configurations.

Speculative decoding systems. We evaluate the following speculative decoding systems.

- **EAGLE.** EAGLE (Li et al., 2024a) performs feature-level drafting with tree-based speculative decoding. We adopt the standard `mc_sim_7b_63` tree configuration to organize and verify candidate token sequences.
- **EAGLE-2.** EAGLE-2 (Li et al., 2024b) extends EAGLE with a dynamic draft-tree construction strategy. To isolate the EAGLE-2 architecture from EAGLE-3-specific components, we disable EAGLE-3 features using `--no-eagle3`.
- **EAGLE-3.** EAGLE-3 (Li et al., 2025) replaces feature prediction with direct token prediction and incorporates multi-layer target-model features for speculative drafting. We enable the EAGLE-3 pathway using `--use-eagle3` and use the corresponding pretrained drafter for each supported target model.
- **Medusa.** Medusa (Cai et al., 2024) adopts a multi-head parallel speculative decoding architecture. We use its native configuration with five Medusa heads, verification over up to ten draft candidates, and a KV-cache margin of 128 tokens. When pre-computing clean reference logits for the KL-divergence objective, we temporarily remove the Medusa attention mask to obtain the target-model reference distribution. Inference-time decoding otherwise follows the standard Medusa pipeline.
- **Hydra.** Hydra (Ankner et al., 2024) uses multiple prediction heads with a posterior acceptance mechanism. We use its default posterior acceptance setting, with threshold 0.09 and coefficient 0.3. As with Medusa, we disable the Hydra attention mask only when pre-computing clean reference logits for the KL objective, while keeping standard Hydra decoding during evaluation.

Target models. We evaluate MISTLETOE across target models from several model families and training paradigms.

- **Vicuna.** Vicuna (Chiang et al., 2023) is an instruction-tuned model derived from LLaMA. We use Vicuna-7B-v1.3 and Vicuna-13B-v1.3 as target models for EAGLE, EAGLE-2, Medusa, and Hydra, enabling controlled comparisons across speculative decoding frameworks with the same target-model family.
- **LLaMA-3.1.** LLaMA-3.1 (Grattafiori et al., 2024) is a more recent generation of the LLaMA family. We use LLaMA-3.1-8B-Instruct with EAGLE-3 to evaluate whether the attack remains effective on a modern instruction-tuned target model.
- **DeepSeek-R1-Distill.** DeepSeek-R1-Distill-LLaMA-8B (DeepSeek-AI, 2025)is a reasoning-oriented model obtained through distillation from DeepSeek-R1 into a LLaMA-based model. We include it with EAGLE-3 to examine acceleration collapse on a target model with reasoning-oriented training.
- **Qwen3.** Qwen3-8B (Yang et al., 2025) provides an additional target from a distinct model family. We pair it with its corresponding EAGLE-3 drafter to evaluate whether MISTLETOE generalizes beyond the LLaMA/Vicuna model family.

Target–speculative model pairing. For each speculative decoding framework, we use the corresponding pretrained speculative model or prediction heads associated with the target model. The complete configurations are summarized in Table 3.

Table 3: Target and speculative model pairs used in our experiments.

Framework	Target model	Speculative model
EAGLE	Vicuna-7B v1.3 Vicuna-13B v1.3	EAGLE-Vicuna-7B-v1.3 EAGLE-Vicuna-13B-v1.3
EAGLE-2	Vicuna-7B v1.3 Vicuna-13B v1.3	EAGLE-Vicuna1.3-7B EAGLE-Vicuna1.3-13B
EAGLE-3	Vicuna-13B v1.3 LLaMA-3.1-8B-Instruct DeepSeek-R1-Distill-LLaMA-8B Qwen3-8B	EAGLE3-Vicuna1.3-13B EAGLE3-LLaMA3.1-Instruct-8B EAGLE3-DeepSeek-R1-Distill-LLaMA-8B Qwen3-8B EAGLE-3
Medusa	Vicuna-7B v1.3 Vicuna-13B v1.3	Medusa-Vicuna-7B-v1.3 Medusa-Vicuna-13B-v1.3
Hydra	Vicuna-7B v1.3 Vicuna-13B v1.3	Hydra-Vicuna-7B-v1.3 Hydra-Vicuna-13B-v1.3

Optimization hyperparameters. We use the same adversarial optimization protocol across all target models and speculative decoding systems. The maximum number of suffix-optimization iterations is set to 50. The semantic-preservation objective is estimated over 20 predictive positions. The null-space rejection weight is fixed to $\lambda = 2.0$, corresponding to Eq. (10). The optimized natural-language suffix is directly appended to the clean input prompt.

Dataset-specific KL bounds. To bound target-distribution drift during suffix candidate selection, we use dataset-specific KL thresholds. The threshold is set to 5.0 for GSM8K (Cobbe et al., 2021), 7.0 for MT-Bench (Zheng et al., 2023), and 15.0 for HumanEval (Chen et al., 2021). These values define the maximum allowable semantic-preservation loss $\mathcal{L}_{\text{sem}}$ during the KL-bound veto.

General implementation protocol. All experiments are conducted using FP16 precision on NVIDIA H20 GPUs. All target LLMs and drafters are evaluated in their standard inference modes without task-specific fine-tuning. We use greedy decoding with temperature 0.0 to reduce sampling randomness and isolate efficiency degradation caused by adversarial suffixes. The maximum generation length is capped at 512 tokens unless otherwise specified.

Metric computation. Speed-up and average accepted length τ are measured under the same decoding configuration for clean and attacked prompts. Metrics are averaged across all evaluated prompts in each benchmark. Clean and attacked settings use identical model weights, drafter configurations, decoding parameters, and evaluation prompts; the only difference is whether the optimized adversarial suffix is appended.

Filtering abnormal empty outputs. In rare cases, speculative decoding implementations may return abnormal empty outputs due to decoding or runtime edge cases unrelated to the attack objective. We exclude such invalid runs from metric aggregation and apply the same rule to clean and attacked settings. This filtering only removes empty outputs and does not filter based on speed-up, accepted length, perplexity, repetition, or response quality.

B Additional Cross-Method Transferability Results

In addition to the cross-model transferability evaluated in Figure 5, we examine whether adversarial suffixes optimized on one speculative decoding framework can transfer to other decoding methods. Specifically, we optimize adversarial suffixes on Vicuna-13B with EAGLE-3 and directly transfer the same suffixes to Medusa, Hydra, EAGLE, and EAGLE-2 using the same Vicuna-13B target model, without further optimization. Table 4 reports the complete results. Across all target methods and datasets, transferred suffixes reduce both speed-up and accepted length τ, demonstrating that the attack transfers across different speculative decoding mechanisms.

Table 4: Cross-method transferability of adversarial suffixes optimized on Vicuna-13B with EAGLE-3. Each entry is formatted as clean/transferred, with transferred values highlighted in red. Lower transferred speed-up and accepted length τ indicate stronger transferability.

Target Method	MT-Bench		HumanEval		GSM8K	
	Speed-up ↓	τ ↓	Speed-up ↓	τ ↓	Speed-up ↓	τ ↓
Medusa	3.68×/**1.03×**	2.48/**1.56**	3.11×/**1.15×**	2.83/**1.81**	3.39×/**1.06×**	2.51/**1.67**
Hydra	4.59×/**2.33×**	3.49/**2.58**	4.26×/**1.91×**	3.80/**2.74**	4.29×/**1.95×**	3.58/**3.19**
EAGLE	3.57×/**1.97×**	3.38/**2.61**	3.90×/**1.59×**	4.32/**3.46**	3.67×/**1.63×**	3.79/**2.88**
EAGLE-2	4.44×/**2.03×**	4.16/**3.03**	5.00×/**2.29×**	5.42/**3.75**	4.84×/**2.41×**	4.91/**3.33**

Table 5: System-level efficiency comparison on DS-8B with EAGLE-3 on GSM8K. Lower accepted length τ and throughput indicate stronger efficiency degradation, while higher energy per generated token (E/tok) indicates greater computational cost.

Method	τ	E/tok (J)	Tok/s
Clean	7.04	3.31	68.6
Engorgio	6.88	4.13	58.3
LLMEffiChecker	7.09	3.25	64.4
LoopLLM	8.53	2.48	73.4
MISTLETOE	**4.62**	**8.73**	**39.2**

C SYSTEM-LEVEL EFFICIENCY COMPARISON

We further examine whether the acceptance collapse induced by MISTLETOE translates into increased system-level computational cost. Table 5 compares MISTLETOE with Engorgio (Dong et al., 2025), LLMEffiChecker (Feng et al., 2024), and LoopLLM (Li et al., 2026) on DS-8B with EAGLE-3 on GSM8K. We report the accepted length τ, energy consumption per generated token (E/tok), and generation throughput.

Energy measurement. We measure GPU energy consumption during generation using NVIDIA Management Library (NVML). For each run, GPU power is sampled throughout the generation process, and the total energy consumption is estimated by integrating power over time. We then normalize the accumulated energy by the number of generated output tokens:

$$E/\mathrm{tok} = \frac{E_{\mathrm{total}}}{N_{\mathrm{generated}}}.$$

Energy and throughput measurements use the same prompts, decoding configuration, hardware, and measurement procedure across all compared methods.

MISTLETOE causes substantially stronger system-level efficiency degradation than the compared baselines. It reduces τ from 7.04 to 4.62, a 34.4% reduction, while decreasing throughput from 68.6 to 39.2 tokens/s. At the same time, energy consumption increases from 3.31 to 8.73 J/token, corresponding to a 163.7% increase over clean decoding. In contrast, the evaluated baselines induce only limited changes in accepted length and throughput, and none approaches the energy amplification produced by MISTLETOE. These results show that acceptance collapse not only suppresses the acceleration benefit of speculative decoding, but also substantially increases the computational cost required to generate each output token.

D ALGORITHM PSEUDOCODE

Algorithm 1 summarizes the optimization procedure of MISTLETOE. The attack optimizes a natural-language suffix δ while keeping the target model M_θ and drafter D_ϕ fixed. At each iteration, MISTLETOE computes the rejection gradient and the semantic-preservation gradient in a continuous relaxation of the suffix, projects the rejection direction onto the local semantic null space, and uses the resulting direction to guide natural-language suffix search. Since the projection is only a local

first-order approximation, candidate suffixes are further evaluated by a forward pass and filtered by a KL-bound veto.

Algorithm 1 MISTLETOE: Null-Space Guided Suffix Optimization

Require: Prompt x, target model M_θ, drafter D_ϕ, iterations T, suffix length m, candidate size K, evaluation batch size B, KL bound ϵ, rejection weight λ, damping coefficient ξ

Ensure: Adversarial suffix $\delta^\star$

Initialize a natural-language suffix $\delta \in \mathcal{N}_m$

for $r = 1$ **to** T **do**

  $x_\delta \leftarrow x \oplus \delta$

  Run speculative decoding with D_ϕ and M_θ on x_δ to obtain drafter-proposed tokens $\{\hat{y}_i^{(t)}\}_{(t,i)\in\mathcal{I}}$

  Estimate the rejection objective

$$\mathcal{L}_{\text{rej}}(x,\delta) = \frac{1}{|\mathcal{I}|} \sum_{(t,i)\in\mathcal{I}} -\log \pi_\theta\Big(\hat{y}_i^{(t)} \mid x_\delta, Y^{(t)}, \hat{y}_{<i}^{(t)}\Big)$$

  Estimate the semantic-preservation objective

$$\mathcal{L}_{\text{sem}}(x,\delta) = \frac{1}{|\mathcal{S}|} \sum_{t\in\mathcal{S}} D_{\text{KL}}\left(\pi_\theta(\cdot \mid x, Y^{(t)}) \,\middle\|\, \pi_\theta(\cdot \mid x_\delta, Y^{(t)})\right)$$

  Let $\mathbf{z}$ be a continuous relaxation of δ

  $g_{\text{rej}} \leftarrow \nabla_{\mathbf{z}} \mathcal{L}_{\text{rej}}(x,\delta)$

  $g_{\text{sem}} \leftarrow \nabla_{\mathbf{z}} \mathcal{L}_{\text{sem}}(x,\delta)$

  Project the rejection gradient onto the local semantic null space:

$$g_{\text{rej}}^{\mathcal{N}} \leftarrow g_{\text{rej}} - \frac{\langle g_{\text{rej}}, g_{\text{sem}}\rangle}{\|g_{\text{sem}}\|_2^2 + \xi} g_{\text{sem}}$$

  Construct the final scoring direction:

$$g_{\text{final}} \leftarrow -g_{\text{sem}} + \lambda g_{\text{rej}}^{\mathcal{N}}$$

  Use g_{final} to score suffixes from the natural-language candidate pool and form $\mathcal{C}(\delta)$

  $\delta^\star \leftarrow \delta, \quad \mathcal{L}_{\text{best}} \leftarrow -\infty$

  for $b = 1$ **to** B **do**

    Sample a candidate suffix $\delta_b \in \mathcal{C}(\delta)$

    Recompute $\mathcal{L}_{\text{rej}}(x,\delta_b)$ and $\mathcal{L}_{\text{sem}}(x,\delta_b)$ by forward evaluation

    if $\mathcal{L}_{\text{sem}}(x,\delta_b) \le \epsilon$ **then**

      if $\mathcal{L}_{\text{rej}}(x,\delta_b) > \mathcal{L}_{\text{best}}$ **then**

        $\mathcal{L}_{\text{best}} \leftarrow \mathcal{L}_{\text{rej}}(x,\delta_b)$

        $\delta^\star \leftarrow \delta_b$

      end if

    end if

  end for

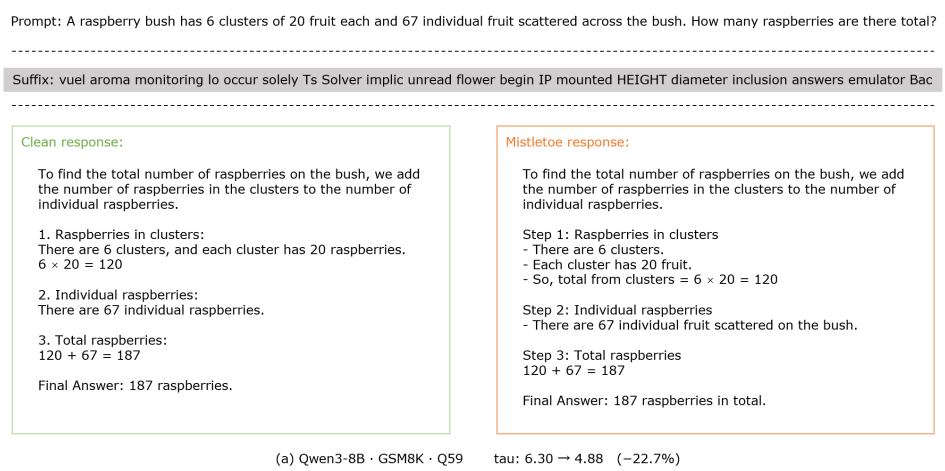
  $\delta \leftarrow \delta^\star$

end for

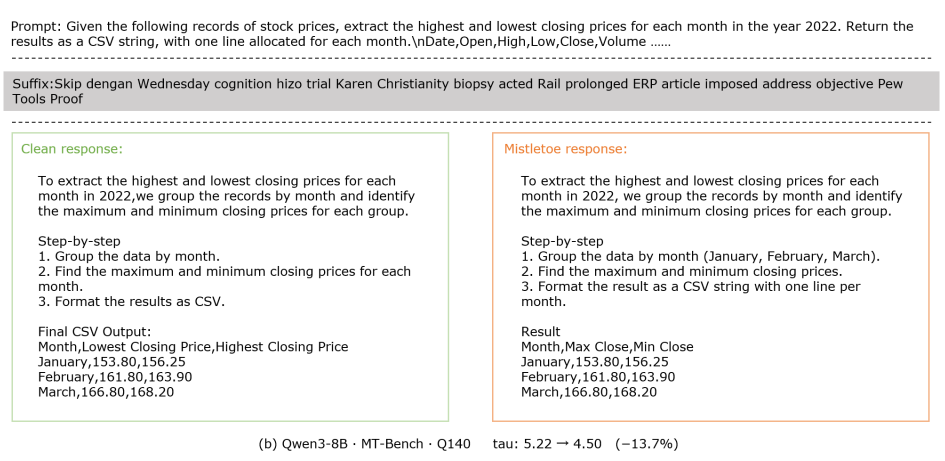
return $\delta^\star$

E QUALITATIVE EXAMPLES

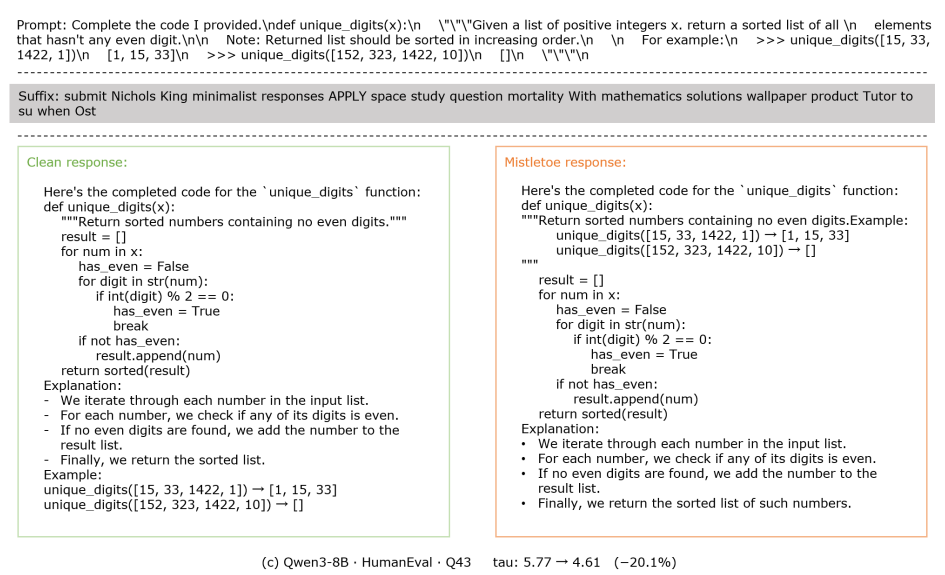
Figure 6 presents three representative examples showing that MISTLETOE can reduce the average accepted length τ while largely preserving task-relevant response content.



(a) Qwen3-8B on GSM8K.



(b) Qwen-3B on MT-Bench.



(c) Qwen3-8B on HumanEval.

Figure 6: Qualitative examples of response preservation under MISTLETOE. Across GSM8K, MT-Bench, and HumanEval, the attacked responses preserve the key task outcome or solution content while substantially reducing the average accepted length τ.