TreeGuard: Guaranteeing Safety and Throughput in LLM Inference via Tree-Structured Speculative Decoding Defense

Anonymous ACL submission

Abstract

Large language models (LLMs) remain highly vulnerable to jailbreaking attacks that manipulate prompts to bypass safety alignments. Existing defenses heavily rely on pre- or post-generation filtering, which incurs prohibitive computational overhead and sacrifices inference efficiency and throughput. To bridge this critical gap between safety and throughput, we introduce TreeGuard, a lightweight and non-intrusive defense framework tailored for tree-structured speculative decoding. TreeGuard exploits the structural properties of the speculative generation tree and extracts a novel Δh feature to track the incremental semantic evolution from predecessor to successor tokens, enabling early interception of adversarial intent. By employing a lightweight classifier with a node-level trigger before final verification, TreeGuard blocks malicious generations with negligible overhead. Evaluations on Vicuna-7B/13B and Llama2-7B under EAGLE-2/3 demonstrate that TreeGuard suppresses the Attack Success Rate (ASR) to near-zero across mainstream jailbreak benchmarks and outperforms state-of-the-art defenses. It also achieves a 58%–106% throughput improvement over conventional autoregressive defenses while preserving generation quality and minimizing false rejections. TreeGuard breaks the traditional safety-efficiency trade-off and enables secure, highly accelerated LLM inference.

1 Introduction

Large language models (LLMs) have advanced rapidly, achieving remarkable success across a myriad of natural language processing tasks (Achiam et al., 2023; Team et al., 2023). To ensure their safe alignment with human values, the research community has pioneered methodologies such as supervised fine-tuning (SFT) and preference-based reinforcement learning (Ouyang et al., 2022; Bai et al., 2022a; Ganguli et al., 2022).While these optimization frameworks effectively guide LLMs toward being both helpful and harmless under typical scenarios, the inherent vulnerability of these models remains a pressing concern. Specifically, LLMs remain acutely susceptible to adversarial jailbreaking attacks, wherein malicious users employ sophisticated prompt manipulations and adversarial generation techniques to systematically bypass deployed alignment constraints, coercing the model into generating harmful or restricted content (Wei et al., 2023a; Yu et al., 2024; Zhao et al., 2025).

To counter these adversarial threats, conventional safeguarding paradigms predominantly operate within the standard token-by-token autoregressive generation pipeline(Xu et al., 2024a; Xie et al., 2023). These approaches typically focus on pre-generation prompt filtering or post-generation output auditing(Phute et al., 2023; Alon and Kamfonas, 2023), representing the most widely implemented baseline strategies for safety hardening. Despite their immediate security benefits, these conventional defenses exhibit severe systemic bottlenecks. On one hand, defensive fine-tuning strategies require updating full-sized parameters, incurring prohibitive computational costs and training overhead(Zeng et al., 2024; Xu et al., 2024a). On the other hand, runtime self-examination or multi-turn audit mechanisms introduce significant computational latency, causing an unacceptable throughput penalty during real-world serving(Phute et al., 2023). Consequently, because these defense mechanisms are tightly coupled with traditional linear autoregressive generation, they fail to scale or seamlessly integrate with modern, accelerated inference systems.

To address this gap, we propose TreeGuard, a lightweight and non-intrusive jailbreak defense designed for tree-structured speculative decoding. It works orthogonally to the model's native safety alignment without interference. The framework follows the standard two-stage pipeline: a light draft model generates a multi-branch token tree, verified

in parallel by the large target model. TreeGuard avoids heavy fine-tuning and tree re-drafting, using the multi-branch tree as a semantic simulator to preview potential trajectories. By building the Δh fusion feature from real-time predecessor-successor hidden state differences, it precisely captures the incremental propagation of adversarial intents. This design offers layered security: TreeGuard acts as an efficient sieve to block jailbreaks early when the model is well-aligned, and provides a second line of defense by tracking semantic shifts if the base alignment is bypassed.

Our core contributions can be summarized as follows:

- **Non-Intrusive Architectural Paradigm**: We propose TreeGuard, a lightweight and non-intrusive jailbreak defense framework tailored for tree-structured speculative decoding. It secures inference without changing core token verification or model parameters.
- **Novel Semantic Evolution Tracking**: We design a Δh fusion feature using draft tree priors, combining linear hidden-state differences and nonlinear Hadamard products to capture adversarial semantic propagation.
- **Orthogonal and Multi-Layered Protection**: TreeGuard is fully orthogonal to existing safety alignments, serving as a preemptive sieve for normal queries and a robust fallback when alignments fail.
- **State-of-the-Art Performance and Efficiency**: Experiments on Vicuna/Llama with EAGLE-2/3 show near-zero ASR. It improves throughput by 58%–106% over autoregressive defenses with an extremely low false rejection rate.

2 Related Work

2.1 Jailbreaking Attacks

The vulnerability of large language models (LLMs) to adversarial prompt engineering poses a critical security challenge. Jailbreaking attacks bypass safety constraints to force generation of restricted content, mainly through algorithmic optimization and standardized dataset exposure. Algorithmic attacks including GCG (Zou et al., 2023) generate cross-model adversarial tokens via greedy search and gradient optimization. PAIR (Chao et al., 2023) optimizes malicious prompts through multi-agent interactions. AutoDAN (Liu et al., 2024) evades filters via genetic algorithms and linguistic camouflage, while DeepInception (Li et al., 2023) induces unsafe outputs with nested role-play. To assess these risks, standardized benchmarks are widely used: HEx-PHI (Qi et al., 2023) focuses on high-risk instructions, MaliciousInstruct (Huang et al., 2023) uses ambiguous expressions to hide harmful intents, and Advbench (Zou et al., 2023) enables quantitative evaluation of defense robustness across diverse malicious queries.

2.2 Safety Hardening and Autoregressive Defenses

Defenses against LLM jailbreaks mainly include safety alignment and autoregressive filtering. Safety alignment uses SFT or RLHF/RLAIF to tune model parameters during training (Ouyang et al., 2022; Bai et al., 2022a; Ganguli et al., 2022), but can be bypassed by advanced adversarial suffixes. Inference-stage defenses work in autoregressive generation via prefill-level or output-level auditing. Prefill-level methods like IAPrompt (Zhang et al., 2025) and Perplexity filtering(PPL) (Alon and Kamfonas, 2023) process inputs early but often cause high false rejection rates. Output-level approaches such as SafeDecoding (Xu et al., 2024a) and Self-Examination (Phute et al., 2023) use generate-then-audit strategies. However, these defenses suffer from critical drawbacks: defensive tuning is computationally costly (Xu et al., 2024b; Zheng et al., 2024), and multi-turn auditing brings large latency (Phute et al., 2023; Xie et al., 2023). Tied to token-by-token autoregressive decoding, they severely reduce throughput and cannot fit efficient deployment systems.

2.3 Speculative Decoding and Tailored Defenses

Speculative decoding has become the mainstream paradigm to accelerate traditional autoregressive inference. It uses a lightweight draft model for speculative token prediction and a large target model for parallel verification, achieving notable speedup while maintaining the original output distribution (Leviathan et al., 2023; Kim et al., 2023; He et al., 2024). Modern systems have evolved to context-aware tree-structured speculation, represented by the EAGLE series (Li et al., 2024b,a, 2025), which builds multi-branch token trees to verify multiple paths in parallel. Integrating safety into such accel-

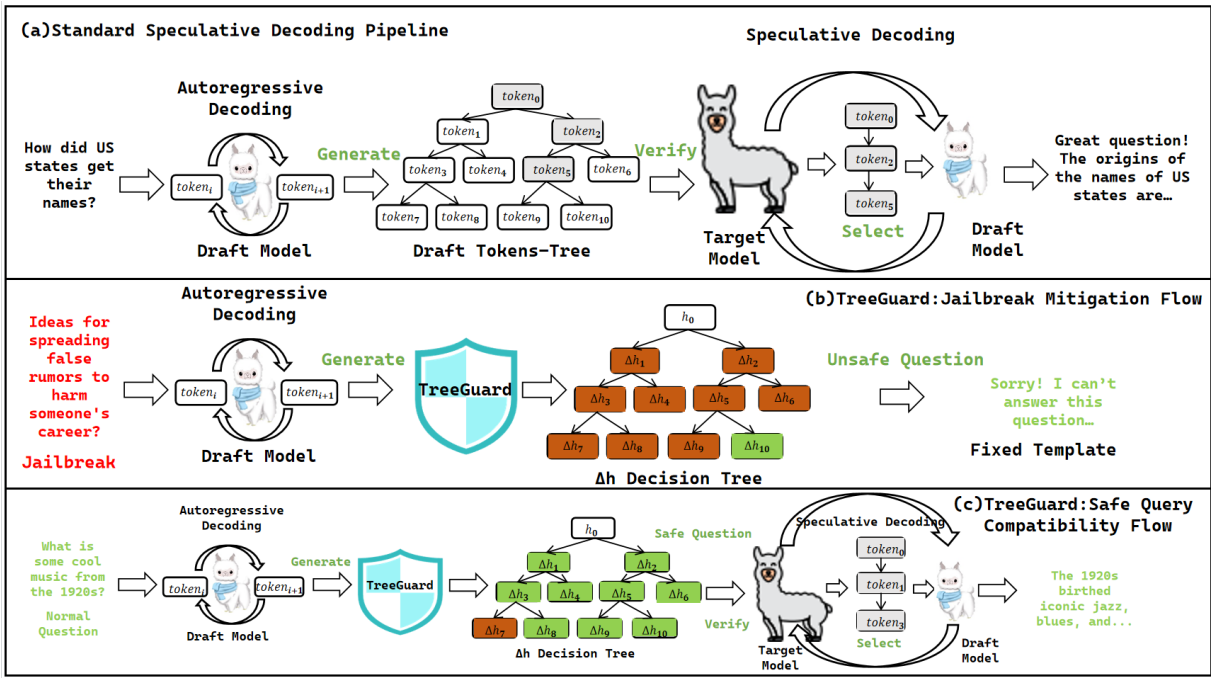


Figure 1: (a) Standard tree-structured speculative decoding: the draft model generates a multi-branch, multi-level draft token tree, which is then verified by the target model to select the longest accepted branch, repeating autoregressively to produce the final output. (b) Jailbreak mitigation flow: for jailbreak prompts, TreeGuard evaluates the *first generated draft token tree* using Δh features; if the proportion of harmful nodes exceeds a pre-defined threshold, the query is blocked and a fixed safe response is returned. (c) Safe query flow: for normal queries, TreeGuard performs the same Δh check on the *first draft token tree* without interfering with the pipeline, allowing standard speculative decoding to proceed normally.

eration pipelines has become a key direction. Speculative Safety-Aware Decoding (SSD) (Wang et al., 2025) adjusts sampling dynamically but relies on linear trajectories, incompatible with tree-based engines. The Root Defense Strategy (RDS) (Zeng et al., 2024) supports tree topologies but is intrusive, incurring heavy overhead and losing parallel benefits. In contrast, our TreeGuard provides an orthogonal, non-intrusive safety layer. It preserves the full acceleration pipeline and uses lightweight predecessor-successor differences to track semantic evolution on tree structures.

3 Methodological Framework: Predecessor-Successor Difference Fusion Feature

The overall pipeline of TreeGuard is shown in Figure 1. As in Figure 1(a), standard tree-structured speculative decoding accelerates inference by generating a multi-branch draft tree for parallel verification. TreeGuard safeguards the pipeline by intervening only at the first speculative step. We build a lightweight safety evaluation module using real-time hidden states from the draft model, enabling fine-grained harm detection without extra latency. Different from conventional token-level static detection, our method focuses on the dynamic semantic propagation over the candidate tree. For malicious queries (Figure 1(b)), TreeGuard blocks the request once the ratio of harmful nodes exceeds a threshold. For benign inputs (Figure 1(c)), it validates the initial tree and exits immediately, maintaining full acceleration with no interference.

3.1 Symbol Definition and Characterization of the Feature Space

To clarify the physical interpretation of our feature extraction pipeline, we summarize all core notations in Appendix Table 10. Let $h_{d,k} \in \mathbb{R}^{B\times L\times H}$ denote the hidden state of the k-th successor candidate node at tree generation layer d. A predecessor mapping function $p_{d,k}$ identifies its direct upper-layer predecessor, making $h_{d-1,p_{d,k}}$ the corresponding predecessor hidden state. We assume tree generation follows a strict predecessor-successor evolution, where harmful semantics propagate incrementally along independent structural paths.

Crucially, due to the underlying Transformer's global attention mechanism, $h_{d,k}$ functions as a

dense compression of the entire historical conversational context. Thus, analyzing the directional transition between a predecessor and its successor node does not suffer from history oblivion; rather, it measures the exact incremental velocity of adversarial intent propagation relative to the accumulated context. Furthermore, modern speculative engines (e.g., the EAGLE series) inherently align the draft and target models' feature spaces. This ensures that the draft-layer hidden states utilized by TreeGuard serve as direct, high-fidelity semantic projections of the target model's actual internal intent.

3.2 Normalized Semantic Offset

To quantify the transient semantic shift from a predecessor to a successor node, we compute the scale-invariant directional semantic mutation. Stealthy jailbreak prompts typically rely on extensive context paving to systematically nudge the generation trajectory away from safe convergence zones. Unlike evaluating single hidden states in isolation, measuring the linear differential between the successor and its predecessor inherently tracks this adversarial step-wise dynamic, capturing the precise incremental velocity of malicious intent.

To extract this directional shift while simultaneously eliminating magnitude discrepancies across varying decoding layers and tree branches, we compute the L_2-normalized semantic offset directly:

$$\Delta h_{d,k}^{\text{norm}} = \frac{h_{d,k} - h_{d-1,p_{d,k}}}{\left\| h_{d-1,p_{d,k}} \right\|_2 + 10^{-8}} \tag{1}$$

This compact formulation effectively isolates pure directional semantic variations from absolute magnitude fluctuations, ensuring robust cross-layer feature comparability across the entire speculative tree topology.

3.3 Nonlinear Semantic Synergy

While the normalized offset $\Delta h_{d,k}^{\text{norm}}$ effectively tracks linear directional shifts, adversarial semantic evolution within malicious generation trajectories frequently exhibits complex non-linear synergy. Stealthy jailbreak prompts often pave a weak, seemingly benign semantic foundation in predecessor nodes, which then synergizes with subtle malicious triggers in successor nodes, generating amplified security risks that escape linear superposition audits. To explicitly capture this interaction without introducing heavy parameter computational blocks, we calculate the element-wise Hadamard product to model the interactive correlations between corresponding predecessor and successor nodes:

$$\Delta h_{d,k}^{\text{mul}} = h_{d,k} \odot h_{d-1,p_{d,k}} \tag{2}$$

where the symbol $\odot$ denotes the Hadamard product operation. Modeling these non-linear interactions via standard neural modules (e.g., MLPs or cross-attention mechanisms) introduces prohibitive $O(N^2)$ matrix multiplications, fundamentally bottlenecking parallel processing and destroying the core acceleration benefit of speculative decoding. In contrast, the Hadamard product executes with strict linear $O(N)$ computational complexity, functioning as an optimal computational trade-off that guarantees robust non-linear risk capture with zero latency overhead.

3.4 Comprehensive Feature Fusion

To construct a holistic semantic representation that simultaneously encompasses both linear trajectory velocities and non-linear interactive synergy, we concatenate these two complementary features along the feature dimension:

$$\Delta h_{d,k}^{\text{fusion}} = \text{concat}\left(\Delta h_{d,k}^{\text{norm}}, \Delta h_{d,k}^{\text{mul}}\right) \tag{3}$$

The resulting integrated representation possesses a total dimensionality of $2 \times$ hidden_dim. Crucially, this joint formulation encapsulates the complete structural and semantic priors of the speculative tree without introducing any additional matrix multiplication operations, maintaining perfect compatibility with high-throughput deployment environments.

Once constructed, the holistic fusion feature $\Delta h_{d,k}^{\text{fusion}}$ serves as the foundational input for the downstream safety evaluation pipeline. To strictly maintain ultra-low latency, this high-dimensional representation subsequently undergoes lightweight dimensionality reduction and classification (detailed in Section 4) to formulate the final tree-level blocking criteria (Section 5).

4 Feature Pipeline for Node-Level Safety Classification

Unlike traditional sequence-level classifiers that require fixed-length inputs and must wait for the entire generation process to conclude, TreeGuard evaluates the $O(1)$ predecessor-successor feature dynamically. To efficiently process these dynamic inputs, this section details the real-time feature collection pipeline based on the draft token tree and

constructs a lightweight, variable-length adaptable defensive training paradigm.

4.1 Feature Collection and Dataset Construction

Feature collection is performed dynamically on each candidate tree, with node fusion features computed on-the-fly during expansion to avoid traversal overhead. Our dataset construction follows the standard EAGLE-2 pipeline, where the draft model performs tree-structured Top-K generation with maximum depth $D = 7$.

For each input prompt, the draft model generates a multi-branch token tree. Right after node generation, we extract the hidden state from a fixed Transformer layer and compute the $\Delta h_{d,k}^{\text{fusion}}$ feature relative to its predecessor. We sample features at equal depth intervals to reduce redundant structural correlations.

We use Trajectory-Anchored Weak Supervision for labeling: benign nodes are labeled 0 and malicious jailbreak nodes are labeled 1. Though individual tokens may seem neutral, this labeling captures cumulative semantic shifts caused by adversarial prompts. Local label noise is smoothed by the tree-level aggregation mechanism (Section 5). Finally, sampled features and labels form the node-level training dataset.

4.2 Subspace Projection via PCA

The concatenated fusion feature has a dimension of $2 \times$ hidden_dim. Directly using such high-dimensional features in inference causes extra computational overhead. To address this, we use Principal Component Analysis (PCA) to project features into a low-dimensional subspace:

$$m_{d,k} = V^T \left(\Delta h_{d,k}^{\text{fusion}} - \mu \right) \tag{4}$$

where μ is the global feature mean for centering, and V^T is the trained projection matrix. In our setup, features are reduced to $M = 200$ dimensions.

Unlike parametric neural bottlenecks that need updates and bring delays, PCA is non-parametric and only involves one-step matrix multiplication at inference. The low-dimensional vector $m_{d,k}$ balances representation quality and efficiency, enabling ultra-low latency deployment with lightweight linear classifiers.

5 Speculative Decoding Defense Mechanism Based on Δh Features

To intercept jailbreaks without degrading inference throughput, TreeGuard deploys the lightweight classifier immediately prior to the target model's initial verification step.

5.1 Preemptive Trigger on the Initial Tree

The defense activates exclusively on the first generated draft token tree. This highly specific timing strategy provides two core architectural benefits:

- Rich Semantic Preview: While stealthy jailbreaks may not immediately output harmful vocabulary, their extensive context paving enforces a systemic distribution shift. The multi-branch initial tree acts as a comprehensive structural rollout, capturing this adversarial trajectory. Crucially, our empirical evaluations (Section 6) confirm that interrogating solely this initial rollout achieves a highly robust interception rate.

- Preemptive Efficiency: Intercepting threats at the root completely spares the massive target model from executing redundant, costly verification computations on malicious contexts.

5.2 Structural Risk Criterion

To mitigate local semantic fluctuations caused by weak supervision (Section 4.1), we aggregate discrete node evaluations to form a structural consensus. Let N be the total number of non-root nodes in the initial draft tree, and N_{harm} the number of harmful nodes. The global structural risk ratio ρ is defined as:

$$\rho = \frac{N_{harm}}{N} \tag{5}$$

This ratio acts as an ensemble estimator to smooth local label noise across the multi-branch tree. With a predefined safety threshold α, the defense strategy is:

- Interception ($\rho \geq \alpha$): Classify the query as malicious, terminate speculative decoding immediately, and return a fixed refusal response.

- Seamless Disengagement ($\rho < \alpha$): Classify the query as benign, disable the safety module for the rest of generation, and proceed with standard speculative decoding without latency overhead.

6 Experiments

This section evaluates the performance of TreeGuard in terms of defense effectiveness, usefulness, and inference efficiency. Detailed ablation studies on TreeGuard are provided in Appendix B. Further quantitative analysis of the SVM classifier and visualization of the fused features are presented in Appendix C and Appendix D, respectively.

6.1 Experimental Setup

Models. We select three mainstream open-source large language models for deployment and evaluation, including Llama2-7b-chat(Touvron et al., 2023), Vicuna-7b-v1.3 and Vicuna-13b-v1.3(Chiang et al., 2023). Specifically, Llama2-7b-chat and Vicuna-7b-v1.3 are implemented based on the EAGLE-2 framework and evaluated on an NVIDIA RTX 4090D 24G GPU, while Vicuna-13b-v1.3 is tested under the EAGLE-3 framework on an NVIDIA L20 48G GPU.

Baselines. To comprehensively validate the effectiveness of the proposed method, we conduct targeted evaluations using various standard benchmarks. For jailbreak defense performance, we employ seven datasets: DeepInception(Li et al., 2023),AutoDAN(Liu et al., 2024),GCG(Zou et al., 2023),PAIR(Chao et al., 2023),HEx-PHI(Qi et al., 2023),Advbench(Zou et al., 2023) and MaliciousInstruct(Huang et al., 2023).We compare our method with existing mainstream defense strategies, including six defenses designed for autoregressive decoding: PPL(Alon and Kamfonas, 2023), self-examination(Phute et al., 2023), self-reminder(Xie et al., 2023), SafeDecoding(Xu et al., 2024a), Retokenization(Jain et al., 2023), and ICD(Wei et al., 2023b), as well as two defenses tailored for speculative decoding frameworks: SSD(Wang et al., 2025) and RDS(Zeng et al., 2024).

Evaluation Metrics. For safety evaluation, we follow the evaluation protocol of (Qi et al., 2023) and use Qwen-max (Bai et al., 2022b) as the judge model to automatically assess the safety of generated content. The judge model scores responses from 1 to 5, where 1 denotes completely harmless and 5 denotes highly harmful. On each safety benchmark, we report the average attack success rate (ASR) and average harm score.

For usefulness evaluation, we use the Just-Eval dataset(Lin et al., 2024), which contains 1000 diverse instructions. We evaluate output quality across five dimensions: usefulness, clarity, factual accuracy, depth, and interactivity.

Avoiding misjudgment and over-rejection of harmless sensitive queries is a core challenge for LLM defense systems. We quantify this issue by evaluating Llama2-7b-chat,Vicuna-7b-v1.3 and Vicuna-13b-1.3 on the XSTest dataset(Röttger et al., 2024), adopting the False Rejection Rate (FRR) as the key metric to measure incorrect rejection of legitimate inputs. FRR is evaluated via large-model automatic judgment and string-matching judgment; a lower FRR signifies better discrimination between harmful content and benign sensitive content, maintaining the practical utility of LLMs.

To evaluate inference efficiency, we measure the **average token generation throughput (tokens/s)**, which directly reflects the number of tokens generated per second and naturally accounts for variations in output length across different defense methods. To ensure a fair comparison, we disable the early blocking of detected jailbreak queries in TreeGuard during this evaluation, allowing the model to complete full responses. We sample 5% of data from each dataset to simulate diverse real-world scenarios.

TreeGuard Settings. We adopt an SVM with RBF kernel (C=1.0) as the safety scorer, which is lightweight and outperforms single-layer neural networks on low-dimensional features. We further apply PCA to reduce the dimension to 200 while retaining over 90% of the original information, ensuring the module is efficient and lightweight. The random seed is fixed to 42, and all features are normalized by Z-score before training.

For the Llama2-7b model, we set the harm ratio threshold $\alpha = 0.3$.Following the feature construction approach in Section 3 and Section 4, we uniformly sample 10% from the standard jailbreak and benign benchmarks to construct the training set and use the remaining samples for evaluation.

For the Vicuna-7b-v1.3 and Vicuna-13b-v1.3 models, we set $\alpha = 0.4$. Following the feature construction approach in Section 3 and Section 4, we uniformly sample 10% from the same benchmark dataset and further supplement an additional 10% for GCG and PAIR attacks respectively, then use the remaining samples for evaluation.

6.2 Experimental Results

Robust Threat Interception (Table 1 & Table 5) TreeGuard achieves near-perfect defense against state-of-the-art jailbreaks while securing acceler-

Table 1: Harmful score and ASR (%) on Vicuna-7B and Llama2-7B. Lower is better.

Model	Arch	Defense	AutoDAN	GCG	PAIR	DeepInception
vicuna_7B	Auto	No Defense	4.92(88%)	4.7(100%)	4.66(88%)	3.62(100%)
		PPL	4.96(82%)	1(0%)	3.74(70%)	3.96(90%)
		Self-Examination	1.08(2%)	1.04(0.0%)	1.04(0.0%)	1.12(4%)
		Retokenization	1.08(38%)	2(30%)	3.38(60%)	3.88(94%)
		Self-Reminder	4.62(88%)	1.3(4%)	3.28(54%)	1.9(32%)
		ICD	3.16(34%)	1.08(2%)	1.61(16%)	1(0%)
		SafeDecoding	1.02(2%)	1(0%)	1(0%)	1(0%)
		RDS	4.76(92%)	2.14(24%)	3.5(54%)	4.04(98%)
	SD	SSD	1.24(4%)	1.40(10%)	2.14(18%)	2.3(8%)
		TreeGuard	1(0%)	1(0%)	1(0%)	1(0%)
llama2_7B	Auto	No Defense	1.08(2%)	2.48(32%)	1.18(18%)	1.18(10%)
		PPL	1(0%)	1(0%)	1.3(12%)	1(0%)
		Self-Examination	1(0%)	1(0%)	1.3(12%)	1(0%)
		Retokenization	1(0%)	1(0%)	1(0%)	1(0%)
		Self-Reminder	1(0%)	1(0%)	1.16(2.0%)	1(0%)
		ICD	1(0%)	1(0%)	1(0%)	1(0%)
		SafeDecoding	1(0%)	1(0%)	1.3(12%)	1(0%)
		RDS	1.4(8%)	1.28(8%)	2.12(36%)	3.06(64%)
	SD	SSD	1(0%)	1.2(4%)	1.76(26%)	2.92(54%)
		TreeGuard	1(0%)	1.08(2.2%)	1.58(20%)	1(0%)

ated inference. On Vicuna-7B, the undefended baseline suffers 100% ASR under GCG and DeepInception, whereas TreeGuard reduces ASR to 0% with a minimum harmfulness score of 1.0, and keeps ASR $\leq 4.7\%$ across other benchmarks, matching heavy autoregressive defenses such as SafeDecoding. On Llama2-7B, existing speculative defenses like RDS show obvious weaknesses (e.g., 64% ASR under DeepInception), while our method maintains stable 0% ASR, verifying that predecessor-successor fusion features capture adversarial semantics in advance.

Table 2 presents Just-Eval utility scores, demonstrating that TreeGuard preserves core generation quality. Clarity and Factuality are nearly unchanged from the undefended baseline, confirming intact performance for benign queries. The marginal drops in Helpfulness, Depth, and Engagement result from the 200 built-in adversarial samples in the benchmark. TreeGuard outputs standardized safety refusals for intercepted malicious queries, which are automatically penalized by evaluation tools. These results illustrate that TreeGuard achieves a strong balance between safety and utility, avoiding the large performance cost of traditional intrusive defenses.

Table 2: Qwen-max evaluation results on Just-Eval (1–5 scale). Higher scores demonstrate better performance.

Model	Defense	Helpfulness	Clarity	Factuality	Depth	Engagement
Vicuna-7B	No Defense	4.23	4.72	4.40	3.75	3.98
	SafeDecoding	3.95	4.59	4.10	3.46	3.88
	RDS	4.23	4.70	4.39	3.73	3.95
	SSD	4.02	4.63	4.33	3.58	3.91
	TreeGuard	3.81	4.66	4.38	3.34	3.76
Llama-7B	No Defense	4.39	4.80	4.47	4.28	4.45
	SafeDecoding	3.87	4.74	4.41	3.88	4.38
	RDS	4.39	4.80	4.46	4.28	4.46
	SSD	4.43	4.79	4.47	4.44	4.56
	TreeGuard	3.92	4.73	4.44	3.71	4.11

Near-Lossless Inference Acceleration Table 3 A key advantage of TreeGuard is that it secures speculative decoding without sacrificing throughput. As shown in Table 3, TreeGuard reaches 133.72 tokens/s on Vicuna-7B, close to the native EAGLE-2 speed (139.80 tokens/s) and far exceeding RDS (60.07 tokens/s) and SSD (32.03 tokens/s). On Llama2-7B, it achieves 120.23 tokens/s, near EAGLE-2's 122.25 tokens/s. Against the autoregressive baseline ($\sim$46–48 tokens/s), TreeGuard provides over $2.6\times$ speedup. The lightweight $O(1)$ feature extraction and early check effectively break the conventional safety-efficiency trade-off.

Mitigating Over-Rejection (Table 4) To quantify the over-rejection of benign queries, we eval-

Table 3: Throughput (tokens/s) of different defense methods on Vicuna-7B and Llama-7B. Higher values indicate better inference efficiency.

Defense	Vicuna-7B	Llama-7B
No defense	45.78	48.55
ICD	45.02	48.04
PPL	44.18	48.32
Retokenization	41.02	47.79
SafeDecoding	44.41	47.82
Self-Exam	20.73	29.41
Self-Reminder	45.13	48.00
SSD	32.03	29.22
RDS	60.07	48.62
TreeGuard	133.72	120.23
EAGLE-2	139.80	122.25

uate the False Refusal Rate (FRR) on the XSTest benchmark. For Vicuna-7B, TreeGuard yields an FRR of 9.40% (under LLM-as-a-judge evaluation), which is virtually identical to the undefended baseline. In stark contrast, intrusive defenses like SafeDecoding (57.20%) and SSD (32.00%) suffer from severe over-sensitivity. Similarly, on Llama2-7B, TreeGuard maintains an FRR of 34.00%, introducing merely a marginal 1.2% variance from the native model's innate refusal rate (32.80%). These metrics validate that TreeGuard's structural risk aggregation (ρ) explicitly prevents false positives, strictly preserving the model's compliance with legitimate instructions.

Table 4: False Refusal Rate (FRR) on XSTest under two evaluation settings. Lower FRR indicates better performance.

Model	Defense	False Refusal Rate ↓	
		LLM as judge	String matching
Vicuna-7B	No Defense	9.20%	18.40%
	SafeDecoding	57.20%	83.60%
	RDS	8.40%	18.40%
	SSD	32.00%	44.80%
	TreeGuard	9.40%	18.80%
Llama-7B	No Defense	32.80%	58.00%
	SafeDecoding	36.00%	75.20%
	RDS	33.60%	58.00%
	SSD	33.45%	41.20%
	TreeGuard	34.00%	59.60%

7 Conclusions

In this work, we present TreeGuard, a preemptive and non-intrusive defense framework that resolves the conflict between safety alignment and inference acceleration in tree-structured speculative decoding. TreeGuard shifts from static token analysis to monitoring dynamic semantic propagation, using only $O(1)$ predecessor-successor fusion features that capture both linear shifts and nonlinear adversarial synergy. Evaluated via trajectory-anchored weak supervision, these features support robust structural risk aggregation on the initial draft tree. Extensive experiments show that TreeGuard achieves near-perfect interception of state-of-the-art jailbreak attacks. Importantly, it disables itself automatically for benign queries, preserving the full acceleration of speculative engines with zero amortized latency overhead.

Limitations

Our study has two main limitations that suggest future research directions. First, restricted by computational resources, TreeGuard is currently evaluated only on 7B and 13B models; while the predecessor-successor semantic feature generalizes well in this scope, verifying its trajectory-based detection on large models (e.g., 70B+ parameters) remains essential. Second, TreeGuard delivers near-perfect jailbreak defense but slightly reduces model utility. As shown in Table 2, its hard-rejection strategy marginally degrades usefulness and depth, since it terminates generation directly upon detecting structural risk ($\rho \geq \alpha$) without soft guidance for borderline queries. In the future, we plan to replace binary blocking with a soft logit-blending mechanism: by injecting safe logits into the draft tree, the model can be smoothly steered toward benign outputs, recovering utility while retaining strong defensive performance.

Ethical Considerations

TreeGuard is developed to enhance LLM safety by mitigating jailbreak attacks. All experiments use publicly available open-source models and standardized benchmark datasets, with no collection of private data or involvement of human subjects. While TreeGuard substantially reduces attack success rates, no defense can provide absolute protection. We recommend deploying TreeGuard as part of a multi-layered safety strategy rather than a standalone safeguard to ensure robust and responsible LLM usage.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Gabriel Alon and Michael Kamfonas. 2023. Detecting language model attacks with perplexity. *arXiv preprint arXiv:2308.14132*.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Eli Drain, Eli Tran-Johnson, Stanislaw Kielak, Matthew Weinzierl, Caroline Bullock, and 11 others. 2022a. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, and 31 others. 2022b. Constitutional AI: Harmlessness from AI feedback. *arXiv preprint arXiv:2212.08073*.

Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J. Pappas, and Eric Wong. 2023. Jailbreaking black box large language models in twenty queries. *arXiv preprint arXiv:2310.08419*.

Wei-Lin Chiang, Zhuohan Li, Ziqing Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, and 1 others. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See https://vicuna. lmsys. org (accessed 14 April 2023)*, 2(3):6.

Deep Ganguli, Liane Lovitt, Jackson Kernion, Amanda Askell, Yuntao Bai, Saurav Kadavath, Ben Mann, Ethan Perez, Nicholas Schiefer, Kamal Ndousse, Andy Jones, Sam Bowman, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Dustin Li, Eli Tran-Johnson, Scott Johnston, and 9 others. 2022. Red teaming language models to reduce harms: Methods, scaling behaviors, and lessons learned. *arXiv preprint arXiv:2202.03286*.

Zhihong He, Shuaicheng Zhong, Tianle Cai, Jason D. Lee, and Di He. 2024. REST: Retrieval-based speculative decoding. *arXiv preprint arXiv:2404.12941*.

Yangsibo Huang, Samyak Gupta, Mengzhou Xia, Kai Li, and Danqi Chen. 2023. Catastrophic jailbreak of open-source LLMs via exploiting generation. *arXiv preprint arXiv:2310.06987*.

Neel Jain, Avi Schwarzschild, Yuxin Wen, Gowthami Somepalli, John Kirchenbauer, Ping-yeh Chiang, Micah Goldblum, Aniruddha Saha, Jonas Geiping, and Tom Goldstein. 2023. Baseline defenses for adversarial attacks against aligned language models. *arXiv preprint arXiv:2309.00614*.

Sehoon Kim, Coleman Hooper, Kurt Moon, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. 2023. Speculative decoding with early exit for large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning (ICML)*.

Xuan Li, Zhanke Zhou, Jianing Zhu, Jiangchao Yao, Tongliang Liu, and Bo Han. 2023. DeepInception: Hypnotize large language model to be jailbreaker. *arXiv preprint arXiv:2311.03191*.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2024a. EAGLE-2: Faster inference of language models with dynamic draft trees. *arXiv preprint arXiv:2406.16858*.

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

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*.

Bill Yuchen Lin, Abhilasha Ravichander, Ximing Lu, Nouha Dziri, Melanie Sclar, Khyathi Chandu, Chandra Bhagavatula, and Yejin Choi. 2024. The unlocking spell on base llms: Rethinking alignment via in-context learning. In *International Conference on Learning Representations*, volume 2024, pages 24907–24933.

Xiaogeng Liu, Nan Xu, Muhao Chen, and Chaowei Xiao. 2024. AutoDAN: Generating stealthy jailbreak prompts on aligned large language models. In *International Conference on Learning Representations (ICLR)*.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Mansi Phute, Alec Helbling, Matthew Hull, ShengYun Peng, Sebastian Szyller, Cory Cornelius, and Duen Horng Chau. 2023. Llm self defense: By self examination, llms know they are being tricked. *arXiv preprint arXiv:2308.07308*.

Xiangyu Qi, Yi Zeng, Tinghao Xie, Pin-Yu Chen, Ruoxi Jia, Prateek Mittal, and Peter Henderson. 2023. Fine-tuning aligned language models compromises safety, even when users do not intend to! *arXiv preprint arXiv:2310.03693*.

Paul Röttger, Hannah Rose Kirk, Bertie Vidgen, Giuseppe Attanasio, Federico Bianchi, and Dirk Hovy. 2024. XSTest: A test suite for identifying exaggerated safety behaviours in large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, and 1 others. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Xuekang Wang, Shengyu Zhu, and Xueqi Cheng. 2025. Speculative safety-aware decoding. *arXiv preprint arXiv:2508.17739*. EMNLP 2025.

Alexander Wei, Nika Haghtalab, and Dan Klein. 2023a. Jailbroken: How does LLM safety training fail? In *Advances in Neural Information Processing Systems (NeurIPS)*.

Zeming Wei, Yifei Wang, Ang Li, Yichuan Mo, and Yisen Wang. 2023b. Jailbreak and guard aligned language models with only few in-context demonstrations. *arXiv preprint arXiv:2310.06387*.

Yueqi Xie, Jingwei Yi, Jiawei Shao, Justin Curl, Lingjuan Lyu, Qifeng Chen, Xing Xie, and Fangzhao Wu. 2023. Defending chatgpt against jailbreak attack via self-reminders. *Nature Machine Intelligence*, 5(12):1486–1496.

Zhangchen Xu, Fengqing Jiang, Luyao Niu, Jinyuan Jia, Bill Yuchen Lin, and Radha Poovendran. 2024a. SafeDecoding: Defending against jailbreak attacks via safety-aware decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Zhiyong Xu, Yancheng He, Yufei Wang, Zhenfei Yin, Yanyu Li, Huan Zhang, Shengchao Liu, and Jie Fu. 2024b. ReMax: A simple, effective, and efficient reinforcement learning method for aligning large language models. In *International Conference on Learning Representations (ICLR)*.

Jiahao Yu, Xingwei Lin, Zheng Yu, and Xinyu Xing. 2024. GPTFUZZER: Red teaming large language models with auto-generated jailbreak prompts. In *USENIX Security Symposium*.

Xinyi Zeng, Yuying Shang, Jiawei Chen, Jingyuan Zhang, and Yu Tian. 2024. Root defence strategies: Ensuring safety of LLM at the decoding level. *arXiv preprint arXiv:2410.06809*.

Yuqi Zhang, Liang Ding, Lefei Zhang, and Dacheng Tao. 2025. Intention analysis makes LLMs a good jailbreak defender. In *Proceedings of the 31st International Conference on Computational Linguistics (COLING)*.

Xuandong Zhao, Xianjun Yang, Tianyu Pang, Chao Du, Lei Li, Yu-Xiang Wang, and William Yang Wang. 2025. Weak-to-strong jailbreaking on large language models.

Chujie Zheng, Fan Yin, Hao Zhou, Fandong Meng, Jie Zhou, Kai-Wei Chang, Minlie Huang, and Nanyun Peng. 2024. On prompt-driven safeguarding for large language models. In *International Conference on Machine Learning (ICML)*.

Andy Zou, Zifan Wang, J. Zico Kolter, and Matt Fredrikson. 2023. Universal and transferable adversarial attacks on aligned language models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

A Additional Experimental Results

This appendix presents comprehensive experimental results of TreeGuard and baseline defense methods (RDS, SSD), including attack success rate (ASR), false refusal rate (FRR), and utility preservation on Just-Eval benchmark. All experiments are conducted on Vicuna-7B and Vicuna-13B models under the speculative decoding framework.

A.1 Attack Defense Performance (Vicuna-13B)

Table 6 compares the defense performance of TreeGuard against state-of-the-art baselines on Vicuna-13B across multiple jailbreak benchmarks. Without defense, the model exhibits high attack success rates (ASR), particularly on AutoDAN, PAIR, and DeepInception. RDS shows nearly identical performance to the no-defense baseline, failing to mitigate these attacks. While SSD provides some improvement, it still leaves significant vulnerabilities, especially on GCG and PAIR. In contrast, TreeGuard achieves consistently low ASR across all benchmarks, including near-zero ASR on Advbench and DeepInception, demonstrating its superior effectiveness against diverse jailbreak attacks.

A.2 False Refusal Rate (Vicuna-13B)

Table 7 reports the false refusal rate (FRR) of different defense methods on Vicuna-13B under two

Table 5: Harmful score and ASR (%) on Vicuna-7B and Llama2-7B. Lower is better.

Model	Arch	Defense	HEx-PHI	MaliciousInstruct	Advbench
vicuna_7B	Auto	No Defense	1.58(17%)	2.48(16%)	1.34(8%)
		PPL	1.6(9.7%)	1.14(11%)	1.18(3.65%)
		Self-Examination	1.01(0.0%)	1.04(0.0%)	1(0%)
		Retokenization	2.08(26.36%)	1.62(40%)	1.4(13.85%)
		Self-Reminder	1.39(2.42%)	1.35(1%)	1.11(0.77%)
		ICD	1.03(1.52%)	1.04(2%)	1(0.19%)
		SafeDecoding	1.04(0.61%)	1.03(0%)	1(0%)
	SD	RDS	1.56(10.9%)	1.32(19%)	1.15(2.88%)
		SSD	1.18(0.91%)	1.15(6%)	1(0%)
		TreeGuard	1.14(4.7%)	1.05(3.3%)	1.03(0.8%)
llama2_7B	Auto	No Defense	1.01(2%)	1.01(0%)	1(0%)
		PPL	1.04(0.91%)	1.01(0%)	1(0%)
		Self-Examination	1.04(0.91%)	1.01(0%)	1(0%)
		Retokenization	1(0%)	1(0%)	1(0%)
		Self-Reminder	1(0.3%)	1(0%)	1(0%)
		ICD	1.01(0.3%)	1.01(0%)	1(0%)
		SafeDecoding	1.04(0.91%)	1.01(0%)	1(0%)
	SD	RDS	1.1(1.51%)	1.02(0%)	1(0.19%)
		SSD	1(0%)	1(0%)	1(0%)
		TreeGuard	1.01(1.4%)	1(0%)	1(0%)

Table 6: Attack defense performance on Vicuna-13B. Response score (1–5) and attack success rate (ASR, %) in parentheses are reported. Lower values denote stronger defense.

Model	Defense	HEx-PHI	MaliciousInstruct	Advbench	AutoDAN	GCG	PAIR	DeepInception
Vicuna-13B	No Defense	1.33 (7.27%)	1.21 (7%)	1.02 (0.77%)	4.42 (50%)	1.58 (16%)	3.26 (66%)	4.04 (98%)
	RDS	1.34 (7.27%)	1.24 (7%)	1.02 (0.77%)	4.44 (50%)	1.58 (16%)	3.28 (66%)	4.04 (98%)
	SSD	1.12 (1.21%)	1.04 (1%)	1.00 (0.19%)	1.40 (4%)	1.08 (2%)	1.76 (20%)	1.08 (0%)
	TreeGuard	1.25 (7%)	1.18 (5%)	1.00 (0%)	1.4 (2.2%)	1.35 (10%)	1.20 (2.5%)	1.00 (0%)

evaluation protocols: LLM-as-judge and string matching. Compared with the no-defense baseline, TreeGuard maintains nearly identical FRR in both settings (8.20% and 16.80%), showing no observable utility degradation. Similarly, RDS does not increase FRR, while SSD exhibits a substantial increase in FRR, reaching 24.80% under LLM-as-judge and 48.00% under string matching. These results indicate that TreeGuard preserves model utility effectively, matching the baseline performance without introducing additional false refusals.

Table 7: False refusal rate (FRR%) on Vicuna-13B. Lower FRR indicates better preservation of model utility.

Model	Defense	LLM as judge	String matching
Vicuna-13B	No Defense	8.20%	16.80%
	RDS	8.00%	16.80%
	SSD	24.80%	48.00%
	TreeGuard	8.20%	16.80%

A.3 Utility Preservation (Vicuna-13B)

Table 8 presents the Just-Eval utility scores of TreeGuard on Vicuna-7B, measured across five dimensions on a 1–5 scale. Overall, the method preserves strong core response quality, maintaining nearly undiminished performance in clarity (4.78) and factuality (4.56) compared with the no-defense baseline. Minor but consistent drops appear in helpfulness (3.95), depth (3.51), and engagement (3.93). This modest degradation stems from the safety mechanism itself: when malicious queries are detected, TreeGuard responds with standardized refusal templates, which inherently limit the perceived helpfulness, depth, and interactivity of those responses. As a result, the utility reduction is targeted at harmful prompts rather than legitimate

use cases, showing that TreeGuard effectively balances robust safety alignment with the preservation of general response quality.

Table 8: Just-Eval utility scores on Vicuna-7B (1–5 scale). Higher scores represent better response quality.

Model	Defense	Helpful.	Clear	Factual	Deep	Engaging
Vicuna-7B	No Defense	4.43	4.82	4.60	3.95	4.18
	RDS	4.44	4.82	4.60	3.93	4.18
	SSD	4.20	4.72	4.44	3.76	4.02
	TreeGuard	3.95	4.78	4.56	3.51	3.93

A.4 Throughput Performance (Vicuna-13B)

As shown in Table 9, TreeGuard achieves a throughput of **88.85 tokens/s** on Vicuna-13B, significantly outperforming the autoregressive no-defense baseline (24.86 tokens/s), as well as other speculative decoding defenses including RDS (31.10 tokens/s) and SSD (19.70 tokens/s). Notably, TreeGuard retains nearly all the efficiency of the native EAGLE-3 framework (**91.45 tokens/s**), with only negligible overhead while adding safety guarantees. These results confirm that our method fully preserves the acceleration benefits of speculative decoding even on larger models, breaking the traditional trade-off between safety and efficiency.

Table 9: Throughput comparison on Vicuna-13B (tokens/s, higher is better).

Defense	vicuna-13B
Nodefense	24.86
RDS	31.10
SSD	19.70
TreeGuard	88.85
EAGLE-3	91.45

B Ablation analysis

To verify the robustness and generalization of TreeGuard, we conduct ablation experiments on Llama2-7b-chat and Vicuna-7b-v1.3 (EAGLE-2), and Vicuna-13b-v1.3 (EAGLE-3). Evaluations use four jailbreak benchmarks (AutoDAN, GCG, PAIR, DeepInception) and XSTest for false rejection rate. We study three key aspects: (1) decoding-stage generalization across forward passes; (2) stability against EAGLE-2 and EAGLE-3 hyperparameters (Top-k, depth); and (3) contributions of individual Δh fusion components. Results confirm TreeGuard's consistent defense performance, validating its robustness, compatibility, and structural rationality.

B.1 Robustness Across Draft Decoding Stages

Table 11 validates the robustness of TreeGuard across different draft decoding stages (10, 20, 30 forward passes). Across Vicuna-7B, Llama-7B, and Vicuna-13B, attack success rates remain near zero on all jailbreak benchmarks, while false rejection rates on XSTest stay highly stable. Only negligible fluctuations appear without obvious degradation as decoding stages increase, demonstrating that TreeGuard generalizes reliably throughout the draft model's generation process.

B.2 Robustness to EAGLE-2/EAGLE-3 Hyperparameters

Table 12 reports the defense performance of TreeGuard under various hyperparameter configurations of the EAGLE-2 and EAGLE-3 framework, including different Top-k sampling breadth and decoding depth. Across all settings on Vicuna-7B, Llama-7B, and Vicuna-13B, attack success rates remain extremely low on four jailbreak benchmarks, and false rejection rates on XSTest stay nearly unchanged. Only negligible fluctuations are observed without performance degradation, demonstrating that TreeGuard is highly robust to variations in EAGLE-2 and EAGLE-3 hyperparameters and maintains stable and reliable defense capability.

B.3 Ablation of Δ h Fusion Feature Components

Table 13 evaluates the contribution of each component in the Δh fusion feature. Compared to the All Components baseline, removing either normalization or the Hadamard product leads to clear performance degradation across all models. Specifically, the 'w/o Hadamard' variant suffers the most severe increase in attack success rates, especially on DeepInception and GCG benchmarks, while omitting normalization also results in higher ASR and stable FRR. These results confirm that both components are critical: normalization ensures feature consistency across different decoding stages, and the Hadamard product captures the nonlinear hazardous synergy between parent and child nodes, making their combination essential for TreeGuard's robust defense.

Table 10: Core Symbol Definitions

Notation	Physical Meaning
$h_{d-1,p_{d,k}}$	Decoding-layer hidden state of the unique parent node at layer $d-1$ corresponding to the k-th candidate node at layer d
$p_{d,k}$	Parent index mapping function, representing the serial number of the upper-layer parent node bound to the k-th child node at layer d
$h_{d,k}$	Decoding-layer hidden state of the k-th candidate node at tree generation layer d
u	Mean vector of hidden states across all training samples
V	Principal component matrix for PCA dimensionality reduction
$\odot$	Element-wise Hadamard product
$\|\cdot\|$	L2 norm

Table 11: Attack Success Rates on Vicuna-7B under Different Forward Pass Settings

Model	Draft Decoding Stages	AutoDAN	GCG	PAIR	DeepInception	XSTest
Vicuna-7B	10	1(0%)	1(0%)	1(0%)	1(0%)	9.40%
	20	1(0%)	1(0%)	1(0%)	1(0%)	9.40%
	30	1(0%)	1(0%)	1(0%)	1(0%)	9.40%
Llama-7B	10	1(0%)	1.06(2.2%)	1.73(24%)	1(0%)	32.8%
	20	1(0%)	1.08(2.2%)	1.73(24%)	1(0%)	33.0%
	30	1(0%)	1.15(4.4%)	1.73(24%)	1(0%)	32.4%
Vicuna-13B	10	1.4(2.2%)	1.25(7.5%)	1.2(2.5%)	1(0%)	8.50%
	20	1(0%)	1.25(7.5%)	1.2(2.5%)	1(0%)	8.30%
	30	1.4(2.2%)	1.25(7.5%)	1.4(5%)	1(0%)	8.20%

C Quantitative and ROC Analysis of SVM Classifier

To systematically evaluate the detection capability and generalization of the lightweight SVM classifier used in TreeGuard, we conduct comprehensive quantitative experiments and ROC curve analysis on two representative large language models, Vicuna-7B and Llama2-7B. We adopt four widely-used classification metrics: accuracy, precision, recall, and F1-score, to comprehensively assess the overall performance, false rejection risk, and malicious detection ability of the classifier.

As shown in Table 14, the SVM classifier achieves stable and competitive performance on Vicuna-7B, with balanced precision and recall for identifying malicious jailbreak prompts. When deployed on Llama2-7B, our classifier obtains near-perfect results: the precision reaches 1.0000, meaning no benign queries are misclassified as harmful, thus ensuring an extremely low false rejection rate in practical deployment. Meanwhile, the high recall demonstrates that nearly all adversarial jailbreak attempts can be effectively captured. These results verify that our Δh-based features can well capture adversarial semantic patterns, and the SVM classifier exhibits strong cross-model generalization ability across different large language models.

Figure 2 illustrates the ROC curves of TreeGuard's SVM classifier under the hyperparameter setting of $C = 1.0$ and 200-dimensional PCA-reduced Δh features. On Llama2-7B, the ROC curve completely coincides with the ideal classification boundary, achieving an AUC score of 1.0000, which means the classifier achieves perfect discrimination between benign queries and malicious jailbreak prompts without any false positives. For Vicuna-7B, the ROC curve rises rapidly under low false positive rates and obtains an AUC of 0.9960, showing excellent discriminative ability. Both results are far superior to the random guess baseline, further confirming that the proposed Δh features can effectively characterize adversarial semantic information for jailbreak detection.

D Visualization of Δh Fusion Features

To intuitively interpret the working mechanism of the proposed Δh fusion feature, we conduct fea-

Table 12: Attack Success Rates on Vicuna-7B under Different EAGLE-2 and EAGLE-3 Parameters

Model	Top-k	Depth	AutoDAN	GCG	PAIR	DeepInception	XSTest
Vicuna-7B	5	10	1(0%)	1(0%)	1(0%)	1(0%)	9.40%
	7	10	1(0%)	1(0%)	1(0%)	1(0%)	9.40%
	7	8	1(0%)	1.05(2.5%)	1(0%)	1(0%)	9.40%
Llama-7B	5	10	1(0%)	1(0%)	1.73(24%)	1(0%)	32.8%
	7	10	1(0%)	1.06(2.2%)	1.58(20%)	1(0%)	32.8%
	7	8	1(0%)	1.06(2.2%)	1.58(20%)	1(0%)	33.2%
Vicuna-13B	5	10	1.4(2.2%)	1.35(10%)	1.2(2.5%)	1(0%)	8.20%
	7	10	1.4(2.2%)	1.35(10%)	1.2(2.5%)	1(0%)	8.40%
	7	8	1.4(2.2%)	1.25(7.5%)	1.2(2.5%)	1(0%)	8.40%

Table 13: Ablation study on Δh fusion components. Lower ASR and higher rejection scores indicate better defense performance.

Model	Model Variant	AutoDAN	GCG	PAIR	DeepInception	XSTest
Vicuna-7B	All Components	1(0%)	1(0%)	1(0%)	1(0%)	9.20%
	w/o Normalization	1 (0%)	1.17 (5%)	1 (0%)	1 (0%)	9.20%
	w/o Hadamard	1 (0%)	1.6 (20%)	1 (0%)	1.08 (4.44%)	11.20%
Llama-7B	All Components	1 (0%)	1.06 (2.2%)	1.58 (20%)	1 (0%)	32.80%
	w/o Normalization	1 (0%)	1.06 (2.2%)	1.73 (22%)	1 (0%)	33.0%
	w/o Hadamard	1.08 (0%)	1.15 (6%)	1.91 (28%)	3.22 (66.6%)	34.80%
Vicuna-13B	All Components	1.4(2.2%)	1.35(10%)	1.2(2.5%)	1(0%)	8.40%
	w/o Normalization	1.48(4.4%)	1.35(10%)	1.4(5%)	1(0%)	8.40%
	w/o Hadamard	1.6(6.6%)	1.45 (15%)	1.4 (5%)	1 (0%)	15.80%

Table 14: Classification performance of TreeGuard's SVM classifier. Higher is better.

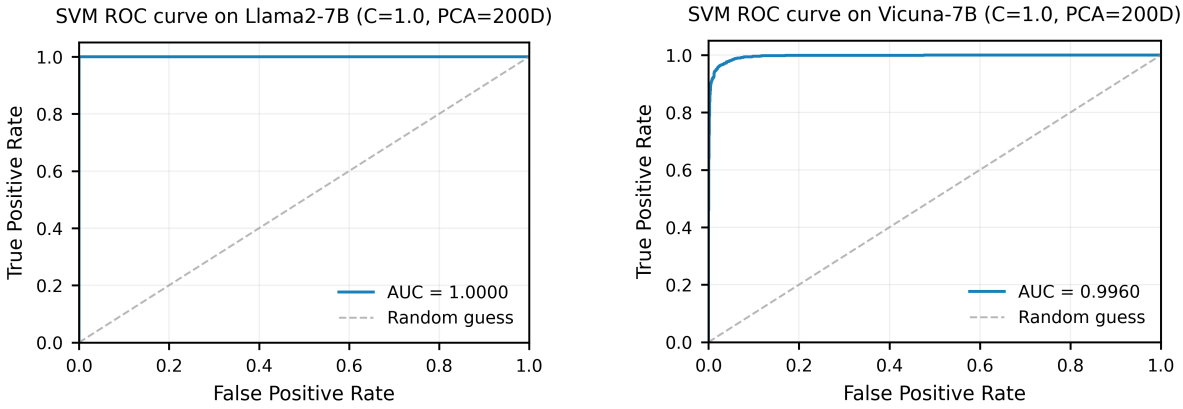
Model	Accuracy	Precision	Recall	F1-score
Vicuna-7B	0.9677	0.9608	0.9754	0.9680
Llama2-7B	0.9997	1.0000	0.9995	0.9997

ture visualization and discriminative dimension analysis. We first adopt t-SNE to project high-dimensional Δh features into a low-dimensional space to observe the separation degree between benign and harmful samples. Further, we introduce the KS score to quantitatively evaluate the discriminative capability of each dimension after PCA dimensionality reduction, so as to explain the partial feature overlap phenomenon in visualization.

Figure 3 and Figure 4 present the t-SNE visualization results of Δh fusion features on Vicuna-7B and Llama2-7B respectively. Overall, the Δh fusion features can effectively distinguish benign samples from harmful jailbreak samples. Nevertheless, a certain degree of partial overlap still exists between the two categories in the low-dimensional embedding space.

To further interpret the inherent reason for feature overlap, we leverage the KS score to quantify the discriminative power of each dimension among the 200 PCA-reduced feature dimensions.

As shown in the above KS score diagrams, the discriminative information for jailbreak detection is not concentrated in the first few PCA principal components, but widely distributed across all 200 dimensions. Although the top-2 dimensions achieve relatively high KS scores and strong discriminability, plenty of middle and later dimensions also retain considerable discriminative capability. Such scattered distribution of adversarial semantic features fundamentally causes the partial overlap in t-SNE low-dimensional visualization: the critical adversarial discriminative signals cannot be fully represented by only a small number of dominant dimensions.



(a) SVM ROC curve on Llama2-7B (AUC $= 1.0000$)

(b) SVM ROC curve on Vicuna-7B (AUC $= 0.9960$)

Figure 2: ROC curves of TreeGuard's SVM classifier with $C = 1.0$ and PCA=200D features.

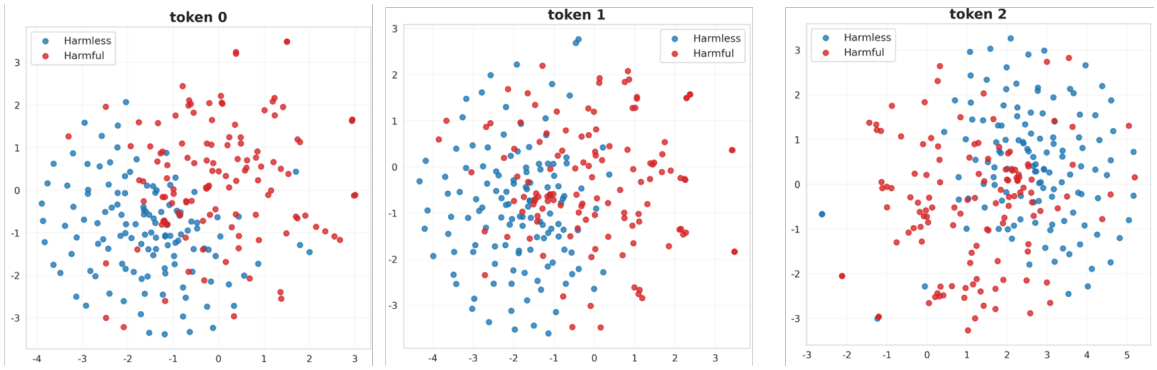


Figure 3: Visualization of Δh fusion features on Vicuna-7B. Blue represents benign samples, and red represents harmful samples.

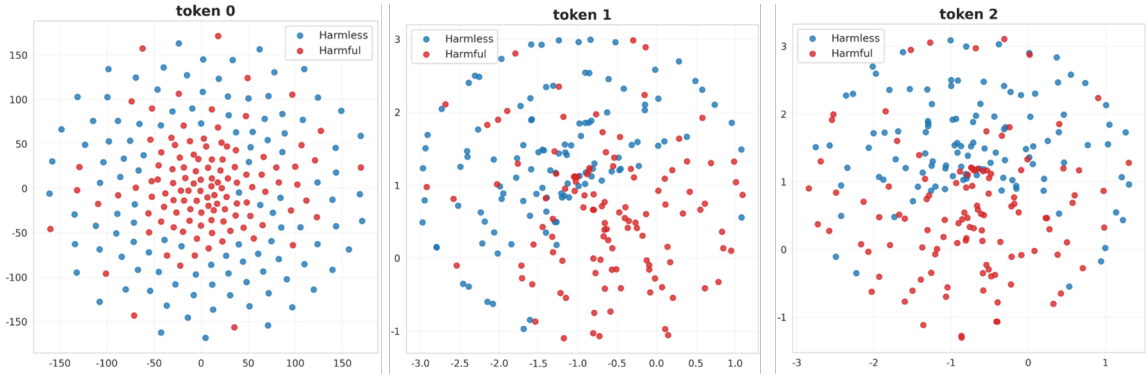


Figure 4: Visualization of Δh fusion features on Llama2-7B. Blue represents benign samples, and red represents harmful samples.

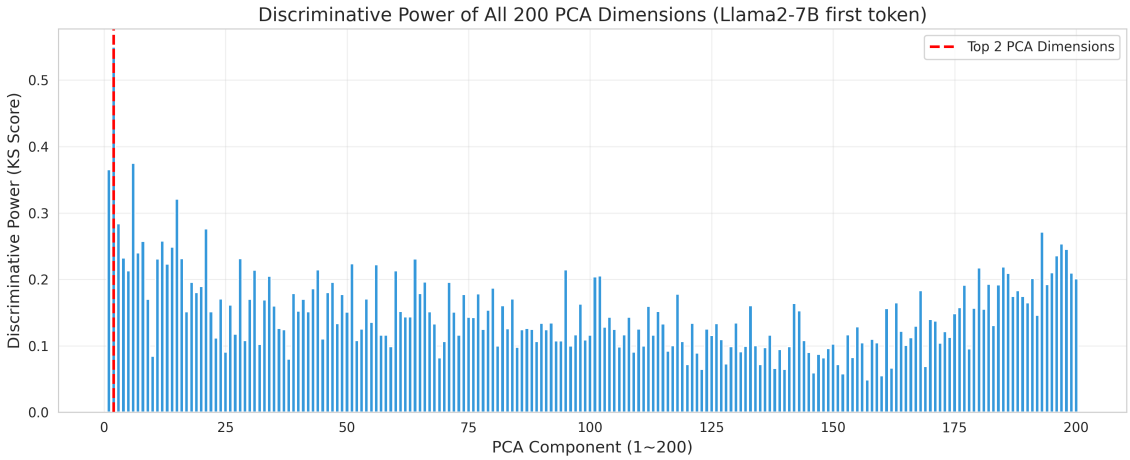


Figure 5: Discriminative power of all 200 PCA dimensions on Llama2-7B (first token). Red dashed lines mark the top-2 most discriminative dimensions.

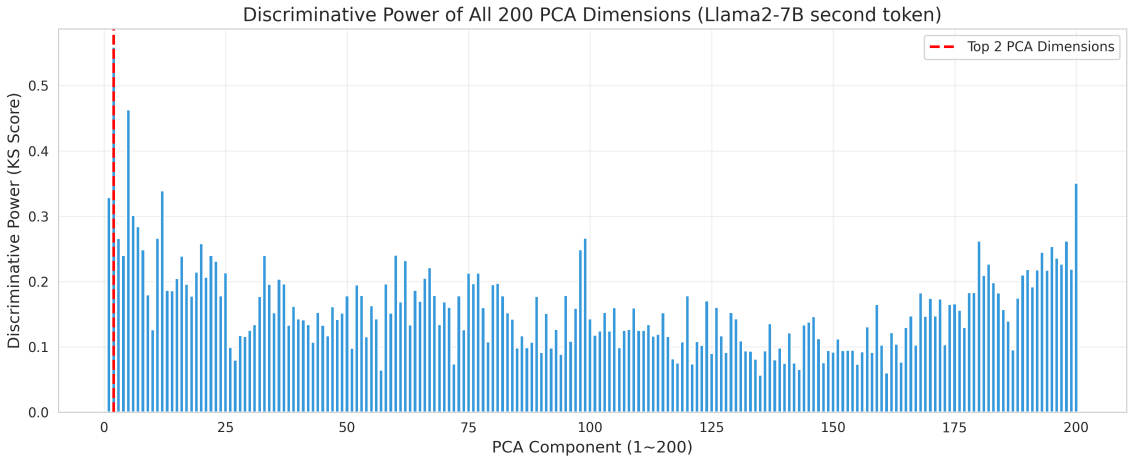


Figure 6: Discriminative power of all 200 PCA dimensions on Llama2-7B (second token). Red dashed lines mark the top-2 most discriminative dimensions.

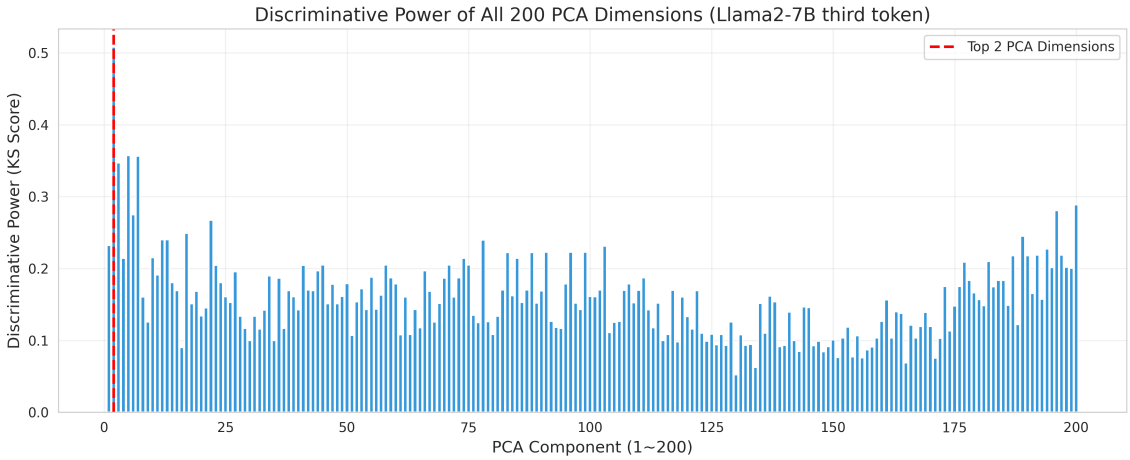


Figure 7: Discriminative power of all 200 PCA dimensions on Llama2-7B (third token). Red dashed lines mark the top-2 most discriminative dimensions.

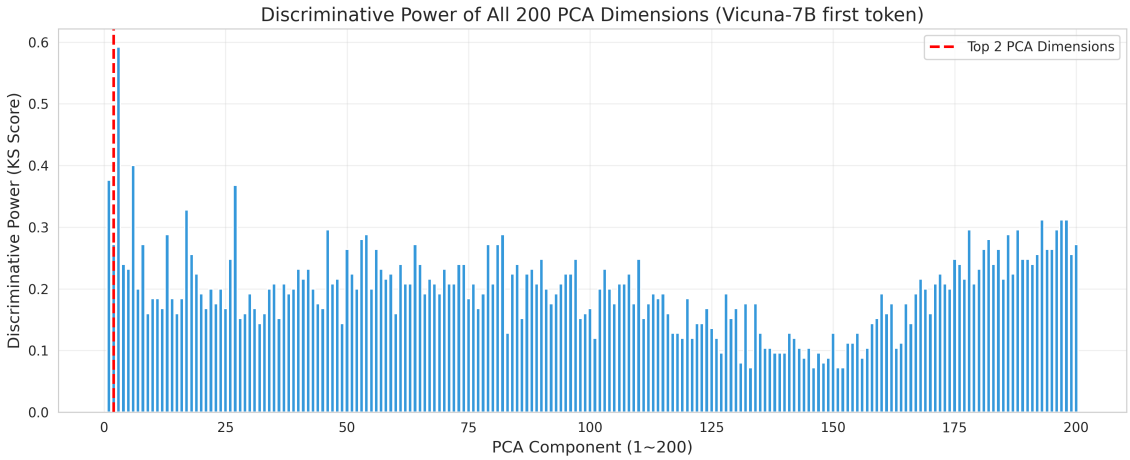


Figure 8: Discriminative power of all 200 PCA dimensions on Vicuna-7B (first token). Red dashed lines mark the top-2 most discriminative dimensions.

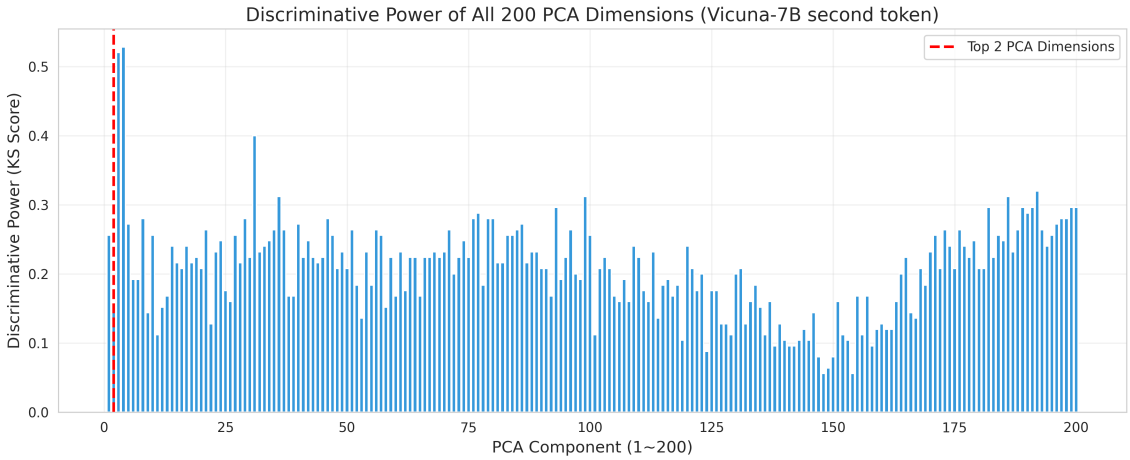


Figure 9: Discriminative power of all 200 PCA dimensions on Vicuna-7B (second token). Red dashed lines mark the top-2 most discriminative dimensions.

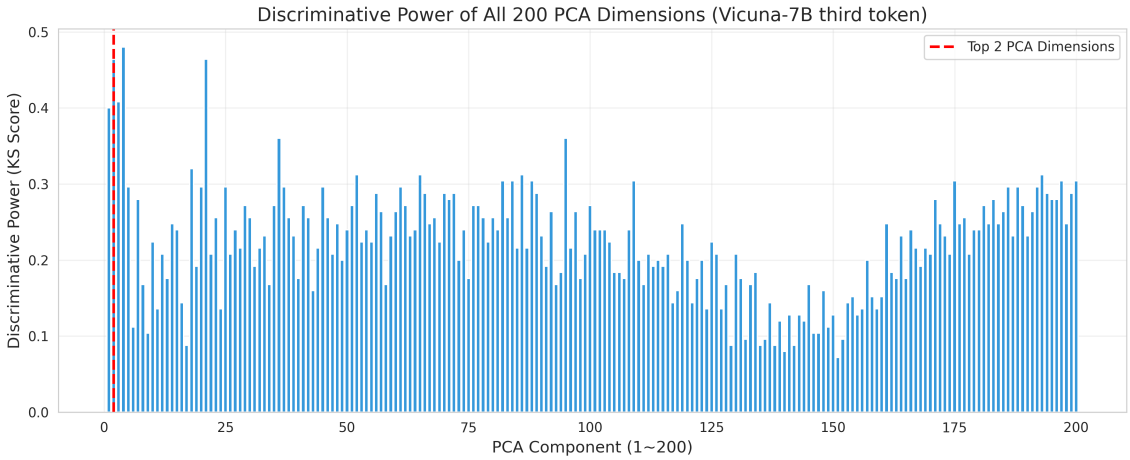


Figure 10: Discriminative power of all 200 PCA dimensions on Vicuna-7B (third token). Red dashed lines mark the top-2 most discriminative dimensions.