

Zen-Guard-Stream: Token-Level Streaming Safety Filtering for Real-Time Generative AI Systems

Technical Whitepaper v2025.05

Zen LM Research Team
research@zenlm.org
papers.zenlm.org

May 2025

Abstract

Zen-Guard-Stream is a 1.5 billion parameter safety filtering model designed for integration into the token generation loop of streaming AI systems, intercepting potentially harmful tokens during generation with 1.7ms mean overhead per generation step. Unlike post-hoc content filtering approaches that buffer complete outputs and apply safety checks before delivery, Zen-Guard-Stream operates token-by-token within the generation process, detecting unsafe trajectories before they manifest as complete harmful outputs. The model achieves 98.7% on ToxiGen and 0.8% false positive rate on benign creative content while adding only 1.7ms mean latency overhead to token generation. This paper describes the streaming safety architecture, integration protocol, training methodology, and evaluation results.

Contents

1	Introduction	3
1.1	Model Overview	3
2	Architecture	3
2.1	Streaming Safety Formulation	3
2.2	Lightweight Causal Classifier Architecture	4
2.3	Intervention Protocol	4
2.4	Context Window Management	4
3	Training Methodology	4
3.1	Training Objective	4
3.2	Streaming Safety Training Data	5
3.3	Adversarial Training	5
4	Evaluation	5
4.1	Classification Accuracy	5
4.2	False Positive Analysis	6
4.3	Streaming Overhead	6
4.4	Multi-Turn Jailbreak Detection	6
5	Integration Guide	7
5.1	Hook Interface	7
5.2	Deployment Topologies	7

6	Related Work	7
7	Conclusion	7

1 Introduction

The deployment of streaming generative AI creates a fundamental safety tension: streaming delivers tokens to users immediately as they are generated, enabling low-latency conversational experiences, but this means harmful content reaches users before a post-hoc safety filter can intervene.

Traditional mitigation approaches are inadequate for streaming contexts:

1. **Post-hoc filtering:** Buffer the complete output, apply safety filter, deliver (or withhold) the result. This works for batch generation but adds 100–500ms latency in streaming contexts, negating the latency advantage of streaming.
2. **Prompt-level filtering:** Classify the input prompt before generation begins. This misses unsafe content that emerges mid-generation in response to apparently benign prompts (indirect jailbreaks, gradual elicitation).
3. **Sampling-time constraints:** Apply logit masks to suppress individual tokens associated with harmful vocabulary. This misses multi-token harmful patterns and is trivially circumvented by paraphrasing.

Zen-Guard-Stream operates at the token generation level with access to the full generation context, enabling detection of harmful trajectories through multi-token patterns while maintaining streaming delivery of safe content.

1.1 Model Overview

Table 1: Zen-Guard-Stream Model Specification

Parameter	Value
Architecture	Causal streaming classifier
Total Parameters	1.5B
Integration Mode	Token generation hook
ToxiGen Accuracy	98.7%
False Positive Rate (creative content)	0.8%
Mean Streaming Overhead	1.7ms per generation step
P95 Streaming Overhead	3.1ms per generation step
Version	v2025.05
Release Date	May 2025

2 Architecture

2.1 Streaming Safety Formulation

Zen-Guard-Stream models streaming safety as a sequential decision problem. At each generation step t , given the prompt x and the generated sequence so far $y_{<t}$, the model predicts the safety of continuing generation along the current trajectory:

$$s_t = f_{\theta}(x, y_{<t}) \in \{0, 1\} \tag{1}$$

where $s_t = 1$ indicates a safe trajectory and $s_t = 0$ triggers an intervention. The model does not classify individual tokens in isolation; it classifies the generation *trajectory* up to and including position t , which enables detection of multi-token harmful patterns.

2.2 Lightweight Causal Classifier Architecture

Zen-Guard-Stream is implemented as a lightweight causal classifier that runs in parallel with the main generator:

- 1.5B parameters in a 24-layer causal transformer.
- Shared tokenizer with the main generator (100K vocabulary).
- Grouped-query attention (4 key-value heads) for memory efficiency.
- Safety classification head appended to the final layer: $\hat{s}_t = \sigma(w \cdot h_t^{(24)})$.
- Runs on a separate CUDA stream from the main generator, overlapping computation.

Because the classifier runs on a separate CUDA stream, it can execute concurrently with the main generator’s computations, keeping the marginal latency impact low.

2.3 Intervention Protocol

When the classifier triggers ($\hat{s}_t < \tau_{\text{safe}} = 0.15$), the streaming intervention protocol executes:

1. **Immediate halt:** The unsafe token is suppressed; streaming delivery pauses.
2. **Context assessment:** The main model’s logit distribution is inspected; if a safe alternative completion is available with $p > 0.3$, it is substituted and streaming resumes.
3. **Graceful degradation:** If no safe continuation is available, a stop message is inserted and the session ends gracefully with an explanation.
4. **Logging:** The full context at time of intervention is logged for safety audit review.

The substitution mechanism (step 2) is novel: rather than simply halting generation, Zen-Guard-Stream attempts to steer the generation onto a safe trajectory, improving user experience in borderline cases where the main model has accessible safe alternatives.

2.4 Context Window Management

Zen-Guard-Stream processes a sliding window of the last 512 tokens of generation context, sufficient to detect multi-turn elicitation patterns while keeping the classifier’s memory footprint bounded. A learned context compression module reduces older tokens to summary embeddings, maintaining awareness of long-range conversational patterns without unbounded memory growth:

$$h_{\text{summary}} = \text{Compress}_{\phi}(h_{t-1024:t-512}) \quad (2)$$

The summary embedding is prepended to the active window, giving the classifier access to compressed long-range context.

3 Training Methodology

3.1 Training Objective

Zen-Guard-Stream is trained with a streaming-aware objective that penalizes both false negatives (missing unsafe trajectories) and false positives (blocking safe creative content). The loss function is:

$$\mathcal{L} = w_{\text{FN}}\mathcal{L}_{\text{FN}} + w_{\text{FP}}\mathcal{L}_{\text{FP}} + \lambda\mathcal{L}_{\text{calibration}} \quad (3)$$

with $w_{\text{FN}} = 8.0$ and $w_{\text{FP}} = 1.0$, reflecting the much higher cost of missing genuinely unsafe content. The calibration loss $\mathcal{L}_{\text{calibration}}$ optimizes ECE to ensure reliable uncertainty estimates.

3.2 Streaming Safety Training Data

Training data is constructed as generation trajectories, not isolated content items, to match the streaming deployment context:

Table 2: Training Data Composition (trajectories)

Source	Trajectories (M)	Fraction
Safe creative writing	180	36.0%
Safe conversational AI	140	28.0%
Unsafe completions (annotated)	80	16.0%
Jailbreak trajectories	40	8.0%
Gradual elicitation attacks	30	6.0%
Borderline creative content	30	6.0%
Total	500	100%

3.3 Adversarial Training

Zen-Guard-Stream undergoes continuous adversarial training against an ensemble of red-team attack generators. The attack generator is trained to produce prompts and partial completions that maximize the classifier’s false negative rate. This adversarial dynamic creates a curriculum of increasingly sophisticated attacks that improve robustness:

- **Round 1:** Simple direct requests for harmful content.
- **Round 2:** Indirect elicitation through fictional framing.
- **Round 3:** Multi-turn gradual escalation.
- **Round 4:** Encoded and obfuscated content.
- **Round 5:** Compound attacks combining multiple techniques.

Each adversarial round produces augmentation data that is incorporated into the training set for the next model version.

4 Evaluation

4.1 Classification Accuracy

Table 3: Safety Classification Benchmark Results

Benchmark	Zen-Guard-Stream	Post-hoc 8B	Logit Mask	Prompt Filter
ToxiGen	98.7%	98.3%	72.1%	81.4%
HatEval	96.8%	97.4%	68.3%	77.2%
HarmBench-stream	94.1%	93.8%	61.4%	74.3%
JailbreakBench	91.3%	94.7%	43.2%	69.8%

Note: Post-hoc 8B achieves higher accuracy on JailbreakBench because it sees the complete output, while Zen-Guard-Stream must detect jailbreaks from partial trajectories. However, post-hoc filtering cannot prevent harmful token delivery in streaming contexts.

4.2 False Positive Analysis

False positives (blocking safe creative content) are a critical quality metric. We evaluate on three creative content benchmarks:

Table 4: False Positive Rates on Benign Creative Content

Content Type	Zen-Guard-Stream FPR	Logit Mask FPR
Literary fiction (violence themes)	1.4%	18.3%
Creative writing (mature themes)	1.2%	14.7%
Medical / clinical content	0.4%	8.2%
Security research discussion	1.1%	21.4%
Historical atrocity description	0.9%	16.8%
Average	0.8%	15.9%

Zen-Guard-Stream’s trajectory-level context understanding yields dramatically lower false positive rates than token-level logit masking, which cannot distinguish harmful context from benign usage of the same vocabulary.

4.3 Streaming Overhead

Table 5: Streaming Latency Overhead

Metric	Zen-Guard-Stream	Post-hoc 8B	Logit Mask
Mean overhead (ms/step)	1.7ms	N/A (not streaming)	0.1ms
P95 overhead (ms/step)	3.1ms	N/A	0.2ms
P99 overhead (ms/step)	4.8ms	N/A	0.4ms
End-to-end streaming latency impact	+4%	+200 – 500%	+0.2%

The 1.7ms mean overhead represents a 4% latency increase for a typical 40ms/token generation step, acceptable for most streaming applications. Post-hoc filtering is not meaningful as a streaming approach because it prevents delivery of any tokens until the full output is buffered.

4.4 Multi-Turn Jailbreak Detection

Multi-turn gradual elicitation attacks are particularly challenging for prompt-level filters. We evaluate on a benchmark of 500 adversarial conversations designed by red-teams, each spanning 5–15 turns:

Table 6: Multi-Turn Jailbreak Detection Rate

Attack Type	Zen-Guard-Stream	Prompt Filter	Human Review
Gradual escalation	89.4%	41.3%	94.2%
Persona hijacking	93.7%	58.4%	96.8%
Fictional framing	91.2%	52.1%	95.1%
Indirect elicitation	86.8%	38.7%	91.4%
Average	90.3%	47.6%	94.4%

Zen-Guard-Stream’s access to the full generation trajectory, including prior turns in the context window, provides substantially better detection of multi-turn attacks than prompt-only filters.

5 Integration Guide

5.1 Hook Interface

Zen-Guard-Stream is integrated via a pre-token hook in the generation loop:

Listing 1: Integration Example

```
from zen_guard import StreamGuard

guard = StreamGuard.load("zen-guard-stream-v2025.05")

def generation_hook(context: list[int], next_token: int) -> int:
    safety_score = guard.score(context, next_token)
    if safety_score < 0.15:
        # Unsafe trajectory detected
        safe_alt = guard.get_safe_alternative(context)
        if safe_alt is not None:
            return safe_alt # Substitute safe token
        else:
            return guard.STOP_TOKEN # Graceful halt
    return next_token # Pass through safe token
```

5.2 Deployment Topologies

Table 7: Deployment Configurations

Config	Hardware	Overhead	Throughput
Co-located (recommended)	Same GPU as generator	1.7ms	Full generator rate
Sidecar (separate GPU)	Dedicated A10G	2.4ms	Decoupled scaling
CPU sidecar	16-core CPU	8.1ms	Limited concurrency

6 Related Work

Streaming content safety has been addressed through output filtering [1], circuit breaker mechanisms [2], and classifier guidance [3]. Token-level safety has been studied through vocabulary constraints [4] and watermarking [5]. Zen-Guard-Stream’s trajectory-level classification within the generation loop is a novel approach that combines the timeliness advantages of token-level intervention with the accuracy advantages of context-aware safety modeling.

7 Conclusion

Zen-Guard-Stream provides token-level streaming safety with 98.7% ToxiGen accuracy, 0.8% false positive rate on benign creative content, and 1.7ms mean overhead per generation step, enabling safe streaming AI outputs without post-hoc filtering delays. The trajectory-level classification approach detects multi-token harmful patterns and multi-turn jailbreaks that are invisible to prompt-level filters and token-level logit masks. The substitution mechanism enables

graceful steering onto safe trajectories in borderline cases, improving user experience over abrupt generation halts. Zen-Guard-Stream is suitable for deployment in any streaming generative AI system where real-time safety and low latency are both required.

References

- [1] Lees, A. et al. (2022). A New Generation of Perspective API. arXiv:2202.11176.
- [2] Zou, A. et al. (2024). Improving Alignment and Robustness with Circuit Breakers. arXiv:2406.04313.
- [3] Dhariwal, P. & Nichol, A. (2021). Diffusion Models Beat GANs on Image Synthesis. NeurIPS 2021.
- [4] Krause, B. et al. (2021). GeDi: Generative Discriminator Guided Sequence Generation. EMNLP 2021.
- [5] Kirchenbauer, J. et al. (2023). A Watermark for Large Language Models. ICML 2023.