

Zen-Guard-Stream: A Streaming Safety Classifier Fine-Tuned from Qwen2.5-3B

Technical Whitepaper v2025.05

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

May 2025

Abstract

Zen-Guard-Stream is a streaming safety classifier built by fine-tuning Alibaba’s **Qwen2.5-3B** base model [1]. It is *not* a from-scratch model and uses no bespoke architecture: the base is Qwen/Qwen2.5-3B, a dense decoder-only transformer (Qwen2ForCausalLM; 3.09B parameters, 36 layers, hidden size 2048, GQA with 16 query / 2 key-value heads, vocab 151,936, 32K native context). The intended use is to run alongside a streaming generator and judge the safety of the generation *trajectory* so far, so that an unsafe continuation can be intercepted before a complete harmful output is delivered — something post-hoc filtering cannot do without buffering. **License caveat:** unlike most Qwen2.5 sizes, Qwen2.5-3B is released under the *Qwen Research License* (non-commercial use only), *not* Apache-2.0; any commercial deployment of a Qwen2.5-3B derivative requires a separate license from Alibaba. We report no safety benchmark numbers: the upstream base is a general LLM with no published safety metrics, and we have not run a rigorous streaming-safety evaluation; the figures (e.g. “ToxiGen 98.7%,” “1.7 ms overhead,” “0.8% FPR”) and the “1.5B-parameter” size in earlier revisions were fabricated and have been removed/corrected.

Contents

1	Introduction	2
1.1	Model Overview	2
2	Architecture	2
2.1	Streaming Safety Formulation	2
2.2	Causal Classifier Architecture	3
2.3	Intervention Protocol	3
2.4	Context Window	3
3	Training Methodology	4
3.1	Approach	4
4	Evaluation	4
4.1	Recommended Evaluation Before Deployment	4
5	Integration Guide	4
5.1	Hook Interface	4
5.2	Deployment Topologies	5

6	Related Work	5
7	Conclusion	5

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 specification. Base-model facts are those of the upstream Qwen2.5-3B [1]; Zen-Guard-Stream is a fine-tune of it for trajectory-level safety scoring.

Parameter	Value
Base model	Qwen2.5-3B (Alibaba)
License	Qwen Research License (non-commercial only) — <i>not</i> Apache-2.0
Architecture	Dense decoder-only transformer (Qwen2ForCausalLM)
Total Parameters	3.09B
Layers / hidden size	36 / 2048
Attention heads (Q / KV, GQA)	16 / 2
Vocabulary	151,936
Native context length	32,768 (32K)
Integration mode	Trajectory scoring alongside a streaming generator
Version	v2025.05

Safety benchmark accuracies, false-positive rates, and per-step overheads are deliberately omitted (see abstract); the values in earlier revisions were not measured.

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:

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

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 Causal Classifier Architecture

Zen-Guard-Stream is the Qwen2.5-3B causal transformer [1] with a safety classification head, run alongside the main generator:

- 3.09B parameters in a 36-layer causal transformer (hidden size 2048).
- Grouped-query attention (16 query / 2 key-value heads), vocabulary 151,936.
- A safety head on the final-layer hidden state: $\hat{s}_t = \sigma(w \cdot h_t^{(36)})$, i.e. a binary safe/unsafe probability for the trajectory through step t .
- Intended to run on a separate CUDA stream from the main generator so the two overlap.

Running the classifier concurrently with the main generator is a design choice to limit the marginal latency impact; we do not claim a specific per-step overhead, as it depends on the hardware, the generator size, and how much computation actually overlaps. The earlier revision’s “1.5B, 24-layer, 100K-vocabulary, 4-KV-head” description did not match the deployed base and has been corrected to the actual Qwen2.5-3B configuration.

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

The classifier sees the recent generation context (prompt plus generated tokens) within the base model’s 32K native context window [1], which is more than enough to capture multi-turn elicitation patterns in a typical conversation. The “learned context compression module” producing a $\text{Compress}_\phi(\cdot)$ summary embedding, described in earlier revisions, did not exist in the deployed model and has been removed; the model simply uses its native context window.

3 Training Methodology

3.1 Approach

Starting from the Qwen2.5-3B base [1], Zen-Guard-Stream is fine-tuned to predict a binary safe/unsafe label for the generation trajectory. The intended objective weights false negatives (missing unsafe trajectories) more heavily than false positives (blocking safe creative content),

$$\mathcal{L} = w_{\text{FN}}\mathcal{L}_{\text{FN}} + w_{\text{FP}}\mathcal{L}_{\text{FP}} + \lambda\mathcal{L}_{\text{calibration}}, \quad (2)$$

so that recall on genuinely harmful trajectories is prioritized. Training data is constructed as generation *trajectories* (safe and unsafe) rather than isolated content items, to match the streaming deployment setting.

We do not publish dataset sizes or composition: the specific figures in earlier revisions (a 500M-trajectory corpus with per-source percentages, and a five-round “continuous adversarial training” curriculum) were fabricated and did not describe a real training run. The honest statements are that the base is Alibaba’s Qwen2.5-3B and that Zen adds a safety fine-tune; we make no quantitative accuracy claim without a rigorous, reproducible evaluation.

4 Evaluation

We intentionally report no benchmark numbers. The upstream Qwen2.5-3B is a general-purpose LLM with no published safety metrics [1], and we have not run a rigorous, reproducible streaming-safety evaluation of the Zen fine-tune. The classification-accuracy, false-positive-rate, streaming-overhead, and multi-turn-jailbreak tables in earlier revisions (e.g. ToxiGen 98.7%, 0.8% FPR, 1.7ms/step overhead, 90.3% multi-turn detection) were fabricated — they did not come from any measured evaluation — and have been removed rather than replaced with invented numbers.

The defensible *qualitative* argument for this design remains: a trajectory-level classifier inside the generation loop can, in principle, intervene before a complete harmful output is delivered, which post-hoc filtering cannot do without buffering, and it can use multi-token context that a per-token logit mask cannot. Whether a given fine-tune actually achieves high recall at an acceptable false-positive rate and overhead is an empirical question that must be answered on the operator’s own workload (Section 4.1).

4.1 Recommended Evaluation Before Deployment

Before relying on Zen-Guard-Stream, operators should measure, on their own traffic: detection recall on unsafe trajectories (including multi-turn elicitation), false-positive rate on benign creative/medical/security content, and the actual per-step latency overhead on their hardware and generator. We provide the model and the integration hook; the validation is the operator’s responsibility.

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

Three topologies are possible; the right choice and its actual overhead depend on the operator’s hardware and generator, which is why we give no overhead numbers (the figures in earlier revisions were not measured):

Table 2: Deployment configurations (qualitative).

Config	Hardware	Trade-off
Co-located (recommended)	Same GPU as generator	Lowest added latency; shares GPU
Sidecar (separate GPU)	Dedicated accelerator	Decoupled scaling; network hop
CPU sidecar	CPU	Cheapest; highest latency, limited concurrency

6 Related Work

Streaming content safety has been addressed through output filtering [2], circuit breaker mechanisms [3], and classifier guidance [4]. Token-level safety has been studied through vocabulary constraints [5] and watermarking [6]. Zen-Guard-Stream applies trajectory-level classification within the generation loop, combining the timeliness of in-loop intervention with multi-token context; the model itself is a fine-tune of Qwen2.5-3B [1].

7 Conclusion

Zen-Guard-Stream is a streaming safety classifier built by fine-tuning Alibaba’s Qwen2.5-3B base model [1]; it is not a from-scratch model and is 3.09B parameters, not the “1.5B” of earlier revisions. Its premise is that a trajectory-level classifier inside the generation loop can intercept an unsafe continuation before a complete harmful output is delivered, which post-hoc filtering cannot do without buffering. We deliberately make no benchmark or latency claims: the upstream base has no published safety metrics, and we have not run a rigorous evaluation, so the inflated accuracy/FPR/overhead figures of earlier revisions have been removed as fabrications. **Adopters must also heed the license:** Qwen2.5-3B is under the non-commercial Qwen Research License, so commercial use of a derivative requires a separate license from Alibaba. Operators should validate the model on their own traffic before relying on it.

Attribution and License

The base weights, training, and capabilities are Alibaba’s Qwen2.5-3B (Qwen/Qwen2.5-3B); Zen contributes a safety fine-tune and packaging. Unlike most Qwen2.5 sizes, Qwen2.5-3B is licensed

under the *Qwen Research License* (non-commercial only), not Apache-2.0. We thank the Qwen team for releasing the base model.

References

- [1] Qwen Team, Alibaba (2024). Qwen2.5 Technical Report. arXiv:2412.15115. Base model: **Qwen/Qwen2.5-3B** (Qwen Research License, non-commercial).
- [2] Lees, A. et al. (2022). A New Generation of Perspective API. arXiv:2202.11176.
- [3] Zou, A. et al. (2024). Improving Alignment and Robustness with Circuit Breakers. arXiv:2406.04313.
- [4] Dhariwal, P. & Nichol, A. (2021). Diffusion Models Beat GANs on Image Synthesis. NeurIPS 2021.
- [5] Krause, B. et al. (2021). GeDi: Generative Discriminator Guided Sequence Generation. EMNLP 2021.
- [6] Kirchenbauer, J. et al. (2023). A Watermark for Large Language Models. ICML 2023.