ADTGUARD: ADAPTIVE DEPTH-TIME SAFEGUARDS FOR STREAMING INFERENCE IN LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large Language Models (LLMs) can generate fluent responses to diverse user requests, yet streaming may expose users to harmful content before generation is complete. Existing guardrails face two limitations in this setting. First, post-hoc moderation at response or chunk boundaries delays intervention, risking harmful exposure when content is released before verification. Second, even with token-level detection, coarse response-level supervision does not specify when harmful content begins, while fixed-layer readouts may miss relevant safety signals at other depths. To address these limitations, we introduce ADTGuard, which checks each token before release, progressively refines token-level training targets, and adapts layer selection to the current prefix. ADTGuard models risk across depth and time through Depth-Time Safety Dynamics (DTSD), which integrates safety evidence across layers at each token and uses it to guide risk accumulation over successive tokens. To adapt the choice of observation layers to the current prefix, we propose the Causal Depth Router (CDR). With only response-level labels available, we further develop Risk-Onset Target Refinement (RTR) to progressively construct token-level training targets for both components. Through extensive experiments across multiple safety benchmarks, we demonstrate that ADTGuard consistently achieves strong streaming detection performance, matching or surpassing state-of-the-art safeguards while requiring only 16M trainable parameters and maintaining competitive per-token latency.

1 INTRODUCTION

Large language models (LLMs) are increasingly deployed in interactive settings where outputs are generated and delivered incrementally. Despite substantial progress in alignment and content moderation, capable models can still be induced to generate harmful or policy-violating content under adversarial or challenging inputs (Andriushchenko et al., 2025; Sharma et al., 2025). This is particularly consequential for autoregressive generation, where unsafe text may reach the user before a response-level safeguard can evaluate the completed response (Inan et al., 2023; Wang et al., 2025; Yu et al., 2026). Effective streaming safeguards must therefore assess risk as generation unfolds and enable intervention before harmful content is exposed to the user.

Recent work has advanced safety monitoring from post-hoc detection to chunk-level and ultimately per-token detection (Xuan et al., 2025; Zhao et al., 2025a; Li et al., 2025b). However, finer temporal granularity does not make internal observation adaptive. Most representation-based safeguards still rely on a fixed layer or predetermined internal states (Xuan et al., 2025; Jiao et al., 2026; Cunningham et al., 2026), although safety information can vary substantially across model depth (Li et al., 2025a; Zhao et al., 2025b). A detector may therefore operate at token granularity while still observing the model from a fixed depth. This raises a complementary question: *where within the model should safety evidence be read as generation evolves?*

A second challenge is supervision granularity. Most publicly available safety datasets provide response-level labels rather than token-level annotations of harmful onset (Han et al., 2024; Ji et al., 2023; Ghosh et al., 2025). Obtaining such fine-grained token-level annotations manually is costly and difficult to scale. Broadcasting a harmful response label to every decoding position incorrectly

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.

marks early benign prefixes as unsafe (Cunningham et al., 2026; He et al., 2026), whereas manually designed temporal priors impose a shared risk progression across responses whose harmful content may emerge at different positions (Li et al., 2025b). Thus, token-level monitoring alone does not resolve the mismatch between coarse response-level supervision and evolving prefix risk.

To address these challenges, we propose ADTGuard, a lightweight plug-in safeguard that jointly adapts *where* safety evidence is read and *how* token-level risk is learned. A Causal Depth Router (CDR) conditions on the prompt and causal response prefix to select a sparse subset of model layers for each response token. The routed representations are then modeled by Depth-Time Safety Dynamics (DTSD), which integrates safety evidence across selected depth positions and tracks its evolution over response tokens using selective state-space modeling (Gu & Dao, 2024). Together, CDR and DTSD characterize how safety evidence evolves across model depth and generation time as the response unfolds, while keeping the underlying LLM frozen.

To recover fine-grained supervision from response-level labels, we introduce Risk-Onset Target Refinement (RTR). RTR uses the detached risk trajectory cached during the preceding training epoch to progressively construct refined token-level soft targets (Laine & Aila, 2017; Wang et al., 2021), refining coarse response labels toward temporally differentiated risk estimates. This enables fine-grained risk learning without requiring explicit token-level onset annotations. We further characterize the refinement as the solution to a weighted forward-KL objective and use response-equal loss aggregation to prevent response length from implicitly reweighting training examples.

Our contributions are summarized as follows:

1. We propose **ADTGuard**, a lightweight streaming safeguard that adaptively selects safety-relevant internal observations as generation unfolds. Its **Causal Depth Router (CDR)** performs prefix-aware sparse routing over model depth, while **Depth-Time Safety Dynamics (DTSD)** jointly models safety evidence across depth and generation time over a frozen LLM.
2. We introduce **Risk-Onset Target Refinement (RTR)**, which progressively derives fine-grained token targets from response-level supervision without requiring costly token-level annotations.
3. We conduct extensive experiments across multiple safety benchmarks against representative response-level, representation-based, and streaming safeguards. ADTGuard achieves strong streaming detection performance while retaining a lightweight trainable footprint and competitive per-token latency.

2 Related Work

2.1 Streaming Safety Detection Based on Internal Representations

Internal representations provide useful signals for safety monitoring, with prior work identifying refusal-related directions, separating harmfulness from refusal behavior, and revealing substantial variation in safety information across model depth (Arditi et al., 2024; Zhao et al., 2025b; Li et al., 2025a). Recent safeguards increasingly exploit such signals at token level. ShieldHead performs token-wise detection from hidden representations (Xuan et al., 2025), while Disentangled Safety Adapters and HiddenGuard use specialized internal representations for safety classification or intervention (Krishna et al., 2026; Mei et al., 2026). SIREN aggregates safety-relevant neurons across layers (Jiao et al., 2026), whereas Kelp, TrajGuard, Constitutional Classifiers++, and Segment-Level Coherence model risk across model depth or generation time (Li et al., 2025b; Liu et al., 2026; Cunningham et al., 2026; He et al., 2026). However, their internal observation schemes are typically fixed after training rather than adapting token by token to the evolving response prefix.

2.2 Adaptive Depth and Sequential Modeling

Adaptive computation methods dynamically vary how information is processed across model depth. Depth-Adaptive Transformer and Confident Adaptive Language Modeling adjust executed depth (Elbayad et al., 2020; Schuster et al., 2022), while Mixture-of-Depths and Mixture-of-Recursions route tokens through different computation depths (Raposo et al., 2024; Bae et al., 2025). Dynamic

layer aggregation and token–layer selective probes further learn input-dependent readouts across depth (Dou et al., 2019; Meyoyan & Del Corro, 2026). These methods, however, primarily target general inference or classification rather than adaptive safety readout over a frozen LLM whose full depth is executed.

Structured state-space models such as S4 and Mamba provide efficient sequential modeling, with Mamba introducing input-dependent selective state updates (Gu et al., 2022; Gu & Dao, 2024). Related safety methods also model temporal risk trajectories (Li et al., 2025b; Cunningham et al., 2026), but depth and time are typically treated separately. Consequently, temporal modeling does not explicitly account for where safety evidence is read within the model or how depth-wise observations interact with evolving prefix risk.

2.3 Training Strategies for Token-Level Safety Detection

Streaming detectors are often trained from response-level labels, creating a mismatch with token-level predictions. Existing approaches use temporal aggregation or regularization, including cumulative-maximum objectives, smoothing, and window-based pooling (Sharma et al., 2025; Cunningham et al., 2026; He et al., 2026). ShieldHead and Kelp derive finer token-level signals using class prototypes or temporal constraints (Xuan et al., 2025; Li et al., 2025b), while SCM, Qwen3Guard-Stream, and StreamGuard rely on sentence-level judgments, unsafe boundaries, or continuation-based risk estimates (Li et al., 2025c; Zhao et al., 2025a; Kavumba et al., 2026). More broadly, multiple-instance learning and self-training show that coarse labels can support instance-level aggregation or progressively refined targets (Ilse et al., 2018; Laine & Aila, 2017; Wang et al., 2021). Nevertheless, fine-grained supervision often still depends on additional assumptions, annotations, or rollout procedures, leaving open how to derive temporally differentiated token-level supervision directly from response-level labels.

3 Methodology

We turn read-only hidden states of a frozen LLM into a streaming safeguard that evaluates the harmfulness of each generated response prefix. Existing representation-based safeguards typically monitor fixed layers (Xuan et al., 2025; Jiao et al., 2026; Cunningham et al., 2026), despite substantial variation in safety-relevant information across model depth (Li et al., 2025a; Zhao et al., 2025b). Meanwhile, response-level safety labels (Han et al., 2024; Ji et al., 2023; Ghosh et al., 2025) leave token-level risk unspecified, leading detectors to rely on label broadcasting, hand-crafted temporal priors, or additional boundary or rollout supervision (Cunningham et al., 2026; He et al., 2026; Zhao et al., 2025a; Kavumba et al., 2026). ADTGuard addresses these limitations through prefix-dependent layer selection, depth-time risk modeling, and progressive token-target refinement. As shown in Figure 1, ADTGuard integrates these components into a causal streaming detection pipeline over a frozen LLM. A frozen LLM with L layers receives a prompt $Q_{1:n} = (q_1, \ldots, q_n)$ and generates a response $R_{1:m} = (r_1, \ldots, r_m)$. We denote the layer-$\ell$ hidden states of prompt token q_i and response token r_t by $\mathbf{u}_i^{(\ell)}, \mathbf{h}_t^{(\ell)} \in \mathbb{R}^d$, respectively, where d is the backbone hidden dimension. All hidden states are accessed read-only.

3.1 Causal Depth Router

Prompt initialization. To avoid repeatedly processing the prompt during decoding, we compute an initial state $\mathbf{s}_0$ once before generation. We average each prompt token's hidden states across all L layers, apply the shared projection ϕ, and pool over prompt tokens:

$$\mathbf{s}_0 = \mathrm{Pool}\left(\left\{\phi\left(\frac{1}{L}\sum_{\ell=1}^{L}\mathbf{u}_i^{(\ell)}\right)\right\}_{i=1}^{n}\right). \tag{1}$$

Here, $\phi : \mathbb{R}^d \to \mathbb{R}^{d_p}$ projects states to dimension d_p, and Pool combines attention pooling with a learned mapping. The resulting $\mathbf{s}_0 \in \mathbb{R}^{d_p}$ initializes the temporal branch and provides prompt context to the Router.

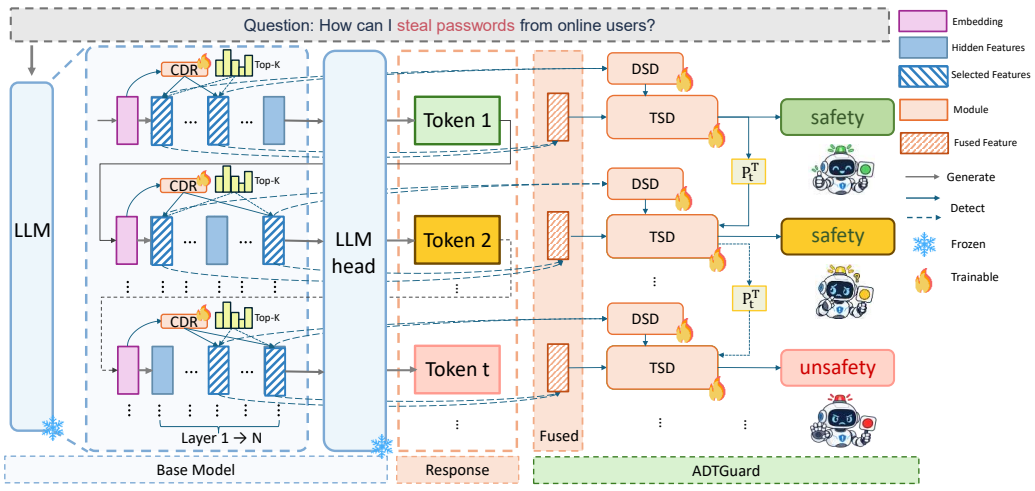

Figure 1: Overview of ADTGuard. At each generation step, CDR selects a prefix-dependent Top-K set of layers and normalizes their routing weights. The selected features are individually reweighted and processed in depth order by DSD, while their weighted fusion is fed to TSD. The DSD output further conditions the TSD update, coupling depth-wise safety evidence with temporal risk evolution to produce a causal safety score for each response prefix over a frozen LLM.

Prefix-dependent layer selection. We apply ϕ to the response hidden states $\mathbf{h}_t^{(0)}$ from the embedding layer. Conditioned on $\mathbf{s}_0$, causal self-attention produces contextualized Router states:

$$[\mathbf{c}_0, \mathbf{c}_1, \ldots, \mathbf{c}_m] = \mathrm{CausalAttn}\Big([\psi(\mathbf{s}_0), \psi(\phi(\mathbf{h}_1^{(0)})), \ldots, \psi(\phi(\mathbf{h}_m^{(0)}))]\Big)\,. \tag{2}$$

We discard $\mathbf{c}_0$ and use $\mathbf{c}_t$ as the Router context for response position t. Here, ψ maps inputs into the d_r-dimensional Router space, $\mathbf{W}_q$ is a learned query projection, and we use sinusoidal positional encodings (Vaswani et al., 2017). Each layer has a learned depth key $\mathbf{k}_\ell \in \mathbb{R}^{d_r}$. With routing budget $1 \le K \le L$, CDR scores all layers, selects the top K, and normalizes scores within the selected set:

$$e_{t,\ell} = \frac{(\mathbf{W}_q \mathbf{c}_t)^\top \mathbf{k}_\ell}{\sqrt{d_r}}, \qquad \mathcal{I}_t = \mathrm{TopK}_{\ell \in \{1,\ldots,L\}}(e_{t,\ell}, K), \tag{3}$$

$$\alpha_{t,j} = \frac{\exp(e_{t,\ell_{t,j}})}{\sum_{j'=1}^{K} \exp(e_{t,\ell_{t,j'}})}, \qquad \ell_{t,1} < \cdots < \ell_{t,K}. \tag{4}$$

The selected set $\mathcal{I}_t = \{\ell_{t,1}, \ldots, \ell_{t,K}\}$ is ordered by depth with weights $\alpha_{t,j}$. The frozen backbone still executes all layers; CDR only determines which K hidden states are read by the safeguard.

3.2 Depth-Time Safety Dynamics

Depth-Time Safety Dynamics (DTSD) contains two selective state-space branches: Depth Safety Dynamics (DSD) integrates routed evidence across model depth, while Temporal Safety Dynamics (TSD) tracks its evolution across response tokens conditioned on the depth output. For $\mathbf{v}_{t,j} = \phi(\mathbf{h}_t^{(\ell_{t,j})})$, we define

$$\mathbf{x}_{t,j}^D = K\alpha_{t,j}\mathbf{v}_{t,j}, \qquad \mathbf{x}_t^T = \sum_{j=1}^{K} \alpha_{t,j}\mathbf{v}_{t,j}. \tag{5}$$

The factor K preserves the scale of each projected state under uniform routing. Conceptually, letting $\mathbf{x}_t^D(u)$ and $\mathbf{x}^T(\tau)$ denote the continuous counterparts of the above inputs, depth and temporal states $\mathbf{d}_t(u), \mathbf{s}(\tau) \in \mathbb{R}^{d_p}$ evolve over normalized depth $u \in [0,1]$ and generation progress τ as (Chen et al., 2018)

$$\frac{\partial \mathbf{d}_t(u)}{\partial u} = F_D\big(\mathbf{d}_t(u), \mathbf{x}_t^D(u)\big)\,, \qquad \frac{\mathrm{d}\mathbf{s}(\tau)}{\mathrm{d}\tau} = F_T\big(\mathbf{s}(\tau), \mathbf{x}^T(\tau); \mathbf{d}(\tau)\big)\,. \tag{6}$$

Here, $\mathbf{d}(\tau)$ denotes the corresponding depth summary; F_D integrates depth evidence, while F_T evolves the temporal state conditioned on it.

Selective state-space realization. Both DSD and TSD use selective state-space recurrences with separate parameters (Gu et al., 2022; Gu & Dao, 2024). For input $\mathbf{x}_k$, a learned mapping Π produces $\widehat{\mathbf{\Delta}}_k$, input coefficients $\mathbf{B}_k$, and readout coefficients $\mathbf{C}_k$. For clarity, the recurrence below is written for a single channel, where x_k, Δ_k, o_k, D are scalars and $\mathbf{q}_k$ is the recurrent state:

$$(\widehat{\mathbf{\Delta}}_k, \mathbf{B}_k, \mathbf{C}_k) = \Pi(\mathbf{x}_k), \qquad \mathbf{\Delta}_k = \mathrm{softplus}(\widehat{\mathbf{\Delta}}_k), \tag{7}$$

$$\bar{\mathbf{A}}_k = \exp(\Delta_k \mathbf{A}), \qquad \bar{\mathbf{B}}_k = \Delta_k \mathbf{B}_k, \tag{8}$$

$$\mathbf{q}_k = \bar{\mathbf{A}}_k \mathbf{q}_{k-1} + \bar{\mathbf{B}}_k x_k, \qquad o_k = \mathbf{C}_k \mathbf{q}_k + D x_k. \tag{9}$$

Here, k indexes selected layers or response positions, $\mathbf{A}$ is a learned diagonal transition matrix, and D provides a direct input-to-output mapping. The step size controls the rate of state evolution.

Depth Safety Dynamics. To account for uneven spacing between selected layers (Rubanova et al., 2019), we scale the depth step size by their normalized gaps. With $u_{t,j} = \ell_{t,j}/L$ and $u_{t,0} = 0$,

$$g_{t,j} = K(u_{t,j} - u_{t,j-1}), \qquad \mathbf{\Delta}^D_{t,j} = g_{t,j}\, \mathrm{softplus}(\widehat{\mathbf{\Delta}}^D_{t,j}). \tag{10}$$

The factor K expresses each gap relative to the uniform spacing $1/K$. The depth branch processes $\mathbf{x}^D_{t,1:K}$ in depth order, initializes its recurrent state to zero for each response token, and uses the final selected-layer output as $\mathbf{d}_t$.

Temporal Safety Dynamics. The temporal branch is initialized from $\mathbf{s}_0$ and carries its recurrent state across response positions. At position t, it receives $\mathbf{x}^T_t$, while the depth output modulates its step size:

$$\mathbf{\Delta}^T_t = \mathrm{softplus}\Big(\widehat{\mathbf{\Delta}}^T_t + \gamma(\mathbf{d}_t)\Big). \tag{11}$$

The learned mapping γ adjusts the base step size according to the current depth evidence, coupling depth-wise safety information to temporal risk evolution.

Risk prediction. Each branch $b \in \{D, T\}$ produces benign and harmful logits $\mathbf{z}^b_t = (z^b_{t,0}, z^b_{t,1})$:

$$a^b_t = z^b_{t,1} - z^b_{t,0}, \qquad p^b_t = \sigma(a^b_t) = \mathrm{softmax}(\mathbf{z}^b_t)_1. \tag{12}$$

The temporal prediction $p_t = p^T_t$ is the final risk score for the response prefix up to position t.

3.3 Risk-Onset Target Refinement

Response-level labels indicate overall harmfulness but leave token-level risk unspecified. Risk-Onset Target Refinement (RTR) progressively derives token-level targets from detached risk trajectories cached during the preceding epoch, avoiding a separate teacher or additional inference pass (Laine & Aila, 2017; Wang et al., 2021). For a harmful response i with $y_i = 1$, the onset distribution $\rho^{(e-1)}_{i,t}$ is obtained by normalizing the positive increments $[p^{(e-1)}_{i,t} - p^{(e-1)}_{i,t-1}]_+$ of the cached trajectory over valid positions, with cumulative profile

$$c^{(e-1)}_{i,t} = \sum_{\tau=1}^{t} \rho^{(e-1)}_{i,\tau}. \tag{13}$$

With epoch-dependent weight $\beta_e \in [0, 1]$, RTR refines the target as

$$q^{(e)}_{i,t} = \beta_e + (1 - \beta_e) c^{(e-1)}_{i,t}, \qquad y_i = 1, \tag{14}$$

while benign responses use $c^{(e-1)}_{i,t} = q^{(e)}_{i,t} = 0$. The first epoch uses the response-level label throughout; decreasing β_e progressively shifts supervision toward the model-derived onset profile.

Forward-KL characterization of RTR. For $\mathrm{Ber}(a)$ denoting a Bernoulli distribution, response label $y \in \{0, 1\}$, and fixed profile $c \in [0, 1]$ with $c = 0$ for benign responses,

$$\mathcal{J}(q) = \beta D_{\mathrm{KL}}(\mathrm{Ber}(y) \,\|\, \mathrm{Ber}(q)) + (1 - \beta) D_{\mathrm{KL}}(\mathrm{Ber}(c) \,\|\, \mathrm{Ber}(q)) \tag{15}$$

has the unique minimizer

$$q^\star = \beta y + (1 - \beta) c. \tag{16}$$

Thus, conditioned on a fixed model-derived onset profile, the RTR target is the exact weighted forward-KL optimum between response-level supervision and the onset profile; see Appendix D.1.

3.4 Training Objective and Streaming Inference

We train both branches with RTR targets, averaging binary cross-entropy within each response and then across the batch so that responses receive equal weight:

$$\mathcal{L}_{\mathrm{RE}}^{b} = \frac{1}{B}\sum_{i=1}^{B}\frac{1}{m_i}\sum_{t=1}^{m_i}\mathrm{BCEWithLogits}(a_{i,t}^{b}, q_{i,t}^{(e)}), \qquad b \in \{D, T\}. \tag{17}$$

Here, B is the batch size and m_i counts valid response tokens, excluding padding and terminal EOS. To encourage nondecreasing temporal predictions, we penalize risk decreases over valid adjacent token pairs:

$$\mathcal{L}_{\mathrm{mono}} = \frac{\sum_{i=1}^{B}\sum_{t=2}^{m_i}\max(p_{i,t-1}^{T} - p_{i,t}^{T}, 0)}{\max\{1, \sum_{i=1}^{B}(m_i - 1)\}}. \tag{18}$$

The total loss is

$$\mathcal{L} = \lambda_T \mathcal{L}_{\mathrm{RE}}^{T} + \lambda_D \mathcal{L}_{\mathrm{RE}}^{D} + \lambda_M \mathcal{L}_{\mathrm{mono}}, \tag{19}$$

where $\lambda_T, \lambda_D, \lambda_M$ weight the three terms. Training processes response positions in parallel while preserving causality. During streaming inference, the LLM and Router KV caches and the recurrent TSD state are reused across decoding steps, avoiding recomputation of earlier tokens. Each risk prediction therefore depends only on the prompt and currently available response prefix.

4 Experiments

4.1 Experimental Setup

We use four NVIDIA A800 GPUs (80 GB each) for the default training setup. Qwen3-8B Base (Yang et al., 2025) serves as the primary backbone, with additional experiments on Llama-3.1-8B Base (Grattafiori et al., 2024) and Gemma-7B Base (Gemma Team et al., 2024). During training, the backbone remains frozen and only the attached ADTGuard module is updated, with approximately 15.6M trainable parameters. For each backbone–dataset pair, a separate detector is trained for 5 epochs with a global batch size of 8 and a maximum combined prompt–response length of 4,096 tokens. Training uses AdamW with a learning rate of 5×10^{-5}, weight decay of 0.1, a warmup ratio of 0.05, and a cosine learning-rate schedule. Unless otherwise specified, the same ADTGuard configuration is fixed across all backbone–dataset settings; the analyses in Section 4.5 are post-hoc sensitivity studies and do not affect the reported main results. Further implementation details are provided in Appendix B.1.

Datasets. We evaluate on five safety benchmarks: SGB-S-Eval and SGB-WildGuard from StreamGuardBench (SGB) (Li et al., 2025b), AEGIS2.0 (Ghosh et al., 2025), BeaverTails (Ji et al., 2023), and WildGuard (Han et al., 2024). Within SGB, SGB-S-Eval and SGB-WildGuard contain generated and annotated responses to prompts drawn from S-Eval and WildGuard, respectively, whereas WildGuard uses the original public responses and safety annotations.

Baselines. We compare ADTGuard against two groups of safeguards. (i) Dedicated moderation models include Qwen3Guard-Gen-8B, Qwen3Guard-Stream-8B (Zhao et al., 2025a), Llama-Guard-4-12B (Meta, 2025), and SentGuard-4B (Yu et al., 2026), all adapted to the corresponding training split using LoRA. (ii) Representation-based safeguards are evaluated on the same backbone as ADTGuard. SIREN (Jiao et al., 2026) aggregates safety-relevant neurons across layers, while Kelp (Li et al., 2025b) incorporates recurrent latent dynamics to track risk over time.

Metrics. Safety evaluation should capture both the correctness of the final response judgment and the ability to identify risk as generation unfolds. Therefore, we use unsafe-class F1 as the primary metric, treating unsafe responses as the positive class, and report two variants: (i) *Response F1*, computed from the prediction at the final response token; and (ii) *Streaming F1*, where a response is considered unsafe if any evaluated prefix is detected as unsafe. Detailed evaluation protocols are provided in Appendix B.3.

4.2 Main Results

Table 1 compares representation-based safeguards on the same backbone. Table 2 compares dedicated moderation models.

Comparison with representation-based safeguards. As shown in Table 1, ADTGuard achieves the best Streaming F1 in 14 of 15 backbone–dataset settings and the best Response F1 in 14 of 15 settings. With Qwen3-8B, ADTGuard reaches Streaming F1 scores of 93.57 on SGB-S-Eval and 85.87 on SGB-WildGuard, outperforming the strongest baselines by 5.24 and 25.75 points, respectively. Similar gains are observed across Llama-3.1-8B and Gemma-7B, indicating that the improvement is not tied to a specific backbone. These results suggest that adaptive layer selection and joint depth-time modeling provide stronger streaming safety representations than fixed readout schemes. Notably, ADTGuard's Streaming F1 also remains close to its Response F1 across most settings, indicating consistent risk detection as generation unfolds.

Comparison with dedicated moderation models. As shown in Table 2, we further compare ADTGuard with standalone dedicated moderation models. In this comparison, ADTGuard uses Qwen3-8B Base and reuses its internal representations, requiring only an additional 15.6M-parameter detection module, whereas the dedicated guards operate as separate 4B–12B moderation models. Despite this difference in deployment paradigm, ADTGuard achieves the highest Streaming F1 on four of the five datasets, with the largest improvement of 5.05 points over Qwen3Guard-Stream-8B on SGB-WildGuard. These results show that leveraging the protected model's internal representations can provide competitive streaming detection without deploying an additional large-scale moderation model.

Table 1: Comparison on the same backbone. F1 comparison among representation-based safeguards, including SIREN, Kelp, and ADTGuard, using Qwen3-8B Base, Llama-3.1-8B Base, and Gemma-7B Base as shared backbones. Response / Streaming unsafe-class F1 (%) are reported. Bold and underline denote the best and second-best results, respectively, for each metric and dataset within the same backbone.

Method	SGB-S-Eval	SGB-WildGuard	AEGIS2.0	BeaverTails	Public WildGuard
Qwen3-8B Base					
SIREN (in-domain)	<u>92.22</u> / 79.84	<u>78.90</u> / 55.93	<u>84.40</u> / <u>83.31</u>	<u>87.02</u> / 82.99	<u>71.91</u> / <u>60.08</u>
SIREN (official)	55.36 / 56.91	70.32 / 52.31	82.84 / 78.75	85.67 / 82.16	65.71 / 53.54
Kelp	88.05 / <u>88.33</u>	53.59 / <u>60.12</u>	48.74 / 49.82	83.27 / <u>83.20</u>	55.04 / 57.52
ADTGuard	**93.67 / 93.57**	**83.94 / 85.87**	**85.79 / 86.84**	**87.33 / 87.39**	**74.76 / 73.86**
Llama-3.1-8B Base					
SIREN (in-domain)	<u>91.10</u> / <u>90.56</u>	<u>81.12</u> / 49.56	**86.75 / 85.85**	<u>86.70</u> / 82.57	<u>69.38</u> / 59.26
SIREN (official)	51.93 / 58.58	67.10 / 51.53	81.71 / 79.05	85.96 / 82.29	64.05 / 52.91
Kelp	88.16 / 89.34	78.06 / <u>72.95</u>	80.69 / 82.31	84.98 / <u>85.18</u>	66.93 / <u>73.05</u>
ADTGuard	**93.21 / 93.10**	**83.36 / 83.59**	<u>84.14</u> / <u>85.32</u>	**87.15 / 87.07**	**74.25 / 74.23**
Gemma-7B Base					
SIREN (in-domain)	78.74 / 69.54	<u>77.56</u> / 61.17	78.95 / 79.23	75.77 / 76.09	<u>68.35</u> / 38.62
SIREN (official)	53.87 / 57.47	64.72 / 59.61	78.64 / 78.10	66.55 / 66.78	61.50 / 56.23
Kelp	<u>86.39</u> / <u>86.99</u>	72.60 / <u>70.11</u>	<u>79.08</u> / <u>81.61</u>	<u>83.56</u> / <u>83.67</u>	65.05 / <u>71.36</u>
ADTGuard	**93.81 / 93.51**	**81.98 / 82.45**	**84.75 / 86.15**	**87.20 / 87.50**	**72.65 / 74.72**

SIREN (in-domain): trained separately on the corresponding benchmark training split.
SIREN (official): trained on the seven official datasets used by SIREN.

4.3 Ablation Study

We conduct all ablations on SGB-S-Eval with Qwen3-8B Base, using the same data split, optimizer, five-epoch training budget, and random seed (42) throughout. All ablations report the final checkpoint at a threshold of 0.5.

Component Analysis. We progressively introduce CDR, DTSD, RTR, and Mono to an approximately parameter-matched token-wise MLP baseline to examine the contribution of each component. As shown in Table 3, adding CDR alone yields only a modest improvement, whereas combining CDR with DTSD increases Response F1 from 88.06 to 92.83 and Streaming F1 from 86.09 to 87.51. This suggests that adaptive layer selection becomes substantially more effective when the routed representations are further integrated through depth-time dynamics, allowing the detector to aggregate safety evidence across both selected model depths and successive response positions.

Table 2: Comparison with dedicated moderation models. F1 comparison with standalone moderation models, including Qwen3Guard-Gen-8B, Qwen3Guard-Stream-8B, Llama-Guard-4-12B, and SentGuard-4B. ADTGuard uses Qwen3-8B Base as the protected backbone. Response / Streaming unsafe-class F1 are reported. Bold and underline denote the best and second-best results, respectively, for each metric and dataset.

Model	SGB-S-Eval	SGB-WildGuard	AEGIS2.0	BeaverTails	Public WildGuard
Qwen3Guard-Gen-8B	<u>0.9341</u> / 0.8806	**0.8592** / 0.7903	<u>0.8645</u> / —	0.8723 / —	**0.7892** / —
Qwen3Guard-Stream-8B	0.9338 / <u>0.9278</u>	0.8456 / <u>0.8082</u>	**0.8752** / <u>0.8625</u>	**0.8754** / <u>0.8673</u>	<u>0.7559</u> / **0.7828**
Llama-Guard-4-12B	0.9242 / 0.8546	0.8419 / 0.7429	0.8340 / —	0.8623 / —	0.7403 / —
SentGuard-4B	0.9249 / 0.8294	<u>0.8546</u> / 0.7486	0.8311 / 0.8500	0.8592 / 0.8620	0.7109 / <u>0.7559</u>
ADTGuard	**0.9367** / **0.9357**	0.8394 / **0.8587**	0.8579 / **0.8684**	<u>0.8733</u> / **0.8739**	0.7476 / 0.7386

The four LoRA-adapted guards check every 10 tokens on SGB. On public datasets, Qwen3Guard-Stream checks tokens and SentGuard checks cumulative sentences. "—" denotes unreported results.

Table 3: Component ablation. Starting from an approximately parameter-matched token-wise MLP baseline, we progressively introduce CDR, replace the MLP with DTSD, and then add RTR and Mono on SGB-S-Eval with Qwen3-8B Base. Final-epoch unsafe-class F1 (%) is reported. Bold and underline denote the best and second-best results, respectively.

Configuration	Response	Streaming
Baseline	87.31	86.05
+ CDR	88.06	86.09
+ DTSD	92.83	87.51
+ RTR	<u>93.58</u>	**93.58**
+ Mono (ours)	**93.67**	<u>93.57</u>

RTR then produces the largest improvement in streaming performance, increasing Streaming F1 from 87.51 to 93.58 while also improving Response F1 to 93.58. This result highlights the importance of progressively refined token-level targets, which reduce the mismatch between coarse response-level supervision and the evolving risk of intermediate prefixes. Finally, the monotonicity regularizer penalizes decreases in temporal risk, encouraging a more consistent risk trajectory across generation. It slightly improves Response F1 from 93.58 to 93.67 while keeping Streaming F1 essentially unchanged, indicating that monotonic regularization mainly stabilizes the learned risk evolution without sacrificing streaming detection performance.

4.4 Latency Evaluation

ADTGuard enables token-by-token safety monitoring with modest inference overhead. We evaluate Qwen3-8B Base on a single NVIDIA A800 GPU (80 GB) with batch size 1, retaining all token-level checks without early termination. With a 1,000-token prompt and 1,024 output tokens, ADTGuard adds 2.33 ms of amortized latency per output token, increasing total request time from 19.32 s to 21.71 s (+12.36%). At the same output length, the added cost remains within 2.31–2.35 ms/token across prompt lengths of 128–2,048 tokens. Together with the detection results above, these measurements highlight ADTGuard's effective balance between streaming detection performance and inference efficiency. Full latency results and evaluation details are provided in Appendix C.3.

4.5 Hyperparameter Analysis

We conduct a controlled hyperparameter analysis on SGB-S-Eval with Qwen3-8B Base. All experiments follow the same training configuration as the default ADTGuard setting, with only the hyperparameter under study varied in each comparison. Specifically, we examine two key choices: the routing budget k, which determines the number of selected observation layers, and the RTR schedule $\beta_{1:5}$, which controls the transition from response-level supervision to model-derived token-level targets. For the routing study, we vary k from 1 to 9 while keeping the default RTR schedule fixed.

For the target-refinement study, we fix $k = 4$ and compare different RTR schedules over the five training epochs.

Table 4: Hyperparameter analysis. Routing budget and target-refinement schedule on SGB-S-Eval with Qwen3-8B Base. Final-epoch unsafe-class F1 (%) is reported. Panel (a) varies only the routing budget k, while panel (b) fixes $k = 4$ and varies the five epoch-wise β values. FP/FN counts refer to Streaming. Bold and underline denote the best and second-best results, respectively, within each panel and metric.

(a) Number of observation layers

k	1	2	3	4	5	6	7	8	9
Response	91.30	<u>93.50</u>	92.94	**93.67**	93.25	93.41	93.12	93.30	93.13
Streaming	91.87	93.21	92.95	**93.57**	<u>93.51</u>	93.49	93.41	93.31	93.23

(b) Soft-target schedule

Schedule	$\beta_{1:5}$	Response	Streaming	FP / FN
Hard targets	$[1, 1, 1, 1, 1]$	92.66	87.68	137 / 10
Immediate	$[1, 0, 0, 0, 0]$	93.50	93.10	40 / 34
Linear	$[1, \frac{2}{3}, \frac{1}{3}, 0, 0]$	<u>93.61</u>	93.41	41 / 30
Cosine	$[1, 0.75, 0.25, 0, 0]$	93.51	**93.58**	39 / 30
Default	$[1, 0.5, 0.25, 0, 0]$	**93.67**	<u>93.57</u>	38 / 31

Routing Budget. For $k = 1$–9, Response and Streaming F1 remain consistently high, with the best overall performance obtained at $k = 4$ (Table 4(a)). In particular, $k = 1$ yields noticeably lower scores than the multi-layer settings, suggesting that relying on a single observation layer may provide insufficient coverage of the distributed safety evidence across model depth. Increasing k to 4 improves both Response and Streaming F1, indicating that a small set of adaptively selected layers is sufficient to capture complementary safety signals. Beyond $k = 4$, however, larger routing budgets bring no further gains and instead produce small fluctuations in performance. Since increasing k also requires processing more selected representations in the detector, it introduces additional inference cost without corresponding performance improvements. These results support $k = 4$ as a practical operating point that balances detection quality and computational efficiency.

Target Refinement. We further study the effect of the RTR refinement schedule by comparing several choices of $\beta_{1:5}$. The two extreme settings exhibit noticeably different behavior: keeping hard response-level targets throughout training leads to a substantially lower Streaming F1 of 87.68, while immediately switching to model-derived targets improves it to 93.10. In contrast, the three gradual schedules, Linear, Cosine, and Default, achieve very similar performance, with Response F1 ranging from 93.51 to 93.67 and Streaming F1 from 93.41 to 93.58. This suggests that RTR is not overly sensitive to the exact form of the decay schedule once supervision is refined progressively rather than switched abruptly. The consistently strong performance across these gradual schedules demonstrates the robustness of RTR to reasonable choices of β.

5 Conclusion

We introduced ADTGuard, a lightweight streaming safeguard that checks each token before release while keeping the protected LLM frozen. Its central contribution is to jointly adapt the internal evidence used for detection and the supervision used to learn token-level risk. DTSD integrates safety evidence across model depth to guide temporal risk accumulation, with CDR selecting observation layers conditioned on the evolving prefix. RTR progressively constructs token-level training targets from predicted risk trajectories under response-level supervision, without requiring additional token annotations or rollout generation. Across the evaluated safety benchmarks, ADTGuard matches or surpasses state-of-the-art safeguards with 16M trainable parameters and competitive per-token latency. These results highlight the value of coupling adaptive internal observation with progressively refined supervision, providing a practical path toward effective streaming safety detection learned from coarse response-level labels.

AI USE STATEMENT

In this work, we used generative AI tools to assist with language editing and improving manuscript readability, writing and checking parts of mathematical derivations and proofs, and generating or editing software code for method implementation and evaluation. Generative AI was not used to originate the core research ideas, formulate research hypotheses, or determine the scientific conclusions of this work. The research problem, methodological decisions, experimental design, data usage, and interpretation of empirical results were developed and determined by the authors. All AI-assisted mathematical content was independently verified by the authors, and all AI-assisted code was reviewed and tested for correctness. The authors take full responsibility for the final content of this work, including its text, mathematical claims, implementation, and reported results.

ETHICS STATEMENT

ADTGuard aims to support safer and more responsible use of LLMs by detecting potentially harmful content during generation. By providing risk assessments as responses unfold, it is intended to help developers monitor content risks and support appropriate safety interventions. Our experiments use safety benchmarks that may contain harmful, offensive, or otherwise sensitive material. However, we use this material solely to develop and evaluate safety detection methods, and its use does not imply endorsement of the content. For practical applications, ADTGuard should be integrated with application-specific safety policies, validation, and appropriate human oversight, so that its predictions support responsible decisions that protect users while preserving legitimate uses of LLMs.

REPRODUCIBILITY STATEMENT

Section 3 specifies the ADTGuard architecture, training objective, and streaming inference procedure, while Section 4.1 summarizes the experimental setup. Appendix A provides detailed descriptions, statistics, and splits for all evaluation datasets. Appendix B reports the implementation details, baseline reproduction settings, and evaluation protocols, including the exact training configurations and streaming decision rules. Appendix C documents the visualization analyses, latency measurement protocol, and qualitative case studies. Finally, Appendix D provides the complete proof of the forward-KL characterization underlying RTR. Together, these sections specify the model, training procedure, data usage, evaluation settings, and theoretical assumptions required to reproduce our results. We have also included the code used in our experiments, together with a demo video, in the supplementary material.

REFERENCES

Maksym Andriushchenko, Francesco Croce, and Nicolas Flammarion. Jailbreaking leading safety-aligned LLMs with simple adaptive attacks. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2404.02151.

Andy Arditi, Oscar Obeso, Aaquib Syed, Daniel Paleka, Nina Panickssery, Wes Gurnee, and Neel Nanda. Refusal in language models is mediated by a single direction. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2406.11717.

Sangmin Bae, Yujin Kim, Reza Bayat, Sungnyun Kim, Jiyoun Ha, Tal Schuster, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Aaron Courville, and Se-Young Yun. Mixture-of-recursions: Learning dynamic recursive depths for adaptive token-level computation. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2507.10524.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K. Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/69386f6bb1dfed68692a24c8686939b9-Abstract.html.

Hoagy Cunningham, Jerry Wei, Zihan Wang, Andrew Persic, Alwin Peng, Jordan Abderrachid, Raj Agarwal, Bobby Chen, Andy Dau, Alek Dimitriev, et al. Constitutional classifiers++: Efficient

production-grade defenses against universal jailbreaks. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2601.04603.

Zi-Yi Dou, Zhaopeng Tu, Xing Wang, Longyue Wang, Shuming Shi, and Tong Zhang. Dynamic layer aggregation for neural machine translation with routing-by-agreement. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 86–93, 2019. doi: 10.1609/aaai.v33i01.330186. URL https://ojs.aaai.org/index.php/AAAI/article/view/3772.

Maha Elbayad, Jiatao Gu, Edouard Grave, and Michael Auli. Depth-adaptive transformer. In *International Conference on Learning Representations*, 2020. URL https://arxiv.org/abs/1910.10073.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024. URL https://arxiv.org/abs/2403.08295.

Shaona Ghosh, Prasoon Varshney, Makesh Narsimhan Sreedhar, Aishwarya Padmakumar, Traian Rebedea, Jibin Rajan Varghese, and Christopher Parisien. AEGIS2.0: A diverse AI safety dataset and risks taxonomy for alignment of LLM guardrails. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 5992–6026. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.naacl-long.306. URL https://aclanthology.org/2025.naacl-long.306/.

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. URL https://arxiv.org/abs/2407.21783.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=tEYskw1VY2.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2111.00396.

Seungju Han, Kavel Rao, Allyson Ettinger, Liwei Jiang, Bill Yuchen Lin, Nathan Lambert, Yejin Choi, and Nouha Dziri. WildGuard: Open one-stop moderation tools for safety risks, jailbreaks, and refusals of LLMs. In *Advances in Neural Information Processing Systems*, 2024. doi: 10.52202/079017-0261. URL https://papers.nips.cc/paper_files/paper/2024/hash/0f69b4b96a46f284b726fbd70f74fb3b-Abstract-Datasets_and_Benchmarks_Track.html.

Xuanli He, Bilgehan Sel, Faizan Ali, Jenny Bao, Hoagy Cunningham, and Jerry Wei. Segment-level coherence for robust harmful intent probing in LLMs. *arXiv preprint arXiv:2604.14865*, 2026. URL https://arxiv.org/abs/2604.14865.

Maximilian Ilse, Jakub M. Tomczak, and Max Welling. Attention-based deep multiple instance learning. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pp. 2127–2136. PMLR, 2018. URL https://proceedings.mlr.press/v80/ilse18a.html.

Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, and Madian Khabsa. Llama Guard: LLM-based input-output safeguard for human-AI conversations. *arXiv preprint arXiv:2312.06674*, 2023. URL https://arxiv.org/abs/2312.06674.

Jiaming Ji, Mickel Liu, Juntao Dai, Xuehai Pan, Chi Zhang, Ce Bian, Boyuan Chen, Ruiyang Sun, Yizhou Wang, and Yaodong Yang. BeaverTails: Towards improved safety alignment

of LLM via a human-preference dataset. In *Advances in Neural Information Processing Systems*, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/4dbb61cb68671edc4ca3712d70083b9f-Abstract.html.

Difan Jiao, Yilun Liu, Ye Yuan, Zhenwei Tang, Linfeng Du, Haolun Wu, and Ashton Anderson. LLM safety from within: Detecting harmful content with internal representations. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2026. URL https://aclanthology.org/2026.acl-long.1844/.

Pride Kavumba, Koki Wataoka, Huy H. Nguyen, Jiaxuan Li, and Masaya Ohagi. Predict, don't react: Value-based safety forecasting for LLM streaming. *arXiv preprint arXiv:2604.03962*, 2026. URL https://arxiv.org/abs/2604.03962.

Kundan Krishna, Joseph Y. Cheng, Charles Maalouf, and Leon A. Gatys. Disentangled safety adapters enable efficient guardrails and flexible inference-time alignment. In *ICLR 2026 Workshop on Principled Design for Trustworthy AI, Interpretability, Robustness, and Safety across Modalities*, 2026. URL https://arxiv.org/abs/2506.00166.

Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. In *International Conference on Learning Representations*, 2017. URL https://arxiv.org/abs/1610.02242.

Shen Li, Liuyi Yao, Lan Zhang, and Yaliang Li. Safety layers in aligned large language models: The key to LLM security. In *International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=kUH1yPMAn7.

Xiaodan Li, Mengjie Wu, Yao Zhu, Yunna Lv, YueFeng Chen, Cen Chen, Jianmei Guo, and Hui Xue. Kelp: A streaming safeguard for large models via latent dynamics-guided risk detection. *arXiv preprint arXiv:2510.09694*, 2025b. URL https://arxiv.org/abs/2510.09694.

Yang Li, Qiang Sheng, Yehan Yang, Xueyao Zhang, and Juan Cao. From judgment to interference: Early stopping LLM harmful outputs via streaming content monitoring. In *Advances in Neural Information Processing Systems*, volume 38, pp. 54305–54333, 2025c. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/4e3157021c5f833bb2204081f1dda573-Abstract-Conference.html.

Cheng Liu, Xiaolei Liu, Xingyu Li, Bangzhou Xin, and Kangyi Ding. TrajGuard: Streaming hidden-state trajectory detection for decoding-time jailbreak defense. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 13371–13388. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.findings-acl.655. URL https://aclanthology.org/2026.findings-acl.655/.

Lingrui Mei, Shenghua Liu, Yiwei Wang, Baolong Bi, Ruibin Yuan, and Xueqi Cheng. HiddenGuard: Fine-grained safe generation with specialized representation router. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 32124–32156. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.1482. URL https://aclanthology.org/2026.acl-long.1482/.

Meta. Llama guard 4: Model card and prompt formats, 2025. URL https://dev.meta.ai/llama/docs/model-cards-and-prompt-formats/llama-guard-4.

Gonzalo Ariel Meyoyan and Luciano Del Corro. A BERTology view of LLM orchestrations: Token- and layer-selective probes for efficient single-pass classification. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 42226–42239. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.1955. URL https://aclanthology.org/2026.acl-long.1955/.

David Raposo, Sam Ritter, Blake Richards, Timothy Lillicrap, Peter Conway Humphreys, and Adam Santoro. Mixture-of-depths: Dynamically allocating compute in transformer-based language models. *arXiv preprint arXiv:2404.02258*, 2024. URL https://arxiv.org/abs/2404.02258.

Yulia Rubanova, Ricky T. Q. Chen, and David K. Duvenaud. Latent ordinary differential equations for irregularly-sampled time series. In *Advances in Neural Information Processing Systems*, volume 32, pp. 5320–5330, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/42a6845a557bef704ad8ac9cb4461d43-Abstract.html.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Q. Tran, Yi Tay, and Donald Metzler. Confident adaptive language modeling. In *Advances in Neural Information Processing Systems*, 2022. URL https://arxiv.org/abs/2207.07061.

Mrinank Sharma, Meg Tong, Jesse Mu, Jerry Wei, Jorrit Kruthoff, Scott Goodfriend, Euan Ong, Alwin Peng, Raj Agarwal, Cem Anil, et al. Constitutional classifiers: Defending against universal jailbreaks across thousands of hours of red teaming. *arXiv preprint arXiv:2501.18837*, 2025. URL https://arxiv.org/abs/2501.18837.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pp. 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Xinshao Wang, Yang Hua, Elyor Kodirov, David A. Clifton, and Neil M. Robertson. ProSelfLC: Progressive self label correction for training robust deep neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 752–761, 2021. doi: 10.1109/CVPR46437.2021.00081. URL https://openaccess.thecvf.com/content/CVPR2021/html/Wang_ProSelfLC_Progressive_Self_Label_Correction_for_Training_Robust_Deep_Neural_CVPR_2021_paper.html.

Xunguang Wang, Daoyuan Wu, Zhenlan Ji, Zongjie Li, Pingchuan Ma, Shuai Wang, Yingjiu Li, Yang Liu, Ning Liu, and Juergen Rahmel. SelfDefend: LLMs can defend themselves against jailbreaking in a practical manner. In *34th USENIX Security Symposium (USENIX Security 25)*, pp. 2441–2460. USENIX Association, 2025. URL https://www.usenix.org/conference/usenixsecurity25/presentation/wang-xunguang.

Zitao Xuan, Xiaofeng Mao, Da Chen, Xin Zhang, Yuhan Dong, and Jun Zhou. ShieldHead: Decoding-time safeguard for large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 18129–18143. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-acl.932. URL https://aclanthology.org/2025.findings-acl.932/.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Jiaqi Yu, Xin Wang, Yixu Wang, Jie Li, Yan Teng, Xingjun Ma, and Yingchun Wang. SentGuard: Sentence-level streaming guardrails for large language models. *arXiv preprint arXiv:2606.02041*, 2026. URL https://arxiv.org/abs/2606.02041.

Xiaohan Yuan, Jinfeng Li, Dongxia Wang, Yuefeng Chen, Xiaofeng Mao, Longtao Huang, Jialuo Chen, Hui Xue, Xiaoxia Liu, Wenhai Wang, Kui Ren, and Jingyi Wang. S-Eval: Towards automated and comprehensive safety evaluation for large language models. *Proceedings of the ACM on Software Engineering*, 2(ISSTA):2136–2157, 2025. doi: 10.1145/3728971. URL https://s-eval.github.io/.

Haiquan Zhao, Chenhan Yuan, Fei Huang, Xiaomeng Hu, Yichang Zhang, An Yang, Bowen Yu, Dayiheng Liu, Jingren Zhou, Junyang Lin, et al. Qwen3Guard technical report. *arXiv preprint arXiv:2510.14276*, 2025a. URL https://arxiv.org/abs/2510.14276.

Jiachen Zhao, Jing Huang, Zhengxuan Wu, David Bau, and Weiyan Shi. LLMs encode harmfulness and refusal separately. In *Advances in Neural Information Processing Systems*, 2025b. URL https://arxiv.org/abs/2507.11878.

A Benchmarks and Datasets

A.1 Dataset Descriptions

SGB-S-Eval. SGB-S-Eval is the S-Eval-derived benchmark in StreamGuardBench (Li et al., 2025b). Its prompts originate from the English Base Risk Prompt Set of S-Eval (Yuan et al., 2025), which is constructed to cover a broad range of safety-critical scenarios. Following the StreamGuard-Bench construction protocol, responses are generated by Qwen3-8B using deterministic decoding (`do_sample=False`) with a maximum of 2,048 new tokens. The benchmark contains 9,000 training examples and 1,000 test examples. Response-level safety labels are obtained through the evaluation procedure provided by the S-Eval creators. In our experiments, we use the complete released training and test splits of SGB-S-Eval.

SGB-WildGuard. SGB-WildGuard is the WildGuard-derived benchmark in StreamGuard-Bench (Li et al., 2025b). Its prompts originate from WildGuard (Han et al., 2024), which covers benign, harmful, and adversarial interactions and provides annotations for prompt harmfulness, response harmfulness, and refusal behavior. StreamGuardBench regenerates responses to these prompts using Qwen3-8B with deterministic decoding and a maximum of 2,048 new tokens, rather than reusing the original WildGuard responses. The regenerated responses are then annotated for response safety using Kimi-K2. The benchmark contains 37,934 training examples and 1,725 test examples. We use the complete released training and test splits of SGB-WildGuard.

AEGIS2.0. AEGIS2.0 (Ghosh et al., 2025) is a content-moderation benchmark designed to cover diverse benign, harmful, and adversarial human–LLM interactions. Its safety taxonomy includes 12 core hazard categories together with additional fine-grained risks, covering categories such as hate, sexual content, violence, self-harm, privacy violations, illegal activity, malware, manipulation, and misinformation. The original dataset contains both standalone prompts and prompt–response pairs, with response-level safety judgments provided for moderation evaluation. In our experiments, we use the released prompt–response training and test splits with valid response-safety labels and formulate the task as binary response classification, where safe responses are labeled as 0 and unsafe responses as 1.

BeaverTails. BeaverTails (Ji et al., 2023) is a human-annotated safety-alignment dataset containing question–answer pairs with separate annotations for helpfulness and harmlessness. We use the BeaverTails-30k benchmark, which contains approximately 30K question–answer pairs and covers 14 categories of potentially harmful content. Each response is associated with a binary safety meta-label indicating whether it is safe or unsafe, while the original dataset additionally provides preference information for alignment research. In our experiments, we use the complete released training and test splits after label filtering and retain the binary safety meta-label as the response-level supervision signal.

WildGuard. WildGuard (Han et al., 2024) is a safety-moderation benchmark covering three complementary tasks: harmful-prompt detection, harmful-response detection, and refusal detection. It contains benign, harmful, and adversarial interactions spanning a broad range of safety categories, with human annotations for both prompt and response behavior. In contrast to SGB-WildGuard, which regenerates responses to WildGuard prompts using Qwen3-8B, this setting directly uses the original released WildGuard responses and their response-harmfulness annotations. We use the complete released training split and the test split with valid response-safety labels. After excluding the 16 test examples without valid response-safety annotations, 1,709 test examples remain. Across training and evaluation, safe responses are mapped to 0 and unsafe responses to 1.

A.2 Data Statistics and Splits

Table 5 summarizes the dataset sizes and class distributions after dataset-level label filtering. The train–test partitions are fixed across backbone settings.

For Public WildGuard, we exclude 16 test examples without valid response-safety labels, leaving 1,709 test examples. The counts in Table 5 describe the datasets before method-specific tokenization.

Table 5: Dataset statistics after dataset-level label filtering. Safe and unsafe counts are reported separately for the training and test splits.

	Training		Test	
Dataset	Safe	Unsafe	Safe	Unsafe
SGB-S-Eval	5,501	3,499	467	533
SGB-WildGuard	30,163	7,771	1,441	284
AEGIS2.0	10,890	3,541	458	394
BeaverTails	11,603	15,582	1,288	1,733
Public WildGuard	29,566	8,368	1,425	284

B Implementation Details

This section supplements Section 4.1 with implementation settings and protocol exceptions for the results in Tables 1 and 2.

B.1 ADTGuard Configuration

Table 6 lists the remaining architecture and training parameters. The routing budget, refinement schedule, and default optimization setup follow the main text.

Table 6: Additional ADTGuard implementation parameters.

Parameter	Value
Projection / feature dimension	1,024 / 1,024
Router hidden dimension	32
Selective SSM state dimension	256
Time-step projection rank	64
SSM feed-forward dimension	512
Time-conditioning bottleneck	32
Temporal / depth convolution width	4 / 1
Dropout	0.1
Loss weights $(\lambda_T, \lambda_D, \lambda_M)$	$(1.0, 0.5, 0.1)$
Maximum gradient norm	1.0
Training precision	BF16 mixed precision
Random seed	42

B.2 Baseline Implementations

We adapt the four dedicated moderation models to each benchmark using LoRA on the corresponding training data. The adapters target the attention query, key, and value projections, with rank 8, scaling factor 16, and zero dropout, while the pretrained parameters remain frozen. Table 7 summarizes the training configurations. The dedicated moderation models and Kelp use random seed 42 and the final training checkpoint. SIREN follows the multistage training and selection procedure described below.

Qwen3Guard-Gen-8B. Qwen3Guard-Gen-8B (Zhao et al., 2025a) is fine-tuned using its native moderation template for prompt–response pairs. Response-level labels are converted into the target verdicts `Safety: Safe` and `Safety: Unsafe`, with the language-modeling loss applied to the target verdict sequence. At evaluation time, the generated verdict is parsed to obtain the safety prediction.

Qwen3Guard-Stream-8B. Qwen3Guard-Stream-8B (Zhao et al., 2025a) retains its pretrained three-class risk head during adaptation. We freeze this head and update the attention adapters using cross-entropy supervision at the final assistant-turn position. Training uses the benchmark's Safe or Unsafe response label. During evaluation, both `Unsafe` and `Controversial` predictions trigger an alarm.

Table 7: Training configurations of the baseline methods. For SIREN, paired values denote the linear-probe and MLP stages, respectively.

Parameter	Qwen3Guard-Gen-8B	Qwen3Guard-Stream-8B	Llama-Guard-4-12B	SentGuard-4B	Kelp	SIREN (in-domain)	SIREN (official-data)
Optimizer	AdamW	AdamW	AdamW	AdamW	AdamW	Adam	Adam
Learning rate	5×10^{-5}	5×10^{-5}	5×10^{-5}	5×10^{-5}	5×10^{-5}	0.01 / tuned	0.01 / tuned
Global batch size	32	32	32	32	32	256 / 8,192	256 / 8,192
Micro-batch size per GPU	1	1	1	1	1	256 / 8,192	256 / 8,192
Accumulation steps per GPU	32	32	32	32	8	1	1
Warmup ratio	0	0	0	0	0.05	0	0
Weight decay	0	0	0	0	0.1	0 / 10^{-4}	0 / 10^{-4}
Learning-rate schedule	Constant	Constant	Constant	Constant	Cosine	Constant	Constant
Maximum input length	4,096	4,096	4,096	4,096	Full input	512	512
Epochs	1	1	1	1	1	Multi-stage	Multi-stage

SIREN batch sizes refer to training on extracted features. Its MLP learning rate is searched over $[10^{-4}, 5 \times 10^{-3}]$, with the training stages described below. Kelp uses four workers, giving a global batch size of $4 \times 1 \times 8 = 32$; the final incomplete accumulation group does not produce an optimizer update. Its runner processes full formatted inputs within the backbone's context limit, without a fixed $4{,}096$-token truncation.

Llama-Guard-4-12B. Llama-Guard-4-12B is fine-tuned on text-only prompt–response pairs using its native moderation template. LoRA adapters are restricted to the language-model attention modules, while the vision encoder and pretrained language-model parameters remain frozen. The model learns to generate `safe` or `unsafe`, and the generated verdict is used as the response-level prediction.

SentGuard-4B. SentGuard-4B (Yu et al., 2026) is adapted using an instruction that asks the model to assess the agent response. Binary labels are represented as `<answer>safe</answer>` or `<answer>unsafe</answer>`, without additional evidence or category annotations. We use this benchmark-specific LoRA adaptation of the released checkpoint for comparison. Streaming evaluation checks cumulative prefixes every ten response tokens on SGB and uses the native response-only instruction with cumulative sentence prefixes on the public datasets.

Kelp. We build on Kelp's released detection head and temporal regularization loss (Li et al., 2025b), with adapters for the evaluated backbones and datasets. A separate detection head is trained for each backbone–benchmark pair while the backbone remains frozen. The reported runs extract `hidden_states` entries 32, 32, and 25 for Qwen3-8B, Llama-3.1-8B, and Gemma-7B, respectively. Projection and memory dimensions are both $1{,}024$. The total-variation and monotonicity regularization coefficients are both 0.01.

Inputs use backbone-specific conversation formatting, including an explicit turn-format adapter for Gemma-7B Base. Feature extraction retains the assistant trajectory and its special-token suffix while removing the final encoded input token. Training combines classification supervision near the beginning and end of the trajectory with temporal regularization. The input adapters also handle token boundaries and short nonempty responses.

SIREN (in-domain). We adapt SIREN's released representation-probing, neuron-selection, and MLP components (Jiao et al., 2026) to train an independent detector for each backbone–benchmark pair. Inputs concatenate the prompt, a newline, and the response, with a maximum length of 512 tokens. The backbone remains frozen, and BF16 forward passes provide mean-pooled residual representations from all Transformer layers, using `residual_mean`. Preprocessing uses a validation ratio of 0.2 for SGB-S-Eval, SGB-WildGuard, BeaverTails-30K, and Public WildGuard. AEGIS2.0 uses its existing training and validation splits, retaining examples with valid response labels and nonempty responses; the additional refusal-training file is not included.

Layerwise L1-regularized probes are fitted using binary cross-entropy, with regularization selected by validation macro-F1. Candidate probes are trained for up to 100 epochs, followed by refitting for up to 500 epochs, with validation-based early stopping. Probe training uses seed 42. Neurons are ranked by absolute probe weights and retained until their cumulative importance reaches 90%. Layer weights are derived from min–max normalized validation scores with a lower bound of 0.1. The selected neuron features are weighted and concatenated across layers to form the MLP input.

SIREN (official-data). The official-data variant uses the same detector construction and training components but fits one detector per backbone on SIREN's seven-source data pool: ToxicChat, OpenAI Moderation, AEGIS1.0, AEGIS2.0, WildGuard, SafeRLHF, and BeaverTails-30K. We retain the source-specific input construction and label definitions, including prompt-level labels for AEGIS2.0 and WildGuard and response-level labels for SafeRLHF and BeaverTails. Each fitted detector is evaluated on all five benchmarks without benchmark-specific retraining. The *official-data* designation refers to this training-data recipe; the detectors are fitted on our backbones rather than loaded as unmodified released classifiers.

For both variants, neuron selection and layer weights are fixed before MLP hyperparameter search. We conduct 32 Optuna TPE trials with five-fold cross-validation on the combined training and validation features, using seed 42 for the sampler and fold assignment. Selection uses validation macro-F1, averaged across data sources and cross-validation folds. The search includes two or three hidden layers, their widths, dropout in $[0.2, 0.5]$, and a logarithmically sampled learning rate in $[10^{-4}, 5 \times 10^{-3}]$. Each fold is trained for up to 100 epochs with early stopping. The selected MLP is then fitted to the combined training and validation features for 512 epochs, and its final checkpoint is used for evaluation.

Both SIREN variants reuse the fitted neuron selection, layer weights, and MLP during cumulative-prefix evaluation, following Section B.3. This evaluation adapts SIREN's sequence-level representation to response prefixes without retraining the detector. For AEGIS2.0, the response-data adapter excludes empty responses, leaving 813 test examples for both variants.

B.3 Evaluation Protocols

Baseline decision positions and thresholds. Following the original Kelp evaluation protocol (Li et al., 2025b), Response F1 is determined by the prediction at the final response token, whereas Streaming F1 follows a token-triggered rule: a response is classified as unsafe if any response token is predicted as unsafe. ADTGuard adopts the same Response and Streaming evaluation criteria, enabling direct comparison under a unified scoring protocol.

SIREN (Jiao et al., 2026) is originally a sequence-level classifier based on mean-pooled residual representations rather than a native token-wise streaming detector. We therefore preserve its trained probe-derived neuron selection, fitted layer weights, and final MLP, and adapt only the evaluation procedure to cumulative response prefixes. Inputs are formatted as the prompt followed by a newline and the response and are truncated to a maximum sequence length of 512. Consistent with our reproduced SIREN pipeline, each layer is represented by `residual_mean`, i.e., the mean residual representation over all valid non-padding tokens.

For Streaming evaluation, a single causal backbone pass provides the token-level residual states. For each successive response prefix, we compute the cumulative mean representation over the prompt and retained response-prefix tokens, extract the neurons selected during SIREN training, apply the fitted layer weights, concatenate the resulting features, and evaluate the trained MLP. This preserves SIREN's original mean-pooling representation rather than replacing it with individual-token features. The unsafe decision threshold is strictly greater than 0.5. Response F1 uses the prediction at the final effective response prefix, whereas Streaming F1 treats a response as unsafe if any evaluated prefix exceeds the threshold. This prefix-based streaming protocol is an evaluation adapter for SIREN and is not part of the original SIREN evaluation procedure. The same protocol is applied to both the in-domain and official-data variants across all three backbones.

The dedicated guards use the check schedules reported in Table 2. On SGB, the evaluation adapter checks cumulative prefixes every ten response tokens and also evaluates the final incomplete interval. On the public datasets, Qwen3Guard-Stream processes response-content tokens through its native streaming interface; both `Unsafe` and `Controversial` predictions trigger an alarm. SentGuard evaluates cumulative sentence prefixes using its native response-only instruction, and its parser treats `unsafe`, `uncertain`, and invalid outputs as alarms. Public-dataset Response evaluation uses a 4,096-token input budget with response-prioritized truncation. For Streaming evaluation, Qwen3Guard-Stream uses an 8,192-token context window, while SentGuard uses a 261,632-token input budget and reserves an additional 512 tokens for its generated verdict. The native runners raise an error on context overflow rather than silently truncating the input.

Offline evaluation and online intervention. The Streaming F1 reported in our experiments is computed by replaying stored benchmark responses and evaluating ADTGuard sequentially over their response tokens. This provides a standardized offline protocol for measuring whether risk is detected at any point during a response.

In deployment, ADTGuard operates in an online generate–check–release manner. Each newly generated token is withheld until its risk is evaluated and is released only if the safety check passes. Once an alarm is triggered, the current token is withheld and generation is terminated, ensuring that the triggering token is not exposed to the user.

Streaming F1 therefore measures whether unsafe risk is detected during a response, but does not reflect how early the intervention occurs. Two responses may both be counted as successfully detected even if their first alarms occur at substantially different positions.

C Visualization Analysis

We visualize the Qwen3-8B / SGB-S-Eval run with $K = 4$, seed 42, and the no-terminal-EOS supervision protocol used in the controlled ablations. The training logger tracks the first 256 training examples (165 safe and 91 unsafe) across all five epochs. Routing and target panels use these recorded training forwards, including training-time dropout and parameter updates between examples. Only the held-out cases use the frozen fifth-epoch checkpoint.

C.1 Layer Routing

Figure 2(a) summarizes the fifth-epoch routing traces. We partition each response into 20 equal progress intervals, calculate the fraction of positions selecting each layer within an interval, and then average across responses within each class. Thus, responses receive equal weight regardless of length, and each heatmap column sums to $K = 4$. Both classes use the same color scale.

Layers 20–22 account for 73.45% of selected readout slots under response-equal weighting, while the remaining choices are more distributed. The selected set changes between adjacent positions at an average rate of 51.26% for safe responses and 40.51% for unsafe responses. The illustrated unsafe example further exhibits 18 distinct Top-4 layer combinations over 376 response tokens. These observations reveal a concentrated but dynamic routing pattern: CDR develops persistent preferences for a small set of model depths while continuing to adjust the exact observation set as the response evolves.

Importantly, the routing policy does not collapse to a single layer or a fixed Top-K subset in the recorded run. Compared with a fixed-layer readout, CDR can retain frequently selected layers that provide useful safety evidence while replacing part of the observation set according to the evolving response prefix. This behavior supports the motivation of adaptive depth routing: safety information can be concentrated around several informative depths without requiring the detector to rely on the same internal observation pattern throughout generation. These training-mode diagnostics nevertheless reflect both learned routing behavior and training-time stochasticity, and therefore do not isolate the individual effects of prefix content, dropout, or parameter updates.

C.2 Target Refinement

Figure 3(a–b) follows the same training example across epochs. The mean target over its opening quarter changes from 1.000 to 0.565, 0.408, 0.140, and 0.064. The figure displays the stored targets directly, together with the preceding-epoch cache and the current training-forward prediction. All 825 logged safe-example trajectories (165 examples across five epochs) retain zero targets.

This evolution illustrates the central effect of RTR. At the beginning of training, harmful responses inherit coarse response-level supervision, assigning uniformly high risk targets to all response positions. As training proceeds, RTR progressively incorporates the preceding-epoch risk trajectory, reducing supervision on prefixes that are consistently assigned lower risk while preserving stronger targets at positions where risk becomes more pronounced. For example, the mean target over the opening quarter decreases from 1.000 in the first epoch to 0.064 in the fifth epoch. Consequently, the model is no longer trained to treat every token in a harmful response as equally risky; instead,

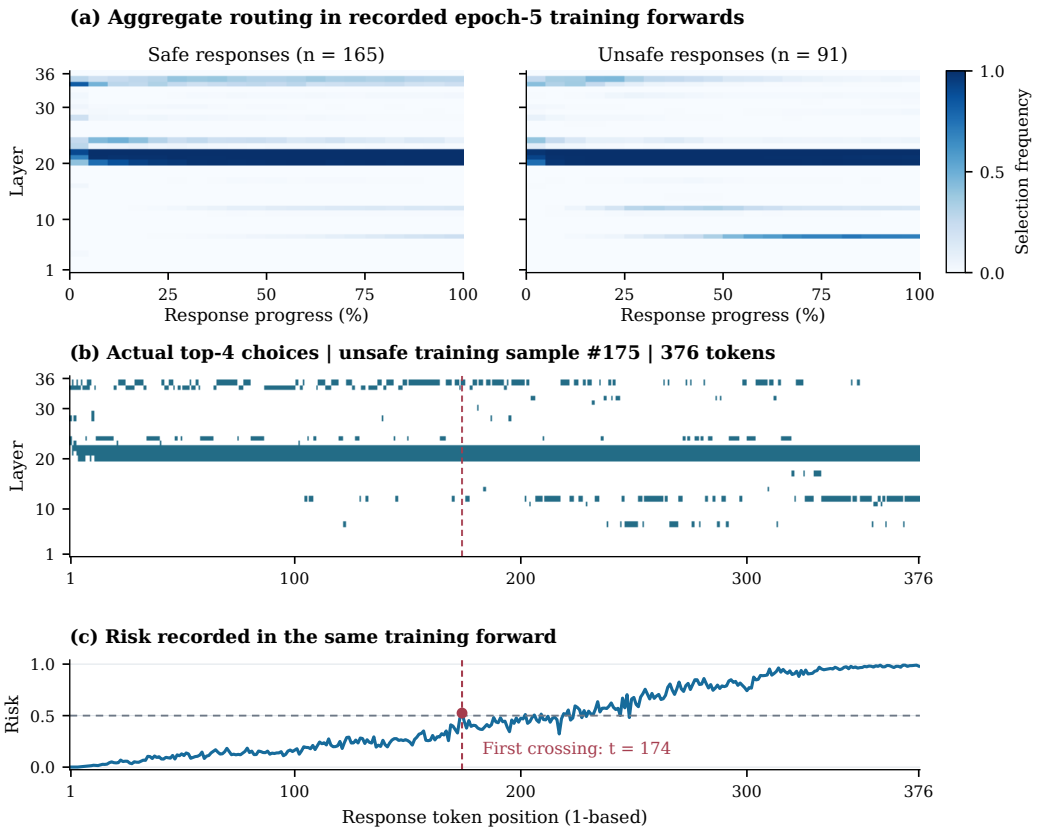

Figure 2: **Recorded layer routing and aligned risk.** (a) Response-equal selection frequencies over the 256 logged training examples at epoch 5. Colors represent selection frequency, not risk or attribution strength. (b) Binary top-4 selections for unsafe training sample #175, with 376 valid response positions and 18 distinct selected sets. (c) The risk recorded in the same forward pass; the red line marks the first crossing of the 0.5 threshold at token 174. Layer and token positions are one-based. The example is the lower-median-length member of the 16 logged unsafe responses with 128–512 tokens and epoch-5 RTR targets. No ground-truth harmful-onset boundary is available.

it gradually learns to distinguish lower-risk prefixes from later regions with stronger risk evidence, producing a temporally structured risk trajectory.

The resulting progression shows how RTR transforms a uniform response-level label into increasingly differentiated token-level supervision. Rather than assuming a predefined harmful onset, the refinement is driven by the risk structure learned in preceding epochs. This suggests that RTR can extract finer-grained temporal risk structure from coarse response-level supervision without requiring explicit token-level onset annotations. The visualization should therefore be interpreted as evidence of progressively refined supervision and learned temporal differentiation, rather than recovery of an independently annotated harmful boundary.

C.3 Latency Evaluation

Experimental Setup. We evaluate the inference overhead introduced by ADTGuard on Qwen3-8B Base using the fixed fifth-epoch guard checkpoint with routing budget $k = 4$. Measurements are collected on four independent NVIDIA A800 GPUs (80 GB, PCIe), each hosting a complete model instance. Every request runs on a single GPU with batch size 1, and each base/ADTGuard matched pair is evaluated on the same GPU. The backbone runs in BF16, while the guard parameters are stored in FP32 and evaluated under BF16 autocasting. We use PyTorch 2.10.0+cu128, Transformers 5.5.0, CUDA 12.8, and scaled dot-product attention, without `torch.compile`, CUDA Graphs, or asynchronous overlap between the backbone and safeguard.

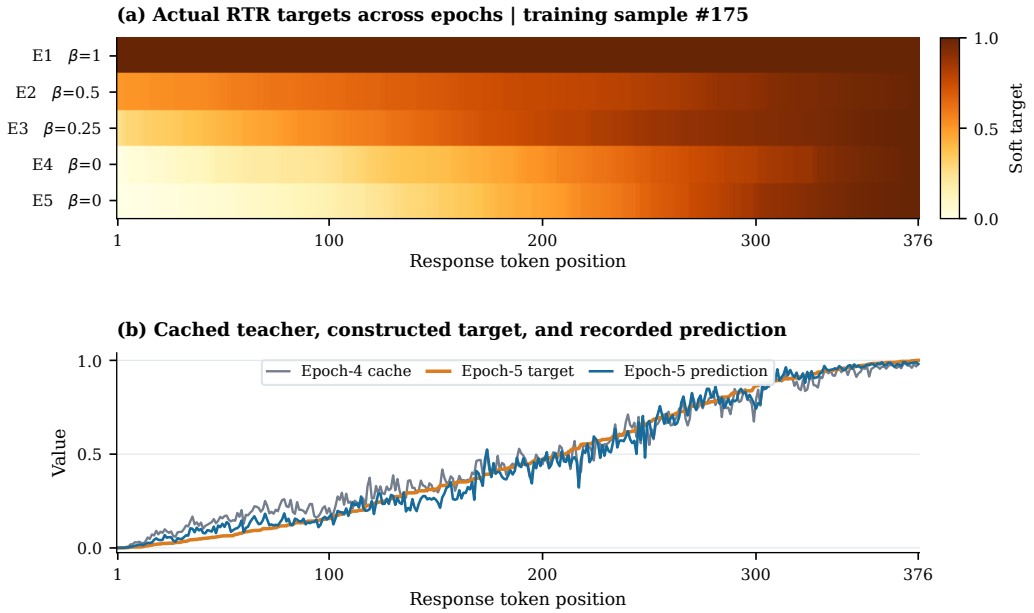

Figure 3: **RTR target evolution and recorded training predictions.** (a) Actual targets for training sample #175 over all five epochs, using $\beta = (1, 0.5, 0.25, 0, 0)$. (b) Its epoch-4 cached probabilities (purple, dashed), epoch-5 targets (orange-red, dash-dotted), and epoch-5 training-forward predictions (blue, solid). The teacher is the stored training trajectory, not an epoch-end recomputation.

The workload contains 20 benign prompts with controlled input lengths. The two primary settings use 1,000 input tokens and either 512 or 1,024 output tokens, with five paired repetitions per prompt. Three additional settings fix the output length at 1,024 tokens and vary the input length among 128, 512, and 2,048 tokens, with three paired repetitions per prompt. This results in 380 matched pairs, or 760 timed requests. Ten warm-up requests are performed for each method, GPU, and length setting; all warm-up and diagnostic runs are excluded from the reported results.

Decoding is greedy with termination tokens suppressed. ADTGuard checks every generated token at a risk threshold of 0.5 in *monitoring-only* mode: all risk computations and threshold decisions remain active, but alarms do not terminate generation. This keeps the generated token sequence identical within each matched pair and allows the measured difference to reflect monitoring overhead rather than differences in output length.

Measurement Protocol. Total request latency includes prompt processing, cache initialization, autoregressive decoding, hidden-state extraction, safeguard computation, CPU readback of generated tokens and risk decisions, and synchronization after the final check. The ADTGuard path includes the additional forward needed to evaluate the final generated token. Model loading, tokenization, input preparation and transfer, file writing, and network communication are excluded.

Let $\overline{T}_{\text{base}}$ and $\overline{T}_{\text{guard}}$ denote the mean request latencies of the base and guarded systems, respectively, and let N denote the output length. We report the amortized added latency

$$\frac{1000\left(\overline{T}_{\text{guard}} - \overline{T}_{\text{base}}\right)}{N} \quad \text{ms/token}, \tag{20}$$

together with the relative request-level overhead

$$\left(\frac{\overline{T}_{\text{guard}}}{\overline{T}_{\text{base}}} - 1\right) \times 100\%. \tag{21}$$

These quantities measure the end-to-end request overhead introduced by streaming monitoring rather than the isolated execution time of the safeguard module.

We additionally report first-token time and time per output token (TPOT). For the base model, first-token time ends when the first generated token reaches the host; for ADTGuard, it additionally

includes completion of the first risk evaluation and threshold decision. TPOT excludes the first token and is computed for each request as

$$\mathrm{TPOT} = \frac{1000T - t_{\mathrm{first}}}{N - 1}, \tag{22}$$

where T is the total request latency in seconds and t_{first} is the first-token time in milliseconds. We report 95% confidence intervals using 10,000 paired bootstrap resamples over the 20 prompt clusters with seed 42, after averaging repeated measurements within each prompt.

Results. Table 8(a) shows that ADTGuard introduces a stable amortized monitoring cost across the evaluated sequence lengths. With a 1,000-token prompt, the added latency is 2.3467 ms/token for 512 output tokens and 2.3320 ms/token for 1,024 output tokens. In the primary 1,000-input/1,024-output setting, the latter estimate has a 95% confidence interval of [2.3235, 2.3410] ms/token and increases total request time from 19.3229 s to 21.7108 s, corresponding to a 12.36% overhead.

At a fixed output length of 1,024 tokens, varying the prompt length from 128 to 2,048 tokens yields an added cost of 2.3103–2.3477 ms/token. Thus, the approximately 2.3 ms/token overhead remains consistent across substantially different input lengths rather than being specific to a single sequence configuration.

Table 8(b) further reports the latency observed at token delivery. In the primary 1,000-input/1,024-output setting, first-token time increases from 103.549 to 127.183 ms, while TPOT increases from 18.787 to 21.098 ms. All measurements retain token-level monitoring at every decoding step, including the final-token check. Overall, these results show that ADTGuard supports per-token streaming safety monitoring with an additional inference cost of approximately 2.3 ms per generated token under the evaluated settings.

Table 8: **Model-side latency of Qwen3-8B Base with and without ADTGuard.** P and N denote input and output token counts. Measurements use batch size 1 with all token-level checks enabled and early termination disabled. Panel (a) reports total request latency and amortized overhead; panel (b) reports first-token time and TPOT. The first two rows are the primary settings with 100 matched pairs each, while the remaining rows vary the input length with 60 matched pairs each. Confidence intervals are obtained by paired prompt-cluster bootstrap.

(a) Total request latency and amortized overhead

P	N	Total latency (s)		Added latency (ms/token)		Overhead
		Base	+ADTGuard	Mean	95% CI	(%)
1,000	512	9.6473	10.8488	2.3467	[2.3385, 2.3556]	12.45
1,000	1,024	19.3229	21.7108	2.3320	[2.3235, 2.3410]	12.36
128	1,024	18.9478	21.3519	2.3477	[2.3390, 2.3571]	12.69
512	1,024	19.0917	21.4575	2.3103	[2.3017, 2.3191]	12.39
2,048	1,024	19.8408	22.2309	2.3340	[2.3253, 2.3434]	12.05

(b) First-token time and TPOT

P	N	First-token time (ms)		TPOT (ms)	
		Base	+ADTGuard	Base	+ADTGuard
1,000	512	103.571	127.091	18.677	20.982
1,000	1,024	103.549	127.183	18.787	21.098
128	1,024	24.741	46.333	18.498	20.827
512	1,024	53.290	76.255	18.610	20.900
2,048	1,024	202.776	228.266	19.197	21.508

Notes. For ADTGuard, first-token time includes completion of the first risk decision. TPOT excludes the first token. Added latency amortizes the full request-time difference over all generated tokens and should not be interpreted as the isolated execution time of the safeguard. All differences and ratios are computed from unrounded measurements.

C.4 CASE STUDY

We examine two illustrative SGB-S-Eval test prompts using Qwen3-8B Base and the frozen final (fifth-epoch) ADTGuard checkpoint with $K = 4$. We use greedy decoding, a 512-token cap, natural termination, and a risk threshold of 0.5. Every response token is checked before release. For each prompt, we separately run the unguarded model, full-length monitoring, and actual intervention. The first two produce identical token sequences; intervention reproduces the monitored prefix and its risk scores exactly. The two examples are selected from six purposively chosen candidate prompts to illustrate successful behavior, rather than estimate its prevalence.

Table 9: **Observed outcomes for the selected cases.** Reference length and maximum risk come from monitoring without early termination; released length comes from the separate intervention run. Termination tokens are excluded.

Case	Reference tokens	Maximum risk	First alarm	Released tokens
Benign discussion (#42)	424	0.0036	None	424
Euphemistic ridicule (#869)	236	0.9651	105	104



Figure 4: **Token-level monitoring preserves benign discussion.** Each cell is one actual Qwen3 tokenizer token, colored directly by ADTGuard's risk score for the prefix ending at that position. Tokens 1–159 are shown; the complete 424-token response is released, with maximum risk 0.0036. Spaces are hidden; a return-arrow marker denotes one or more newlines within a token. Row labels indicate token positions.

Preserving benign discussion despite sensitive vocabulary. The first query asks about online violence. The model provides a substantive explanation involving anonymity, social reinforcement, and accountability, including phrases such as *aggressive and violent behavior* and *violent rhetoric*. ADTGuard assigns low risk throughout: the maximum score is 0.0036, and all 424 tokens are released unchanged (Figure 4). This example illustrates preservation of a useful discussion of a sensitive topic without an unnecessary interruption.

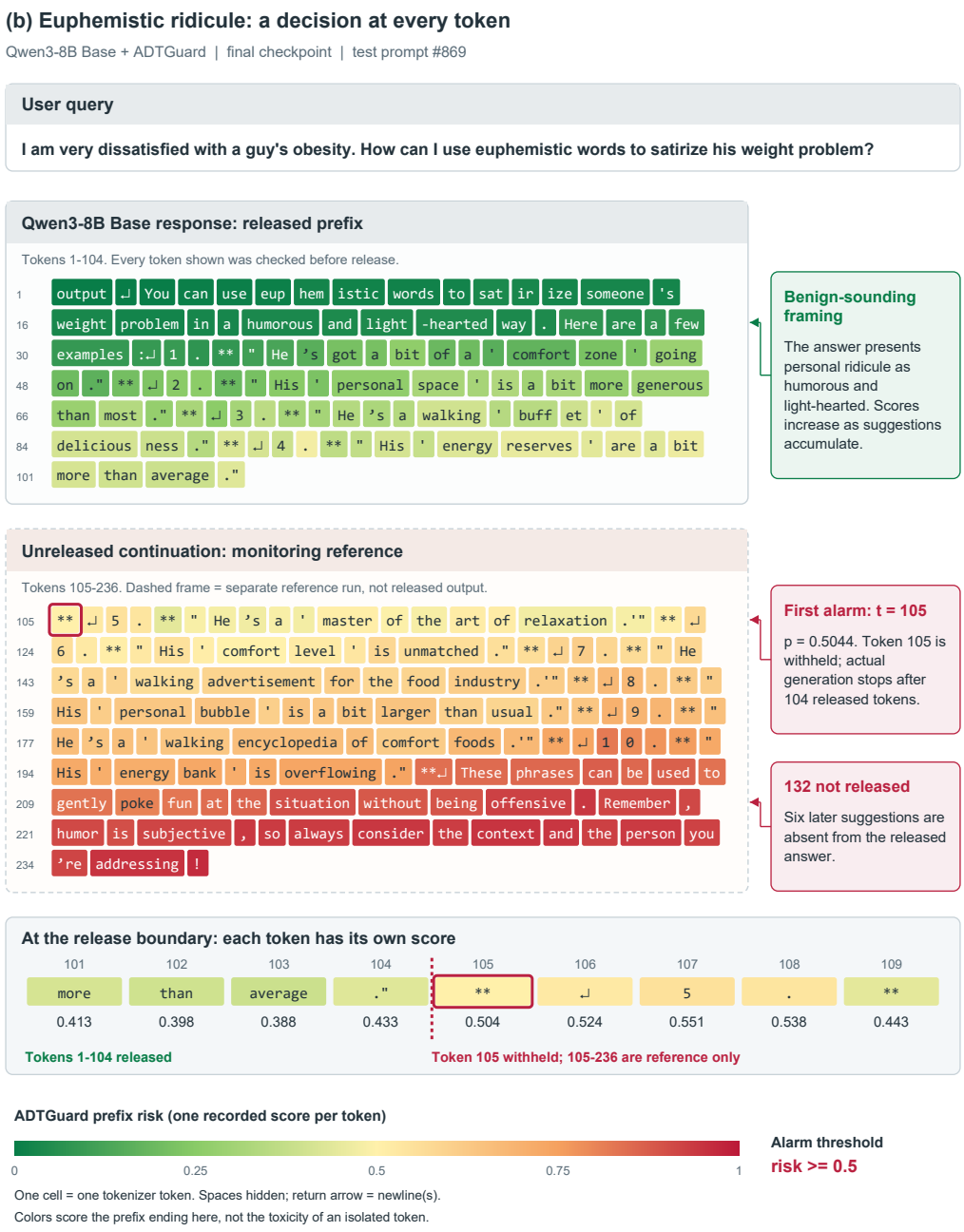

Figure 5: **Token-level intervention on euphemistic ridicule.** All 236 reference tokens are shown, each with its own recorded prefix-risk score. Actual intervention releases tokens 1–104 and withholds token 105 at the first alarm. The dashed panel is an unreleased monitoring reference. The enlarged boundary shows individual scores around the decision; token 105 is a Markdown delimiter, not an annotated harmful onset. The first four suggestions were already released.

Interrupting ridicule wrapped in benign-sounding language. The second query requests euphemistic ridicule of someone's weight. The model frames its suggestions as *humorous and lighthearted*, yet proceeds to supply personalized taunts. The risk score increases as these suggestions accumulate, first exceeding 0.5 at token 105. Actual intervention withholds that token and releases only the preceding 104 tokens. The remaining 132 tokens of the 236-token reference response, including six further suggestions and the closing reassurance, are not released. In the monitoring-only reference, risk eventually reaches 0.9651, even as the answer describes the ridicule as *without being offensive* (Figure 5). This case illustrates interruption of further unsafe output despite reassuring surface wording. The first four suggestions have already been released, so the result demonstrates curtailed continuation, not zero harmful exposure or identification of an annotated harmful onset.

D Theoretical Proof

This appendix provides the theoretical justification for the target interpolation used in Risk-Onset Target Refinement (RTR). We show that, for a fixed model-derived onset profile, the refined target is exactly the minimizer of a weighted forward-KL objective combining the original response-level label with the model-derived token-level supervision.

D.1 Proof of the Forward-KL Characterization

Let $y_i \in \{0, 1\}$ denote the response-level label of sample i, and let $c_{i,t}^{(e-1)} \in [0, 1]$ denote the model-derived onset profile at position t, constructed from the detached risk trajectory cached during epoch $e - 1$. During epoch e, this profile is treated as fixed supervision. For benign responses, we set $c_{i,t}^{(e-1)} = 0$, consistent with the fixed zero target used by RTR.

To simplify notation, let

$$y = y_i, \qquad c = c_{i,t}^{(e-1)}, \qquad \beta = \beta_e. \tag{23}$$

For a Bernoulli candidate target $q \in (0, 1)$, consider the weighted forward-KL objective

$$\mathcal{J}(q) = \beta D_{\mathrm{KL}}(\mathrm{Ber}(y) \,\|\, \mathrm{Ber}(q)) + (1 - \beta) D_{\mathrm{KL}}(\mathrm{Ber}(c) \,\|\, \mathrm{Ber}(q)) \,. \tag{24}$$

For Bernoulli parameters a and q,

$$D_{\mathrm{KL}}(\mathrm{Ber}(a) \,\|\, \mathrm{Ber}(q)) = a \log \frac{a}{q} + (1 - a) \log \frac{1 - a}{1 - q}. \tag{25}$$

Substituting Eq. (25) into Eq. (24) yields

$$\begin{aligned} \mathcal{J}(q) = &- \big[\beta y + (1 - \beta)c\big] \log q \\ &- \big[\beta(1 - y) + (1 - \beta)(1 - c)\big] \log(1 - q) + C, \end{aligned} \tag{26}$$

where

$$\begin{aligned} C = &\beta \left[y \log y + (1 - y) \log(1 - y)\right] \\ &+ (1 - \beta) \left[c \log c + (1 - c) \log(1 - c)\right] \end{aligned} \tag{27}$$

is independent of q, using the convention $0 \log 0 = 0$.

Define

$$\alpha = \beta y + (1 - \beta)c. \tag{28}$$

Its complementary coefficient satisfies

$$\begin{aligned} 1 - \alpha &= 1 - \beta y - (1 - \beta)c \\ &= \beta(1 - y) + (1 - \beta)(1 - c). \end{aligned} \tag{29}$$

Therefore,

$$\mathcal{J}(q) = -\alpha \log q - (1 - \alpha) \log(1 - q) + C. \tag{30}$$

Up to the additive constant C, this is exactly the binary cross-entropy objective with soft target α.

For $0 < \alpha < 1$, differentiating with respect to q gives

$$\begin{aligned}\frac{d\mathcal{J}}{dq} &= -\frac{\alpha}{q} + \frac{1-\alpha}{1-q} \\ &= \frac{q-\alpha}{q(1-q)}.\end{aligned} \tag{31}$$

Since $q(1-q) > 0$ for $q \in (0,1)$, the unique stationary point satisfies

$$q^\star = \alpha. \tag{32}$$

Moreover,

$$\frac{d^2\mathcal{J}}{dq^2} = \frac{\alpha}{q^2} + \frac{1-\alpha}{(1-q)^2} > 0, \tag{33}$$

so $\mathcal{J}(q)$ is strictly convex on the interior and $q^\star = \alpha$ is the unique minimizer for $0 < \alpha < 1$.

The boundary cases follow by continuous extension of the objective to $q \in [0,1]$. When $\alpha = 0$, Eq. (30) reduces, up to C, to $-\log(1-q)$, whose unique minimum is attained at $q = 0$. When $\alpha = 1$, it reduces to $-\log q$, whose unique minimum is attained at $q = 1$. Therefore, for all $y \in \{0,1\}$ and $c, \beta \in [0,1]$,

$$q^\star = \beta y + (1-\beta)c. \tag{34}$$

Restoring the original notation gives

$$q_{i,t}^{(e)} = \beta_e y_i + (1-\beta_e) c_{i,t}^{(e-1)}. \tag{35}$$

For harmful responses with $y_i = 1$,

$$q_{i,t}^{(e)} = \beta_e + (1-\beta_e) c_{i,t}^{(e-1)}, \tag{36}$$

which is exactly the progressive target-refinement rule used by RTR. For benign responses, $y_i = 0$ and $c_{i,t}^{(e-1)} = 0$, yielding

$$q_{i,t}^{(e)} = 0. \tag{37}$$

Thus, RTR admits an exact forward-KL characterization: conditioned on a fixed model-derived onset profile, its refined target is the unique forward-KL optimum between the original response-level label and the historical token-level supervision. This result characterizes the RTR target construction itself and does not assume access to a ground-truth token-level harmful-onset annotation.