KVGUARD: Catastrophe-Aware KV-Cache Compression via Closed-Loop Hazard Control

Anonymous Author(s)

Abstract

KV-cache compression enables long-context inference under memory constraints, but existing compressors operate without feedback: they evict tokens irreversibly and never verify whether eviction degraded generation quality. We show that compression induces *catastrophic failures*—looping, non-termination, and reasoning corruption—that standard average-accuracy metrics hide. We present KVGUARD, a closed-loop controller inspired by TCP congestion control that wraps any existing KV-cache compressor. KVGUARD extracts a 42-dimensional per-token feature vector from the model's own logit distribution at near-zero cost, trains a hazard predictor to forecast catastrophe within the next 32 tokens (AUROC 0.945), and uses a hysteresis-based state machine to dynamically relax compression when danger is detected. Evaluated on Qwen2.5-3B-Instruct with GSM8K across three compressors (StreamingLLM, SnapKV, ObservedAttention) at five compression ratios, KVGUARD reduces the Catastrophic Failure Rate by 43.2% overall (63.9% excluding immediate-onset cases) with only 6.9% false positive rate. The controller exceeds 50% CFR reduction on all three compressors, validating its compressor-agnostic design. Ablations confirm that each component—trained predictor (+42.6pp over random), hysteresis (−53pp false positives), and graduated response—contributes independently.

1 Introduction

Autoregressive language models cache key-value (KV) pairs from every attention layer at every decoding step, producing memory requirements that grow linearly with sequence length. For long-context inference—multi-step reasoning, document summarization, agentic workflows—this cache can dominate GPU memory, prompting a line of work on KV-cache compression: methods that evict, quantize, or merge cache entries to reduce the memory footprint (Xiao et al., 2024; Li et al., 2024; Zhang et al., 2023; Cai et al., 2024). These methods report average accuracy or perplexity under compression and show that significant memory savings are possible with modest average-case degradation.

However, average metrics conceal a dangerous failure pattern. At moderate-to-aggressive compression ratios, a fraction of prompts exhibit *catastrophic failures*: the model enters infinite loops, fails to terminate, or silently corrupts its chain-of-thought reasoning. A compressor reporting 95% average accuracy may achieve 0% accuracy on the 5% of cases where compression causes catastrophe—a tail that standard evaluation completely hides. We characterize three distinct failure profiles across three compressors: StreamingLLM exhibits a cliff (stable accuracy that collapses at a threshold), SnapKV shows catastrophic cliffs (sudden universal looping), and ObservedAttention degrades progressively. These diverse profiles make the problem particularly challenging: a safety mechanism must handle both sudden and gradual failure modes.

All existing KV-cache compression methods are *open-loop*: they decide which tokens to evict based on importance scores or attention patterns, execute the eviction irreversibly, and never check whether the decision was correct. Recent work has begun to recognize this gap. ASR-KF-EGR (Metinov et al., 2024) proposes but does not implement an entropy-guided recovery system. RefreshKV (Xu et al., 2024) maintains the full cache for periodic recalibration, sacrificing memory savings entirely. ERGO (Khalid et al., 2025) demonstrates that entropy signals can trigger corrective action but employs only a full context reset—a sledgehammer response to what may be a localized problem. ThinKV (Ramachandran et al., 2025) classifies reasoning phases to vary compression aggressiveness, but provides no mechanism to detect or recover from misclassification. None of these systems close the loop: none monitor compressed generation quality and feed that information back into cache management.

The detection signals, however, already exist. The HALT framework (Shapiro et al., 2026) shows that a model's top-k logit features cheaply predict generation quality. Work on reasoning dynamics (Qian et al., 2025) identifies a small set of "thinking tokens" whose suppression disproportionately degrades chain-of-thought. Entropy monitoring, used by ERGO for post-hoc detection, can serve as a proactive leading indicator. Repetition patterns directly signal looping. What is missing is not signal—it is *control*: connecting these signals to a feedback mechanism that adjusts compression before failures become irreversible.

We present KVGUARD, a catastrophe-aware controller inspired by TCP congestion control (Jacobson and Karels, 1988). Like TCP, the controller starts aggressively (high compression), monitors for trouble (hazard prediction), and backs off when problems are detected (graduated response from alert through safe mode to recovery). Unlike existing systems, KVGUARD is a *wrapper*: it wraps any existing KV-cache compressor without modification, making it orthogonal to compression research—every advance in compression methods automatically benefits from the safety layer.

Our contributions are:

1. **Characterization of compression-induced catastrophic failures.** We collect 800 generation traces across three compressors and five compression ratios, identifying three distinct failure profiles and showing that 44% of traces at moderate-to-aggressive compression exhibit catastrophe.

2. **A zero-cost hazard predictor.** We extract a 42-dimensional per-token feature vector from quantities already computed during decoding (logit distribution, entropy, repetition counts) and train an XGBoost classifier that achieves AUROC 0.945 with 100% trace-level detection and 34-token mean lead time before catastrophe onset. Leave-one-compressor-out cross-validation (mean AUROC 0.898) validates cross-compressor generalization.

3. **A closed-loop controller with hysteresis-based state machine.** The controller reduces CFR by 43.2% overall (63.9% excluding immediate-onset cases) with only 6.9% false positive rate, exceeding 50% CFR reduction on all three compressors. The key mechanism is cumulative detection: requiring k consecutive high-risk tokens converts modest per-token recall into reliable per-trace triggering.

4. **Systematic ablation validating each component.** We show that the trained predictor adds 42.6 percentage points over a random baseline, hysteresis reduces false positives by 53pp at a 17pp prevention cost, and the balanced configuration achieves 70.5% of the theoretical maximum prevention ceiling while triggering on only 8.1% of prompts.

2 Related Work

We organize related work along three axes: KV-cache compression methods (the systems kvguard wraps), recovery and adaptation mechanisms (the closest relatives), and generation quality monitoring (the signal sources kvguard builds on).

2.1 KV-Cache Compression

KV-cache compression methods reduce the memory footprint of autoregressive decoding by selectively retaining a subset of key-value pairs. StreamingLLM (Xiao et al., 2024) preserves a fixed window of recent tokens plus a small set of "attention sink" initial tokens, achieving constant memory at the cost of discarding intermediate context. H2O (Zhang et al., 2023) dynamically evicts tokens with the lowest cumulative attention score, adapting the cache to the model's evolving focus. SnapKV (Li et al., 2024) clusters attention patterns during prefill to select a compact representative set per head. PyramidKV (Cai et al., 2024) allocates different cache budgets across layers based on attention entropy, assigning more cache to high-entropy (broadly attending) layers.

All of these methods are *open-loop*: they make eviction decisions based on importance heuristics and never verify whether those decisions degraded downstream generation. When compression is too aggressive, failures are silent—the model loops, fails to terminate, or produces corrupted reasoning—and standard average-accuracy evaluation hides these tails. kvguard is designed to wrap any of these compressors, adding a feedback loop without modifying the compression algorithm itself.

2.2 Recovery and Adaptation Mechanisms

Several recent systems recognize that static compression can fail and attempt various forms of recovery or adaptation. We discuss each in terms of its detection mechanism, response strategy, and architectural relationship to the underlying compressor.

ASR-KF-EGR (Metinov et al., 2024) proposes a four-level entropy-guided recovery system (soft reset, window reset, full reset, rewalk regeneration) atop a reversible soft-freeze compression mechanism. However, the recovery system is described only as future work: no entropy thresholds are defined, no triggering mechanism is implemented, and no evaluation is performed. Furthermore, ASR-KF-EGR is itself a compressor (soft-freeze with sublinear scheduling), not a wrapper—it competes with StreamingLLM or SnapKV rather than making them safer. The 5$\times$ overhead from CPU–GPU token transfers (Metinov et al., 2024) contrasts with kvguard's near-zero cost from logit-derived features.

RefreshKV (Xu et al., 2024) alternates between partial-cache and full-cache attention steps, refreshing token selection at each full-attention step based on query similarity. This provides genuine recovery (52% of lost performance recovered on structured extraction tasks) but requires maintaining the complete KV cache throughout inference, yielding zero memory savings. The refresh schedule is periodic or similarity-triggered, not failure-triggered: RefreshKV pays the refresh cost even when compression is performing correctly.

ERGO (Khalid et al., 2025) monitors rolling entropy delta between generation windows and triggers a full context reset when ΔH exceeds a calibrated threshold (85–90th percentile on $\sim$80 examples). This achieves 56.6% average performance gain and 35.3% unreliability reduction, validating entropy as a detection signal. However, the response is a sledgehammer: complete regeneration from scratch, discarding all generated text. ERGO also operates at the generation level rather than the cache level—it does not adjust compression ratios or protect specific tokens. kvguard builds on ERGO's validation of entropy signals while providing graduated responses and operating directly on the cache.

ThinKV (Ramachandran et al., 2025) classifies reasoning into three modes (Reasoning, Transition, Execution) via attention sparsity metrics and applies different compression policies per mode (4-bit quantization for reasoning, 2-bit for execution). This acknowledges that different reasoning phases tolerate different compression—an insight shared with kvguard's multi-mode controller. However, ThinKV provides no mechanism to detect *misclassification*: if a reasoning-critical token is classified as execution and aggressively compressed, there is no correction. Classification also operates at 128-token granularity versus kvguard's per-token risk scoring.

Rethinking KV-Cache Compression (Gao et al., 2025) uses a pre-generation complexity evaluator to route inputs to either compressed or full-cache inference. This is an effective form of risk screening, but the routing decision is static—once made, the compression policy is fixed for the entire generation. Within-generation dynamics (reasoning state evolving, compression failure emerging mid-sequence) cannot be detected or addressed.

DefensiveKV (Feng et al., 2025) replaces snapshot-based importance scoring with max-over-history aggregation, preventing premature eviction of tokens whose importance is non-stationary. Their finding that retained-token importance can drop to 0.34$\times$ its initial value provides strong empirical motivation for closed-loop monitoring: if importance scoring is fundamentally unstable, passive insurance cannot guarantee safety. DefensiveKV is applied once at prefill and provides no runtime detection or recovery.

UNComp (Xiong et al., 2025) uses matrix entropy to drive inter-layer and inter-head compression budgets, allocating more cache to low-entropy (focused-attention) heads. Like DefensiveKV, this optimizes the initial compression policy but does not monitor generation quality at runtime.

Summary. Table 1 positions kvguard against these systems. The key differentiators are: (i) kvguard is a *wrapper* that makes any compressor safer, not a competing compressor; (ii) it uses a *trained* hazard predictor on a 42-dimensional feature vector rather than a single-signal threshold; (iii) it provides *graduated* response (alert $\rightarrow$ safe $\rightarrow$ recovery) rather than all-or-nothing reset; and (iv) it evaluates on a *catastrophe-specific* metric (CFR) rather than average accuracy.

Table 1: Comparison of kvguard with related systems along five key dimensions. **Bold** indicates a unique capability.

System	Wrapper?	Runtime Detection	Recovery	Memory Savings	Overhead
ASR-KF-EGR	No	Proposed (unimpl.)	Proposed (unimpl.)	55–67%	5×
RefreshKV	No	None	Full-cache refresh	None	∼0
ERGO	Partial	Entropy threshold	Full reset	N/A	Reset cost
ThinKV	No	None	None	Yes	∼0
Rethinking KV	Partial	Pre-generation only	Route to full cache	Varies	Evaluator
DefensiveKV	No	None	None	Yes	∼0
UNComp	No	None	None	Yes	∼0
kvguard	**Yes**	**Trained predictor**	**Graduated**	**Preserved**	**∼0**

2.3 Generation Quality Monitoring

kvguard's hazard predictor draws on three lines of work that provide the detection signals.

Logit-based quality prediction. The HALT framework (Shapiro et al., 2026) demonstrates that top-k log-probabilities from a model's own output distribution predict generation quality with minimal computational overhead. These features are a byproduct of the softmax already computed during decoding, making them essentially free. kvguard uses 25 HALT-derived features (top-20 log-probabilities, max, sum, mean, entropy, and var-entropy of the logit distribution) as the core of its 42-dimensional feature vector.

Thinking tokens and reasoning dynamics. Recent work on reasoning dynamics (Qian et al., 2025) identifies a small vocabulary of "thinking tokens" (e.g., "so", "therefore", "wait", "hmm") comprising 1–5% of generated tokens whose suppression disproportionately degrades chain-of-thought reasoning. kvguard uses a thinking-token flag as a feature and, in alert mode, protects these tokens from compression.

Receiver heads and attention anchors. Thought anchors (Bogdan et al., 2025) identifies attention heads that consistently concentrate on a few "anchor" tokens carrying disproportionate information. kvguard's recovery mechanism uses receiver-head identification to perform selective KV recomputation for anchor tokens, enabling surgical recovery rather than full-cache restoration.

Together, these signals enable kvguard's near-zero-cost monitoring: the features are extracted from quantities already computed during standard autoregressive decoding, requiring no additional forward passes or attention recomputation.

3 Method

We present kvguard, a closed-loop controller that wraps any existing KV-cache compressor and dynamically adjusts compression aggressiveness based on runtime signals. The key insight is that catastrophic compression failures—looping, non-termination, reasoning corruption—produce detectable precursor signals in the model's own logit distribution, and these signals can be monitored at near-zero cost during decoding. The system has four components: a trace collection pipeline (§3.2), a per-token feature extractor (§3.3), a trained hazard predictor (§3.4), and a state-machine controller (§3.5).

3.1 Problem Formulation

Let $\mathcal{C}$ be a KV-cache compressor (e.g., StreamingLLM, SnapKV, H2O) parameterized by a compression ratio $r \in [0, 1]$ denoting the fraction of KV entries removed. At each decoding step t, the model produces a token x_t and a logit vector $\ell_t \in \mathbb{R}^{|V|}$. We define a *catastrophic failure* as any of: (i) looping—a 20-token window repeated $\geq$3 times; (ii) non-termination—generation reaches `max_new_tokens` without producing EOS; or (iii) answer failure—the final answer is incorrect. The *Catastrophic Failure Rate* (CFR) at ratio r is the fraction of prompts exhibiting any catastrophe.

The controller's goal is to minimize CFR while preserving compression savings. Formally, we seek a policy π that observes per-token signals s_t and outputs a (possibly adjusted) compression ratio r_t, such that CFR under π is substantially lower than CFR under the static policy $r_t = r$ for all t.

3.2 Trace Collection

We generate a behavior dataset by running Qwen2.5-3B-Instruct on 50 GSM8K prompts (3-shot chain-of-thought) under each combination of compressor and ratio. Three compressors from the `kvpress` library are used—StreamingLLM, SnapKV, and ObservedAttention—at five compression ratios (0.25, 0.50, 0.625, 0.75, 0.875), plus an uncompressed baseline, yielding 16 configurations and 800 total traces.

For each trace, we record:

- The full generated token sequence under greedy decoding (`max_new_tokens` = 512).
- Per-token logit-derived signals (entropy, top-k probabilities, rank of chosen token, top-20 log-probabilities); see §3.3.
- Per-token repetition counts: a sliding 20-token window tracks how many times the current window has appeared so far.
- Catastrophe labels and onset positions. For *looping*, the onset is the token index where the second occurrence of a repeated window begins. For *non-termination*, the true onset (the last token) is uninformative; we use a proxy onset at $\lfloor 0.75 \cdot \texttt{max\_new\_tokens} \rfloor = 384$. *Wrong-answer* failures have no per-token onset and are excluded from the hazard prediction target.

Each compressor–ratio configuration runs in an isolated process with a 300-second per-prompt timeout and per-prompt checkpointing for fault tolerance.

3.3 Per-Token Feature Extraction

At each decoding step t, we extract a 42-dimensional feature vector $\mathbf{f}_t$ from the logit distribution ℓ_t and decoding history. The features are designed to be *zero additional cost*: they are computed from quantities already available during standard autoregressive generation.

Base features (30 dimensions). From the softmax distribution $p_t = \text{softmax}(\ell_t)$:

- **Entropy** $H_t = -\sum_v p_t(v) \log p_t(v)$: overall uncertainty.
- **Top-1 and top-5 probabilities**: confidence in the greedy choice and the top-5 tokens.
- **Rank of chosen token**: always 0 under greedy decoding; informative under sampling.
- **Top-20 log-probabilities** ($\log p_t(v)$ for the 20 highest-probability tokens): the HALT feature set (Shapiro et al., 2026), which captures the shape of the logit distribution beyond the argmax.
- $h_{\textbf{alts}}$: average log-probability of the top-20 alternatives, measuring how concentrated probability mass is.
- $\Delta H_t = H_t - H_{t-1}$: entropy change between consecutive tokens, with a validity flag for $t = 0$.
- **Repetition count**: number of times the current 20-token window has appeared so far.
- **Thinking-token flag**: binary indicator for tokens in a 31-word reasoning vocabulary (e.g., “so”, “wait”, “therefore”, “hmm”), drawn from work showing that suppressing these 1–5% of tokens disproportionately degrades reasoning (Qian et al., 2025).

Rolling features (11 dimensions). To capture temporal dynamics without future leakage, we compute *causal* rolling statistics over a lookback window of $w = 8$ tokens:

- Rolling mean and standard deviation of entropy, top-1 probability, h_{alts}, and ΔH (8 features).
- Rolling sum of repetition count (1 feature).

For positions $t < w$, the window grows from 1 to $w - 1$ tokens, ensuring no padding artifacts.

Context features (2 dimensions).

- **Token position**: $t/(T-1)$, normalized to $[0, 1]$. Captures position-dependent failure risk (e.g., non-termination signals strengthen toward the end).
- **Compression ratio**: the static ratio r applied to this trace. This is known at inference time and lets the predictor learn compressor-specific risk profiles.

All features are causal (depend only on tokens $\leq t$), zero additional cost (derived from quantities already computed during decoding), and compressor-agnostic (no access to internal compressor state).

3.4 Hazard Prediction

We train a binary classifier to predict, at each token t, whether a catastrophe will occur within the next H tokens. This framing—predicting a hazard window rather than instantaneous failure—gives the controller lead time to intervene.

Label construction. For a trace with earliest catastrophe onset at token t^*, the hazard label is:

$$y_t = \begin{cases} 1 & \text{if } t \geq t^* - H \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

We use horizon $H = 32$. Only looping and non-termination receive per-token labels; wrong-answer failures are excluded because they produce no detectable per-token onset signal. When multiple catastrophe types co-occur, the earliest onset determines t^*.

Model. We use XGBoost with 200 trees, max depth 6, learning rate 0.1, and subsampling at 0.8. Class imbalance (13% positive rate) is handled via `scale_pos_weight`. The choice of gradient-boosted trees over neural alternatives is deliberate: XGBoost trains in seconds on 263K tokens, requires no GPU, produces calibrated probabilities, and provides interpretable feature importance—properties that matter for a safety-critical runtime component.

Evaluation protocol. We split at the *trace level* (not token level) to prevent data leakage: a trace cannot appear in both train and validation. The split is stratified by catastrophe presence to ensure both classes appear in each partition. Cross-compressor generalization is evaluated via *leave-one-compressor-out* cross-validation: for each compressor, we train on traces from the other two and evaluate on the held-out compressor. This tests whether the predictor transfers to an unseen compression method—a prerequisite for the compressor-agnostic claim.

3.5 Controller Design

The controller is a per-token state machine inspired by TCP congestion control: it begins with aggressive compression, monitors for signs of trouble, and backs off when hazard is detected. Unlike TCP, which adjusts a single parameter (window size), the controller adjusts both the compression ratio and token-protection policy.

Modes. The controller operates in four modes, ordered by severity:

Mode	Ratio	Action
NORMAL	r_{base}	Standard compression
ALERT	r_{base}	Protect thinking tokens from eviction
SAFE	r_{safe}	Relax compression ratio
RECOVERY	0	Selective KV recomputation

In the evaluated configuration, $r_{\text{base}} = 0.875$ and $r_{\text{safe}} = 0$ (fall back to no compression).

Escalation and de-escalation. Let $\hat{p}_t$ denote the hazard predictor's output probability at token t. Escalation requires k *consecutive* tokens above a threshold (hysteresis):

NORMAL $\to$ ALERT:	$\hat{p}_t > \tau_{\text{low}}$	for k consecutive toker
ALERT $\to$ SAFE:	$\hat{p}_t > \tau_{\text{high}}$	for k consecutive toker
SAFE $\to$ RECOVERY:	rep_count ≥ 3	(immediate)

De-escalation requires j consecutive tokens *below* the threshold, with $j > k$ to implement conservative hysteresis—it is easier to raise an alarm than to lower it.

Hysteresis as cumulative detection. The consecutive-token requirement is the key design choice. A single high-risk token may be noise; k consecutive high-risk tokens indicate a genuine trend. This converts the per-token predictor (which may have modest recall) into a reliable per-trace trigger. In our evaluation, the predictor has only 69% token-level recall at 5% FPR, yet the controller with $k = 8$ achieves 100% trace-level detection—because every catastrophe trace produces *some* run of elevated hazard probabilities.

Operating points. The controller exposes three tunable parameters: τ_{low}, τ_{high}, and k. We evaluate three operating points spanning the prevention–selectivity tradeoff:

	$\tau_{\text{low}}/\tau_{\text{high}}$	k	Character
Conservative	0.7 / 0.8	5	Low FP, moderate prevention
Balanced	0.3 / 0.7	8	Best prevention-to-FP ratio
Aggressive	0.3 / 0.6	5	High prevention, higher FP

3.6 Anchor-Aware Token Protection

When the controller enters ALERT or SAFE mode, it must decide *which* KV entries to protect from eviction. We identify critical positions using *receiver heads*—attention heads that consistently concentrate attention on a few "anchor" tokens carrying disproportionate information (Bogdan et al., 2025).

Calibration. Given attention tensors from calibration traces (shape: layers × heads × seq × seq), we compute a *vertical attention score* for each key position j through each head:

$$v_j = \frac{1}{|\{i : i \geq j + d\}|} \sum_{i \geq j+d} A_{i,j} \qquad (2)$$

where $A_{i,j}$ is the attention weight from query i to key j and $d = 32$ is a minimum distance that filters out local attention patterns. Heads with high excess kurtosis in their vertical attention scores—i.e., "spiky" heads that attend strongly to a few positions—are selected as the top-K receiver heads.

Runtime anchor identification. During generation, we compute vertical attention scores through only the calibrated receiver heads, average across heads, and threshold at the 90th percentile. Positions above this threshold are marked as *anchors* and protected from eviction in ALERT/SAFE mode. In RECOVERY mode, anchor KV entries are candidates for selective recomputation.

3.7 Evaluation Framework

We evaluate the controller via *offline simulation* on the 800 collected traces. For each trace at ratio r where the controller triggers SAFE mode (transitions before catastrophe onset), we check whether the same prompt at $r_{\text{safe}} = 0$ (no compression) avoids catastrophe. If so, the catastrophe is counted as *prevented*.

This simulation makes a simplifying assumption: that switching to r_{safe} mid-generation recovers the same behavior as never compressing. This may not hold if early compressed tokens have already corrupted the reasoning state. The simulation therefore provides an *upper bound* on live controller performance.

We report:

- **CFR reduction**: $(\text{CFR}_{\text{baseline}} - \text{CFR}_{\text{controlled}})/\text{CFR}_{\text{baseline}}$, the relative decrease in catastrophic failures.
- **False positive rate**: fraction of non-catastrophe traces where the controller triggers unnecessarily.
- **Per-compressor breakdown**: verifying the controller's benefit generalizes across compression methods.

Table 2: Baseline accuracy and CFR across compressors and compression ratios (fraction of KV cache removed). $n = 50$ prompts per configuration. Dominant failure modes: WA = wrong answer, NT = non-termination, LP = looping.

Compressor	Ratio	Acc.%	CFR%	Dominant
None (baseline)	0.000	74	26	WA
StreamingLLM	0.250	80	20	WA
	0.500	80	20	WA
	0.625	80	20	WA
	0.750	74	26	WA
	0.875	14	86	WA+LP
SnapKV	0.250	78	22	WA
	0.500	76	24	WA
	0.625	66	34	WA
	0.750	66	34	WA+NT
	0.875	0	100	LP (49/50)
Obs. Attention	0.250	74	26	WA
	0.500	56	44	WA
	0.625	36	64	WA
	0.750	24	76	WA+NT
	0.875	12	88	WA

4 Results

We evaluate kvguard on GSM8K (50 prompts, 3-shot CoT) using Qwen2.5-3B-Instruct on Apple MPS, with three kvpress compressors (StreamingLLM, SnapKV, ObservedAttention) at five compression ratios (0.25, 0.50, 0.625, 0.75, 0.875). Our primary metric is *Catastrophic Failure Rate* (CFR): the fraction of prompts exhibiting looping, non-termination, or answer failure. We report both absolute CFR and the controller's relative CFR reduction.

4.1 Compression Induces Distinct Failure Profiles

Table 2 characterizes how each compressor degrades under increasing compression. The uncompressed baseline achieves 74% accuracy with 26% CFR (13 wrong answers, 2 non-terminations out of 50 prompts).

Three distinct degradation profiles emerge:

- **StreamingLLM** exhibits a cliff profile: accuracy remains stable at 80% through 0.625, then collapses to 14% at 0.875. Failures are predominantly quiet (wrong answers).
- **SnapKV** shows a catastrophic cliff at 0.875: from 66% accuracy at 0.750 to 0% accuracy with 49/50 prompts exhibiting looping behavior. This is the most signal-rich failure mode.

Table 3: Hazard predictor performance. Val = held-out 20% split (within-compressor). CV = leave-one-compressor-out (cross-compressor generalization). Trace detection = fraction of catastrophe traces where the predictor fires at least once before or at onset.

Metric	Val	CV (mean)
AUROC	0.945	0.898
F1	0.643	0.398
Recall @ 5% FPR	0.693	—
Recall @ 10% FPR	0.785	—
Trace detection rate	100% (22/22 traces)	
Mean lead time (tokens)	34.1 from onset	

- **ObservedAttention** degrades progressively from 74% to 12% with no sharp cliff. Failures are predominantly quiet wrong answers at all compression levels.

This diversity is critical: a compressor-agnostic controller must handle both sudden catastrophic failures (SnapKV looping) and gradual quality degradation (ObservedAttention wrong answers).

4.2 Hazard Predictor Performance

We train an XGBoost binary classifier to predict catastrophe within the next $H = 32$ tokens using a 41-dimensional per-token feature vector: 25 HALT logit features, 8 rolling statistics, repetition count, token position, and compression ratio. Training uses 800 traces (263K tokens, 13% positive rate) with leave-one-compressor-out cross-validation.

Feature importance. The top-5 features by gain are: `compression_ratio` (1645), `rep_count_sum_8` (1272), `rep_count` (926), `token_position` (322), and `h_alts_mean_8` (314). The predictor combines compression context, repetition detection, positional risk, and reasoning-quality signals — matching the hypothesized signal structure.

Per-compressor generalization. Cross-validation reveals uneven generalization: SnapKV (AUROC 0.97, F1 0.69) generalizes best because looping produces strong repetition signals. StreamingLLM (AUROC 0.89) and ObservedAttention (AUROC 0.87) are harder because their quiet failures (wrong answers, gradual drift) produce weaker signal contrast. Crucially, the predictor is strongest where CFR is highest.

Trace-level detection. Despite modest token-level recall (69% at 5% FPR), the predictor achieves 100% trace-level detection: every catastrophe trace fires at least one alarm token, which is sufficient for the controller's cumulative trigger mechanism.

Table 4: Controller CFR reduction at three operating points. CFR reduction = (baseline − controlled) / baseline. FP = false positive rate (triggers on non-catastrophe traces). Results on 800 traces (50 prompts × 16 configurations). "Excl. extreme" excludes SnapKV@0.875 where looping onset is immediate (<5 tokens) and no online controller can react.

	Conservative	Balanced	Aggressive
$\tau_{\text{low}}/\tau_{\text{high}}$	0.7 / 0.8	0.3 / 0.7	0.3 / 0.6
k (consecutive)	5	8	5
Overall CFR red.	36.9%	43.2%	47.7%
excl. extreme	50.8%	63.9%	70.5%
FP rate	2.3%	6.9%	25.2%
Per-compressor CFR reduction (balanced):			
StreamingLLM	52.4%	66.7%	76.2%
SnapKV	23.1%	27.7%	29.2%
excl. 0.875	53.3%	60.0%	66.7%
Obs. Attention	60.0%	64.0%	72.0%

4.3 Controller Evaluation

We evaluate the controller via offline simulation on all 800 traces. For each trace where the controller triggers (transitions to SAFE mode), the simulation checks whether the same prompt at the safe compression ratio ($r_{\text{safe}} = 0.0$, i.e., no compression) avoids catastrophe. Table 4 reports results for three operating points.

Main result. The balanced configuration reduces overall CFR by 43.2% (111→63 catastrophes) with only 6.9% false positive rate, meaning compression remains active for 93.1% of prompts. Excluding the extreme case of SnapKV@0.875 — where looping onset occurs within the first 5 tokens, making any online controller ineffective — CFR reduction rises to 63.9%.

Orthogonality. The controller exceeds 50% CFR reduction on all three compressors when excluding the extreme SnapKV case: StreamingLLM 66.7%, SnapKV 60.0% (at ratios $\leq$0.75), ObservedAttention 64.0%. This validates the compressor-agnostic design.

The controller is not "turning off compression." At 6.9% FP, the controller triggers on only 52 of 750 total prompts. Of the 698 non-triggered prompts, compression proceeds exactly as without the controller. The safety net has negligible

Table 5: Ablation study. Each row modifies one component of the balanced configuration. "Ceiling" = theoretical maximum prevention (always trigger on every trace). "Gap to ceiling" = what fraction of preventable catastrophes the balanced controller prevents.

Configuration	**CFR Red.%**	**FP Rate%**
Always-safe (ceiling)	61.3	100.0
Balanced (ours)	43.2	8.1
gap to ceiling	70.5% of max at 8.1% FP	
Random predictor	0.6	1.1
Trained predictor	43.2	8.1
Δ	+42.6pp from trained predictor	
No hysteresis ($k = 1$)	60.4	61.5
With hysteresis ($k = 8$)	43.2	8.1
tradeoff	-17pp prevention, -53pp FP	

overhead in the common case.

Per-budget analysis. Of 15 compressor-ratio configurations, 12 achieve $\geq$50% CFR reduction with the balanced config. The three exceptions are: SnapKV@0.250 (33%, only 3 baseline catastrophes — small-sample noise), SnapKV@0.875 (18%, immediate onset), and StreamingLLM@0.750 (60%, just above threshold). The controller is most effective at moderate-to-high compression (0.5–0.75) where failures are common but gradual enough to detect.

4.4 Ablation Study

We isolate the contribution of each controller component through four ablations (Table 5).

Always-safe ceiling. Triggering SAFE mode on every trace prevents 61.3% of catastrophes (111$\rightarrow$43). The remaining 38.7% represent model-intrinsic failures that occur even without compression. The balanced controller achieves 70.5% of this theoretical maximum while maintaining 8.1% false positive rate — the safety net is selective, not indiscriminate.

Trained predictor is essential. A random predictor with matched trigger probability achieves only 0.6% CFR reduction. The trained hazard predictor adds 42.6 percentage points, demonstrating that the ML component — not just the state machine architecture — drives the controller's effectiveness.

Hysteresis is critical for precision. Without hysteresis ($k = 1$: trigger on any single high-risk token), CFR reduction reaches 60.4% but at 61.5% false positive rate — the controller triggers on most prompts regardless of actual risk. Requiring $k = 8$ consecutive high-risk tokens reduces FP by 53 percentage points at a cost of 17 percentage points prevention. This is the key design tradeoff: the state machine converts weak per-token predictions into reliable per-trace triggers.

Threshold sensitivity. A sweep over 45 configurations ($\tau_{\text{low}} \in \{0.1, 0.3, 0.5, 0.7\}$, $\tau_{\text{high}} \in \{0.4, 0.6, 0.7, 0.8\}$, $k \in \{1, 3, 8\}$) reveals a smooth Pareto frontier from 17.1%/0.0%FP to 48.6%/100%FP with no cliff edges. The controller is robust to parameter choice — practitioners can tune for their risk appetite without fragile sensitivity.

4.5 Key Insight: Cumulative Detection

The most surprising finding is that the controller achieves 66.7% CFR reduction on StreamingLLM despite the hazard predictor having only 41% token-level recall at 5% FPR for that compressor. The state machine with $k = 8$ hysteresis accumulates marginal per-token signals: even if any individual token has low detection probability, requiring 8 consecutive high-risk tokens provides reliable per-trace triggering. This cumulative detection principle means the controller does not require a near-perfect per-token classifier — it needs only that catastrophe traces produce *some* elevated signal that accumulates over time.

4.6 Limitations

Immediate-onset failures. SnapKV at 0.875 compression produces looping within the first 5 tokens. No online controller can prevent failures with <5 token lead time. This limitation is inherent to online monitoring and applies to any reactive approach.

Quiet failures. The controller is weakest against wrong-answer failures that produce no detectable signal (no looping, no entropy spike, no repetition). These "quiet" failures account for the gap between our 43.2% overall reduction and the 61.3% ceiling.

Offline simulation. Our evaluation uses offline trace replay rather than live token-by-token generation with dynamic compression switching. The simulation assumes that switching to $r_{\text{safe}} = 0.0$ recovers the same behavior as never compressing, which may not hold if early compressed tokens have already corrupted the reasoning state.

Single model and task. All experiments use Qwen2.5-3B-Instruct on GSM8K. Generalization to other models, tasks, and scales remains to be validated, though the compressor-agnostic architecture and leave-one-compressor-out CV provide evidence of transferability within this setup.

5 Discussion

5.1 Why Cumulative Detection Works

The most important finding is not the controller's aggregate CFR reduction but *how* it achieves it. The per-token hazard predictor has modest recall (69% at 5% FPR overall, 41% for StreamingLLM), yet the controller achieves 100% trace-level detection and 63.9% CFR reduction (excluding immediate-onset cases). The gap between weak per-token performance and strong per-trace performance is entirely due to hysteresis: requiring $k = 8$ consecutive high-risk tokens before escalation.

This mechanism mirrors a well-known principle in signal processing: noisy detectors become reliable when integrated over time. A single token above threshold is often noise. Eight consecutive tokens above threshold is a trend. Catastrophe traces produce extended runs of elevated hazard probability—not because every individual token is high-confidence, but because the underlying failure mode (looping, entropy drift, repetition) creates a persistent signal that the predictor partially captures at each step. The state machine accumulates this partial signal into a reliable trigger.

This has a practical implication: the hazard predictor need not be a near-perfect per-token classifier. It needs only to produce *some* elevation in risk score during pre-catastrophe windows. This relaxes the accuracy requirement on the ML component and makes the system robust to moderate predictor degradation—a desirable property for safety-critical applications.

5.2 The Closed-Loop Paradigm

All KV-cache compressors to date operate in open loop: they score tokens, evict, and proceed without feedback. This is analogous to early network congestion control, where senders transmitted at a fixed rate without monitoring whether packets were arriving. TCP congestion control (Jacobson and Karels, 1988) solved that problem by closing the loop—adjusting the sending rate based on observed loss—and the analogy transfers directly to cache compression.

The wrapper architecture makes this concrete: kvguard does not compete with compressors, it *makes them safer*. Every advance in compression methods (better importance scoring, smarter eviction policies, new quantization schemes) automatically inherits the safety layer. This orthogonality is not incidental; it is the core architectural choice. A safety mechanism that is entangled with a specific compressor must be re-engineered whenever the compressor changes. A wrapper that monitors output quality and adjusts a single knob (compression ratio) remains valid regardless of the underlying compression strategy.

5.3 CFR as a Complementary Metric

Standard KV-cache compression evaluation reports average accuracy or perplexity under compression. This hides catastrophic failures: in our experiments, ObservedAttention at 0.625 compression reports 36% accuracy—but this single number conceals that 64% of prompts fail, many through looping or non-termination rather than graceful quality degradation. The Catastrophic Failure Rate separates *how often* compression causes complete failure from *how much* it degrades average quality, and the two tell different stories.

We do not argue that CFR should replace average accuracy; both are needed. But for safety-critical deployments—medical reasoning, financial analysis, agentic systems where a single looping generation can block a pipeline—CFR is the metric that matters. A compressor with 90% average accuracy and 2% CFR is safer than one with 92% average accuracy and 8% CFR, even though standard benchmarks would prefer the latter.

5.4 Limitations

Immediate-onset failures remain undetectable. SnapKV at 0.875 compression produces looping within the first 5 tokens—before any online monitor can accumulate sufficient signal. This is an inherent limitation of reactive monitoring: some compression configurations are so aggressive that failure is instantaneous. The practical mitigation is pre-generation screening (Gao et al., 2025): route known-aggressive configurations to full-cache inference and apply the controller only at moderate ratios where it can be effective. The two approaches are complementary.

Quiet failures produce weak signals. The gap between our 43.2% overall CFR reduction and the 61.3% always-safe ceiling is primarily due to wrong-answer failures that produce no detectable signal: no looping, no entropy spike, no repetition. These "quiet" failures corrupt the reasoning chain without overt behavioral markers. Detecting them likely requires deeper signals—attention patterns, internal representation drift, or semantic coherence monitoring—that go beyond the zero-cost logit features used in this work.

Offline simulation overstates live performance. Our evaluation assumes that switching to no compression mid-generation recovers the same behavior as never compressing. In practice, tokens generated under aggressive compression may have already corrupted the reasoning state, making recovery impossible regardless of subsequent compression policy. The reported CFR reduction is therefore an upper bound. Live token-by-token evaluation is needed to quantify the gap between simulation and reality.

Single model and task. All experiments use Qwen2.5-3B-Instruct on GSM8K. While leave-one-compressor-out cross-validation validates signal transferability across compression methods, generalization to other model families (Llama, Phi), scales (7B, 70B), and tasks (coding, document QA, agentic workflows) remains unvalidated. Larger models may produce richer logit signals, improving predictor performance; alternatively, they may fail in qualitatively different ways that the current feature set does not capture.

Fixed safe ratio. The controller currently falls back to $r_{\text{safe}} = 0$ (no compression), which maximizes recovery probability but sacrifices all memory savings during intervention. A more graduated approach—reducing compression from 0.75 to 0.50 rather than to 0—might preserve partial savings while still preventing catastrophe. This requires understanding the relationship between compression reduction and recovery probability, which our current dataset cannot quantify.

5.5 Future Work

Live token-by-token integration. The most immediate priority is validating the controller in a live autoregressive loop where compression ratio changes take effect at the next decoding step. This will reveal whether mid-generation compression switching recovers generation quality as the offline simulation assumes, and will quantify the true CFR reduction in deployment conditions.

Quiet failure detection. The current feature set is optimized for loud failures (looping, non-termination) that produce strong behavioral signals. Detecting quiet wrong-answer failures—where the model follows a plausible but incorrect reasoning path—likely requires attention-derived features (lookback ratio, attention entropy over receiver heads) or semantic coherence signals. These were deferred from the current scope but represent the most impactful extension for closing the gap to the always-safe ceiling.

Multi-model and multi-task evaluation. Extending to additional model families, scales, and reasoning tasks would strengthen generalization claims. Of particular interest is whether the feature importance ranking (compression ratio $>$ repetition $>$ position $>$ HALT alternatives) is model-specific or universal. If universal, a single predictor could serve as a general-purpose safety layer.

Adaptive safe ratio. Rather than a binary switch between base compression and no compression, an adaptive controller could search for the minimum compression relaxation needed to prevent catastrophe, preserving partial memory savings during intervention periods.

6 Conclusion

KV-cache compression enables long-context inference under memory constraints, but existing compressors operate in open loop: they evict tokens without feedback and never verify whether eviction degraded generation quality. We have shown that this open-loop approach produces catastrophic failures—looping, non-termination, and reasoning corruption—that standard average-accuracy metrics hide, affecting 44% of traces at moderate-to-aggressive compression ratios across three compressors.

We presented kvguard, a closed-loop controller inspired by TCP congestion control that wraps any existing KV-cache compressor. The system extracts a 42-dimensional per-token feature vector from the model's own logit distribution at near-zero cost, trains a hazard predictor that achieves 0.945 AUROC with 34-token mean lead time before catastrophe onset, and uses a hysteresis-based state machine to dynamically relax compression when danger is detected.

The controller reduces the Catastrophic Failure Rate by 43.2% overall (63.9% excluding immediate-onset cases) with only 6.9% false positive rate, exceeding 50% CFR reduction on all three compressors. Systematic ablations confirm that each component contributes independently: the trained predictor adds 42.6 percentage points over random, and hysteresis reduces false positives by 53 percentage points at a 17 percentage point prevention cost.

The key mechanism is cumulative detection: requiring k consecutive high-risk tokens converts a modest per-token classifier into a reliable per-trace trigger. This principle—that a weak detector integrated over time becomes a strong one—suggests that closing the loop on KV-cache compression does not require perfect per-token prediction, only persistent signal during pre-catastrophe windows. We believe this closed-loop paradigm will become a standard component of KV-cache management as compression is deployed in safety-critical applications.

References

Paul C. Bogdan, Uzay Macar, Neel Nanda, and Arthur Conmy. 2025. Thought anchors: Which LLM reasoning steps matter? *arXiv preprint arXiv:2506.19143*.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, and Wen Xiao. 2024. PyramidKV: Dynamic KV cache compression based on pyramidal information funneling. *arXiv preprint arXiv:2406.02069*.

Yuan Feng, Haoyu Guo, JunLin Lv, S. Kevin Zhou, and Xike Xie. 2025. Taming the fragility of KV cache eviction in LLM inference. *arXiv preprint arXiv:2510.13334*.

Wei Gao, Xinyu Zhou, Peng Sun, Tianwei Zhang, and Yonggang Wen. 2025. Rethinking key-value cache compression techniques for large language model serving. *arXiv preprint arXiv:2503.24000*.

Van Jacobson and Michael J. Karels. 1988. Congestion avoidance and control. In *Proceedings of the ACM SIGCOMM Conference on Communications Architectures and Protocols*, pages 314–329. ACM.

Haziq Mohammad Khalid, Athikash Jeyaganthan, Timothy Do, Yicheng Fu, Sean O'Brien, Vasu Sharma, and Kevin Zhu. 2025. ERGO: Entropy-guided resetting for generation optimization in multi-turn language models. In *Proceedings of the 2nd Workshop on Uncertainty Aware NLP (UncertaiNLP)*, pages 273–286. Association for Computational Linguistics.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. 2024. SnapKV: LLM knows what you are looking for before generation. In *Advances in Neural Information Processing Systems*, volume 37.

Adilet Metinov, Gulida M. Kudakeeva, Bolotbek uulu Nursultan, and Gulnara D. Kabaeva. 2024. Adaptive soft rolling KV freeze with entropy-guided recovery: Sublinear memory growth for efficient LLM inference. *arXiv preprint arXiv:2512.11221*.

Chen Qian, Dongrui Liu, Haochen Wen, Zhen Bai, Yong Liu, and Jing Shao. 2025. Demystifying reasoning dynamics with mutual information: Thinking tokens are information peaks in LLM reasoning. *arXiv preprint arXiv:2506.02867*.

Akshat Ramachandran, Marina Neseem, Charbel Sakr, Rangharajan Venkatesan, Brucek Khailany, and Tushar Krishna. 2025. ThinKV: Thought-adaptive KV cache compression for efficient reasoning models. *arXiv preprint arXiv:2510.01290*.

Ahmad Shapiro, Karan Taneja, and Ashok Goel. 2026. HALT: Hallucination assessment via log-probs as time series. *arXiv preprint arXiv:2602.02888*.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*.

Jing Xiong, Jianghan Shen, Fanghua Ye, Chaofan Tao, Zhongwei Wan, Jianqiao Lu, Xun Wu, Chuanyang Zheng, Zhijiang Guo, Min Yang, Lingpeng Kong, and Ngai Wong. 2025. UNComp: Can matrix entropy uncover sparsity? A compressor design from an uncertainty-aware perspective. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Fangyuan Xu, Tanya Goyal, and Eunsol Choi. 2024. RefreshKV: Updating small KV cache during long-form generation. *arXiv preprint arXiv:2411.05787*.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. 2023. H_2O: Heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems*, volume 36.