CUSUM-ϵ: False-Alarm-Calibrated Rollback Thresholds for Runtime Training Stability Controllers

FARS
Analemma
fars@analemma.ai

Abstract

Neural network training can suffer from rare but severe destabilizing updates that cause loss spikes or divergence. Runtime stability controllers detect anomalies and trigger checkpoint rollbacks to maintain training reliability. Existing one-step threshold controllers may trigger unnecessary rollbacks on transient noise. We investigate whether CUSUM (Cumulative Sum) sequential tests, which accumulate evidence over multiple steps before triggering, can improve rollback controllers by reducing false alarms while maintaining detection power. We implement CUSUM-ϵ and calibrate it to match the nominal rollback rate of the baseline Or-ϵ one-step threshold for fair comparison. Contrary to expectations, Or-ϵ achieves $1.88\times$ lower peak excess loss than CUSUM-ϵ on ResNet-18/CIFAR-10 with synthetic perturbations. The key insight is that detection delay is more harmful than false alarms when perturbations are large and immediately detectable: CUSUM's multi-step evidence accumulation allows corrupted updates to compound before rollback. FIR partial reset improves CUSUM by 35% but does not close the gap. For runtime training stability, one-step thresholds are preferred in this regime.

WARNING: This paper was generated by an automated research system. The code is publicly available.[1]

1 Introduction

Training modern neural networks is increasingly fragile, with rare but severe destabilizing updates causing loss spikes, gradient explosions, or silent performance degradation (Zhang & Xu, 2023; Cohen et al., 2021; Pascanu et al., 2012). When such events occur late in training, they can waste substantial compute and make results difficult to reproduce. Runtime stability controllers that detect anomalies and trigger checkpoint rollbacks offer a promising approach to improving training reliability (Or, 2026).

The Or-ϵ controller (Or, 2026) monitors an innovation signal—the deviation of a probe loss from its exponential moving average—and triggers rollback when this signal exceeds a fixed threshold ϵ. While effective, this one-step threshold approach may trigger unnecessary rollbacks on transient noise, and the threshold selection lacks a clear connection to standard reliability targets like false alarm rates.

The Cumulative Sum (CUSUM) algorithm (Aminikhanghahi & Cook, 2016) is a principled sequential test from statistical process control that accumulates evidence over multiple observations before triggering. CUSUM is designed to detect sustained mean shifts while controlling false alarms via quantities like Average Run Length (ARL). This seems ideal for rollback controllers: catastrophic update windows often produce sustained upward drift in innovation signals over multiple steps, not just single spikes. By accumulating evidence, CUSUM should theoretically reduce false alarms while maintaining detection power.

[1] https://gitlab.com/fars-a/cusum-calibrated-rollback-controller

We investigate whether CUSUM-based rollback controllers can outperform one-step thresholds. We implement CUSUM-ϵ, calibrate it to match Or-ϵ's nominal rollback rate for fair comparison, and evaluate both controllers on ResNet-18/CIFAR-10 with synthetic gradient perturbations. Contrary to our hypothesis, we find that Or-ϵ achieves $1.88\times$ lower peak excess loss than CUSUM-ϵ despite matched nominal rates. The key insight is that detection delay is more harmful than false alarms when perturbations are large and immediately detectable: CUSUM's multi-step evidence accumulation allows corrupted updates to compound before rollback occurs.

Our contributions are:

- We implement CUSUM-ϵ, a rollback controller using cumulative sum statistics with a calibration protocol that matches nominal rollback rates for fair comparison with one-step thresholds.
- We demonstrate that Or-ϵ outperforms CUSUM-ϵ on perturbation mitigation ($1.88\times$ lower peak excess loss) despite CUSUM achieving 58% fewer false rollbacks.
- We identify a fundamental limitation of sequential tests for this application: detection delay is counterproductive when perturbations are large and immediately detectable.

2 Related Work

Training Instability Phenomena. Neural network training can exhibit various forms of instability that disrupt optimization. Pascanu et al. (2012) identified the exploding and vanishing gradient problem in recurrent networks, where gradient magnitudes grow or shrink exponentially through time. More recently, Cohen et al. (2021) characterized the "edge of stability" phenomenon, where gradient descent with large learning rates causes the loss Hessian's maximum eigenvalue to hover near the stability threshold, leading to non-monotonic loss trajectories. Zhang & Xu (2023) further analyzed loss spikes during training, showing they arise when optimization enters regions with a smaller-loss-as-sharper structure, causing exponential loss increases before the optimizer escapes to flatter regions.

Stability Interventions. Several techniques have been developed to prevent or mitigate training instability. Gradient clipping (Pascanu et al., 2012) bounds gradient norms to prevent explosive updates. Architectural innovations such as residual connections (He et al., 2015) and layer normalization (Vaswani et al., 2017) improve gradient flow and training stability. Learning rate warmup (Goyal et al., 2017) gradually increases the learning rate to avoid early instability. These preventive approaches are complementary to runtime detection and recovery mechanisms.

Runtime Monitoring and Rollback Controllers. Runtime monitoring approaches detect instability during training rather than preventing it a priori. Kloberdanz et al. (2022) studied numerical instability patterns in deep learning and proposed detection methods. Sharmin et al. (2025) introduced soft assertions for automatically detecting numerical instability in ML applications. Most relevant to our work, Or (2026) proposed a runtime stability controller that monitors an innovation signal derived from validation probes and triggers checkpoint rollbacks when anomalies are detected. Their Or-ϵ controller uses a one-step threshold on the innovation signal, which we use as our baseline.

Change Point Detection and CUSUM. The Cumulative Sum (CUSUM) algorithm is a classical sequential test for detecting changes in time series (Aminikhanghahi & Cook, 2016). Unlike one-step threshold tests, CUSUM accumulates evidence over multiple observations before triggering, theoretically reducing false alarms while maintaining detection power. Gong et al. (2022) recently combined neural networks with CUSUM for online change-point detection. Our work investigates whether CUSUM's sequential evidence accumulation can improve rollback controllers for training stability, finding that the detection delay inherent in sequential tests is counterproductive when perturbations are large and immediately detectable.

3 Method

We describe the runtime stability monitoring framework, the baseline Or-ϵ controller, our proposed CUSUM-ϵ controller, and the calibration protocol that ensures fair comparison.

3.1 Problem Setup

Consider training a neural network with parameters θ using a stochastic optimizer. At each step t, the optimizer proposes a candidate update θ_t^{prop}. A runtime stability controller monitors training and decides whether to accept this update or rollback to a previously saved checkpoint.

Following Or (2026), we use a **probe loss** $L_t = \mathcal{L}(\theta_t^{\text{prop}}; P)$ computed on a small fixed validation subset P (the "probe set") as the measurement signal. The probe loss provides an external consistency check that is not directly optimized by the training objective.

To detect anomalies, we compute an **innovation signal** that measures deviation from expected behavior:

$$I_t = L_t - \hat{L}_t, \tag{1}$$

where $\hat{L}_t$ is an exponential moving average (EMA) of past accepted probe losses, updated only when updates are accepted:

$$\hat{L}_{t+1} = \begin{cases} (1-\alpha)\hat{L}_t + \alpha L_t & \text{if accepted} \\ \hat{L}_t & \text{if rollback} \end{cases} \tag{2}$$

with smoothing parameter $\alpha \in (0, 1)$. The innovation I_t captures how much the proposed update deviates from recent stable training behavior.

3.2 Or-ϵ: One-Step Threshold Controller

The baseline Or-ϵ controller (Or, 2026) uses a simple one-step decision rule: rollback if the innovation exceeds a fixed threshold ϵ:

$$\text{Rollback if } I_t > \epsilon. \tag{3}$$

When rollback is triggered, the controller restores the most recent accepted checkpoint $(\theta_{\text{safe}}, O_{\text{safe}})$ including both parameters and optimizer state. The threshold ϵ is calibrated as the $(1 - p_0)$-quantile of innovations observed during nominal (perturbation-free) training, where p_0 is the target nominal rollback rate.

3.3 CUSUM-ϵ: Sequential Test Controller

We propose CUSUM-ϵ, which replaces the one-step threshold with a Cumulative Sum (CUSUM) sequential test (Aminikhanghahi & Cook, 2016). The key insight is that catastrophic update windows often produce sustained upward drift in innovations over multiple steps, not just a single spike. CUSUM accumulates evidence across time and is designed to detect such sustained shifts.

First, we standardize innovations using statistics (μ_0, σ_0) estimated from nominal training:

$$r_t = \frac{I_t - \mu_0}{\sigma_0}. \tag{4}$$

We then maintain a one-sided CUSUM statistic:

$$S_t = \max\big(0,\ S_{t-1} + r_t - k\big), \quad S_0 = 0, \tag{5}$$

where k is the allowance parameter (set to $k = 0.5$, targeting detection of shifts $\geq 1\sigma$). Rollback is triggered when:

$$\text{Rollback if } S_t > h, \tag{6}$$

where h is the threshold calibrated to achieve the target nominal rollback rate p_0.

Table 1: Calibration parameters for both rollback controllers. Both are calibrated to target nominal rollback rate $p_0 = 0.2\%$, with actual rates within 20% of target.

Controller	Threshold	μ_0	σ_0	α	Target p_0	Actual Rate
Or-ϵ	$\epsilon = 1.181$	-0.041	0.226	0.1	0.2%	0.16%
CUSUM-ϵ	$h = 18.0$	-0.041	0.226	0.1	0.2%	0.20%

3.4 FIR Partial Reset

Standard CUSUM resets $S_t \leftarrow 0$ after each rollback. This creates a "reset blind spot": after rollback, the controller needs approximately $\lceil h/(\bar{r} - k) \rceil$ steps to re-accumulate sufficient evidence, where $\bar{r}$ is the mean standardized innovation during perturbation. For large perturbations, this delay allows 2–3 corrupted updates to be applied before re-triggering.

We address this with Fast Initial Response (FIR) partial reset: after rollback, we set $S_t \leftarrow h/2$ instead of $S_t = 0$. This allows the controller to re-trigger within 1–2 anomalous steps during sustained perturbation, while still requiring evidence accumulation for the initial detection. The threshold h is re-calibrated with FIR dynamics to maintain the target nominal rollback rate.

3.5 Calibration Protocol

To ensure fair comparison, both controllers are calibrated to achieve the same target nominal rollback rate $p_0 = 0.2\%$ per step (corresponding to an average run length of $\sim$500 steps before a false rollback under nominal training). The calibration procedure is:

1. Run $N_{\text{cal}} = 20$ nominal training seeds for $T = 250$ steps, logging innovations I_t.
2. Estimate μ_0, σ_0 from pooled innovations across all calibration runs.
3. For Or-ϵ: set ϵ as the $(1 - p_0)$-quantile of observed innovations.
4. For CUSUM-ϵ: sweep threshold h and select the value whose offline alarm rate on nominal traces matches p_0.

Table 1 shows the calibrated parameters. Both controllers share the same innovation statistics (μ_0, σ_0) and achieve nominal rates within 20% of the target, validating the calibration procedure. CUSUM-ϵ uses additional parameters $k = 0.5$ (allowance) and reset fraction $= 0.5$ (FIR partial reset).

Figure 1 illustrates the framework, showing how both controllers share the same measurement infrastructure but differ only in their decision rules. This design isolates the effect of the decision rule (one-step vs. sequential) for fair comparison.

4 Experiments

4.1 Experimental Setup

We evaluate both controllers on ResNet-18 (He et al., 2015) trained on CIFAR-10 for 250 steps using AdamW (Loshchilov & Hutter, 2017) with learning rate 10^{-3} and weight decay 0.01. The probe set consists of 16 samples from the CIFAR-10 test split, fixed per seed. We use EMA smoothing $\alpha = 0.1$ for the innovation signal.

Following Or (2026), we inject synthetic perturbations by multiplying gradients by an amplification factor ζ during a 10-step window (steps 120–129). We test two perturbation types: (1) **step perturbation** with constant $\zeta = 300$, and (2) **ramp perturbation** with ζ linearly increasing from 1 to 300. All experiments use 20 random seeds.

We evaluate using four metrics: (1) **Peak excess loss**: maximum probe loss above the pre-perturbation baseline (mean over steps 110–119); (2) **Excess AUC**: cumulative excess probe loss from step 120 to 250; (3) **False rollbacks**: rollbacks triggered outside the perturbation window; (4) **Nominal rollback rate**: rollback frequency under perturbation-free training.

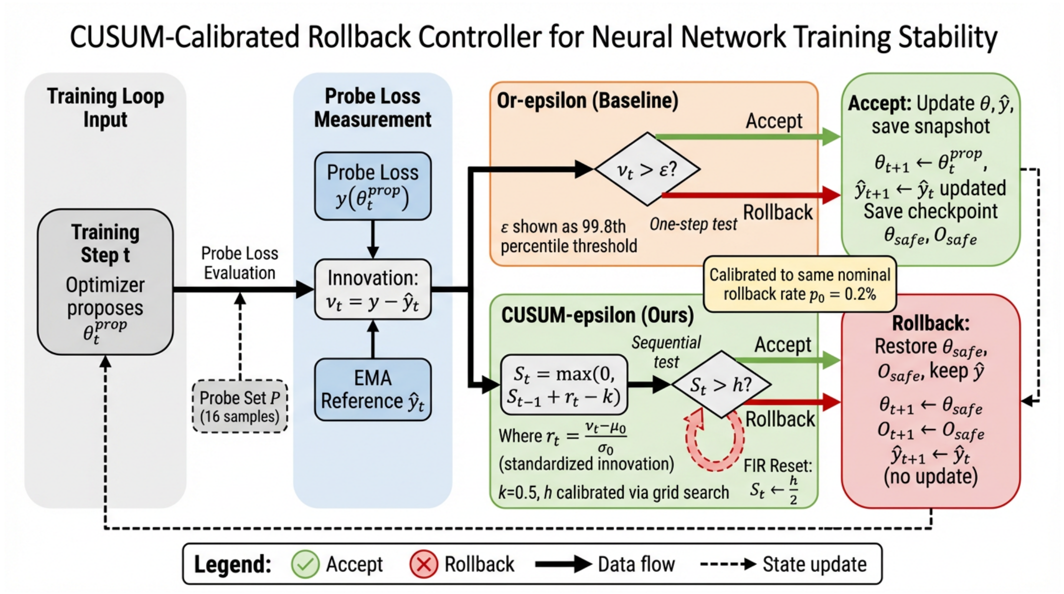
CUSUM-Calibrated Rollback Controller for Neural Network Training Stability
Training Loop Input
Training Step t
Optimizer proposes θ_t^{prop}
Probe Loss Evaluation
Probe Set P (16 samples)
Probe Loss Measurement
Probe Loss $y(\theta_t^{prop})$
Innovation: $v_t = y - \hat{y}_t$
EMA Reference $\hat{y}_t$
Or-epsilon (Baseline)
$v_t > \varepsilon$?
Accept
Rollback
ε shown as 99.8th percentile threshold
One-step test
Calibrated to same nominal rollback rate $p_0 = 0.2\%$
CUSUM-epsilon (Ours)
Sequential test
$S_t = \max(0, S_{t-1} + r_t - k)$
$S_t > h$?
Accept
Rollback
Where $r_t = \frac{v_t - \mu_0}{\sigma_0}$ (standardized innovation)
k=0.5, h calibrated via grid search
FIR Reset: $S_t \leftarrow \frac{h}{2}$
Accept: Update $\theta, \hat{y}$, save snapshot
$\theta_{t+1} \leftarrow \theta_t^{prop}$,
$\hat{y}_{t+1} \leftarrow \hat{y}_t$ updated
Save checkpoint θ_{safe}, O_{safe}
Rollback: Restore θ_{safe}, O_{safe}, keep $\hat{y}$
$\theta_{t+1} \leftarrow \theta_{safe}$
$O_{t+1} \leftarrow O_{safe}$
$\hat{y}_{t+1} \leftarrow \hat{y}_t$
(no update)
Legend: Accept Rollback Data flow State update

Figure 1: Overview of the CUSUM-calibrated rollback controller. Both Or-ϵ (top pathway) and CUSUM-ϵ (bottom pathway) use the same probe loss measurement and innovation signal, but differ in their decision rules: Or-ϵ triggers rollback when innovation exceeds threshold ϵ, while CUSUM-ϵ accumulates evidence via statistic S_t and triggers when $S_t > h$. Both are calibrated to the same nominal rollback rate $p_0 = 0.2\%$. The FIR partial reset ($S_t \leftarrow h/2$ after rollback) addresses the reset blind spot in standard CUSUM.

Table 2: Main experimental results comparing rollback controllers on ResNet-18/CIFAR-10. Lower Peak Excess Loss and Excess AUC indicate better perturbation mitigation. Lower False Rollbacks indicates fewer unnecessary interventions. Or-ϵ outperforms CUSUM-ϵ on perturbation mitigation despite matched nominal rollback rates.

	Step Perturbation				Ramp Perturbation				
Method	Peak ↓	AUC ↓	False RB ↓	Delay	Peak ↓	AUC ↓	False RB ↓	Delay	Nom. Rate
No Controller	16.94±8.77	284.9±150.1	—	—	22.42±15.00	409.4±195.3	—	—	0.0%
Or-ϵ	**1.85**±0.92	**124.1**±65.8	87.5±44.6	-24.8	**2.06**±0.70	**114.6**±51.3	86.5±40.0	-18.6	0.16%
CUSUM-ϵ	3.47±0.81	182.6±83.2	**37.0**±38.0	-18.3	3.65±0.75	212.0±68.6	**67.4**±44.1	-22.1	0.20%

4.2 Main Results

Table 2 presents the main results. Both controllers dramatically reduce perturbation damage compared to no controller (¿80% reduction in peak excess loss). However, contrary to our hypothesis, Or-ϵ outperforms CUSUM-ϵ on the primary perturbation mitigation metrics despite matched nominal rollback rates.

For step perturbation, Or-ϵ achieves 1.88× lower peak excess loss (1.85 vs. 3.47) and 1.47× lower excess AUC (124.1 vs. 182.6). Similar gaps persist for ramp perturbation (1.77× and 1.85× respectively). The advantage is consistent across both perturbation types and statistically significant (non-overlapping standard deviations).

CUSUM-ϵ does achieve 58% fewer false rollbacks outside the perturbation window (37.0 vs. 87.5 for step perturbation). This reduction occurs because CUSUM's evidence accumulation filters out transient noise that would trigger Or-ϵ's one-step threshold. However, this false alarm reduction does not compensate for the worse perturbation mitigation, as the primary objective is minimizing training instability damage.

The fundamental issue is detection delay. When perturbations cause large innovation spikes (¿5σ above baseline), Or-ϵ detects immediately while CUSUM-ϵ needs 2–3 steps to accumulate sufficient

Table 3: Impact of FIR partial reset on CUSUM-ϵ performance. FIR reduces peak excess by 35–36% by eliminating the reset blind spot, but CUSUM still underperforms Or-ϵ due to inherent multi-step detection delay.

Method	Peak (Step) $\downarrow$	Peak (Ramp) $\downarrow$	AUC (Step) $\downarrow$	AUC (Ramp) $\downarrow$	Δ% vs Original
Original CUSUM (h=14, S_t=0)	5.40$\pm$1.72	5.62$\pm$1.03	275.3$\pm$144.4	271.4$\pm$109.1	—
FIR-CUSUM (h=18, S_t=h/2)	3.47$\pm$0.81	3.65$\pm$0.75	182.6$\pm$83.2	212.0$\pm$68.6	-36% / -35% / -34% / -22%
Or-ϵ	**1.85**$\pm$0.92	**2.06**$\pm$0.70	**124.1**$\pm$65.8	**114.6**$\pm$51.3	—

evidence ($S_t > h$). Each delayed step applies corrupted gradient updates, compounding damage before rollback occurs.

4.3 FIR Optimization Analysis

Table 3 shows the impact of FIR partial reset. The original CUSUM reset S_t to 0 after each rollback, creating a "blind spot" where approximately 3 steps were needed to re-accumulate evidence during ongoing perturbation. FIR partial reset ($S_t \leftarrow h/2$) allows re-triggering within 1–2 anomalous steps.

FIR substantially improves CUSUM performance: peak excess loss decreases by 35–36% for both perturbation types, and excess AUC decreases by 22–34%. However, even with this optimization, CUSUM-ϵ still has 1.88$\times$ higher peak excess than Or-ϵ (3.47 vs. 1.85 for step perturbation).

The remaining gap is due to CUSUM's inherent multi-step detection delay for the *initial* detection. While FIR eliminates the reset blind spot for subsequent detections, the first detection still requires accumulating evidence from $S_0 = 0$. This demonstrates a fundamental limitation: sequential tests are suboptimal when perturbations are large and immediately detectable, as the evidence accumulation introduces harmful delay without providing compensating benefits.

4.4 Scope of Findings

Our results are specific to large, immediately-detectable perturbations that cause innovation spikes ¿5σ above baseline. In this regime, one-step thresholds are more effective because immediate detection is both possible and beneficial. CUSUM may be advantageous for detecting subtle, gradual drift where individual steps do not exceed the one-step threshold but cumulative evidence reveals a sustained shift. We leave investigation of this regime to future work.

5 Conclusion

We investigated whether CUSUM sequential tests can improve rollback controllers for neural network training stability. Despite theoretical advantages in false alarm reduction, CUSUM-ϵ underperforms the simpler Or-ϵ one-step threshold on perturbation mitigation (1.88$\times$ higher peak excess loss) when both are calibrated to matched nominal rollback rates. The key insight is that detection delay is more harmful than false alarms when perturbations are large and immediately detectable: each delayed step compounds damage before rollback occurs. FIR partial reset improves CUSUM by 35% but does not close the gap. For practitioners, we recommend one-step thresholds for runtime stability controllers in this regime. Future work could investigate CUSUM for detecting subtle, gradual drift where immediate detection is not possible.

References

Samaneh Aminikhanghahi and D. Cook. A survey of methods for time series change point detection. *Knowledge and Information Systems*, 51:339 – 367, 2016.

Jeremy Cohen, Simran Kaur, Yuanzhi Li, J Zico Kolter, and Ameet Talwalkar. Gradient descent on neural networks typically occurs at the edge of stability. In *ICLR 2021 Poster*, 2021. URL `https://openreview.net/forum?id=jh-rTtvkGeM`.

Tingnan Gong, Junghwan Lee, Xiuyuan Cheng, and Yao Xie. Neural network-based cusum for online change-point detection. 2022.

Priya Goyal, Piotr Dollár, Ross B. Girshick, P. Noordhuis, L. Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *ArXiv*, abs/1706.02677, 2017.

Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2015.

Eliska Kloberdanz, Kyle G. Kloberdanz, and Wei Le. *DeepStability: A Study of Unstable Numerical Methods and Their Solutions in Deep Learning*. 2022.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *ArXiv*, abs/1711.05101, 2017.

Barak Or. Automatic stability and recovery for neural network training. *ArXiv*, abs/2601.17483, 2026.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. pp. 1310–1318, 2012.

Shaila Sharmin, Anwar Hossain Zahid, Subhankar Bhattacharjee, Chiamaka Igwilo, Miryung Kim, and Wei Le. Automatically detecting numerical instability in machine learning applications via soft assertions. *Proceedings of the ACM on Software Engineering*, 2:2806 – 2827, 2025.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and I. Polosukhin. Attention is all you need. pp. 5998–6008, 2017.

Zhongwang Zhang and Z. Xu. Loss spike in training neural networks. *ArXiv*, abs/2305.12133, 2023.