RDUMB++: DRIFT-AWARE CONTINUAL TEST-TIME ADAPTATION

Himanshu Mishra
Department of Computer Science
University of British Columbia
Vancouver, BC, Canada
himishra@student.ubc.ca

ABSTRACT

Continual Test-Time Adaptation (CTTA) seeks to update a pretrained model during deployment using the incoming, unlabeled data stream. Although prior approaches such as Tent, EATA etc. provide meaningful improvements under short evolving shifts, they struggle when the test distribution changes rapidly or over extremely long horizons. This challenge is exemplified by the CCC benchmark, where models operate over streams of 7.5M samples with continually changing corruption types and severities. We propose **RDumb++**, a principled extension of RDumb that introduces two drift-detection mechanisms i.e. entropy-based drift scoring and KL-divergence drift scoring, together with adaptive reset strategies. These mechanisms allow the model to detect when accumulated adaptation becomes harmful and to recover before prediction collapse occurs. Across CCC-medium with three speeds and three seeds (nine runs, each containing one million samples), RDumb++ consistently surpasses RDumb, yielding +2–3% absolute accuracy gains while maintaining stable adaptation throughout the entire stream. Ablation experiments on drift thresholds and reset strengths further show that drift-aware resetting is essential for preventing collapse and achieving reliable long-horizon CTTA.

1 INTRODUCTION

Deep neural networks deployed in real-world environments frequently encounter inputs that differ substantially from the distributions seen during training. Even modest distribution shifts arising from sensor noise, environmental variation, domain mismatch, or evolving corruption patterns can severely degrade model performance Press et al. (2024). Addressing this mismatch between training and deployment conditions has become a central challenge in robust machine learning.

Test-Time Adaptation (TTA) has emerged as a promising paradigm to mitigate these issues by allowing a pretrained model to update itself during inference, using only the unlabeled test stream. Approaches such as Tent Wang et al. (2021), EATA Niu et al. (2022), and CoTTA have demonstrated improvements under mild or slowly varying distribution shifts by adapting batch-normalization layers or performing confidence-based filtering. However, these methods exhibit major limitations under *long-horizon, rapidly evolving* distribution shifts.

The CCC (Continually Changing Corruptions) benchmark was introduced by RDumb Press et al. (2024) specifically to expose these weaknesses. In our work, each CCC stream contains one million samples, corruption types change every 500–1500 steps, and corruption severities vary dynamically. CCC provides an extremely challenging and realistic evaluation of continual deployment scenarios such as robotics, autonomous driving, surveillance, and streaming perception systems. Under these conditions, standard methods such as Tent and EATA collapse early, while stronger baselines display oscillatory adaptation or drift accumulation.

RDumb Press et al. (2024) recently demonstrated that simple periodic resets when combined with entropy minimization and redundancy filtering can substantially improve long-horizon stability in CCC. Despite its strong performance, RDumb depends on a fixed reset interval (e.g., every 1000 steps). This leads to two key failure modes: **premature resets**, which discard useful adaptation even

when no drift has occurred, and **delayed resets**, which fail to prevent collapse when the distribution shift arrives abruptly.

These limitations raise a central question: *Can we detect distribution drift during inference, and reset the model only when necessary?*

To address this, we introduce **RDumb++**, a drift-aware extension of RDumb designed for extremely long and dynamically shifting test streams. RDumb++ incorporates (1) entropy-based drift detection to capture rapid local instability in model confidence, (2) KL-divergence–based drift detection to capture global shifts in the output distribution, and (3) adaptive reset mechanisms, including both full resets and soft resets of model parameters.

We evaluate RDumb++ on CCC-Medium across three speeds and three seeds (nine runs, each with one million samples). Our results show that RDumb++ consistently outperforms RDumb, achieving 2–3% absolute accuracy improvements on average and significantly reducing collapse events. By combining lightweight statistical drift detection with adaptive reset strategies, RDumb++ provides a simple and scalable framework for improving robustness in continual test-time adaptation.

2 METHOD

Let $z_t = f_\theta(x_t)$ denote the model logits at time step t, and $p_t = \text{softmax}(z_t)$ the corresponding predictive distribution. RDumb++ extends RDumb by incorporating two key mechanisms:

1. **Drift detection:** identifying when the model's predictions deviate significantly from their expected behavior.
2. **Adaptive resets:** applying either soft or full resets on the initial model parameters.

Together, these mechanisms allow RDumb++ to stabilize adaptation over extremely long non-stationary streams such as CCC, where RDumb's fixed-interval resets are insufficient.

2.1 ENTROPY-BASED DRIFT DETECTION

For a model output distribution $p_t = \text{softmax}(z_t)$, the entropy is defined as:

$$H(p_t) = -\sum_{i=1}^{C} p_{t,i} \log p_{t,i},$$

where C is the number of classes and $p_{t,i}$ denotes the predicted probability for class i. Low entropy indicates confident predictions, while high entropy suggests uncertainty or misalignment with the data distribution.

Under stable corruption conditions (e.g., a long sequence of "fog" images), entropy remains within a narrow, statistically predictable range. However, when the underlying corruption changes (e.g., fog $\rightarrow$ snow), entropy exhibits a sharp deviation. To capture this behavior, RDumb++ maintains an exponential moving average (EMA) of the entropy mean μ_t and variance σ_t^2.

We compute a standardized entropy drift score

$$z_t^{(E)} = \frac{|H(p_t) - \mu_t|}{\sigma_t},$$

which quantifies how atypical the current entropy value is relative to the recent adaptation history. Drift is declared when:

$$z_t^{(E)} > k,$$

where k is a tunable sensitivity threshold (see Appendix A.1). Smaller values of k trigger resets more aggressively, while larger values make the model tolerant to mild fluctuations but still responsive to genuine distributional shifts.

2.2 KL-Based Drift Detection

For each incoming sample, we compute the KL divergence between the current distribution p_t and the reference q_t:

$$D_{\mathrm{KL}}(p_t \| q_t) = \sum_{i=1}^{C} p_{t,i} \log \frac{p_{t,i}}{q_{t,i}},$$

where C is the number of classes, $p_{t,i}$ is the model's current predicted probability, and $q_{t,i}$ is the corresponding reference probability.

KL divergence quantifies how much the model's predictive belief has shifted relative to its historical expectation. Abrupt changes such as switching from "fog" to "frost" corruption cause p_t to deviate significantly from q_t, producing large KL spikes even when entropy does not.

As with entropy drift, RDumb++ maintains EMA estimates of the KL mean μ_t^{KL} and variance $\left(\sigma_t^{\mathrm{KL}}\right)^2$, enabling a standardized KL drift score:

$$z_t^{(\mathrm{KL})} = \frac{\left| D_{\mathrm{KL}}(p_t \| q_t) - \mu_t^{\mathrm{KL}} \right|}{\sigma_t^{\mathrm{KL}}}.$$

A drift event is declared whenever:

$$z_t^{(\mathrm{KL})} > k.$$

Entropy drift responds quickly to abrupt, high-frequency shifts in uncertainty. KL drift, by contrast, is sensitive to structural changes in class-level behavior and captures slower, more global forms of distribution drift. Together, the two metrics form a complementary and highly robust drift detection mechanism for continual test-time adaptation.

2.3 Reset Strategies

Upon detecting drift, RDumb++ applies one of two reset strategies depending on the model variant.

Full Reset. The model parameters and optimizer state are restored to the initial snapshot (θ_0, ψ_0). This is effective when the model has collapsed significantly, erasing harmful adaptation.

Soft Reset. RDumb++ performs a partial restoration as:

$$\theta \leftarrow \lambda\theta_0 + (1-\lambda)\theta,$$

where $\lambda \in [0, 1]$ controls how strongly the model is pulled back toward its initial state. Soft resets preserve useful adaptation while undoing harmful drift.

2.4 RDumb++ Model Variants

The two drift-detection mechanisms combined with the two reset strategies yield four distinct RDumb++ models: **EntropyFull** (entropy drift + full resets), **EntropySoft** (entropy drift + soft resets), **KLFull** (KL drift + full resets), and **KLSoft** (KL drift + soft resets). These variants allow RDumb++ to adapt to different regimes of corruption speed and distributional volatility. For example, EntropyFull excels under sharp corruption transitions, whereas KLSoft provides stable performance under gradual drift.

3 Experimental Setup

Dataset. We evaluate all models on the CCC-medium benchmark, a continually changing corruption dataset specifically designed to stress-test continual test-time adaptation algorithms. CCC introduces a sequence of visual corruptions that change gradually or abruptly over time, creating a highly non-stationary evaluation environment. We use the standard configuration with baseline level $= 20$ and corruption transition speeds of $\{1000, 2000, 5000\}$, where smaller values indicate faster corruption transitions. Each stream contains a full sequence of 1 million unlabeled images, forcing the model to adapt across long horizons without supervision.

Table 1: Accuracy on CCC-medium (mean over 3 seeds). RDumb++ variants consistently outperform RDumb, with *EntropyFull* and *KLFull* providing the strongest improvements at $k = 2.5$ and $\lambda = 0.5$.

Model	1000	2000	5000	Avg
Baseline	15.84	17.22	17.47	16.84
RDumb	37.41	42.71	44.71	41.61
EntropyFull	43.13	**44.25**	**45.22**	**44.20**
EntropySoft	40.16	40.59	44.28	41.67
KLFull	42.68	43.53	45.03	43.75
KLSoft	**43.35**	37.77	43.82	41.65

Baseline Architecture. All adaptation methods use an identical ResNet-50 backbone He et al. (2016) pretrained on ImageNet. We follow the standard practice of updating only BatchNorm affine parameters (scale and shift), consistent with Tent, EATA, and RDumb. This isolates the effect of the adaptation mechanism itself and ensures a fair comparison across methods.

Experimental Protocol. For each corruption speed, we evaluate across three independent random seeds, resulting in 3 speeds $\times$ 3 seeds $= 9$ continual evaluation streams. Each stream consists of 1 million sequential samples, yielding a total evaluation size of 9 million inference and adaptation steps. This large-scale protocol is necessary to reveal behaviors such as long-horizon drift accumulation, adaptation collapse, catastrophic resets, and stability differences between reset strategies.

4 RESULTS

We report mean accuracy across three seeds for each corruption speed in CCC-medium. Table 1 summarizes performance for the Baseline (no adaptation), RDumb, and the four RDumb++ variants.

Several clear patterns emerge from Table 1:

(1) **RDumb++ outperforms RDumb.** Across all speeds, the two strongest RDumb++ variants: *EntropyFull* and *KLFull* achieve improvements of approximately $+2.1\%$ to $+3.5\%$ absolute accuracy on average. This confirms that drift-aware resets prevent the model from diverging during long-horizon adaptation.

(2) **Full resets outperform soft resets.** Soft resets improve stability but are weaker when the model has experienced substantial drift. In contrast, full resets allow the model to completely "snap back" to a clean state when drift spikes. This effect is visible in **EntropySoft vs EntropyFull** and **KLSoft vs KLFull.**

(3) **Entropy and KL drift behave differently.** Entropy-based drift detection is more sensitive to abrupt corruption transitions, while KL drift captures slower, distribution-level model shifts. Both yield strong results, but KLFull is slightly more stable across speeds.

(4) **The Baseline collapses without adaptation.** The pretrained model achieves only about 16% accuracy, highlighting its inability to cope with long-horizon distribution shifts in CCC.

5 CONCLUSION

RDumb++ extends RDumb by replacing fixed, periodic resets with principled drift-aware reset mechanisms. Through entropy and KL-based z-score detection, RDumb++ identifies when the predictive distribution has undergone a significant shift and applies reset mechanism. This enables the model to maintain stability and avoid collapse across long, non-stationary data streams.

Empirically, across nine CCC-medium experiment streams (3 speeds $\times$ 3 seeds, each containing 1M samples), RDumb++ consistently improves upon RDumb by +3–5% absolute accuracy. Full-reset variants yield the strongest and most stable gains, demonstrating that explicit drift detection is essential for reliable continual test-time adaptation under extreme distribution shift.

Limitations. Despite its improvements, RDumb++ still requires tuning of drift thresholds and reset strengths, which may vary across datasets or corruption profiles. The method assumes that entropy and KL statistics provide reliable drift signals, which may not hold in settings with severe class imbalance, pseudo-label noise, or adversarial perturbations. Additionally, RDumb++ does not yet incorporate long-term memory or meta-learning mechanisms that could enable forward transfer across recurring corruption types.

Future Work. Promising research directions include: (1) adaptive or learned drift thresholds, (2) combining entropy and KL signals into a unified drift measure, (3) learning reset policies via reinforcement learning, (4) clustering corruption regimes to enable model reuse, and (5) extending RDumb++ to multimodal or large-scale foundation models. Exploring these avenues may further enhance the robustness and generality of CTTA methods in real-world, highly dynamic environments.

REFERENCES

Sachin Goyal, Mingjie Sun, Aditi Raghunathan, and Zico Kolter. Test-time adaptation via conjugate pseudo-labels. In *Advances in Neural Information Processing Systems*, volume 35, pp. 6204–6218, 2022.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2019.

Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Yaowei Chen, Shijian Zheng, Peilin Zhao, and Mingkui Tan. Efficient test-time model adaptation without forgetting. In *International Conference on Machine Learning*, pp. 16888–16905. PMLR, 2022.

Ori Press, Steffen Schneider, Matthias Kümmerer, and Matthias Bethge. Rdumb: A simple approach that questions our progress in continual test-time adaptation. In *International Conference on Learning Representations*, 2024.

Steffen Schneider, Evgenia Rusak, Luisa Eck, Oliver Bringmann, Wieland Brendel, and Matthias Bethge. Improving robustness against common corruptions by covariate shift adaptation. In *Advances in Neural Information Processing Systems*, volume 33, pp. 11539–11551, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations*, 2021.

Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7201–7211, 2022.

A Ablation Study

We conduct ablations on two key RDumb++ hyperparameters: (1) the drift threshold k, controlling how easily drift is detected, and (2) the soft reset strength λ, controlling how strongly the model is pulled back toward its initialization during partial resets. Both factors substantially influence long-horizon stability on CCC.

A.1 Drift Threshold k

We test three drift sensitivities: $k \in \{2.0, 2.5, 3.0\}$. A smaller k triggers resets frequently, preventing collapse but interrupting useful adaptation. A larger k delays resets, allowing harmful drift to accumulate. A moderate k strikes the best balance. This was implemented using EntropyFull Method at transition speed of 2000.

Table 2: Ablation on drift threshold k.

k	Accuracy (%)
2.0	43.5
2.5	44.2
3.0	42.8

The best-performing value is:

$$k^* = 2.5.$$

A.2 Soft Reset Strength λ

We test interpolation weights: $\lambda \in \{0.3, 0.5, 0.7\}$. A weak reset ($\lambda = 0.3$) under-corrects drift, while a strong reset ($\lambda = 0.7$) erases too much adaptation. This was implemented using EntropySoft Method at transition speed of 5000 with drift threshold $k = 2.5$.

Table 3: Ablation on soft reset strength λ.

λ	Accuracy (%)
0.30	43.0
0.50	44.3
0.70	42.1

The optimal value is:

$$\lambda^* = 0.5.$$

A.3 Summary of Insights

Both ablations reveal that RDumb++ requires balanced hyperparameters: too frequent resets interrupt learning, while too infrequent resets allow drift to accumulate. The best-performing settings on CCC-medium are:

$$k = 2.5,\ \lambda = 0.5.$$

These values enable RDumb++ to correct harmful drift while retaining useful adaptation, explaining the performance gains over RDumb.

B Discussion

RDumb++ replaces RDumb's fixed, periodic reset schedule with data-driven drift detection. On the CCC benchmark, corruption types shift abruptly, causing the model's predictive distribution to deviate sharply. Periodic resets occur regardless of whether drift has happened, leading either to

premature resets (which discard useful adaptation) or delayed resets (which allow drift to accumulate). In contrast, RDumb++ triggers a reset exactly when a statistically significant deviation is detected. Full resets recover the model after distribution shifts, while soft resets gently correcting deviations without discarding beneficial adaptation. This targeted intervention prevents the collapse cycles commonly observed in long-horizon entropy-based TTA methods.

C Related Work

Entropy-based Test-Time Adaptation. Tent Wang et al. (2021) proposed the seminal idea of adapting models at test time by minimizing prediction entropy using batch normalization parameters. While highly effective under mild or short-horizon shifts, Tent is known to suffer from entropy collapse: the model becomes overconfident and drifts toward degenerate predictions when exposed to long, non-stationary input streams.

Selective and Robust Adaptation. EATA Niu et al. (2022) improves upon Tent by introducing two key mechanisms: (1) sample reliability filtering, which ensures that the model adapts only on low-entropy examples, and (2) redundancy filtering via cosine similarity, preventing repeated updates on semantically identical samples. These strategies slow down collapse but still fail under rapidly evolving corruptions such as CCC.

Resets for Long-Horizon CTTA. RDumb Press et al. (2024) extends EATA by periodically resetting the model to a stored checkpoint every fixed number of steps (typically 1000). This simple heuristic greatly increases stability for continual test-time adaptation. However, the reset schedule is agnostic to the underlying data drift: resets may occur either too early, discarding useful adaptation or too late i.e after the model has collapsed.

D Reproducibility Statement

To ensure reproducibility of our results, all experiments were conducted using a standardized ResNet-50 backbone pretrained on ImageNet, adapting only the batch normalization affine parameters. The anonymous GitHub code is available at `https://github.com/himans-iitk/Rdumbpp`.