Test Time Adaptation Using Adaptive Quantile Recalibration

Paria Mehrbod [1 2] **Pedro Vianna** [3 2] **Geraldin Nanfack** [1 2] **Guy Wolf** [3 2] **Eugene Belilovsky** [1 2]

Abstract

Domain adaptation methods have emerged as effective mechanisms to improve the generalizability and robustness of deep learning models, particularly in real-world scenarios where test data may differ significantly from the training domain. However, traditional domain adaptation techniques often require prior knowledge of target domains or model retraining, which limits their applicability in dynamic settings where such information is unavailable or retraining is impractical. Approaches based on updating batch normalization statistics at test-time have been gaining traction, as it allows for unsupervised adaptation based on the target data. Some of these approaches only adjust batch normalization statistics and do not fully capture complex distributions and are restricted to specific normalization types. To address this, we propose Adaptive Quantile Recalibration (AQR), a novel test-time adaptation method based on quantile recalibration, which modifies the pre-activation distributions by aligning quantiles on a channel-by-channel basis. AQR captures the complete shape of activation distributions and works across diverse architectures regardless of normalization type (BatchNorm, GroupNorm, or LayerNorm).We demonstrate that our method provides robust adaptation across diverse settings, outperforming baseline test-time adaptation methods.

1. Introduction

Deep neural networks have demonstrated remarkable success across numerous computer vision tasks, including image classification, object detection, and segmentation. However, these models often suffer significant performance degradation when deployed in real-world environments that differ from their training conditions. This phenomenon, known as distribution shift or domain gap, poses a major challenge for the practical deployment of deep learning systems in applications where reliability and robustness are critical.

Several domain adaptation methods have been proposed to address this issue, although they often assume prior knowledge of the target domain or require retraining, which hinders their applicability across different tasks and scenarios. Test-time adaptation (TTA) techniques have emerged as promising approaches that adapt models to target distributions during inference, relying solely on test data batches. These techniques are particularly suitable for real-world applications where distribution shifts may occur unexpectedly or evolve over time.

A popular approach to TTA is based on test-time normalization (TTN), where batch normalization statistics are modified to match the target data distribution. This method has been demonstrated to be particularly effective for convolutional neural networks (CNNs) (Nado et al., 2020; Schneider et al., 2020; Vianna et al., 2024), and recently has achieved notable success when applied to vision transformers (ViT) (Marsden et al., 2023; Lee & Chang, 2024; Lee et al., 2024; Wang et al., 2024). However, TTN implicitly assumes the neuron-level activations approximate a Gaussian distribution, which may not hold for complex, multi-modal distributions encountered in practice. Furthermore, TTN is limited to architectures that employ batch normalization layers (BatchNorm), making it inapplicable to models using other normalization schemes such as group normalization or layer normalization (GroupNorm and LayerNorm).

We thus propose Adaptive Quantile Recalibration (AQR), a novel TTA method that aligns the distributions of internal features between source and target domains, without relying on parametric distribution assumptions. Our approach leverages nonparametric quantile-based transformations to map target domain activations to their corresponding source domain distributions on a channel-by-channel basis. Unlike methods that only adjust the mean and variance of pre-activations, AQR captures and preserves the complete shape of pre-activation distributions, making it effective for handling complex distribution patterns commonly found in deep neural networks (Figure 1). A critical advantage of our method is that it does not degrade over time, as we are adapt-

[1]Concordia University [2]Mila – Quebec AI Institute [3]Université de Montréal. Correspondence to: Paria Mehrbod <paria.mehrbod@mila.quebec>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025. Copyright 2025 by the author(s).

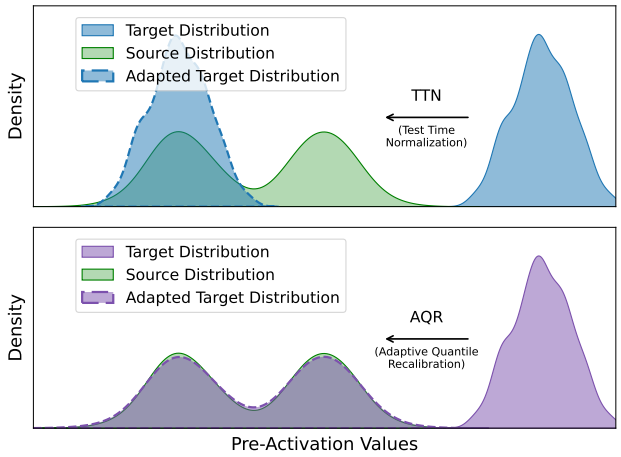

Figure 1. Comparing AQR and TTN in preserving complex distribution shapes at test-time using synthetic data.

ing to source activations that are precomputed and remain fixed throughout testing. Unlike entropy-based methods that continuously update model parameters and can drift toward suboptimal solutions, AQR provides a stable reference point derived from the source domain statistics, ensuring consistent and stable adaptation performance even in challenging test scenarios. Our key contributions are as follows:

- We propose a novel method that calibrates pre-activations at test-time to align with train-time pre-activations by leveraging statistics computed at the end of model training.
- We demonstrate our method's applicability across diverse model architectures, independent of specific types of normalization layer.
- We identify and address challenges associated with varying batch sizes in computing statistical information and propose strategies for the accurate estimation of distribution tails.
- Our experiments on two datasets across three architectures show that our method outperforms current state-of-the-art approaches and shows potential for real-world applications.

2. Related Work

TTA methods frequently operate by updating the parameters associated with BatchNorm layers in response to covariate shifts in the input distribution, as is the case of the popular approach TTN (Schneider et al., 2020; Nado et al., 2020). This strategy has demonstrated effectiveness in mitigating the effects of varying degrees of image corruption. However, a key limitation lies in their dependency on BatchNorm, which restricts their applicability to architectures using this specific normalization scheme. Additionally, BatchNorm is considered a major contributor to instability in standard TTA pipelines (Niu et al., 2023).

The rising use of alternative normalization layers like GroupNorm and LayerNorm necessitates the development of TTA algorithms compatible with various architectures. Addressing these challenges, sharpness-aware and reliable entropy minimization (SAR) (Niu et al., 2023) was developed as an online TTA method that supports all types of normalization layer.

Another popular approach, often combined with TTN, requires adapting the affine parameters of normalization layers using entropy minimization loss, as seen in methods like (Wang et al., 2021) and (Niu et al., 2023). However, these methods face stability challenges in wild test scenarios. Specifically, they can produce faulty feedback if an incorrect selection of samples is used to calculate the loss and may suffer performance degradation over time.

For easier deployment across multiple architectures, marginal entropy minimization with one test point (MEMO) (Zhang et al., 2022) was proposed as an approach needing only the trained model and a single test input. However, it is computationally expensive due to per-sample backpropagation and test-time augmentation, making it unsuitable for latency-sensitive tasks.

Neuron editing (Amodio et al., 2019) addresses the problem of generating transformed versions of data based on the pre- and post-transformation versions observed of a small subset of the available data. Rather than learning distribution-to-distribution mappings, it reframes the problem as learning a general edit function that can be applied to other datasets. The method applies piecewise linear transformations to neuron activations in autoencoder latent spaces, computing percentile-based differences between source and target distributions. This nonparametric approach preserves data variability and avoids issues like mode collapse seen in generative models. Inspired by this approach, we adapt their percentile-based transformation strategy to the test-time adaptation setting while making it applicable to diverse neural network architectures independent of specific normalization layers.

3. Methodology

In the current work, we propose a TTA method based on aligning the distributions of the intermediate features of a neural network. Our key insight is that distribution shifts between training and testing data manifest as shifts in the distributions of intermediate features of neural networks. By transforming these internal distributions to match those observed during training, we can improve the model's performance on out-of-distribution test data without requiring access to training data or modifying the training process. Our method consists of two phases: First, in the setup phase, we compute the statistical information of the internal lay-

ers of the model when given the source data. Second, in the inference phase, we transform the model's internal pre-activation values to correct for distribution shifts that occur when processing test data.

3.1. Setup Phase: Source Distribution Statistics

Let f_θ denote a neural network with parameters θ trained on source distribution $P(x)$. After training is complete and before any inference, we perform a one-time setup phase to capture the statistical information of the source distribution. In this phase, we apply the following steps:

1) Process a subset of source/training data S through the trained model.

2) For each layer l, and each channel c within that layer, store the pre-activation values denoted as a_c^l (outputs of normalization layers before activation function)

3) Compute percentiles p_i^S where $i \in \{0, 1, \ldots, 100\}$ from the stored pre-activation values a_c^l, doing this separately for each channel in each layer.

These stored percentiles (p_i^S) serve as a memory of the distribution characteristics of the model's internal values when processing in-distribution data, and will be used during inference to guide the adaptation process.

3.2. Inference Phase: Distribution Alignment

During inference, when out-of-distribution test samples are processed through the network, the distribution of pre-activation values (a_c^l) deviates from what was observed during training. We propose to transform these values to match their training-time distributions.

Our method is agnostic to the specific type of normalization layer used in the network (batch, layer, or group normalization). For each batch of test samples, we: 1) compute percentiles p_i^T of the pre-activation values for each channel, 2) transform these values using a piecewise linear transformation adapted from (Amodio et al., 2019) that we denote **AQR**. This transformation is applied as follows:

$$\text{AQR}(x) = p_j^S + \left(\frac{x - p_j^T}{\Delta_j^T} \right) \cdot \Delta_j^S \quad \text{for } x \in [p_j^T, p_{j+1}^T) \tag{1}$$

where, $\Delta_j^T = p_{j+1}^T - p_j^T$ and x represents the pre-activation values of a specific channel and a specific layer, p_i^T represents the i-th percentile of the test samples' pre-activation values (computed on-the-fly), and p_i^S represents the i-th percentile of the source/training pre-activation values (previously computed during the setup phase). This transformation uses 100 percentile intervals, with $j \in \{0, 1, 2, ..., 99\}$ covering the entire distribution range from the 0th to the 100th percentile, and maps the test-time distribution back to the distribution observed during training. This transformation is applied to all channels of a given model.

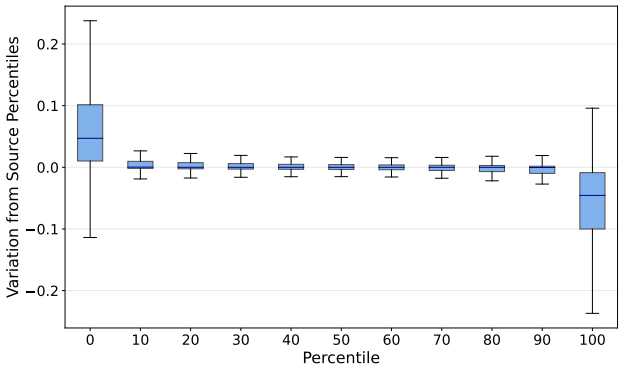

Figure 2. Distribution of deviations between small-batch (128 samples) and reference (10,000 samples) percentiles across 20 trials.

3.3. Calibrating the tail of distribution

The size of source dataset S can affect how accurately the source distribution is estimated. More samples lead to better overall estimation, but can produce extreme values in the distribution tails. Figure 2 demonstrates the instability of tail percentile estimation using small batches. We drew 20 different batches of 128 samples from the source/training distribution and computed percentiles for each batch. For each percentile level, we calculated the deviation from the source/training percentile computed using 10,000 training samples. Each boxplot shows the distribution of these deviations across the 20 batches. The results reveal that tail percentiles show substantial variability: the 0th percentile (minimum) consistently overestimates the true minimum, while the 100th percentile (maximum) consistently underestimates the true maximum. This bias and high variability in tail estimation motivates our tail calibration strategy. Instead of using actual minimum and maximum values of the source data, we estimate the first and last percentiles through sampling. We compute these statistics over a batch of 100, repeat the sampling 1,000 times, and then average the results. This approach provides more reliable estimates of the distribution tails. We evaluate the impact of this strategy in the following section.

4. Experiments

Datasets. We evaluate AQR on two standard benchmarks: CIFAR-10/CIFAR-10-C and ImageNet-1K/ImageNet-1K-C (Hendrycks & Dietterich, 2019), which apply 19 corruption types at 5 severity levels to test robustness under distribution shift. **Models.** We test diverse architectures with different normalization schemes: For CIFAR-10: ResNet18 (BatchNorm), ResNet26-GN (GroupNorm), and ViT-Patch4-32 (LayerNorm). For ImageNet: ResNet50 (BatchNorm) and ResNet50-GN (GroupNorm). All experiments use 10 random seeds for statistical significance. **Baselines.** We com-

Table 1. Classification accuracy (%) of different test-time adaptation methods on ImageNet-C

Method	Batch Size = 128			Batch Size = 512		
	Severity 1	Severity 3	Severity 5	Severity 1	Severity 3	Severity 5
ResNet50 (BatchNorm)						
Not Adapted	61.75 ± 7.20	41.45 ± 14.86	19.78 ± 14.47	61.53 ± 6.33	41.87 ± 14.00	19.63 ± 13.98
TTN	66.84 ± 6.00	52.36 ± 12.48	33.01 ± 16.17	67.21 ± 4.49	53.10 ± 11.68	33.34 ± 15.94
TENT	66.89 ± 5.96	52.45 ± 12.45	33.19 ± 16.13	67.28 ± 4.46	53.22 ± 11.47	33.52 ± 15.92
SAR	66.84 ± 6.01	52.37 ± 12.48	33.05 ± 16.14	67.22 ± 4.48	53.19 ± 11.64	33.36 ± 15.94
AQR	**67.39 ± 5.48**	**53.43 ± 12.40**	**33.94 ± 16.68**	**67.85 ± 4.05**	**54.36 ± 11.59**	**34.47 ± 16.32**
ResNet50 (GroupNorm)						
Not Adapted	69.65 ± 6.23	55.11 ± 12.72	32.87 ± 16.33	69.72 ± 4.92	55.42 ± 12.23	32.85 ± 15.95
TENT	**69.68 ± 6.22**	**55.16 ± 12.71**	32.86 ± 16.36	70.14 ± 6.89	55.50 ± 12.22	32.85 ± 15.95
SAR	69.64 ± 6.23	55.12 ± 12.71	32.87 ± 16.33	69.72 ± 4.92	55.42 ± 12.23	32.85 ± 15.95
AQR	68.78 ± 5.57	55.00 ± 12.17	**35.61 ± 16.75**	**70.81 ± 3.89**	**58.36 ± 11.39**	**37.69 ± 17.15**

Table 2. Classification accuracy (%) of different tail calibration strategies on ImageNet-C with ResNet50. Results are averaged over 10 random seeds.

Tail Calibration Strategy	Batch Size	
	128	512
AQR (standard)	30.8±16.3	33.7±16.3
Not Calibrated	29.7±16.6	33.3±16.3
Average Sample Tails	**33.7±16.6**	**34.6±16.1**

pare against established TTA methods: TTN, TENT, and SAR, alongside unadapted models. All methods are evaluated in offline TTA mode, processing each batch independently. Additional experimental details are provided in Appendix A.

Ablation Study on Tail Handling We conducted an ablation study to evaluate different strategies for calibrating extreme tails. The standard AQR method serves as our baseline approach. We then tested our proposed enhancement, AQR with "Average Sample Tails", which uses the sampling technique described in the previous section to better estimate extreme percentiles. We also explored a simpler alternative: AQR with no tail adaptation. Since the extreme ends ($[p^0, p^1]$ and $[p^{99}, p^{100}]$) contain only 2% of the data, we tested whether simply not adapting these regions would be effective. The results in Table 2 show the impact of our strategies. Additional details and alternative approaches are provided in more depth in Appendix B.

5. Results

Our experimental evaluation shows the effectiveness of AQR across multiple architectures, datasets, and corruption severities. On the challenging ImageNet-C dataset, AQR demonstrates superior adaptation capabilities as shown in Table 1. With ResNet50 (BatchNorm), our method surpasses TTN, TENT, and SAR across all test conditions, with advantages becoming more significant at higher corruption levels. For ResNet50 with GroupNorm, AQR shows especially notable improvements at severity level 5. On CIFAR-10-C, AQR consistently outperforms baseline methods across all three tested architectures, as shown in Table 4 in Appendix C.

6. Conclusion

In this study, we introduced Adaptive Quantile Recalibration (AQR), a novel test-time adaptation approach that aligns the distributions of internal features between source and target domains through nonparametric quantile-based transformations. Our approach offers several key advantages: (1) it captures the complete shape of activation distributions rather than just mean and variance, enabling more effective adaptation for complex distribution patterns; (2) our tail calibration strategy effectively handles the challenges of estimating distribution extremes with varying batch sizes; (3) it maintains stability during extended test sessions by using fixed source distribution statistics as reference points. Experiments on CIFAR-10-C and ImageNet-C across multiple architectures demonstrate that AQR outperforms state-of-the-art TTA methods. However, while AQR provides stability through fixed reference statistics, this design choice means it processes batches independently rather than accumulating knowledge across sequential batches, which could potentially enhance adaptation in some online scenarios. Future work could explore combining AQR with other TTA methods and extending AQR to online settings while preserving its stability advantages.

Acknowledgment

We acknowledge funding from FRQNT-NOVA grant 2023-NOVA- 329759.

References

Amodio, M., van Dijk, D., Montgomery, R., Wolf, G., and Krishnaswamy, S. Out-of-sample extrapolation with neuron editing, 2019.

Hendrycks, D. and Dietterich, T. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

Lee, J., Jung, D., Lee, S., Park, J., Shin, J., Hwang, U., and Yoon, S. Entropy is not enough for test-time adaptation: From the perspective of disentangled factors. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=9w3iw8wDuE.

Lee, J.-H. and Chang, J.-H. Continual momentum filtering on parameter space for online test-time adaptation. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=BllUWdpIOA.

Marsden, R. A., Döbler, M., and Yang, B. Universal test-time adaptation through weight ensembling, diversity weighting, and prior correction, 2023. URL https://arxiv.org/abs/2306.00650.

Nado, Z., Padhy, S., Sculley, D., D'Amour, A., Lakshminarayanan, B., and Snoek, J. Evaluating prediction-time batch normalization for robustness under covariate shift, 2020.

Niu, S., Wu, J., Zhang, Y., Wen, Z., Chen, Y., Zhao, P., and Tan, M. Towards stable test-time adaptation in dynamic wild world, 2023.

Schneider, S., Rusak, E., Eck, L., Bringmann, O., Brendel, W., and Bethge, M. Improving robustness against common corruptions by covariate shift adaptation. *Advances in Neural Information Processing Systems*, 33, 2020.

Vianna, P., Chaudhary, M., Mehrbod, P., Tang, A., Cloutier, G., Wolf, G., Eickenberg, M., and Belilovsky, E. Channel-selective normalization for label-shift robust test-time adaptation, 2024. URL https://arxiv.org/abs/2402.04958.

Wang, D., Shelhamer, E., Liu, S., Olshausen, B. A., and Darrell, T. Tent: Fully test-time adaptation by entropy minimization. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=uXl3bZLkr3c.

Wang, Z., Luo, Y., Zheng, L., Chen, Z., Wang, S., and Huang, Z. In search of lost online test-time adaptation: A survey, 2024. URL https://arxiv.org/abs/2310.20199.

Yoshioka, K. vision-transformers-cifar10: Training vision transformers (ViT) and related models on CIFAR-10. https://github.com/kentaroy47/vision-transformers-cifar10, 2024. GitHub repository.

Zhang, M., Levine, S., and Finn, C. MEMO: Test Time Robustness via Adaptation and Augmentation, October 2022. URL http://arxiv.org/abs/2110.09506.

A. Detailed Experimental Setup

A.1. Dataset Specifications

CIFAR-10/CIFAR-10-C: CIFAR-10 contains 50,000 training images and 10,000 validation images across 10 classes. **ImageNet-1K/ImageNet-1K-C:** ImageNet-1K is a large-scale image classification dataset with 1.2 million training images and 50,000 validation images across 1,000 classes. For experiments with AQR, we use 10,000 training samples from each dataset to estimate source percentiles.

A.2. Model Implementation Details

For CIFAR-10 experiments, we obtain pre-trained weights for ResNet26-GN from (Zhang et al., 2022) and for ViT-Patch4-32 from (Yoshioka, 2024). For ImageNet experiments, we load pre-trained weights from PyTorch's model zoo for ResNet50, while for ResNet50-GN we use pre-trained weights from the timm library.

A.3. Baseline Implementation

For consistency, we implement episodic versions of TENT and SAR by resetting their model parameters to the original pre-trained weights after processing each batch. For TENT and SAR, we perform one forward-backward pass to update the model parameters, then conduct a second forward pass with the updated parameters to generate the final predictions. This approach ensures these parameter-updating methods have the opportunity to adapt before evaluation.

B. Discussion of Strategies for Calibrating Tails of Distributions

We evaluated five distinct approaches for handling the extreme percentiles:

1. **Standard AQR**: Our baseline approach that maps all percentiles using the linear transformation in Equation 1. The border cases are shown in Equation 2.

2. **Average Sample Tails**: This approach estimates the first and last percentiles through statistical sampling. We compute the first and last percentiles over a batch of 100, repeat the sampling 1,000 times, and then average the results to obtain more reliable estimates of the distribution tails.

3. **Not Calibrated**: In this strategy, no remapping occurs at the extremes. Values below the 1st percentile or above the 99th percentile remain unchanged, as shown in Equation 3.

4. **Clipping**: This implements a simple thresholding mechanism where values below the 1st percentile are set exactly to p_1^T, and values above the 99th percentile are set to p_{99}^T, as formalized in Equation 4.

5. **Gaussian Estimation**: This leverages parametric assumptions about the distribution tails. Rather than using empirical extrema, this approach models both source and target distributions as Gaussian and estimates their theoretical tail values, as shown in Equation 5.

6. **Interval Estimation**: This approach uses the standard deviation of the distribution as a normalizing factor, instead of the intervals at tails to determine the remapping scale according to Equation 6.

Table 3 shows the classification accuracy of different tail handling methods across various batch sizes and datasets using ResNet50. We conducted experiments with batch sizes of 128 and 512, evaluating performance on three datasets: standard training data, test data, and corrupted data (severity level 5). Each experiment was repeated with 10 different random seeds. While the "Not Calibrated" approach performs competitively on clean test data, it shows reduced robustness on corrupted data compared to our tail calibration strategies. The "Clipping" method significantly underperforms across all scenarios. Both "Gaussian Estimation" and "Average Sample Tails" methods performs well but "Average Sample Tails" performs particularly well on corrupted datasets.

Table 3. Classification accuracy (%) of different tail calibration strategies using ResNet50 on various ImageNet datasets. Results are averaged over 10 random seeds with the best results in bold.

Calibration Strategy	**Batch Size = 128**			**Batch Size = 512**		
	ImageNet-C	Clean Test	Clean Train	ImageNet-C	Clean Test	Clean Train
AQR (standard)	30.8±16.3	73.8±3.3	84.4±1.9	33.7±16.3	74.1±1.6	86.2±1.9
Average Sample Tails	**33.7±16.6**	73.6±4.2	85.9±2.2	**34.6±16.1**	74.2±1.6	86.7±2.1
Gaussian Estimation	33.6±16.6	73.8±3.8	**86.6±2.7**	33.5±16.1	74.3±1.4	**87.0±2.1**
Not Calibrated	29.7±16.6	**74.1±4.1**	86.2±2.4	33.3±16.3	**74.6±1.5**	86.9±2.1
Interval Estimation	29.3±15.6	71.6±4.9	83.2±2.1	30.8±15.2	72.2±1.2	84.1±2.6
Clipping	3.4±4.8	52.1±5.3	59.1±5.6	3.6±4.7	53.3±2.0	61.0±2.4

B.1. AQR

$$\text{AQR}(x) = \begin{cases} p_0^S + \left(\dfrac{x - p_0^T}{\Delta_0^T}\right) \cdot \Delta_0^S & x < p_1^T, \\ p_{99}^S + \left(\dfrac{x - p_{99}^T}{\Delta_{99}^T}\right) \cdot \Delta_{99}^S & x \geq p_{99}^T, \end{cases} \tag{2}$$

B.2. Not Calibrated

$$\text{AQR}(x) = \begin{cases} x & x < p_1^T \\ x & x \geq p_{99}^T \end{cases} \tag{3}$$

B.3. clipping

$$\text{AQR}(x) = \begin{cases} p_1^T & x < p_1^T \\ p_{99}^T & x \geq p_{99}^T \end{cases} \tag{4}$$

B.4. Gaussian Estimation

$$\text{AQR}(x) = \begin{cases} \left(\dfrac{x - Q(0)^T}{Q(1)^T - Q(0)^T} \cdot \left(Q(1)^S - Q(0)^S\right)\right) + Q(0)^S & x < p_1^T \\ \left(\dfrac{x - Q(99)^T}{Q(100)^T - Q(99)^T} \cdot \left(Q(100)^S - Q(99)^S\right)\right) + p_{99}^S & x \geq p_{99}^T \end{cases} \tag{5}$$

$$Q(p)^S = \beta + \gamma \Phi^{-1}(p)$$

and

$$Q(p)^T = \mu(X) + \sigma(X)\Phi^{-1}(p)$$

where $\Phi^{-1}(p) = \sqrt{2} \cdot \text{erf}^{-1}(2p - 1)$

(X is all points in a specific channel of a specific layer of a test input.)

B.5. Interval Estimation

$$\text{AQR}(x) = \begin{cases} \left(\dfrac{a - p_0^T}{std(X)} \cdot (\gamma)\right) + p_0^S & x < p_1^T \\ \left(\dfrac{a - p_{99}^T}{std(X)} \cdot (\gamma)\right) + p_{99}^S & x \geq p_{99}^T \end{cases} \tag{6}$$

C. Experiments on CIFAR-10-C

Table 4. Classification accuracy (%) of different test-time adaptation methods on CIFAR-10-C

Method	Batch Size = 128			Batch Size = 512		
	Severity 1	Severity 3	Severity 5	Severity 1	Severity 3	Severity 5
ResNet18 (BatchNorm)						
Not Adapted	59.54 ± 4.35	55.89 ± 8.22	46.78 ± 8.30	59.54 ± 4.35	55.89 ± 8.22	46.78 ± 8.30
TTN	77.97 ± 2.05	75.13 ± 2.41	70.24 ± 4.37	78.29 ± 2.01	75.46 ± 2.41	70.59 ± 4.40
TENT	77.97 ± 2.05	75.21 ± 2.44	70.24 ± 4.36	78.29 ± 2.01	75.53 ± 2.43	70.61 ± 4.38
SAR	77.97 ± 2.05	75.21 ± 2.44	70.24 ± 4.36	78.29 ± 2.01	75.53 ± 2.43	70.61 ± 4.38
AQR	77.91 ± 2.01	75.20 ± 2.36	70.61 ± 4.22	78.28 ± 1.99	75.50 ± 2.36	70.94 ± 4.24
ResNet26 (GroupNorm)						
Not Adapted	81.83 ± 4.33	77.26 ± 5.03	68.17 ± 6.98	81.83 ± 4.33	77.26 ± 5.03	68.17 ± 6.98
TENT	81.86 ± 4.30	77.32 ± 4.99	68.24 ± 6.91	81.85 ± 4.31	77.31 ± 4.99	68.23 ± 6.92
SAR	81.84 ± 4.31	77.29 ± 4.99	68.20 ± 6.93	81.84 ± 4.31	77.29 ± 4.99	68.20 ± 6.93
AQR	81.72 ± 3.68	78.68 ± 4.42	73.24 ± 5.68	81.96 ± 3.54	78.94 ± 4.37	73.65 ± 5.56
ViT-Patch4-32 (LayerNorm)						
Not Adapted	75.66 ± 3.52	68.93 ± 7.41	58.73 ± 14.93	75.66 ± 3.52	68.93 ± 7.41	58.73 ± 14.93
TENT	74.37 ± 3.11	67.24 ± 6.49	57.53 ± 12.64	74.42 ± 3.17	67.26 ± 6.55	57.68 ± 12.87
SAR	74.48 ± 3.17	67.29 ± 6.55	57.70 ± 12.88	74.37 ± 3.12	67.24 ± 6.53	57.71 ± 12.87
AQR	77.47 ± 1.71	74.05 ± 2.56	67.30 ± 9.24	77.80 ± 1.73	74.41 ± 2.50	67.65 ± 9.29