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]

[1]Concordia University [2]Mila – Quebec AI Institute [3]Université de Montréal

Abstract

Domain adaptation is a key strategy for enhancing the generalizability of deep learning models in real-world scenarios, where test distributions often diverge significantly from the training domain. However, conventional approaches typically rely on prior knowledge of the target domain or require model retraining, limiting their practicality in dynamic or resource-constrained environments. Recent test-time adaptation methods based on batch normalization statistic updates allow for unsupervised adaptation, but they often fail to capture complex activation distributions and are constrained to specific normalization layers. We propose Adaptive Quantile Recalibration (AQR), a test-time adaptation technique that modifies pre-activation distributions by aligning quantiles on a channel-wise basis. AQR captures the full shape of activation distributions and generalizes across architectures employing BatchNorm, GroupNorm, or LayerNorm. To address the challenge of estimating distribution tails under varying batch sizes, AQR incorporates a robust tail calibration strategy that improves stability and precision. Our method leverages source-domain statistics computed at training time, enabling unsupervised adaptation without retraining models. Experiments on CIFAR-10-C, CIFAR-100-C, and ImageNet-C across multiple architectures demonstrate that AQR achieves robust adaptation across diverse settings, outperforming existing test-time adaptation baselines. These results highlight AQR's potential for deployment in real-world scenarios with dynamic and unpredictable data distributions.

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 [11].

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) [15,18,19], and recently has achieved notable success when applied to vision transformers (ViTs) [8, 9, 14, 21]. 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 [6]), making it inapplicable to models using other normalization schemes such as group normalization or layer normalization (GroupNorm [24] and LayerNorm [2]).

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 adapting to source activations that are

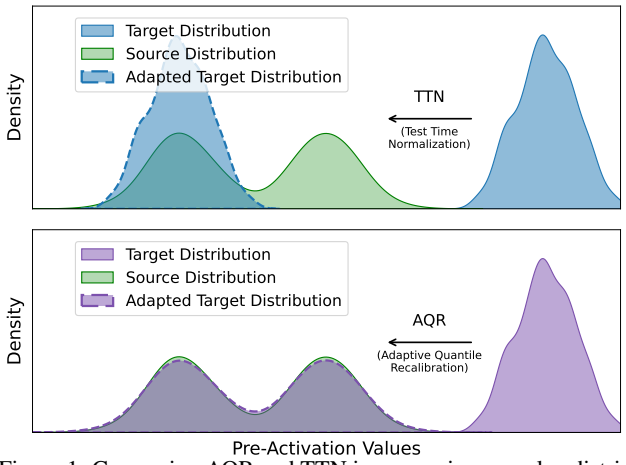

Figure 1. Comparing AQR and TTN in preserving complex distribution shapes at test-time using synthetic data.

precomputed and remain fixed throughout testing. Unlike entropy-based methods that continuously update model parameters and can drift toward suboptimal solutions [28], 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 layers.
- 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 three datasets across four 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 [3, 4, 12, 15, 16, 18, 20, 23, 26], as is the case of the popular approach TTN [15, 18]. This strategy is closely related to the unsupervised domain adaptation technique AdaBN [10], and 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 [17].

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) [17] was developed as an online TTA method that supports all types of normalization layers.

Another popular approach, often combined with TTN, requires adapting the affine parameters of normalization layers using entropy minimization loss, as seen in previous research [17, 20]. 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) [27] 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 [1] 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-

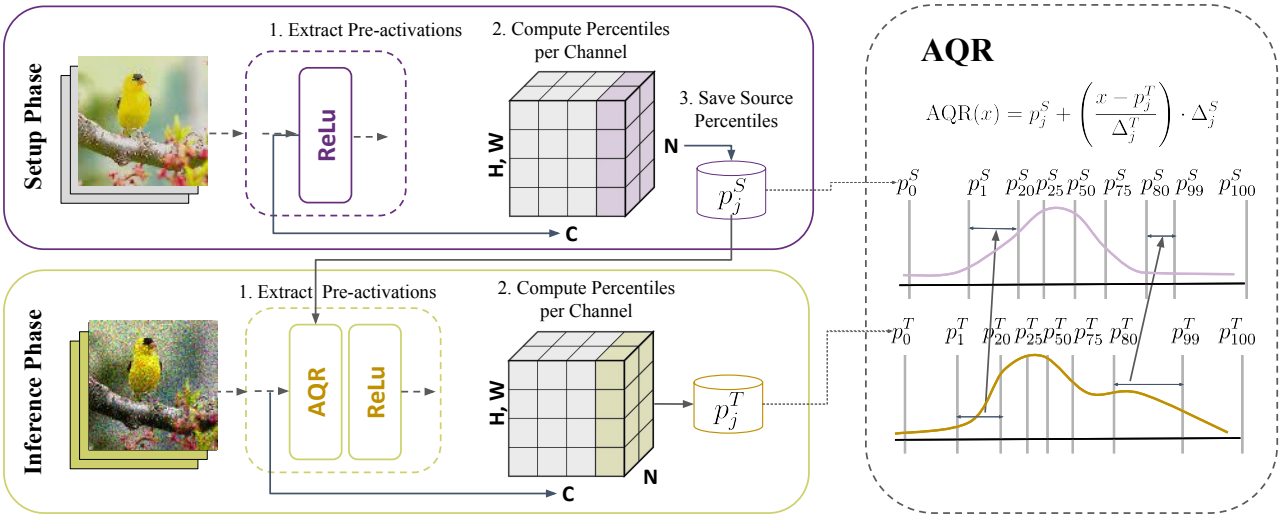

Figure 2. Overview of the Adaptive Quantile Recalibration (AQR) method. During the setup phase (top), source data is processed to extract pre-activations and compute percentiles per channel, which are saved as reference statistics. During inference (bottom), target data pre-activations are similarly processed, and AQR transforms target percentiles to match source percentiles using piecewise linear transformation, enabling distribution alignment without architectural constraints.

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 [1] that we denote **AQR**. This transformation is applied as follows:

$$\mathrm{AQR}(x) = p_i^S + \left(\frac{x - p_i^T}{\Delta_i^T}\right) \cdot \Delta_i^S \quad \text{for } x \in [p_i^T, p_{i+1}^T) \quad (1)$$

where, $\Delta_i^T = p_{i+1}^T - p_i^T$ 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 $i \in \{0, 1, 2, ..., 99\}$ 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.

3.3. Calibrating the Tail of Distribution

The size of the source dataset 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 3 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

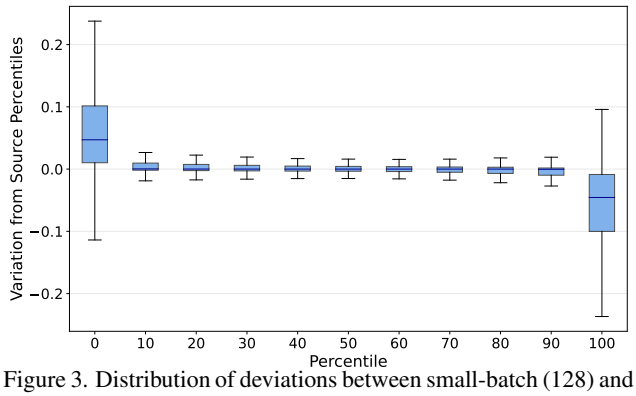

Figure 3. Distribution of deviations between small-batch (128) and reference (10,000) percentiles across 20 trials.

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 motivate 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.

3.4. Analysis of a One-Hidden-Layer Model

In this section, we provide a simple theoretical motivation under a simplified architecture and corruption model for advantages of AQR's piecewise-linear transformation. This proof holds under specific assumptions and is intended to provide intuition for how AQR works compared to TTN and similar methods that only update affine parameters. In App. **??**, we present a complementary analysis under finite samples and quantile discretization.

Main objective. We compare two adaptation strategies, AQR and TTN, which attempt to recover the source hidden representation h^S from the corrupted test-time h^T. We measure adaptation quality using the total mean squared error (MSE)

$$\mathrm{MSE}(T) \;:=\; \sum_{i=1}^{m} \mathbb{E}\big[(T_i(h_i^T) - h_i^S)^2\big].$$

Our goal is to show that under the assumptions below,

$$\mathrm{MSE}\big(T^{\mathrm{AQR}}\big) \;=\; 0,\; \mathrm{MSE}\big(T^{\mathrm{TTN}}\big) \;>\; 0 \text{ for non-affine } k_i.$$

Setting. We consider a one-hidden-layer MLP on the **source domain**. Let the input be a random vector $x \in \mathbb{R}^d$ with

$$x \sim P \text{ on } \mathbb{R}^d, \qquad \mathrm{supp}(P) = \mathbb{R}^d.$$

Network parameters are $W \in \mathbb{R}^{m\times d}$ and $b \in \mathbb{R}^m$. The pre-activations and post-activations are

$$a^S = Wx + b \in \mathbb{R}^m, \qquad h^S = \phi(a^S) \in \mathbb{R}^m,$$

with final prediction $y_S = w^\top h^S$ for some $w \in \mathbb{R}^m$. The activation $\phi : \mathbb{R} \to \mathbb{R}$ is strictly increasing and continuous (e.g., identity or leaky-ReLU with positive slope), so ϕ^{-1} is well-defined on its image. Index neurons by $i \in \{1, \dots, m\}$. For each i, the source hidden h_i^S has marginal distribution P_i with continuous density and strictly increasing CDF F_{P_i}.

Corruption model. Test-time corruptions are modeled as strictly increasing transformations that, although induced by the input shift, are observed at pre-activations/activations. Concretely, for each i there exists strictly increasing $g_i : \mathbb{R} \to \mathbb{R}$ such that

$$a_i^T \;\overset{d}{=}\; g_i(a_i^S).$$

Since ϕ is strictly increasing, we define

$$k_i \;:=\; \phi \circ g_i \circ \phi^{-1},$$

which is strictly increasing. Then the corrupted hidden representation satisfies

$$h_i^T \;=\; k_i(h_i^S), \tag{2}$$

and has target marginal $Q_i = (k_i)_{\#}P_i$ (pushforward of P_i through k_i). If F_{Q_i} is the CDF of Q_i, then for all $x \in \mathrm{supp}(Q_i)$,

$$F_{Q_i}(x) = F_{P_i}\big(k_i^{-1}(x)\big).$$

Comparison of methods. **AQR** uses the exact CDFs to define the quantile transform

$$T_i^{\mathrm{AQR}}(z) := F_{P_i}^{-1}\big(F_{Q_i}(z)\big).$$

Applying this to h_i^T gives

$$\begin{aligned} T_i^{\mathrm{AQR}}(h_i^T) &= F_{P_i}^{-1}\big(F_{Q_i}(h_i^T)\big) \\ &= F_{P_i}^{-1}\big(F_{P_i}(k_i^{-1}(h_i^T))\big) \\ &= k_i^{-1}(h_i^T) \\ &= k_i^{-1}\big(k_i(h_i^S)\big) \quad \text{by Eq. (2)} \\ &= h_i^S. \end{aligned}$$

Table 1. Comparing the performance of test-time adaptation methods on ImageNet-C at corruption severity level 3 and batch size 128. Results show classification accuracy (%) per corruption type. Bold indicates best performance per model. "NA" denotes the non-adapted (original) model. Standard Error does not exceed 0.27.

Model	Method	Noise				Blur				Weather				Distortion					Digital		Average
		Gauss.	Impul.	Shot	Speck.	Defoc.	G.Blur	Motion	Zoom	Bright.	Contr.	Satur.	Fog	Elastic	Frost	Glass	Pixel	Snow	JPEG	Spatt.	Avg
ResNet50 (BN)	NA	27.6	25.1	25.1	31.7	**38.0**	41.6	37.7	35.2	69.6	46.0	71.4	46.6	55.6	32.1	16.9	46.2	35.2	59.3	49.4	41.6
	TTN	45.5	44.5	43.2	47.4	37.2	42.3	50.2	51.2	72.1	64.1	73.8	62.2	66.4	41.6	32.4	64.4	47.5	63.0	59.4	53.1
	TENT	45.5	44.5	43.2	47.4	37.2	42.3	50.3	51.2	72.1	64.1	73.8	62.3	66.4	41.6	32.4	64.4	47.5	63.0	59.4	53.1
	SAR	45.6	44.6	43.3	47.5	37.4	**42.5**	50.4	51.3	72.1	64.1	**73.8**	62.3	66.5	41.7	32.6	64.5	47.6	63.0	59.5	53.2
	AQR	**48.3**	**47.2**	**47.4**	**50.2**	35.1	39.6	**51.3**	**52.1**	**72.1**	**64.8**	73.3	**63.1**	**68.0**	**45.5**	**33.8**	**65.0**	**51.0**	**64.6**	**61.5**	**54.4**
ResNet50 (GN)	NA	54.5	53.1	52.8	57.7	44.3	49.8	49.7	39.2	75.4	69.8	76.9	55.8	59.6	**54.0**	21.2	59.7	54.8	66.3	63.3	55.7
	TENT	54.5	**53.1**	52.8	57.7	44.3	49.8	49.7	39.2	75.4	69.8	76.9	55.8	59.6	54.0	21.2	59.7	54.8	66.3	63.3	55.7
	SAR	**54.5**	53.1	**52.9**	**57.7**	**44.3**	**49.8**	49.7	39.2	**75.4**	69.8	**76.9**	56.1	59.6	54.0	21.3	59.8	54.8	**66.3**	63.2	55.7
	AQR	50.9	48.0	48.1	50.1	41.7	45.4	**59.2**	**55.5**	74.9	**71.3**	76.1	**68.7**	**71.1**	51.4	**36.9**	**67.9**	**57.1**	64.9	**65.0**	**58.1**
ViT-Base (FT)	NA	62.7	62.0	60.9	63.3	51.9	54.6	58.2	45.0	72.6	76.8	75.4	71.0	67.9	34.4	37.1	69.2	45.3	67.8	64.1	60.0
	TENT	53.8	53.1	50.1	53.8	48.6	51.6	53.3	40.8	68.4	74.3	71.0	67.7	64.9	31.6	35.4	65.4	35.8	64.7	59.5	54.9
	SAR	62.7	58.5	60.3	63.3	52.0	54.7	58.3	45.1	72.6	76.8	75.4	71.0	67.3	34.5	37.2	69.0	45.6	67.7	64.1	59.8
	AQR	**63.9**	**63.3**	**62.3**	**64.9**	**55.2**	**57.8**	**59.7**	**52.2**	**75.1**	**77.7**	**77.5**	**72.9**	**73.5**	**41.8**	**48.1**	**72.6**	**52.6**	**71.7**	**69.7**	**63.8**
ViT-Base (TS)	NA	39.3	37.9	36.5	43.8	42.5	46.1	48.8	40.8	64.9	66.7	67.9	56.5	67.9	30.2	37.6	68.0	31.5	64.6	53.8	49.7
	TENT	39.3	37.9	36.5	43.7	42.5	46.1	48.8	40.8	64.9	66.7	67.9	56.5	67.9	30.2	37.6	68.0	31.5	64.6	53.8	49.7
	SAR	39.4	37.9	36.5	43.8	42.5	46.1	48.8	40.9	64.9	66.7	67.9	56.5	67.9	30.3	37.6	68.0	31.6	64.6	53.8	49.8
	AQR	**43.8**	**43.2**	**41.6**	**47.7**	**44.9**	**48.3**	**51.1**	**44.7**	**65.6**	**67.9**	**68.3**	**58.5**	**68.9**	**35.3**	**42.4**	**68.7**	**37.7**	**65.4**	**57.3**	**52.7**

TTN applies the affine transform

$$T_i^{\mathrm{TTN}}(z) \;=\; \mu_i^S \;+\; \sigma_i^S \,\frac{z-\mu_i^T}{\sigma_i^T},$$

where (μ_i^S, σ_i^S) and (μ_i^T, σ_i^T) are the (population) mean and standard deviation of P_i and Q_i, respectively. This matches first and second moments, and exactly inverts the corruption only when k_i is affine; for nonlinear k_i, TTN leaves a nonzero distortion.

Result. From the AQR derivation above,

$$\mathrm{MSE}\big(T^{\mathrm{AQR}}\big) = \sum_{i=1}^{m} \mathbb{E}\big[(T_i^{\mathrm{AQR}}(h_i^T) - h_i^S)^2\big] \;=\; 0.$$

In contrast,

$$\mathrm{MSE}\big(T^{\mathrm{TTN}}\big) \;>\; 0 \quad \text{whenever some } k_i \text{ is non-affine.}$$

Therefore, AQR achieves perfect recovery under these idealized conditions, whereas TTN cannot unless each k_i is affine. A complementary finite-sample/discretization analysis is provided in App. **??**.

Remark (MSE aggregation). We sum MSE across neurons; equivalently, one may average by m or write $\mathbb{E}\big[\|T(h^T) - h^S\|_2^2\big]$. The conclusions are unchanged.

4. Experiments and Results

4.1. Datasets and Models

We evaluate our methods on CIFAR-10, CIFAR-100, and ImageNet-1K together with their corresponding corruption benchmarks, CIFAR-10-C, CIFAR-100-C, and ImageNet-C [5, 7]. Each corruption suite contains 19 corruption types, and each type is provided at five severity levels. We report results at severity levels 1, 3, and 5. CIFAR-10 and CIFAR-100 each contain 50,000 training images and 10,000 test images at a resolution of 32×32, with 10 and 100 classes respectively. ImageNet-1K contains 1,280,000 training images and 50,000 validation images at a resolution of 224×224 across 1,000 classes.

We evaluate four architecture families per dataset: ResNets with BatchNormalization (BN) or GroupNormalization (GN), and Vision Transformers (ViT) with LayerNormalization (LN). For ViTs, we consider two regimes: trained from scratch (TS) on the target dataset and fine-tuned (FT) after pre-training on a larger corpus. All fine-tuned ViTs are pre-trained on ImageNet-21K and then fine-tuned on the dataset at hand. We denote ViT-Base-patch16-224 as ViT-Base and ViT-patch4-32 as ViT-Small. Below we list, for each dataset, the specific models and the ViT training regime (TS/FT).

- **CIFAR-10:** ResNet-18 (BN); ResNet-26 (GN); ViT-Small (TS); ViT-Base (FT).
- **CIFAR-100:** ResNet-50 (BN); ResNet-50 (GN); ViT-

Small (TS); ViT-Base (FT).

- **ImageNet-1K:** ResNet-50 (BN); ResNet-50 (GN); ViT-Base (trained from scratch with Sharpness Aware Minimization); ViT-Base (FT).

Each set of experiments is evaluated over three random seeds to ensure statistical significance.

We obtain pre-trained ResNet-26 (GN) weights from [27] and ImageNet models from the PyTorch model zoo and the `timm` library [22]. ViT-Small models are trained from scratch using the approach described in [25]. For ResNet models on CIFAR-100, we used pre-trained ResNet-50 (BN) and ResNet-50 (GN) models and fine-tuned them on CIFAR-100, achieving test accuracies of 78.3% and 72.6% following the training methodology of [13]. For the ViT-B (FT) models, images were resized to 224×224. We fine-tuned these models that were pre-trained on ImageNet21K, reaching test accuracies of 98.6% for CIFAR-10 and 90.5% for CIFAR-100.

4.2. Experiment Setup

For experiments with AQR, we used 10,000 training samples from each dataset to estimate the source percentiles. These statistics are frozen for all test-time evaluations. We experimented with applying AQR transformation throughout the network as well as in just the top half. For more complex datasets (Imagenet) with ViT, we found it was beneficial only to include it in the top half. See ablations in Appendix **??**. We hypothesize that for ViT, the layer normalization is placed at the beginning of the block without influencing the residual stream. Therefore, placing AQR after layer normalization can leave the residual stream unadapted. This could pass noisy input to subsequent layers through the residual stream, even with AQR placed within the block; thus, limiting the use of AQR towards the top of the network can be beneficial.

Episodic Setting of TTA: We compare AQR to TTN [15, 18], TENT [20], and SAR [17], along with the unadapted source models. Since AQR is designed for stateless test-time use, we evaluate all methods in an **offline** (episodic) mode. Specifically, the inference on each batch is independent; This is an ideal setting for when distribution shift is highly variable.

For TENT and SAR, we reset model weights after every batch to their pre-trained values. Within each batch, a single forward–backward step is used for adaptation, followed by a second forward pass for evaluation. This approach ensures these parameter-updating methods have the opportunity to adapt before evaluation. This explains why in some cases, the models that have been adapted with SAR or TENT perform worse than the model that has not been adapted. TTN is applied as originally proposed. However, TTN is only applicable to model architectures that use BatchNorm, therefore TTN is excluded from experiments with ViTs or with ResNets that use GroupNorm.

4.3. Main Results

Our experimental evaluation shows the effectiveness of AQR across multiple architectures, datasets, and corruption severities. On the challenging ImageNet-C dataset, Table 1 highlights the key ImageNet-C experiments (severity 3, batch size 128), where AQR consistently surpasses TTN, TENT, and SAR across corruption types and architectures. The detailed experimental results, including comprehensive per-dataset summaries across all batch sizes and severity levels, are provided in the Appendix **??** (Tables **??**, **??**, **??**).

Cross-Dataset Performance Our method consistently outperforms existing approaches across all three benchmark datasets. As illustrated in Figure 4, AQR's advantage increases substantially with corruption severity, demonstrating its robustness to challenging test conditions. At severity level 1 (mild corruptions), AQR maintains competitive performance with existing methods. However, as corruption intensity increases to severity 3 and 5, AQR provides increasing improvements over baselines. Since TTN is not applicable to all architectures, it is omitted from this figure; a corresponding plot limited to ResNet models with BatchNorm is provided in the Appendix (Figure **??**).

Architecture-Specific Analysis Figure 5 reveals that the effectiveness of AQR spans diverse architectural designs. For Vision Transformers, which do not employ batch normalization layers, AQR provides improvements through its quantile-based normalization approach. ResNet architectures benefit significantly from AQR across both batch normalization and group normalization variants.

Batch Size Effects Our analysis reveals that AQR's advantages are particularly pronounced with larger batch sizes, which provide more robust estimation of quantiles for the incoming batch. At batch size 512, the performance gaps between AQR and baselines are consistently larger than at batch size 128 across all datasets.

This observation adds more evidence for the need to use the tail calibration strategy, as the batch size gets smaller and insufficient for estimating the target distribution tails.

Robustness Across Corruption Types The per-corruption analysis in Table 1 shows AQR's consistent effectiveness across diverse corruption categories. It performs particularly well on noise-based corruptions (Gaussian, impulse, shot noise) and distortion-based ones (elastic transform, pixelate). For digital corruptions (JPEG compression, spatter) and weather distortions (fog, brightness, contrast), AQR remains competitive.

4.4. Ablation on Mechanisms to Calibrate the Tail

We conducted an ablation study to evaluate different strategies for calibrating extreme tails. The standard AQR

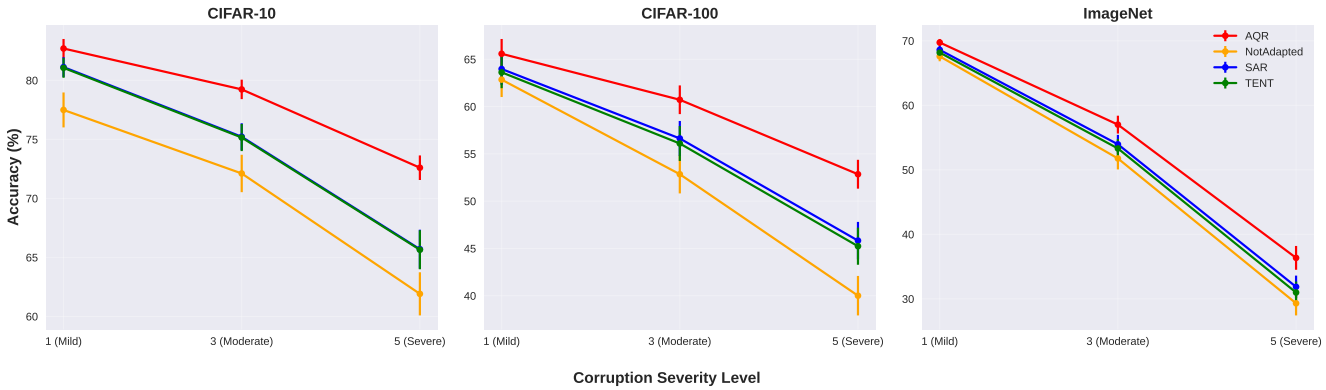

Figure 4. Performance comparison across corruption severity levels. AQR consistently outperforms baseline methods on all datasets, with larger performance gains at higher severities. Results averaged across all corruption types, batch sizes, and architectures. Error bars represent the standard error of the mean for different experimental conditions.

method serves as our baseline approach.

Average Sample Tails. We tested our proposed enhancement, AQR with the average of samples of tails, which uses the sampling technique described in the previous section to better estimate extreme percentiles. Specifically, we sample batches of 100 points, compute the minimum (p_0) and maximum (p_100) for each batch, repeat this 1,000 times, and average the results to obtain stable tail estimates. Subsequently, at inference time, these estimated values will be used to perform the recalibration.

AQR without Tail Adaptation. We also explored a simpler alternative: AQR with no tail adaptation, which we denote *Not Calibrated*. 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. Expressed as

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

This approach leaves values below the first percentile or above the 99th percentile unchanged.

Clipping. This approach implements simple thresholding. Expressed as

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

it sets extreme values to fixed boundaries - values below the 1st percentile are set exactly to the 1st percentile value, and values above the 99th percentile are set to the 99th percentile value.

Gaussian Estimation. Another approach of calibrating the tails, which we call *Gaussian Estimation*, assumes both source and target data follow normal distributions. Instead of using unreliable minimum and maximum values from small batches, it fits Gaussian curves to the data. It then uses these fitted curves to predict what the true 0^{th} and 100^{th} percentiles should theoretically be. A complete mapping formula of this strategy is provided in Appendix **??**.

Table 2. Classification accuracy (%) of different tail calibration strategies using ResNet50 on various ImageNet datasets. Results are averaged over 3 random seeds with the best results in bold.

Tail Calibration Strategy	Batch Size	
	128	512
AQR (standard)	30.8±16.3	33.7±16.3
Average Sample Tails	**33.7±16.6**	**34.6±16.1**
Gaussian Estimation	33.6±16.6	33.5±16.1
Not Calibrated	29.7±16.6	33.3±16.3
Interval Estimation	29.3±15.6	30.8±15.2
Clipping	3.4±4.8	3.6±4.7

Interval Estimation. This approach aims to estimate the magnitude of extreme intervals (Δ_0^T and Δ_{99}^S). Rather than relying on percentile intervals for scaling transformations, it uses the standard deviation of the distribution. The *interval estimation* can be expressed as follows

$$\text{AQR}(x) = \begin{cases} \left(\frac{a - p_0^T}{std(X)} \cdot (\gamma)\right) + p_0^S & x < p_1^T \\ \left(\frac{a - p_{99}^T}{std(X)} \cdot (\gamma)\right) + p_{99}^S & x \geq p_{99}^T \end{cases}, \tag{5}$$

where X is all points in a specific channel of a specific layer of a batch at test time and $std(X)$ is the standard deviation of X. The idea is that standard deviation provides a more stable measure of data spread and applies the remapping rule of Eq. 5.

Table 2 reports classification accuracies for several tail-calibration strategies evaluated with ResNet-50 (BN) on the ImageNet-C dataset, across two batch sizes on all corruption types. The results demonstrate that precise model-

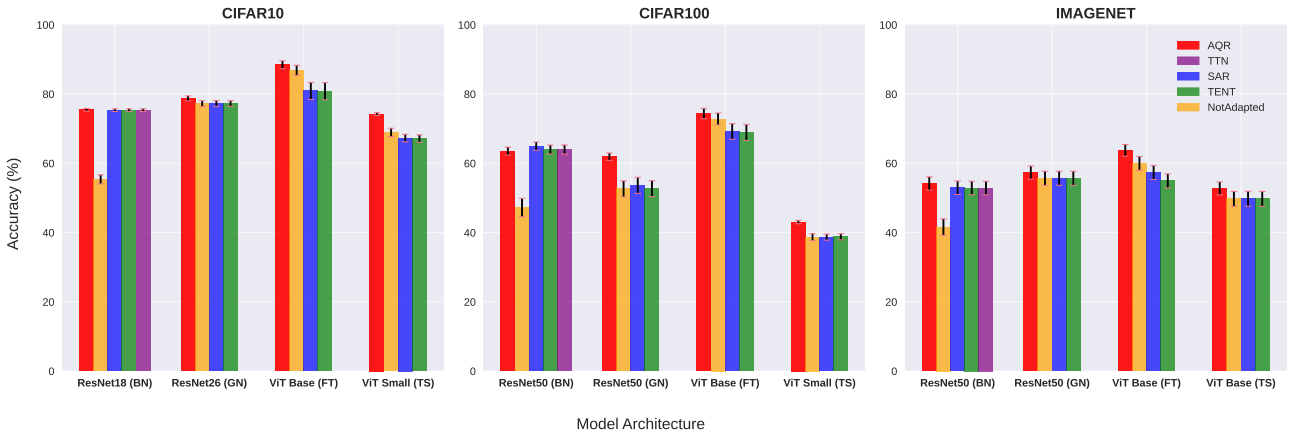

Figure 5. Architecture-specific performance comparison at corruption severity level 3. AQR demonstrates consistent improvements across diverse architectures on three datasets (CIFAR-10-C, CIFAR-100-C, ImageNet-C), including ResNets with different normalization schemes (BN, GN) and ViTs (LN). Error bars represent standard error across corruption types and batch sizes

Table 3. Impact of percentile granularity on AQR performance. Comparison between using 101 percentiles (standard AQR) versus 11 percentiles on ResNet-50 (BN) on ImageNet-C.

Method	**Batch Size = 128**		**Batch Size = 512**	
	Severity 3	Severity 5	Severity 3	Severity 5
AQR	53.92 ± 11.79	33.72 ± 16.33	54.40 ± 11.69	34.19 ± 16.37
AQR (10 qds)	46.37 ± 13.06	27.00 ± 16.07	50.68 ± 12.54	29.77 ± 16.49

ing of extreme percentiles is essential for robustness. "Average Sample Tails" attains the highest accuracy at both batch sizes, marginally outperforming "Gaussian Estimation" and considerably exceeding the standard AQR baseline; this gap is most pronounced at the smaller batch size (128), where tail statistics are most volatile. "Interval Estimation" offers only a modest benefit, whereas aggressive "Clipping" severely degrades performance. Collectively, these findings confirm that nuanced calibrating of the values at the tails of the distribution is required and suggest "Average Sample Tails" as the most dependable approach.

4.5. Granularity of Percentiles

We also performed an ablation study to investigate how the granularity of the percentiles affects AQR performance. To this end, we computed fewer percentiles at setup phase to understand the trade-off between computational complexity and the performance of adaptation. Specifically, we compared using 11 percentiles ($p_0, p_{10}, p_{20}, \ldots, p_{100}$) and our standard 101 percentiles ($p_0, p_1, p_2, \ldots, p_{100}$) on ResNet50 (BN) with ImageNet-C. The results show that finer granularity leads to better performance, with 101 percentiles achieving a much better accuracy.

5. 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, CIFAR-100-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

This work was partially funded by FRQNT-NSERC grant 2023-NOVA-329759 [E.B.]; FRQNT-NSERC grant 2023-NOVA-329125 [E.B. & G.W.]; Canada CIFAR AI Chair, NSF DMS grant 2327211 and NSERC Discovery grant 03267 [G.W.]. G.N. is supported by NSERC grant number 599788-2025. This work is also supported by resources from Compute Canada and Calcul Québec. The content is solely the responsibility of the authors and does not necessarily represent the views of the funding agencies.

References

[1] Matthew Amodio, David van Dijk, Ruth Montgomery, Guy Wolf, and Smita Krishnaswamy. Out-of-sample extrapolation with neuron editing, 2019. 2, 3

[2] Lei Jimmy Ba, Ryan Kiros, and Geoffrey E Hinton. Layer normalization. corr abs/1607.06450 (2016). *arXiv preprint arXiv:1607.06450*, 2016. 1

[3] Xin Gu. Less is more: Revisiting the role of batch normalization in test-time adaptation. In *2024 3rd International Conference on Cloud Computing, Big Data Application and Software Engineering (CBASE)*, pages 49–53, 2024. 2

[4] Humza Wajid Hameed, Geraldin Nanfack, and Eugene Belilovsky. Not only the last-layer features for spurious correlations: All layer deep feature reweighting. *arXiv preprint arXiv:2409.14637*, 2024. 2

[5] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019. 5

[6] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. corr abs/1502.03167. *arXiv preprint arXiv:1502.03167*, 2015. 1

[7] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. 5

[8] Jonghyun Lee, Dahuin Jung, Saehyung Lee, Junsung Park, Juhyeon Shin, Uiwon Hwang, and Sungroh Yoon. Entropy is not enough for test-time adaptation: From the perspective of disentangled factors. In *The Twelfth International Conference on Learning Representations*, 2024. 1

[9] Jae-Hong Lee and Joon-Hyuk Chang. Continual momentum filtering on parameter space for online test-time adaptation. In *The Twelfth International Conference on Learning Representations*, 2024. 1

[10] Yanghao Li, Naiyan Wang, Jianping Shi, Jiaying Liu, and Xiaodi Hou. Revisiting batch normalization for practical domain adaptation. *arXiv preprint arXiv:1603.04779*, 2016. 2

[11] Jian Liang, Ran He, and Tieniu Tan. A comprehensive survey on test-time adaptation under distribution shifts. *International Journal of Computer Vision*, 133(1):31–64, 2025. 1

[12] Hyesu Lim, Byeonggeun Kim, Jaegul Choo, and Sungha Choi. Ttn: A domain-shift aware batch normalization in test-time adaptation. *arXiv preprint arXiv:2302.05155*, 2023. 2

[13] Kuang Liu. pytorch-cifar. https://github.com/kuangliu/pytorch-cifar, 2020. Accessed: 2024-07-18. 6

[14] Robert A. Marsden, Mario Döbler, and Bin Yang. Universal test-time adaptation through weight ensembling, diversity weighting, and prior correction, 2023. 1

[15] Zachary Nado, Shreyas Padhy, D Sculley, Alexander D'Amour, Balaji Lakshminarayanan, and Jasper Snoek. Evaluating prediction-time batch normalization for robustness under covariate shift, 2020. 1, 2, 6

[16] Geraldin Nanfack and Eugene Belilovsky. Fairdropout: Using example-tied dropout to enhance generalization of minority groups. *arXiv preprint arXiv:2502.06695*, 2025. 2

[17] Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Zhiquan Wen, Yaofo Chen, Peilin Zhao, and Mingkui Tan. Towards stable test-time adaptation in dynamic wild world, 2023. 2, 6

[18] Steffen Schneider, Evgenia Rusak, Luisa Eck, Oliver Bringmann, Wieland Brendel, and Matthias Bethge. Improving robustness against common corruptions by covariate shift adaptation. *Advances in Neural Information Processing Systems*, 33, 2020. 1, 2, 6

[19] Pedro Vianna, Muawiz Chaudhary, Paria Mehrbod, An Tang, Guy Cloutier, Guy Wolf, Michael Eickenberg, and Eugene Belilovsky. Channel-selective normalization for label-shift robust test-time adaptation, 2024. 1

[20] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno A. Olshausen, and Trevor Darrell. 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. 2, 6

[21] Zixin Wang, Yadan Luo, Liang Zheng, Zhuoxiao Chen, Sen Wang, and Zi Huang. In search of lost online test-time adaptation: A survey, 2024. 1

[22] Ross Wightman. Pytorch image models. https://github.com/rwightman/pytorch-image-models, 2019. 6

[23] Yanan Wu, Zhixiang Chi, Yang Wang, Konstantinos N. Plataniotis, and Songhe Feng. Test-time domain adaptation by learning domain-aware batch normalization. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'24/IAAI'24/EAAI'24. AAAI Press, 2024. 2

[24] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 3–19, 2018. 1

[25] Kentaro Yoshioka. vision-transformers-cifar10: Training vision transformers (ViT) and related models on CIFAR-10. https://github.com/kentaroy47/vision-transformers-cifar10, 2024. GitHub repository. 6

[26] Longhui Yuan, Binhui Xie, and Shuang Li. Robust test-time adaptation in dynamic scenarios. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15922–15932, 2023. 2

[27] Marvin Zhang, Sergey Levine, and Chelsea Finn. MEMO: Test Time Robustness via Adaptation and Augmentation, Oct. 2022. 2, 6

[28] Bowen Zhao, Chen Chen, and Shu-Tao Xia. Delta: Degradation-free fully test-time adaptation. In *The Eleventh International Conference on Learning Representations*. 2