NanoAdapt: Mitigating Negative Transfer in Test Time Adaptation with Extremely Small Batch Sizes

Shiji Zhao , Shao-Yuan Li *, Sheng-Jun Huang
College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics
MIIT Key Laboratory of Pattern Analysis and Machine Intelligence
{zhaosj, lisy, huangsj}@nuaa.edu.cn

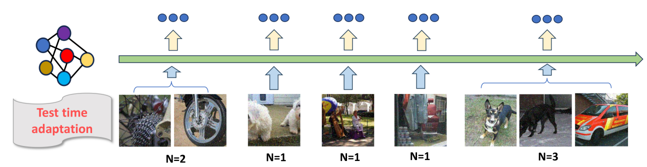
Abstract

Test Time Adaptation (TTA) has garnered significant attention in recent years, with the research focus on addressing distribution shifts during test time. As one fundamental component of many TTA methods, the Batch Normalization (BN) layer plays a crucial role in enabling the model adaptability. However, existing BN strategies can prove detrimental when the batch size is (extremely) small. In numerous real-world scenarios, limited hardware resources or just-in-time demand often necessitates adjusting models with very small batch sizes, making existing methods less practical. In this paper, we first showcase and thoroughly analyze the negative transfer phenomenon in previous TTA methods encountering extremely small batch sizes. Subsequently, we propose a novel batch size-agnostic method called NanoAdapt to effectively mitigate the negative transfer even with batch size 1. NanoAdapt is composed of three key components: a dynamic BN calibration strategy that leverages historical information and the Taylor series to refine the statistics estimations, an entropy-weighted gradient accumulation strategy that uses the entropy of each sample's label prediction to weigh and accumulate the loss for backpropagation, and a novel proxy computation graph to capture the sample interactions. Extensive experiments are conducted to validate the superiority of NanoAdapt, showing its consistent efficacy in improving existing TTA methods.

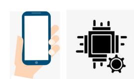
1 Introduction

Deep neural networks have demonstrated remarkable performance across various machine learning problems, particularly in image classification. However, they frequently exhibit brittleness and vulnerability to challenges arising from data distribution shifts. Instances of these challenges include a rapid decline in accuracy for deep image classifiers when confronted with input perturbations, such as noise or blur

*Corresponding author: Shao-Yuan Li.



Batch size may be small !



Cifar100-C Error Rate with Different Batch Sizes

	N=200	N=100	N=10	N=5	N=2	N=1
Tent	35.52	34.58	45.27	79.58	97.61	99.00
+NanoAdapt	36.05	35.38	34.20	34.61	38.37	40.05

Figure 1: Top: resource-constrained test time adaptation scenarios with small batch size. Bottom: Noticeable performance decline of one representative pioneering TTA approach TENT [Wang *et al.*, 2021] as the batch size decreases from N=200 to N=1. Whereas our proposed NanoAdapt effectively enhances TENT even in case of extremely small batch size N=1.

[Hendrycks and Dietterich, 2019]. Consequently, the robustification of deep models against such test shifts constitutes a crucial and actively researched area.

Test time adaptation (TTA) is an emerging frontier that addresses the challenge of adjusting pre-trained models to unforeseen shifts in distribution. It is particularly crucial as it enables models to perform effectively on unlabeled data encountered during deployment. In the pursuit to adapt models during testing, various methodologies have been explored, including test time normalization [Nado *et al.*, 2020; Schneider *et al.*, 2020], entropy minimization [Wang *et al.*, 2021], self-supervised learning [Sun *et al.*, 2020a; Liu *et al.*, 2021], contrastive learning [Chen *et al.*, 2022], data augmentation [Zhang *et al.*, 2022a], uncertainty-aware optimization [Niu *et al.*, 2022], etc.

Recently, some studies have focused on addressing the challenges of TTA in intricate test environments, particularly in Out-of-Distribution scenarios [Sun *et al.*, 2020b; Li *et al.*, 2023; Zhou *et al.*, 2023], as well as in online continual adaptation [Boudiaf *et al.*, 2022; Wang *et al.*, 2022; Niu *et al.*, 2022; Zhang *et al.*, 2023]. However, there is a noticeable gap in the literature concerning the issue of TTA in resource-constrained scenarios. As depicted in Fig. 1, resource-constrained industrial environments with limited hardware resources such as mobile devices, or just-in-time adaptation needs for individualized models, often necessitate adjusting models with very small batch sizes. However, existing TTA methods relying on the Batch Normalization layer encounter significant challenges in such situations, as the conventional BN relies on a sufficiently large enough batch size to accurately estimate the data distribution statistics. When the batch sizes are small, they tend to degrade drastically, leading to severe negative transfer. Fig. 1 shows the noticeable performance decline of one representative pioneering TTA approach TENT [Wang *et al.*, 2021] as the batch size decreases from N=200 to N=1. With extremely small batch sizes N=2, 1, the model collapses with error rates exceeding 97%.

Previously only few works have explored the small batch size challenge of BN in TTA. SAR [Niu *et al.*, 2023] highlighted that existing TTA methods struggle to handle small batch sizes. It sidestepped the issues associated with BN by opting for Group Normalization (GN) over BN in the pre-trained training model. However, what if a pre-trained model already includes BN modules and how do we address this issue? Delta [Zhao *et al.*, 2023] introduced test time renormalization to refine the statistical estimation in BN. While this proves helpful in addressing situations with small batch sizes, it still falls short in extremely small batch size cases. Besides, gradient accumulation is a common technique during the training phase to tackle challenges arising from small batch sizes. It entails accumulating gradients computed across multiple batches and updating them collectively. However, simple gradient accumulation alone does not mitigate the negative transfer phenomenon in TTA methods when confronted with small batch sizes.

To the best of our knowledge, our study is the first to investigate TTA with extremely small batch sizes. We first conduct a thorough analysis to comprehend why previous approaches relying on Batch Normalization struggle to adapt the model during test time. We found that when the batch size is particularly small, there is significant discrepancy between the BN statistics of the current batch and the entire test dataset. Additionally, the gradient computed from small batches is also highly unreliable. Then we introduce our NanoAdapt approach to mitigate this issue. NanoAdapt comprises three main components: Dynamic BN Calibration (DBC), Entropy-Weighted Gradient Accumulation (EWGA), and Proxy Computation Graph (PCG). DBC harnesses historical information and employs the Taylor series to correct the statistics estimations. EWGA uses the entropy of each sample's label prediction to weigh the loss and accumulates them for backpropagation. PCG employs different data augmentations to construct a proxy computation graph for the forward process, capturing the sample interactions for backpropagation.

In summary, our contributions are as follows:

- We showcase and give a thorough analysis of the negative transfer phenomenon in prior TTA methods with extremely small batch sizes, showing that inaccurate statistical estimates and the unreliable gradients collectively contribute to this phenomenon.
- We have designed three powerful tools: DBC, EWGA, and PCG to respectively solve these issues. DBC aims to correct BN statistics at each time step. EWGA focuses on reduce noise in gradients and PCG is designed to capture sample interactions in gradients.
- We evaluate NanoAdapt on three distribution shift benchmark datasets, showing that it consistently mitigates negative transfer for existing TTA methods with (extremely) small batch sizes.

2 Related Works

Unsupervised Domain Adaptation (UDA) Unsupervised domain adaptation (UDA) seeks to enhance models' capacity for generalization to target domain data, even when no labeled data is available in the target domain. UDA commonly involves methods such as feature alignment [Zellinger *et al.*, 2017; Long *et al.*, 2017], adversarial training [Ganin and Lempitsky, 2015; Yi-Hsuan Tsai, 2018; Ganin *et al.*, 2016] and self-supervised training [Hoyer *et al.*, 2022; Zou *et al.*, 2018]. In more stringent scenarios, such as Source-Free Domain Adaptation (SFDA), it is not feasible to acquire source domain data, and only the source domain model is available [Liang *et al.*, 2020; Qiu *et al.*, 2021; Ding *et al.*, 2022]. However, SFDA still faces limitations in addressing a more realistic scenario where the distribution of the target domain is unknown before testing commences.

Domain Generalization (DG) Domain generalization (DG) focuses on developing models capable of learning from multiple domains and performing well on unseen testing domains. Techniques such as meta-learning [Li *et al.*, 2019; Li *et al.*, 2018a], data augmentation [Prakash *et al.*, 2019; Nam *et al.*, 2021], and domain alignment [Li *et al.*, 2018b], style augmentation [Li *et al.*, 2022; Kang *et al.*, 2022; Zhang *et al.*, 2022b] are employed. However, DG is not capable of generalizing to all target domains without any prior knowledge of the specific target domain.

Test Time Adaptation (TTA) Test Time Adaptation (TTA) focuses on dynamically adapting a pre-trained model in real-time. This adaptation occurs as a test data stream in batches, offering a more agile and online approach to model refinement. Pred BN [Nado *et al.*, 2020] replaces the normalization statistics computed during training with those derived from the test mini-batch. TENT [Wang *et al.*, 2021] optimizes the affine parameters in Batch Normalization through entropy minimization during testing. Long-term test time adaptation in dynamically changing environments is explored by EATA [Niu *et al.*, 2022] and CoTTA [Wang *et al.*, 2022]. AdaContrast [Chen *et al.*, 2022] employs contrastive learning to enhance feature learning, incorporating a pseudo label refinement mechanism. LAME [Boudiaf *et al.*, 2022] adjusts the model's output probabilities via the Laplacian adjusted maximum-likelihood estimation. SAR [Niu *et al.*, 2023] replaces BN with GN in the training phase and introduces a sharpness-aware and reliable entropy minimization method. OWTTT [Li *et al.*, 2023] develops adaptive strong OOD pruning and proposes a method to dynamically expand prototypes to represent robust OOD samples. In various real-world scenarios, the demand for constrained hardware resources or just-in-time adaptation often necessitates adapting models with small batch sizes. However, prior research has largely overlooked the challenges associated with adaptation under small batch sizes. Delta [Zhao *et al.*, 2023] introduces a test time batch renormalization technique to enhance estimated normalization statistics, but it falls short of fully resolving this issue. Our proposed method, NanoAdapt, effectively addresses this problem and mitigates the occurrence of the negative transfer phenomenon.

3 Problem Statement and Analysis

3.1 TTA Problem Definition

In the context of Test Time Adaptation, we denote the training data as $D_{\text{train}} = \{(x_i, y_i)\}_{i=1}^{N_{\text{train}}} \sim P_{\text{train}}(X, Y)$, where $x \in X$ represents the input, and $y \in Y$ represents the label. We denote the test data as $D_{\text{test}} = \{(x_j, y_j)\}_{j=1}^{N_{\text{test}}} \sim P_{\text{test}}(X, Y)$, where the labels $\{y_j\}$ are not available. It should be noted that the test data is delivered in a streaming fashion, organized into batches. We consider the common covariate distribution shift, which can be formalized as $P_{\text{train}}(X) \neq P_{\text{test}}(X)$ while $P_{\text{train}}(Y|X) = P_{\text{test}}(Y|X)$. A pre-trained model $f_{\{\theta_0, s_0\}}$ is characterized by model weights θ_0 and normalization statistics s_0 learned from D_{train}. It requires updates at each time step t to attain $f_{\{\theta_t, s_t\}}$ and generate label predictions for samples x_t. In real-world scenarios, the batch size of incoming test data may vary at each time step. In the most extreme scenario, each batch may comprise only one sample.

3.2 The Negative Transfer Phenomenon With Small Batch Sizes

We are familiar with the forward process of BN layers, which unfolds as follows:

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i, \quad \sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2,$$
$$\hat{x}_i = \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}}, \quad y_i = \gamma \hat{x}_i + \beta, \tag{1}$$

where x_i is the input and y_i is the output of the BN layer. μ and σ are the normalization statistics and γ and β are the affine parameters.

Traditionally, each BN layer records $\{\mu_0, \sigma_0\}$ in the training phase and directly applies them in the testing phase:

$$\mu = \mu_0, \quad \sigma = \sigma_0. \tag{2}$$

In previous TTA methods, notable works have implemented treatments specifically tailored for Batch Normalization layers. Taking the two most representative works as examples, Pred BN [Nado *et al.*, 2020] entails substituting the normalization statistics computed during training with those obtained from the test mini-batch. Meanwhile, TENT [Wang *et al.*, 2021] further minimize entropy and selectively updates only the affine parameters of the BN layers. Their BN statistics are all calculated using the information from the current batch at time step t:

$$\mu = \mu_t, \quad \sigma = \sigma_t. \tag{3}$$

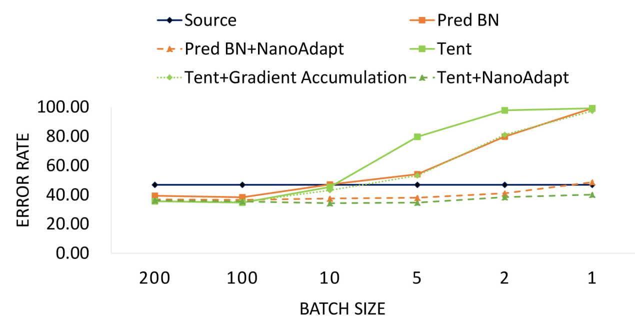


Figure 2: Error Rate on Cifar100-C. "Source" denotes the performance of the unadapted pre-trained ResNet-50 model.

When dealing with small batch sizes, the calculated statistics are not accurate, and the computed gradient is also unreliable.

As illustrated in Fig. 2, when the batch size exceeds 100, both Pred BN and Tent perform well. However, as the batch size decreases to 1, their performance declines. In the most severe scenarios when N=1, their error rate reaches 99%. Since Tent requires gradient updates, we employ gradient accumulation to assist it, which helps reduce the classification error but falls short of avoiding the negative transfer phenomenon. With the implementation of NanoAdapt, the negative transfer phenomenon can be effectively circumvented.

4 The Proposed Method: NanoAdapt

4.1 Dynamic BN Calibration

To address inaccurate statistics within each testing batch, a straightforward solution arises: employing the Exponential Moving Average(EMA) to update the statistics in each time step t:

$$\hat{\mu}_{\theta_t}(x_{1..t}) = m\hat{\mu}_{\theta_{t-1}}(x_{1..t-1}) + (1-m)\mu_{\theta_t}(x_t),$$
$$\hat{\sigma}^2_{\theta_t}(x_{1..t}) = m\hat{\sigma}^2_{\theta_{t-1}}(x_{1..t-1}) + (1-m)\sigma^2_{\theta_t}(x_t), \tag{4}$$

where m denotes the smoothing coefficient, $\mu_{\theta_t}(x_t)$ represents the mean value of each input x before the BN layers at time step t, $\hat{\mu}_{\theta_{t-1}}(x_{1..t-1})$ represents the aggregated mean estimates from past time steps 1 to t-1. $\hat{\mu}_{\theta_t}(x_{1..t})$ represents the final mean estimation with current model θ_t. The notation for the corresponding variance σ^2 has a similar interpretation.

Due to the second-order nature of σ^2, a more effective approach is to maintain the mean values of both x and x^2 for each feature map before the BN layer at time step t and calculate $\hat{\sigma}^2_t$ based on the corresponding relationship.

$$\hat{\mu}_{\theta_t}(x_{1..t}) = m\hat{\mu}_{\theta_{t-1}}(x_{1..t-1}) + (1-m)\mu_{\theta_t}(x_t),$$
$$\hat{\mu}_{\theta_t}(x^2_{1..t}) = m\hat{\mu}_{\theta_{t-1}}(x^2_{1..t-1}) + (1-m)\mu_{\theta_t}(x^2_t), \tag{5}$$
$$\hat{\sigma}^2_{\theta_t}(x_{1..t}) = \hat{\mu}_{\theta_t}(x^2_{1..t}) - \hat{\mu}^2_{\theta_t}(x_{1..t}).$$

Nonetheless, a notable issue arises with this computation that both $\hat{\mu}_{\theta_{t-1}}(x_{1..t-1})$ and $\hat{\mu}_{\theta_{t-1}}(x^2_{1..t-1})$ are computed using the previous model weights θ_{t-1}, For a more accurate estimation, these values should be computed under the current

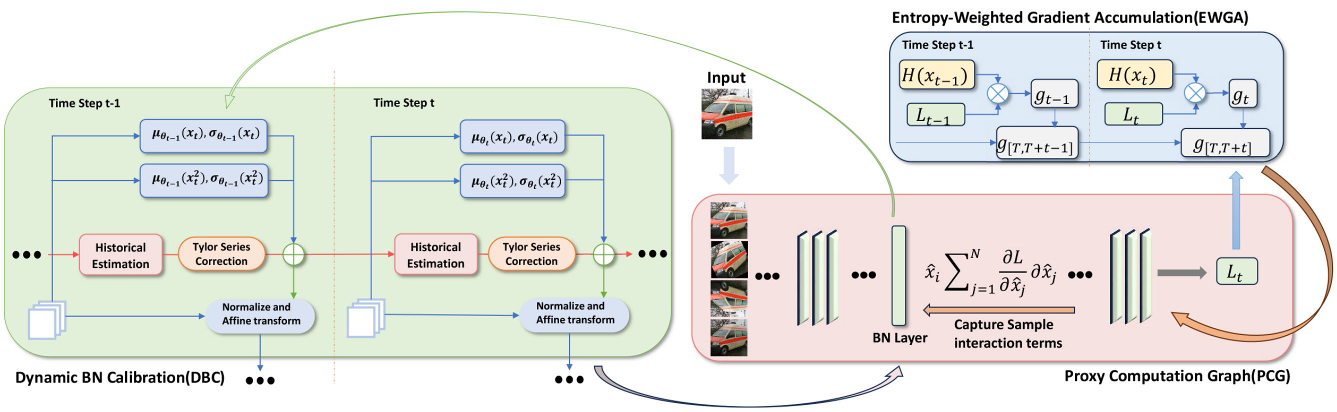


Figure 3: Overall structure of NanoAdapt. NanoAdapt consists of three components: Dynamic BN Calibration (DBC), Entropy-Weighted Gradient Accumulation (EWGA) and Proxy Computation Graph (PCG). DBC harnesses historical information and employs the Taylor series to correct the statistics estimations. EWGA uses the entropy of each sample's label prediction($H(x_t)$) to weigh the loss(L_t) and accumulates the gradients(g_t) for backpropagation. PCG employs different data augmentations to construct a proxy computation graph for the forward process, capturing the sample interaction terms for backpropagation.

model weights θ_t. In essence, what we seek are $\hat{\mu}_{\theta_t}(x_{1..t-1})$ and $\hat{\mu}_{\theta_t}(x^2_{1..t-1})$.

Given that the model weights θ_t are updated through gradient descent, when the learning rate is sufficiently small, the alterations in parameters at each time step t can be perceived as a continuous function. Hence, The Taylor polynomials can be employed to approximately compute $\hat{\mu}_{\theta_t}(x_{1..t-1})$ and $\hat{\mu}_{\theta_t}(x^2_{1..t-1})$ derived from past model weights θ_{t-1} under the current model weights θ_t.

$$
\begin{aligned}
\hat{\mu}_{\theta_t}(x_{1..t-1}) &= \hat{\mu}_{\theta_{t-1}}(x_{1..t-1}) + \frac{\partial \hat{\mu}_{\theta_{t-1}}(x_{1..t-1})}{\partial \theta_{t-1}}(\theta_t - \theta_{t-1}) \\
&\quad + \mathcal{O}(\|\theta_t - \theta_{t-1}\|^2), \\
\hat{\mu}_{\theta_t}(x^2_{1..t-1}) &= \hat{\mu}_{\theta_{t-1}}(x^2_{1..t-1}) + \frac{\partial \hat{\mu}_{\theta_{t-1}}(x^2_{1..t-1})}{\partial \theta_{t-1}}(\theta_t - \theta_{t-1}) \\
&\quad + \mathcal{O}(\|\theta_t - \theta_{t-1}\|^2),
\end{aligned} \tag{6}
$$

where $\mathcal{O}(\cdot)$ represents higher-order terms.

To expedite computation, we neglect higher-order terms. Additionally, empirical observations suggest a swift decline in the gradients of layers preceding the BN layer [Yao *et al.*, 2021]. So we focus solely on computing the gradients of the convolutional layer preceding the BN layer. We simplify Eq. (6) as follows:

$$
\begin{aligned}
\hat{\mu}_{\theta^l_t}(x_{1..t-1}) &\approx \hat{\mu}_{\theta^l_{t-1}}(x_{1..t-1}) + \frac{\partial \hat{\mu}_{\theta^l_{t-1}}(x_{1..t-1})}{\partial \theta^l_{t-1}}(\theta^l_t - \theta^l_{t-1}), \\
\hat{\mu}_{\theta^l_t}(x^2_{1..t-1}) &\approx \hat{\mu}_{\theta^l_{t-1}}(x^2_{1..t-1}) + \frac{\partial \hat{\mu}_{\theta^l_{t-1}}(x^2_{1..t-1})}{\partial \theta^l_{t-1}}(\theta^l_t - \theta^l_{t-1}),
\end{aligned} \tag{7}
$$

where the superscript l indicates that only the parameters of the layer preceding the BN layer l are considered. Therefore, the final dynamic BN calibration can be formalized as the following:

$$
\begin{aligned}
\hat{\mu}_{\theta_t}(x_{1..t}) &= m(\hat{\mu}_{\theta^l_{t-1}}(x_{1..t-1}) + \frac{\partial \hat{\mu}_{\theta^l_{t-1}}(x_{1..t-1})}{\partial \theta^l_{t-1}}(\theta^l_t - \theta^l_{t-1})) \\
&\quad + (1-m)\mu_{\theta_t}(x_t), \\
\hat{\mu}_{\theta_t}(x^2_{1..t}) &= m(\hat{\mu}_{\theta^l_{t-1}}(x^2_{1..t-1}) + \frac{\partial \hat{\mu}_{\theta^l_{t-1}}(x^2_{1..t-1})}{\partial \theta^l_{t-1}}(\theta^l_t - \theta^l_{t-1})) \\
&\quad + (1-m)\mu_{\theta_t}(x^2_t), \\
\hat{\sigma}^2_{\theta_t}(x_{1..t}) &= \hat{\mu}_{\theta_t}(x^2_{1..t}) - \hat{\mu}^2_{\theta_t}(x_{1..t}).
\end{aligned} \tag{8}
$$

The implementation is depicted in Figure 2. In each time step t, for each BN layer l, we apply the correction outlined in Eq. (8) to obtain estimates of the statistics $\hat{\mu}_{\theta_t}(x_{1..t})$ and $\hat{\sigma}_{\theta_t}(x_{1..t})$. These estimates are then applied to the BN layer, and the estimates at the current time step t are integrated into the subsequent time step t+1.

4.2 Entropy-Weighted Gradient Accumulation with Proxy Computation Graph

To deal with unreliable gradient with small batch sizes, we propose to employ an entropy-weighted gradient accumulation strategy that leverages entropy of each sample's label prediction to assess the reliability of the gradient computed by the loss. Subsequently, these gradients are accumulated for backpropagation. This can be formulated as follows:

$$
\begin{aligned}
L &= \sum_{i=1}^{S} \frac{r}{S} \cdot min(\frac{1}{H(x)}, 1) \cdot L_o(x), \\
r &= \frac{log(S) + log(i)}{2 \cdot log(S)},
\end{aligned} \tag{9}
$$

where $H(x)$ denotes the entropy of the sample's label prediction, L_o represents the original loss computed by TTA methods, S denotes the steps for gradient accumulation, L represents the corrected loss of x for gradient descent, and r is employed to progressively increase the weighting of subsequent step computations since their BN statistics are more accurate.

When combined with dynamic BN calibration proposed in Section 4.1, this approach is sufficiently effective in achieving results nearly comparable to those obtained with larger batch sizes. However, this success diminishes when the batch size is extremely small, such as N=1. In the following, we will give a formal analysis, and propose our *proxy* computation graph strategy to compensate this issue.

Analysis for Batch Size 1 Due to the forward process of BN layers shown in Eq. (1), we can compute the gradients across the BN layers as follows:

$$\begin{aligned}
&\frac{\partial L}{\partial \gamma} = \sum_{i=1}^{N} \frac{\partial L}{\partial y_i}\hat{x}_i, \quad \frac{\partial L}{\partial \beta} = \sum_{i=1}^{N} \frac{\partial L}{\partial y_i}, \quad \frac{\partial L}{\partial \hat{x}_i} = \sum_{i=1}^{N} \frac{\partial L}{\partial y_i}\gamma, \\
&\frac{\partial L}{\partial \sigma^2} = \sum_{i=1}^{N} \frac{\partial L}{\partial \hat{x}_i} \cdot (x_i - \mu) \cdot -\frac{1}{2}(\sigma^2+\epsilon)^{-\frac{3}{2}}, \\
&\frac{\partial L}{\partial \mu} = \frac{\partial L}{\partial \sigma^2}\frac{\sum_{i=1}^{m} -2(x_i-\mu)}{m} + \sum_{i=1}^{N} \frac{\partial L}{\partial \hat{x}_i}\frac{-1}{\sqrt{\sigma^2+\epsilon}}, \\
&\frac{\partial L}{\partial x_i} = \frac{1}{\sqrt{\sigma^2+\epsilon}}\frac{\partial L}{\partial \hat{x}_i} + \frac{1}{N}\frac{\partial L}{\partial \mu} + \frac{2(x_i-\mu)}{N}\frac{\partial L}{\partial \sigma^2}.
\end{aligned} \tag{10}$$

We can further simplify the gradient over x_i it as follows:

$$\frac{\partial L}{\partial x_i} = \frac{1}{N\sqrt{\sigma^2+\epsilon}}\left(N\frac{\partial L}{\partial \hat{x}_i} - \sum_{j=1}^{N}\frac{\partial L}{\partial \hat{x}_j} - \hat{x}_i\sum_{j=1}^{N}\frac{\partial L}{\partial \hat{x}_j}\hat{x}_j\right). \tag{11}$$

The gradient includes the interaction term $\hat{x}_i \sum_{j=1}^{N} \frac{\partial L}{\partial \hat{x}_j}\hat{x}_j$ between different samples within one batch. When N=1, the interaction term disappears, indicating that even if we employ dynamic BN calibration and accumulate entropy-weighted gradients, the computation graph remains different.

Proxy Computation Graph Strategy To build a proxy computation graph with sample interaction terms, we propose employing data augmentation to generate various views of a single sample x. As illustrated in Fig. 3, in the forward process, we randomly select k distinct data augmentations including rotation, random crop, flipping, grayscale, saturation to generate $\{\hat{x}_1, \hat{x}_2, \cdots, \hat{x}_k\}$ and combine them with the original image x to feed into the network and construct the proxy computation graph. Then we employ the dynamic BN calibration to exclusively replace the statistics while preserving the forward proxy computation graph.

$$\hat{\mu}, \hat{\sigma} = DBC(\mu, \sigma) \tag{12}$$

In other words, we fool the model into believing that the statistics are computed by the proxy computation graph. During the backpropagation process, the gradient will traverse through the substituted statistics in the proxy computation graph.

5 Experiments

5.1 Experimental Setup

Datasets We conduct experiments on three distribution shift datasets: Cifar10-C, Cifar100-C and ImageNet-C [Hendrycks and Dietterich, 2019], which are commonly used to evaluate the performance of test time adaptation. These datasets cover a comprehensive range of 15 distinct corruption types. For each corruption type, there are 5 severity levels, providing a nuanced evaluation of the models under varying degrees of data corruption.

Baselines Pred BN [Nado *et al.*, 2020] and Tent [Wang *et al.*, 2021] represent pioneering approaches in TTA, specifically addressing the adaptation of BN layers. Notably, Pred BN does not require gradient updates, while Tent necessitates gradient updates. Subsequent TTA works often build upon these two studies to make modifications in other directions. Our paper specifically addresses the issue that TTA methods relying on BN may exhibit negative transfer when the batch size is small. Therefore, we choose these two as baselines. Furthermore, we compare with Delta [Zhao *et al.*, 2023], the latest and so far the only method that addresses the inaccuracy of BN statistics.

Implementation Details We evaluate the results at the most severe corruption level (level 5). We use the ResNet-50 [He *et al.*, 2016] architecture as the pre-trained model for our experiments. The performance of ResNet-50 on each dataset is denoted as Source. For optimization, we utilize the Adam optimizer [Kingma and Ba, 2014] with a fixed learning rate of 0.001. In the dynamic BN calibration, we set the smoothing coefficient to $m = 0.98$. For the construction of the proxy computation graph, we opt for a value of $k = 3$ as the number of augmentations.

5.2 Comparative Analysis

Table 1 - 3 respectively report the results on the three distribution shift datasets. As observed in Table 1 and Table 2, when N=200, Pred BN and Tent perform well, and their error rates exhibited no significant change when augmented with Delta or NanoAdapt. However, with reduced batch size (N=10), both Pred BN and Tent experience a decline in performance, albeit still outperforming Source (without any adaptation). Applying Delta results in a partial improvement in performance, while applying NanoAdapt can nearly match or even slightly surpass the performance with N=200. When N=1, both Pred BN and Tent exhibit a negative transfer phenomenon. Delta proves effective in reducing the error rate with Pred BN, as it doesn't require gradient updates. However, its impact is limited with Tent, as the gradient from a single sample may introduce excessive noise. On the other hand, when NanoAdapt is applied, both Pred BN and Tent experience a substantial decrease in error rate, surpassing the Source results. This indicates that NanoAdapt can effectively address the negative transfer phenomenon even with a minimal batch size. ImageNet-C is a more challenging dataset. As shown in Table 3, NanoAdapt consistently outperforms Delta. The only pity is that when N=1, NanoAdapt can only assist Tent in reducing its error rate from 99.97% to 74.10%, which is still slightly lower than the Source. This suggests that on

Cifar10-C error rate on various batch sizes with corruption level 5																
Method	gauss	shot	impulse	defocus	glass	motion	zoom	snow	frost	fog	bright	contrast	elastic	pixel	jpeg	Average
Source	28.77	22.95	26.18	9.45	20.59	10.56	9.25	14.15	15.27	17.49	7.60	20.95	14.73	41.31	14.67	18.26
$N = 200$																
Pred BN	18.52	16.17	22.27	8.97	21.85	10.47	9.68	12.80	13.33	15.02	7.56	11.90	16.33	15.00	17.46	14.49
+Delta	18.45	16.05	21.73	10.24	22.77	10.69	9.62	13.32	13.22	16.10	7.74	12.82	16.40	16.72	17.25	14.87
+NanoAdapt	**15.22**	**13.15**	**18.40**	**9.11**	**19.12**	**9.30**	**7.87**	**10.46**	**9.63**	**12.89**	**6.34**	**9.18**	**13.93**	**13.93**	**15.25**	**12.25**
Tent	15.60	13.24	18.78	**7.93**	18.08	**8.98**	8.01	10.39	10.88	12.28	6.64	10.05	14.05	**11.40**	14.75	12.07
+Delta	14.83	13.71	19.99	10.30	19.74	10.09	8.19	11.05	11.15	13.87	7.29	12.31	14.77	14.34	14.79	13.09
+NanoAdapt	**13.51**	**12.33**	**17.16**	8.60	**16.87**	9.08	**7.59**	**9.46**	**9.40**	**10.50**	**6.35**	**8.37**	**12.87**	11.65	**13.35**	**11.14**
$N = 10$																
Pred BN	23.26	21.17	26.71	13.32	27.09	14.78	13.87	17.78	17.69	19.70	11.57	16.83	21.28	19.72	22.47	19.15
+Delta	19.56	17.16	23.48	9.72	22.99	11.24	10.30	13.68	14.35	16.11	8.24	13.76	17.08	15.88	18.38	15.46
+NanoAdapt	**15.76**	**13.91**	**19.34**	**8.43**	**19.46**	**9.69**	**8.31**	**10.71**	**10.05**	**12.92**	**6.61**	**9.10**	**14.24**	**13.16**	**16.05**	**12.52**
Tent	21.29	21.41	24.47	13.52	29.12	15.45	15.77	16.71	16.81	18.04	12.25	14.63	21.71	17.28	21.31	18.65
+Delta	16.83	15.79	20.84	10.09	24.14	11.66	9.83	13.44	12.44	14.36	8.80	13.59	17.08	14.51	16.70	14.67
+NanoAdapt	**13.90**	**11.99**	**16.95**	**7.91**	**17.03**	**8.67**	**7.55**	**9.76**	**9.15**	**10.29**	**6.32**	**8.02**	**13.02**	**10.73**	**13.66**	**11.00**
$N = 1$																
Pred BN	90.00	90.00	90.00	90.00	90.00	90.00	90.00	90.00	90.00	90.00	90.00	90.00	90.00	90.00	90.00	90.00
+Delta	30.04	28.02	33.75	19.22	34.26	21.61	20.07	24.35	26.22	25.83	18.12	60.53	28.16	26.39	30.66	28.48
+NanoAdapt	**23.01**	**20.91**	**25.95**	**14.53**	**27.31**	**16.20**	**14.28**	**17.54**	**16.54**	**19.62**	**11.94**	**15.98**	**22.13**	**19.99**	**22.62**	**19.24**
Tent	90.00	90.00	90.00	90.00	90.00	90.00	90.00	90.00	90.00	90.00	90.00	90.00	90.00	90.00	90.00	90.00
+Delta	78.67	80.59	80.72	76.42	79.00	75.69	70.63	72.06	70.35	78.94	66.82	81.49	79.81	74.80	77.83	76.25
+NanoAdapt	**17.91**	**15.79**	**21.37**	**9.39**	**21.42**	**10.67**	**9.53**	**12.20**	**11.22**	**14.35**	**7.55**	**9.09**	**16.59**	**14.97**	**17.80**	**13.99**

Table 1: Error rate on Cifar10-C. Bold value represents the best result.

Cifar100-C error rate on various batch sizes with corruption level 5																
Method	gauss	shot	impulse	defocus	glass	motion	zoom	snow	frost	fog	bright	contrast	elastic	pixel	jpeg	Average
Source	65.58	60.05	59.10	32.05	50.94	33.56	32.53	41.39	45.15	51.38	31.63	55.53	40.28	59.72	42.45	46.76
$N = 200$																
Pred BN	44.31	44.03	47.35	32.11	45.85	32.84	33.03	38.36	37.90	45.38	29.86	36.48	40.58	36.71	44.10	39.26
+Delta	44.31	43.52	46.81	33.79	46.90	33.28	32.65	38.90	38.00	46.68	29.85	37.52	41.03	38.75	44.01	39.73
+NanoAdapt	**41.28**	**40.92**	**43.92**	**31.73**	**43.56**	**31.05**	**29.77**	**35.59**	**34.22**	**43.13**	**27.51**	**33.75**	**38.71**	**35.12**	**41.29**	**36.77**
Tent	40.11	39.50	41.95	**29.65**	41.85	30.72	29.67	34.45	34.76	39.04	27.43	32.99	37.55	32.88	40.23	35.52
+Delta	39.72	40.30	42.93	32.00	43.60	32.07	30.16	35.73	35.36	41.76	29.26	35.56	39.01	37.08	40.79	37.02
+NanoAdapt	**38.51**	**38.29**	**39.80**	29.77	**40.62**	**30.04**	**28.31**	**33.47**	**33.12**	**36.98**	**26.59**	**30.97**	**36.53**	**32.61**	**38.67**	**34.29**
$N = 10$																
Pred BN	52.62	50.78	54.64	39.62	53.67	40.22	40.83	46.16	45.77	52.73	37.80	44.17	48.71	45.44	51.63	46.99
+Delta	45.71	45.31	48.40	33.56	47.62	34.32	33.93	40.30	39.68	46.77	31.58	39.56	41.90	38.34	45.74	40.85
+NanoAdapt	**42.44**	**41.98**	**45.13**	**31.29**	**43.68**	**31.57**	**31.15**	**36.63**	**35.14**	**43.15**	**28.25**	**34.29**	**39.08**	**34.78**	**42.40**	**37.40**
Tent	52.68	48.24	49.14	38.85	52.06	40.59	39.39	43.42	45.69	46.25	37.39	44.59	47.78	41.98	51.01	45.27
+Delta	42.65	42.05	41.20	31.78	45.12	33.47	32.11	36.58	38.03	39.47	31.07	40.22	39.67	33.84	42.40	37.98
+NanoAdapt	**38.92**	**38.50**	**40.61**	**29.08**	**40.27**	**29.72**	**28.41**	**33.26**	**33.33**	**37.18**	**26.59**	**30.00**	**36.25**	**31.89**	**38.95**	**34.20**
$N = 1$																
Pred BN	99.04	98.97	98.92	99.13	99.01	99.01	99.08	99.05	99.05	99.05	99.06	99.10	98.99	99.11	99.00	99.04
+Delta	61.18	60.56	62.98	50.12	62.79	51.79	51.12	56.01	57.24	60.05	49.16	88.39	58.48	54.96	61.87	59.11
+NanoAdapt	**52.60**	**51.99**	**55.39**	**41.63**	**54.87**	**43.15**	**42.57**	**48.22**	**46.42**	**53.11**	**39.26**	**48.42**	**50.56**	**46.68**	**53.40**	**48.55**
Tent	99.00	99.00	99.00	99.00	99.00	99.00	99.00	99.00	99.00	99.00	99.00	99.00	99.00	99.00	99.00	99.00
+Delta	93.86	94.35	94.17	91.00	93.54	91.87	90.76	91.99	91.87	93.61	90.36	96.38	92.58	91.32	93.09	92.72
+NanoAdapt	**46.38**	**45.65**	**49.51**	**33.45**	**46.72**	**33.85**	**33.17**	**38.85**	**37.65**	**46.52**	**30.30**	**34.47**	**41.98**	**37.08**	**45.15**	**40.05**

Table 2: Error rate on Cifar100-C. Bold value represents the best result.

ImageNet-C error rate on various batch sizes with corruption level 5																
Method	gauss	shot	impulse	defocus	glass	motion	zoom	snow	frost	fog	bright	contrast	elastic	pixel	jpeg	Average
Source	94.32	88.44	94.64	74.40	82.94	70.92	64.66	76.82	72.12	76.10	39.36	84.32	71.40	59.26	52.14	73.46
$N = 50$																
Pred BN	71.68	68.54	71.66	**70.30**	69.52	54.24	48.08	**54.54**	58.58	47.76	31.06	**69.60**	**44.28**	41.04	50.70	56.77
+Delta	70.70	68.84	70.92	85.82	68.30	**53.98**	47.46	55.08	61.16	47.68	36.58	73.90	45.44	40.76	49.42	58.40
+NanoAdapt	**69.34**	**66.86**	**68.94**	75.80	**67.44**	54.16	**46.76**	55.04	**56.98**	**47.36**	**30.82**	70.36	47.46	**40.20**	**48.82**	**56.42**
Tent	61.26	**58.06**	**59.10**	61.04	60.16	**45.44**	42.28	45.78	**50.82**	40.16	30.92	**53.90**	40.48	37.10	43.52	48.67
+Delta	59.46	58.94	62.36	97.32	60.44	47.14	41.74	47.20	52.20	42.38	33.88	70.98	44.38	38.82	43.16	53.36
+NanoAdapt	**58.72**	59.34	59.60	**60.18**	**58.62**	45.48	**41.14**	**44.82**	52.02	**39.20**	**30.12**	54.22	**39.74**	**36.62**	**42.14**	**48.13**
$N = 10$																
Pred BN	77.98	75.82	77.26	77.58	76.86	63.62	57.10	61.68	64.86	55.82	40.18	74.88	53.08	50.40	59.68	64.45
+Delta	74.20	72.76	72.48	76.12	72.34	59.46	51.14	58.54	60.84	49.98	33.00	79.44	49.34	46.96	53.78	60.69
+NanoAdapt	**69.64**	**67.42**	**68.68**	**72.86**	**66.94**	**54.06**	**47.44**	**56.20**	**57.02**	**46.86**	**32.50**	**73.98**	**45.84**	**40.26**	**51.06**	**56.72**
Tent	88.90	86.92	88.58	86.72	89.88	74.74	71.28	69.74	79.52	66.76	47.48	85.42	63.16	55.86	70.70	75.04
+Delta	74.42	74.70	72.82	95.08	75.96	58.28	52.68	55.36	65.16	48.80	39.10	89.82	51.76	47.72	51.94	63.57
+NanoAdapt	**61.00**	**62.64**	**60.88**	**68.30**	**59.20**	**46.84**	**42.54**	**48.42**	**52.38**	**43.02**	**30.06**	**70.06**	**41.52**	**37.58**	**44.12**	**51.24**
$N = 1$																
Pred BN	99.96	99.96	99.94	99.98	99.92	99.96	99.92	99.96	99.94	99.94	99.94	99.96	99.96	99.94	99.92	99.95
+Delta	91.14	91.90	88.02	92.34	92.70	91.16	83.38	8.04	90.22	81.16	68.72	99.72	82.46	79.46	83.06	81.57
+NanoAdapt	**73.22**	**70.60**	**71.34**	**70.20**	**74.34**	**62.76**	**54.80**	**59.92**	**61.02**	**50.62**	**37.44**	**78.14**	**51.86**	**48.24**	**56.90**	**61.43**
Tent	99.88	99.98	99.98	99.98	99.98	99.98	99.86	99.98	99.98	99.98	99.98	99.98	99.98	99.98	99.98	99.97
+Delta	99.68	99.72	99.40	99.64	99.52	99.58	98.86	99.04	99.56	99.12	98.10	99.60	99.26	99.06	99.06	99.28
+NanoAdapt	**77.72**	**77.82**	**85.96**	**82.84**	**86.20**	**73.44**	**70.88**	**72.02**	**76.62**	**68.92**	**48.30**	**89.82**	**68.04**	**61.94**	**70.92**	**74.10**

Table 3: Error rate on ImageNet-C. Bold value represents the best result.

more challenging datasets like ImageNet-C, NanoAdapt prevents Tent from experiencing negative transfer but may not enable it to achieve nearly equivalent results to larger batch sizes.

5.3 Ablation Study of Different Components

We conducted a comprehensive ablation study on each component within NanoAdapt. As illustrated in Table 4, Tent performs poorly as the batch size decreases in the absence of NanoAdapt. Both Dynamic BN Calibration (DBC) and Entropy-Weighted Gradient Accumulation (EWGA) prove to be beneficial, although they may not assist in extreme situations when N=1. Combining DBC and EWGA effectively resolves the negative transfer phenomenon. Furthermore, incorporating the Proxy Computation Graph (PCG) results in improved performance.

DBC	EWGA	PCG	N=200	N=10	N=1
-	-	-	35.52	46.99	99.04
✓	-	-	36.07	36.89	90.95
-	✓	-	35.51	43.62	97.20
✓	✓	-	36.07	35.91	43.66
✓	✓	✓	**34.29**	**34.20**	**40.05**

Table 4: Average Error Rate on Cifar100-C (Corruption Level 5) when Tent Combines Each Component of NanoAdapt

5.4 Hyperparameter Sensitivity

As depicted in Table 5, we evaluate Tent+NanoAdapt with N=1, varying DBC smoothing factor m from 0.9, 0.92, 0.94, 0.96 to 0.98. In our experiments, $m = 0.98$ yields optimal performance. Additionally, we explore different selections of augmentation views k, ranging from 1 to 4. Notably, we observe that when k reaches 3, the performance peaks.

m	0.9	0.92	0.94	0.96	0.98
Error Rate	52.74	50.11	47.28	44.32	40.05
k	0	1	2	3	4
Error Rate	43.66	41.48	41.06	40.05	40.99

Table 5: Average Error Rate on Cifar100-C (Corruption level 5, N=1) with Tent+NanoAdapt under different hyperparameters

6 Conclusion

In this paper, we showcase the presence of negative transfer in previous test time adaptation methods with extremely small batch sizes. We attribute this issue to inaccurate BN statistics estimates and unreliable gradient updates. To address these challenges, we propose NanoAdapt, comprising three key components: dynamic BN calibration, entropy-weighted gradient accumulation and proxy computation graph. Dynamic BN calibration utilizes historical information to refine the estimation of Batch Normalization statistics, incorporating the Taylor series for additional correction. Entropy-weighted gradient accumulation employs entropy to weigh the loss and accumulate the gradients for backpropagation. The proxy computation graph is constructed to capture sample interaction terms in backpropagation, enhancing NanoAdapt's capabilities. We evaluate NanoAdapt on three distribution shift datasets, demonstrating its efficiency in mitigating the negative transfer phenomenon with minimal batch sizes.

Acknowledgments

This work is supported by National Science and Technology Major Project (2022ZD0114801).

References

[Boudiaf *et al.*, 2022] Malik Boudiaf, Romain Mueller, Ismail Ben Ayed, and Luca Bertinetto. Parameter-free online test-time adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8344–8353, June 2022.

[Chen *et al.*, 2022] Dian Chen, Dequan Wang, Trevor Darrell, and Sayna Ebrahimi. Contrastive test-time adaptation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[Ding *et al.*, 2022] Ning Ding, Yixing Xu, Yehui Tang, Chao Xu, Yunhe Wang, and Dacheng Tao. Source-free domain adaptation via distribution estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7212–7222, 2022.

[Ganin and Lempitsky, 2015] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 1180–1189, 2015.

[Ganin *et al.*, 2016] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *J. Mach. Learn. Res.*, 17(1):2096–2030, jan 2016.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

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

[Hoyer *et al.*, 2022] Lukas Hoyer, Dengxin Dai, and Luc Van Gool. DAFormer: Improving network architectures and training strategies for domain-adaptive semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9924–9935, 2022.

[Kang *et al.*, 2022] Juwon Kang, Sohyun Lee, Namyup Kim, and Suha Kwak. Style neophile: Constantly seeking novel styles for domain generalization. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7120–7130, 2022.

[Kingma and Ba, 2014] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. 12 2014.

[Li *et al.*, 2018a] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M. Hospedales. Learning to generalize: Meta-learning for domain generalization. AAAI'18/IAAI'18/EAAI'18, 2018.

[Li *et al.*, 2018b] Haoliang Li, Sinno Jialin Pan, Shiqi Wang, and Alex C. Kot. Domain generalization with adversarial feature learning. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5400–5409, 2018.

[Li *et al.*, 2019] Da Li, Jianshu Zhang, Yongxin Yang, Cong Liu, Yi-Zhe Song, and Timothy Hospedales. Episodic training for domain generalization. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1446–1455, 2019.

[Li *et al.*, 2022] Xiaotong Li, Yongxing Dai, Yixiao Ge, Jun Liu, Ying Shan, and LINGYU DUAN. Uncertainty modeling for out-of-distribution generalization. In *International Conference on Learning Representations*, 2022.

[Li *et al.*, 2023] Yushu Li, Xun Xu, Yongyi Su, and Kui Jia. On the robustness of open-world test-time training: Self-training with dynamic prototype expansion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2023.

[Liang *et al.*, 2020] Jian Liang, Dapeng Hu, and Jiashi Feng. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *International Conference on Machine Learning (ICML)*, pages 6028–6039, 2020.

[Liu *et al.*, 2021] Yuejiang Liu, Parth Kothari, Bastien van Delft, Baptiste Bellot-Gurlet, Taylor Mordan, and Alexandre Alahi. Ttt++: When does self-supervised test-time training fail or thrive? In *Advances in Neural Information Processing Systems*, volume 34, pages 21808–21820. Curran Associates, Inc., 2021.

[Long *et al.*, 2017] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I. Jordan. Deep transfer learning with joint adaptation networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2208–2217, 06–11 Aug 2017.

[Nado *et al.*, 2020] Zachary Nado, Shreyas Padhy, D. Sculley, Alexander D'Amour, Balaji Lakshminarayanan, and Jasper Snoek. Evaluating prediction-time batch normalization for robustness under covariate shift. *ArXiv*, abs/2006.10963, 2020.

[Nam *et al.*, 2021] Hyeonseob Nam, HyunJae Lee, Jongchan Park, Wonjun Yoon, and Donggeun Yoo. Reducing domain gap by reducing style bias. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8686–8695, 2021.

[Niu *et al.*, 2022] Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Yaofo Chen, Shijian Zheng, Peilin Zhao, and Mingkui Tan. Efficient test-time model adaptation without forgetting. In *The Internetional Conference on Machine Learning*, 2022.

[Niu *et al.*, 2023] Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Zhiquan Wen, Yaofo Chen, Peilin Zhao, and Mingkui Tan. Towards stable test-time adaptation in dynamic wild world. In *Internetional Conference on Learning Representations*, 2023.

[Prakash *et al.*, 2019] Aayush Prakash, Shaad Boochoon, Mark Brophy, David Acuna, Eric Cameracci, Gavriel State, Omer Shapira, and Stan Birchfield. Structured domain randomization: Bridging the reality gap by context-aware synthetic data. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7249–7255, 2019.

[Qiu *et al.*, 2021] Zhen Qiu, Yifan Zhang, Hongbin Lin, Shuaicheng Niu, Yanxia Liu, Qing Du, and Mingkui Tan. Source-free domain adaptation via avatar prototype generation and adaptation. In *International Joint Conference on Artificial Intelligence*, 2021.

[Schneider *et al.*, 2020] 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, pages 11539–11551, 2020.

[Sun *et al.*, 2020a] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 9229–9248, 13–18 Jul 2020.

[Sun *et al.*, 2020b] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A. Efros, and Moritz Hardt. Test-time training for out-of-distribution generalization, 2020.

[Wang *et al.*, 2021] 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.

[Wang *et al.*, 2022] Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7191–7201, 2022.

[Yao *et al.*, 2021] Zhuliang Yao, Yue Cao, Shuxin Zheng, Gao Huang, and Stephen Lin. Cross-iteration batch normalization. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12326–12335, 2021.

[Yi-Hsuan Tsai, 2018] Samuel Schulter Kihyuk Sohn Ming-Hsuan Yang Manmohan Chandraker Yi-Hsuan Tsai, Wei-Chih Hung. Learning to adapt structured output space for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[Zellinger *et al.*, 2017] Werner Zellinger, Thomas Grubinger, Edwin Lughofer, Thomas Natschläger, and Susanne Saminger-Platz. Central moment discrepancy (CMD) for domain-invariant representation learning. In *International Conference on Learning Representations*, 2017.

[Zhang *et al.*, 2022a] Marvin Zhang, Sergey Levine, and Chelsea Finn. Memo: Test time robustness via adaptation and augmentation. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 38629–38642. Curran Associates, Inc., 2022.

[Zhang *et al.*, 2022b] Yabin Zhang, Minghan Li, Ruihuang Li, Kui Jia, and Lei Zhang. Exact feature distribution matching for arbitrary style transfer and domain generalization. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[Zhang *et al.*, 2023] Yifan Zhang, Xue Wang, Kexin Jin, Kun Yuan, Zhang Zhang, Liang Wang, Rong Jin, and Tieniu Tan. AdaNPC: Exploring non-parametric classifier for test-time adaptation. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pages 41647–41676, 23–29 Jul 2023.

[Zhao *et al.*, 2023] Bowen Zhao, Chen Chen, and Shu-Tao Xia. DELTA: DEGRADATION-FREE FULLY TEST-TIME ADAPTATION. In *The Eleventh International Conference on Learning Representations*, 2023.

[Zhou *et al.*, 2023] Zhi Zhou, Lan-Zhe Guo, Lin-Han Jia, Dingchu Zhang, and Yu-Feng Li. ODS: Test-time adaptation in the presence of open-world data shift. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 42574–42588, 23–29 Jul 2023.

[Zou *et al.*, 2018] Yang Zou, Zhiding Yu, BVK Vijaya Kumar, and Jinsong Wang. Unsupervised domain adaptation for semantic segmentation via class-balanced self-training. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 289–305, 2018.