mathematics

MDPI

Article

Advancing Model Generalization in Continuous Cyclic Test-Time Adaptation with Matrix Perturbation Noise

Jinshen Jiang [1], Hao Yang [2], Lin Yang [3,*] and Yun Zhou [1,*]

[1] National Key Laboratory of Information Systems Engineering, National University of Defense Technology, Changsha 410073, China; jiangjinshen22@nudt.edu.cn
[2] National Key Laboratory on Blind Signal Processing, Chengdu 610041, China; yanghao@nudt.edu.cn
[3] School of Information and Intelligent Engineering, University of Sanya Academician Guoliang Chen Team Innovation Center, University of Sanya, Sanya 572000, China
* Correspondence: susan_yanglin@sohu.com (L.Y.); zhouyun@nudt.edu.cn (Y.Z.)

Abstract: Test-time adaptation (TTA) aims to optimize source-pretrained model parameters to target domains using only unlabeled test data. However, traditional TTA methods often risk overfitting to the specific, localized test domains, leading to compromised generalization. Moreover, these methods generally presume static target domains, neglecting the dynamic and cyclic nature of real-world settings. To alleviate this limitation, this paper explores the continuous cyclic test-time adaptation (CycleTTA) setting. Our unique approach within this setting employs matrix-wise perturbation noise in batch-normalization statistics to enhance the adaptability of source-pretrained models to dynamically changing target domains, without the need for additional parameters. We demonstrated the effectiveness of our method through extensive experiments, where our approach reduced the average error by 39.8% on the CIFAR10-C dataset using the WideResNet-28-10 model, by 38.8% using the WideResNet-40-2 model, and by 33.8% using the PreActResNet-18 model. Additionally, on the CIFAR100-C dataset with the WideResNet-40-2 model, our method reduced the average error by 5.3%, showcasing significant improvements in model generalization in continuous cyclic testing scenarios.

Keywords: deep learning; model generalization; distribution shift; test-time adaptation; model robustness

MSC: 68T07

Citation: Jiang, J.; Yang, H.; Yang, L.; Zhou, Y. Advancing Model Generalization in Continuous Cyclic Test-Time Adaptation with Matrix Perturbation Noise. *Mathematics* **2024**, *12*, 2800. https://doi.org/10.3390/math12182800

Academic Editor: Florin Leon

Received: 5 August 2024
Revised: 26 August 2024
Accepted: 5 September 2024
Published: 10 September 2024

1. Introduction

The training of deep learning models is typically based on a key assumption that the data in the model's source domain (training set) and target domain (test set) are independent and identically distributed (iid) [1]. However, in practical applications, this iid assumption often proves to be overly idealistic, leading to suboptimal performance in the testing phase. In the real world, data distributions frequently undergo dynamic changes [2], posing challenges to pre-trained deterministic models in coping with continuously evolving test tasks. For instance, in the face of emerging new viruses or evolving treatment methods, pre-trained medical diagnostic models struggle with data featuring constant changes. Similarly, in the field of supply chain management, demand patterns and available resources may shift due to seasonal variations and market demand fluctuations, rendering pre-trained management prediction models ineffective.

To tackle this challenge, researchers have recently introduced the concept of TTA, aimed at enhancing the adaptability of models to real-world data environments [3,4]. The essence of TTA is to use target domain data and source domain pre-trained model parameters for real-time online adjustments of the model, effectively improving performance in downstream applications under domain shift. It's important to note that existing TTA

methods usually assume that the target domain remains static [3]. However, in reality, machine systems are situated in non-static and continually evolving environments, where the data distribution of the target domain may experience shifts and cyclical changes over time.

In this paper, we introduce the concept of CycleTTA, which allows the evaluation of pre-trained models in a continuous cyclic testing environment. Additionally, to address potential overfitting issues in continuous testing scenarios, we developed a lightweight yet highly effective parameter-free approach. This approach incorporates matrix-level perturbation noise into batch normalization statistical parameters, significantly enhancing the source pre-trained model's adaptability to target domain shifts. An illustrative diagram of this method is provided in Figure 1. To further understand the effect of different types of noise during the testing phase, we examined variants incorporating uniform noise and Gaussian noise distributions.

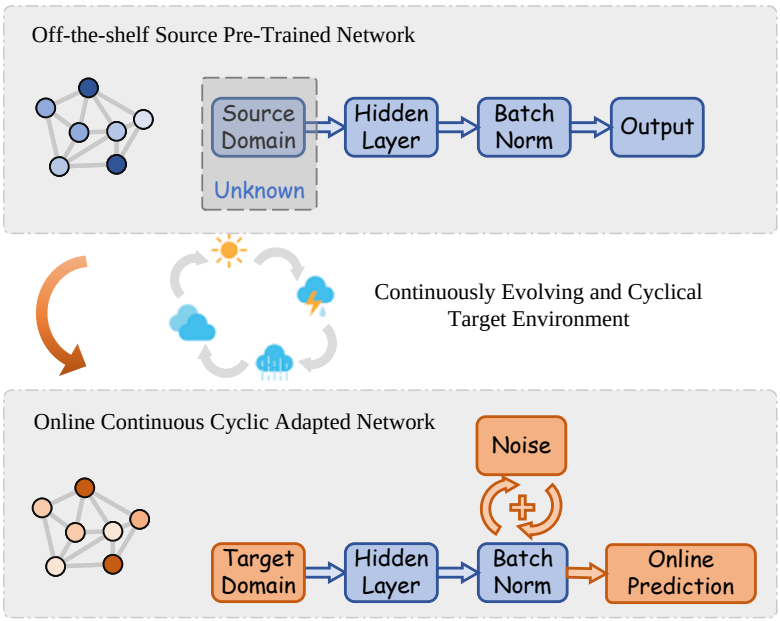

Figure 1. An illustration of our proposed method. We consider the scenario of online continuous cyclic test-time adaptation. The target data are not only provided in a sequence but also exhibit periodicity, originating from a continually changing environment.

We summarize our contributions as follows:

- To our knowledge, we are the first to propose a continuous cyclic test-time domain-adaptation setting, named CycleTTA.
- We are the first to explore the relationship between perturbation noise and model generalization in continuous cyclic testing scenarios, and we introduce a lightweight yet effective parameter-free optimization method by adding random noises to the batch-normalization (BN) statistics.
- Our experiments have proven the effectiveness of the proposed method across various models and datasets: on the CIFAR10-C dataset with the WideResNet-28-10 model, the average error was reduced by 39.8%; on the WideResNet-40-2 model, it was reduced by 38.8%; and on the PreActResNet-18 model, it was reduced by 33.8%. On the CIFAR100-C dataset with the WideResNet-40-2 model, the average error was reduced by 5.3%.

2. Related Work

2.1. Batch Normalization

Overfitting and prolonged training time are significant hurdles in the field of multi-layer neural network learning [5]. To overcome these challenges, batch normalization has become a widely recognized approach [6–8].

Prior to the introduction of BN, network convergence during training relied heavily on meticulous hyperparameter initialization, including suboptimal initial weights and small learning rates, leading to delays in the training process [9]. Additionally, the intricate interdependencies between layers added complexity to the learning process, as even minor changes in one layer could be amplified as they propagate through the network [10]. By normalizing the inputs for each layer, including the input layer, BN effectively reduces training time [11,12]. This normalization allows for higher learning rates, resulting in fewer training steps required for convergence [13–15]. Due to the simplicity of implementing batch normalization by introducing a BN layer into the network, batch normalization is a natural choice for expediting training across various hyperparameter combinations and a basic component in most state-of-the-art architectures [16–21].

Li et al. [22] introduced the pioneering work of integrating BN into domain adaptation, known as Adaptive batch normalization (AdaBN). AdaBN calculates BN statistics for the source domain during training and adapts these statistics for the target domain during testing. Coupling the network parameters of both target and source samples during the training stage has been the primary focus of subsequent studies inspired by AdaBN [23–25]

The concept of normalization from domain adaptation can also be applied to model robustness when handling corruption, where different types of corruption can be viewed as distinct domains. Several studies have explored normalization techniques to enhance corruption robustness under covariate shift [26–29]. The BN-based approach for learning domain-invariant representations is also employed to defend against multiple adversarial examples [30–32]. Wang et al. [3] proposed Full Test Time Adaptive (Tent) for unsupervised domain adaptation or image corruption. Tent adjusts features during testing by estimating normalization statistics (μ and σ) and optimizing affine parameters (γ and β) by using entropy minimization. The normalization statistics and affine parameters are updated exclusively based on target data, without employing source data.

Despite advancements in the field of deep learning, traditional BNs still play a pivotal role due to their simplicity and proven effectiveness [3,22,26,33–36]. Our research aims to investigate the impact of noise within the BN layer, specifically during test time adaptation. Our objective is to demonstrate how the introduction of noise could enhance the model's adaptability to transition between domains, particularly during the testing phase.

2.2. Test-Time Adaptation

TTA is a technique aimed at mitigating the distribution disparity between training and testing data, without the need for additional data collection or labeling expenses [37–44]. Unlike other methods of domain generalization, TTA does not depend on source domain data and is occasionally referred to as unsupervised domain adaptation in certain literature [45–50]. In practical terms, TTA optimizes model parameters using a single epoch or a small batch of test data [51–55]. The strategies employed to handle TTA tasks can be categorized into feature alignment [50,56,57] and fine-tuning [58,59], with the latter gaining increasing popularity.

BN is widely incorporated into neural network architectures as it reduces training complexity by encoding domain-specific knowledge through normalized statistical information. Consequently, many TTA methods focus on the utilization of BN layers [22,60–63]. A lightweight method for adapting during testing involves replacing the statistical data in BN with the statistics from the current testing batch. Test-time Self-Training (TST) [64] is a TTA method based on self-supervised learning, employing two networks: one for source domain training and another for unsupervised self-training. During testing, a small batch of unlabeled target data is employed to update the model and adapt to the target domain distribution. Additionally, the Test Entropy Minimization technique [3] employs entropy minimization during testing to eliminate inconsistent features in predictions. It serves as a means of fine-tuning the source model without explicitly performing domain alignment. Many TENT-based methods inherit this advantage [65,66]. Moreover, the Continuous Test Time Adaptation (CoTTA) method [4] has been proposed to adapt to dynamically chang-

ing target environments. This method enhances mean prediction and random recovery strategies to achieve the desired outcome.

In recent years, research related to TTA has shown interest in dynamic scenarios encountered in real-world environments, where the types and intensities of continuous noise or corruption faced by testing data vary [4,67–71]. Related researches have demonstrated that BN's performance in dynamic scenarios needs improvement [65]. Although CoTTA addresses adaptation to dynamically changing target environments, it operates in a non-cyclic environment and necessitates updating the entire model during adaptation, leading to significantly increased computational complexity and overall cost.

Our work tackles the limitations present in current TTA research. Firstly, our proposed CycleTTA explicitly addresses continuous cyclic testing within domain adaptation settings. Secondly, we employ a parameter-free approach by directly introducing matrix perturbation noise into BN layers, which enhances convenience and cost-effectiveness.

3. Methodology

3.1. CycleTTA Setup

In our proposed CycleTTA scheme, the approach is based on the fundamental premise that the source model is already pretrained. This setup is straightforward and uncomplicated, as it does not impose any additional assumptions on the source model. Specifically, it requires no modifications to the existing training architecture or methodologies employed for the source model. This characteristic ensures that the integrity and original design of the source model are preserved throughout the entire adaptation process.

CycleTTA intentionally avoids any form of manipulation of the target data. This means it does not employ typical procedures like data augmentation or data replay on the target dataset. The rationale behind this approach is rooted in practical considerations relevant to real-world applications. In many real-world scenarios, the resources required for extensive data manipulation are often limited or unavailable, possibly due to constraints in computational power, time, or access to additional data. Moreover, privacy and data-protection concerns play a crucial role in this decision. In various contexts, especially those involving sensitive or personal data such as medical records or private communications, applying transformations or augmentations to data could present significant privacy risks or even violate data-protection laws. By avoiding these data manipulations, CycleTTA not only respects privacy and ethical considerations but also ensures that the adaptation process is feasible and compliant across a wide range of real-world applications. This approach, therefore, maintains the integrity and original distribution of the target data, ensuring that the adaptation is both practical and in line with real-world constraints and ethical standards.

Furthermore, this approach does not require any modifications to the architecture of the target model when adapting the source model to the target domain. This method ensures a seamless integration of the source model with the target domain, eliminating the need for reconfiguring the architecture of the target model. CycleTTA sidesteps these common complexities, offering a more efficient and less resource-intensive solution for domain adaptation. It focuses solely on adapting the pretrained source model to effectively operate in the new target-domain environment.

3.2. Test-Time Entropy Minimization

Our experiments add correlated noise on top of the TENT setting [3], which tunes the model via test-time optimization by modulating its features to minimize the entropy of its predictions. The test-time target $L(x_t)$ is the entropy $H(\hat{y})$ of the model predictions during the test:

$$\hat{y} = f_\theta(x_t) \tag{1}$$

In particular, the Shannon entropy of the probability $\hat{y}_c$ of class c is

$$H(\hat{y}) = -\sum_c p(\hat{y}_c) \log p(\hat{y}_c) \tag{2}$$

As a metric for task prediction, it belongs to supervised training, and the function of the model parameters is $H(\hat{y} = f_\theta(x))$. During test-time adaptation, TENT is encapsulated by

$$\mathcal{L}_{TENT}(f_\theta(x^t), y^t) = \frac{1}{N}\sum_{i=1}^{C}(-q_i \log(q_i)) \tag{3}$$

Here, regarding the probability of class i among C classes, N stands for the number of samples and q_i for the softmax-transformed model-output logits.

3.3. Batch-Normalization Perturbation

Consider a mini-batch $\mathcal{B}$ of size m. Since the normalization is applied to each activation independently, let us focus on a particular activation $x^{(k)}$ and omit k for clarity. We have m values of this activation in the mini-batch:

$$\mathcal{B} = \{x_{1\ldots m}\} \tag{4}$$

Let the normalized values be $\widehat{x}_{1\ldots m}$ and their linear transformations be $h_{1\ldots m}$. We refer to the transform

$$\mathrm{BN}_{\gamma,\beta} : x_{1\ldots m} \rightarrow h_{1\ldots m} \tag{5}$$

as the batch-normalizing transform. It can be represented as

$$\begin{aligned}
\mu_{\mathcal{B}} &\leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i; && \text{mini-batch mean} \\
\sigma^2_{\mathcal{B}} &\leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2; && \text{mini-batch variance} \\
\widehat{x}_i &\leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}}; && \text{normalize} \\
h_i &\leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i); && \text{scale and shift}
\end{aligned} \tag{6}$$

where ϵ is an infinitesimal value introduced to prevent division by zero, while γ and β are learnable parameters utilized for scaling and shifting the normalized samples.

For each mini-batch $\mathcal{B}_i$, we compute raw batch-normalization statistics $\mathbf{B}_i$ and add noise to $\mathbf{B}_i$ using either a uniform distribution or a Gaussian distribution as the prior. This process can be represented as $\tilde{\mathbf{B}}_i = \mathbf{B}_i + U\left(-\frac{\lambda}{2}, \frac{\lambda}{2}\right) * \mathrm{std}(\mathbf{B}_i)$ or $\tilde{\mathbf{B}}_i = \mathbf{B}_i + G\left(0, \frac{\lambda}{2}\right) * \mathrm{std}(\mathbf{B}_i)$, where $U(a, b)$ represents uniform-distribution noise ranging from a to b and $G(0, c)$ represents Gaussian distribution noise with zero mean and c standard variance. The added noise is typically chosen to be small, with a magnitude controlled by a hyperparameter λ.

The proposed method is a lightweight and general plug-and-play parameter-free technique that can be broadly applied to continuous-testing scenarios. In practice, we can simply implement it using the following Pytorch-style code Listing 1:

Listing 1. CycleTTA Pytorch-style code.

```
for name, param in model.named_parameters():
    layer = dict(model.named_modules())[name.split('.')[0]]
    if isinstance(layer, nn.BatchNorm2d):
        if 'weight' in name or 'bias' in name:
            noise = (torch.rand(param.size()) - 0.5) * lambda_value * torch.
    std(param)
            #noise= (torch.randn(param.size())) * lambda_value * torch.std(
    param)
            param.data.add_(noise)
```

and the proposed training methodology is delineated in Algorithm 1.

Algorithm 1: Forward propagation with proposed method

Input: Source-pretrained model $f(\cdot)$, target data x^t, batch size B, noise strength λ;
Parameter: Maximum epochs E, model parameters θ;
Output: Model parameters θ;
Initialize $E = 1$.
while *epoch* $\leq E$ **do**
  for each batch normalization statistic $\mathbf{B}_i$ **do**
    Generate noise from $U\left(-\frac{\lambda}{2}, \frac{\lambda}{2}\right) * \text{std}(\mathbf{B}_i)$ or $G\left(0, \frac{\lambda}{2}\right) * \text{std}(\mathbf{B}_i)$
    $\tilde{\mathbf{B}}_i = \mathbf{B}_i + U\left(-\frac{\lambda}{2}, \frac{\lambda}{2}\right) * \text{std}(\mathbf{B}_i)$ or $\tilde{\mathbf{B}}_i = \mathbf{B}_i + G\left(0, \frac{\lambda}{2}\right) * \text{std}(\mathbf{B}_i)$;
  end for
end while

4. Experiments

4.1. Experimental Settings and Implementation Details

Datasets: For our experiments, we chose CIFAR10-C [72] and CIFAR100-C [70] as our datasets. The CIFAR datasets consist of a training set containing 50,000 images and a test set with 10,000 color images. All the images have a resolution of 32 × 32 pixels. CIFAR-10 is composed of 10 classes, whereas CIFAR-100 has 100 classes. CIFAR10-C and CIFAR100-C expand the primary dataset by introducing 15 unique types of corruptions, with five levels of severity. In our experiment, we will assess our model's efficacy against all the different types and levels of corruption.

Network Architectures: The three network architectures we utilized are as follows: (1) **WideResNet-28-10** is an enhanced version of the classic ResNet model, which increases the network's width by ten times the number of channels in each layer compared to the baseline ResNet. After each convolutional layer, BN and rectified linear unit (ReLU) activation functions are employed to ensure stable learning and introduce non-linearity. (2) **WideResNet-40-2** is characterized by its 40 layers, making it deeper than WideResNet-28-10 but with a narrower width, having a width factor of two. (3) **PreActResNet-18:** consisting of 18 layers, this network employs Preactivation Residual Blocks, where activation functions are applied before the convolutional layers, diverging from the standard ResNet design.

Baselines: During the experiments, we will compare three methods: TENT, TENT-U, and TENT-G. In this context, TENT [3] serves as the backbone method, where no noise is introduced to batches, while TENT-U and TENT-G incorporate uniform and Gaussian noise, respectively; the process is detailed in Algorithm 1.

Training Details: In our CycleTTA setup, we cycle models from corruption level 5 to 1 and add noises between batches. The classifier is trained using SGD with a learning rate of 0.001, exponential decay rate of 0.9, no weight decay, and batch size of 200. All comparative experiments were conducted using all the proposed baselines with the CycleTTA setup. For CIFAR10-C, we conducted experiments utilizing all network architectures. For CIFAR100-C, our methods were evaluated using the WideResNet-40-2 model. All code is implemented with Pytorch on NVIDIA GeForce RTX 3090 GPU.

Robustness Evaluation: to assess the adversarial robustness of these models, we employ the per-sample classification error rate (%) as proposed in [3].

4.2. Results on CIFAR10-C

In this section, we conducted experiments on the CIFAR10-C dataset to assess the generalization capability of the proposed method, utilizing three network architectures for performance evaluation, including WideResNet-28-10, WideResNet-40-2, and PreActResNet-18.

4.2.1. Results on WideResNet-28-10

Table 1 presents the average error rates of models for five levels of corruption across 15 types of corruption. The lowest error rates achieved using different network architectures are highlighted in bold in Table 1.

Table 1. The average classification error rate (%) for 5 levels of corruption on CIFAR10-C TTA tasks. Results are evaluated on TENT, TENT-U, and TENT-G methods using WideResNet-28-10, WideResNet-40-2, and PreActResNet-18 architectures. **Bold** indicates the best performance under each network architecture.

Architectures	**WideResNet-28-10**			**WideResNet-40-2**			**PreActResNet-18**		
Corruptions	**TENT**	**TENT-U**	**TENT-G**	**TENT**	**TENT-U**	**TENT-G**	**TENT**	**TENT-U**	**TENT-G**
Gaussian	30.2	**21.7**	24.8	22.6	15.7	**14.5**	17.6	12.2	**11.9**
Shot	28.7	**19.4**	21.7	21.1	14.0	**12.9**	16.2	10.8	**10.5**
Impulse	33.4	**23.6**	26.7	24.0	15.6	**15.4**	19.1	14.2	**13.4**
Defocus	24.2	**13.9**	16.5	18.9	11.0	**10.3**	14.8	10.0	**9.2**
Glass	39.2	**28.6**	31.9	26.8	18.5	**17.7**	22.3	17.6	**16.3**
Motion	28.7	**17.3**	20.2	21.2	12.9	**11.8**	17.0	12.2	**11.3**
Zoom	27.2	**15.4**	18.5	20.1	11.9	**11.0**	15.5	11.2	**10.2**
Snow	31.2	**19.5**	22.9	22.2	13.8	**13.1**	18.0	13.4	**12.5**
Frost	29.9	**17.4**	21.4	21.3	12.7	**12.5**	17.0	11.6	**11.4**
Fog	28.3	**14.5**	19.1	21.6	12.5	**11.8**	17.3	11.4	**11.2**
Brightness	26.7	**12.3**	17.4	20.1	10.8	**10.2**	16.0	9.7	**9.3**
Contrast	28.5	**14.4**	19.1	21.2	11.6	**11.4**	17.0	10.4	**9.9**
Elastic_trans	32.9	**18.1**	23.7	24.0	14.1	**13.6**	20.2	13.3	**13.1**
Pixelate	30.5	**16.3**	22.1	22.4	12.3	**11.5**	17.7	11.3	**11.1**
Jpeg	35.8	**22.4**	27.3	28.2	17.5	**16.4**	20.5	14.1	**13.6**
Mean	30.4	**18.3**	22.2	22.4	13.7	**12.9**	17.7	12.2	**11.7**

According to Table 1, among the network architectures, WideResNet-28-10 performs the worst on the CIFAR10-C task, exhibiting the highest mean error rate of 30.4% across all corruption types. Its low accuracy indicates the challenge of using the WideResNet-28-10 for CIFAR10-C classification tasks. Meanwhile, TENT-G demonstrates enhanced robustness, as it reduced the mean error rate to 22.2% (reduced by 8.2% compared to the mean error rate of TENT). Notably, TENT-U indicates the effectiveness of our proposed method by reducing the mean error rate by 12.1% compared to the backbone method (TENT) and outperforming other methods across all corruption types, especially in corruption types like Brightness, Elastic_trans, and Pixelate (reduced by over 5% compared to the error rates of TENT-G in the corresponding corruption type).

To comprehensively assess the performance of each strategy across different corruption levels, we present detailed experimental results for each level in Figure 2. Figure 2 illustrates that at the highest corruption level (level = 5), three methods exhibit low robustness to specific damage types such as Gaussian, Impulse, and Glass, with high classification error rates. At this stage, the data contain less noise, leading to a smaller robustness gap among the three models. Within the CycleTTA setup, as the model iterates, it introduces more noise into the data, and the robustness of the TENT model naturally decreases. This is evident from the classification error rate, which ranges from no more than 25% to no less than 40% for most corruption types. However, when the model reaches the corruption level of one, both TENT(1)-U and TENT(1)-G demonstrate significant improvements in model robustness compared to TENT(1). This can be observed by a decrease in the classification error rate to approximately 25% and 20%, respectively, for TENT(1)-U and TENT(1)-G. In summary, our extensive experiments on the WideResNet-28-10 architecture validate that the method within the CycleTTA setup can improve the robustness of models when exposed to data with escalating levels of noise.

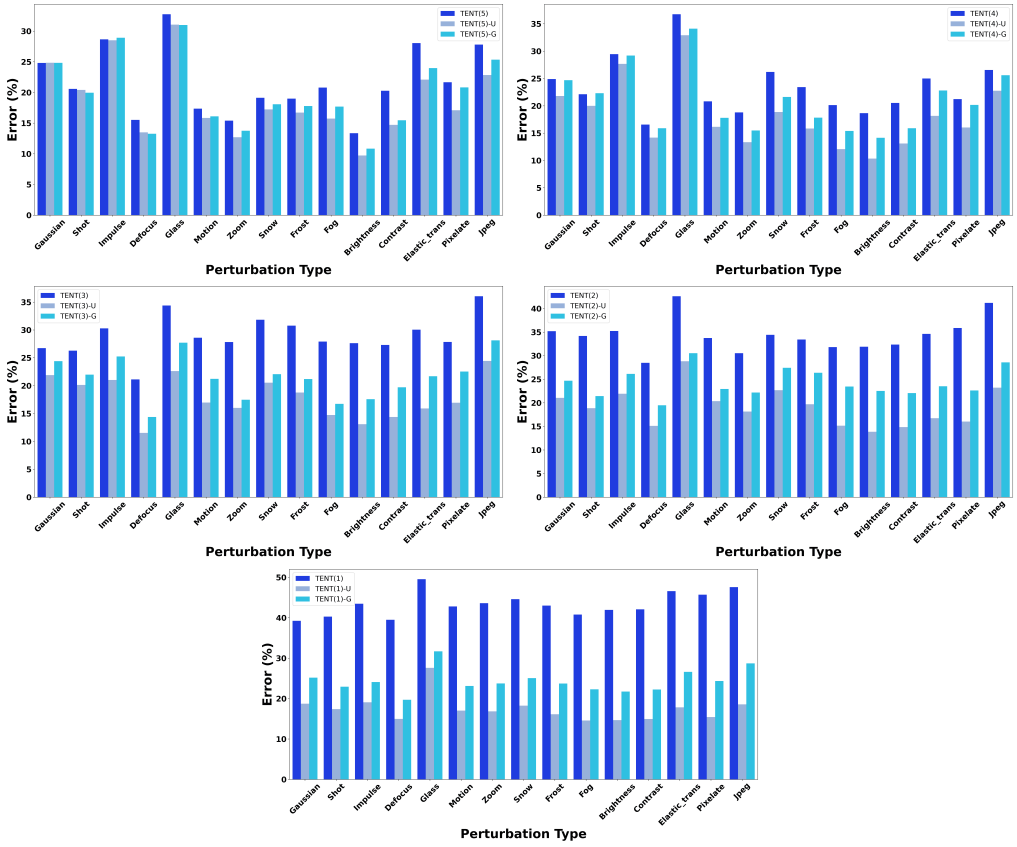

Figure 2. Results for five continuous perturbation levels on WideResNet-28-10.

4.2.2. Results on WideResNet-40-2

The average error rates for models under five levels of corruption when employing the WideResNet-40-2 architecture are presented in columns five to seven of Table 1. The lowest error rate for each corruption type is highlighted in bold. According to Table 1, the TENT model, serving as the backbone, indicates the initial challenges in CIFAR10-C TTA tasks with a mean error rate of 22.4%. Furthermore, TENT-U exhibits an improved robustness by reducing error rates across all corruption types. In contrast to the results obtained with the WideResNet-28-10 architecture, TENT-G outperforms other methods with the lowest error rate across all corruption types. Notably, there is a significant enhancement in robustness observed for corruption types involving Elastic_trans, Pixelate, and Jpeg, resulting in an error rate reduction of over 10% compared to TENT.

To provide a comprehensive comparison of the results for different strategies across varying levels of corruption severity, we present the experimental findings separately for each of the five levels in Figure 3. At a corruption severity level of five, TENT(5) exhibits an average error rate of 13.6%. In contrast, TENT(5)-U produces an average error rate of 12.9%, resulting in a 5% decrease compared to TENT. Likewise, TENT(5)-G achieves an average error rate of 13.2%, representing a 3% reduction compared to TENT. At a corruption level of three, TENT(3) yields an average error rate of 22.9%. However, the average error rate for TENT(3)-U is 12.7%, demonstrating a 45% decrease compared to TENT. Similarly, the average error rate for TENT(3)-G is 13.4%, which is 42% lower than TENT. At a corruption level of one, the average error rate for TENT(1) is 32.3%. In contrast, TENT(1)-U exhibits an average error rate of 13.8%, showcasing a 57% reduction compared to TENT. Additionally, TENT(1)-G has an average error rate of 12.6%, which demonstrates a 61% decrease compared to TENT. Upon further analysis of Figure 3, it becomes evident that both in comparison to TENT, TENT-U, and TENT-G demonstrate a gradual increase in their effectiveness at optimizing model robustness as the severity of corruption decreases.

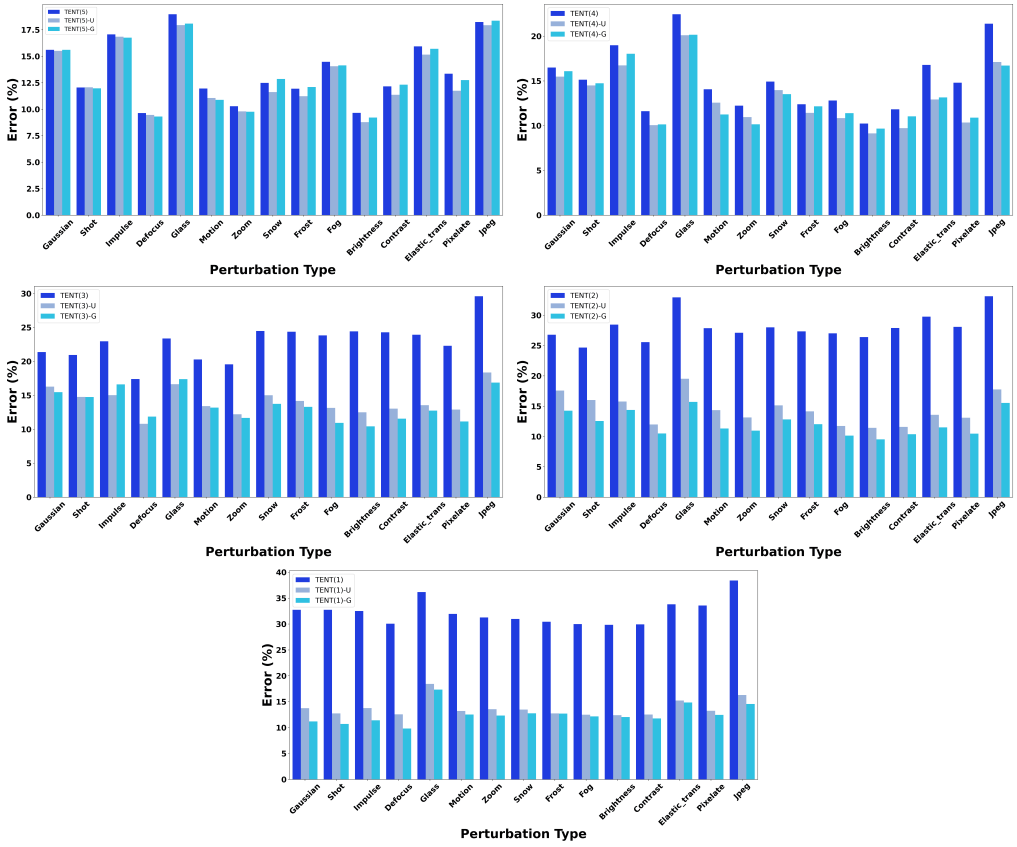

Figure 3. Results for five continuous perturbation levels on WideResNet-40-2.

4.2.3. Results on PreActResNet-18

The results of the PreActResNet-18 neural network architecture within the CycleTTA setting are detailed in the fourth column of Table 1, with a focus on the lowest error rates achieved by the PreActResNet-18 network.

The overall efficacy of TENT-G was highlighted by an average error rate of 11.7% compared to 12.2% for TENT-U and 17.7% for TENT. Compared to the results of the models using the other two network architectures, WideResNet-28-10 and WideResNet-40-2, the model image-classification error rates using the PreActResNet-18 architecture are both reduced, which can be attributed to the design of the network architecture; PreActResNet-18 utilizes a preactivation residual cell structure, which helps to mitigate the gradient vanishing problem by placing the activation function before the convolutional layer, improving the efficiency of information transfer. This design allows the network to better learn and represent image features, resulting in an improved classification performance.

The experimental results for various strategies at the five levels of corruption severity are displayed individually in Figure 4. When the corruption level is at five, indicating a relatively low amount of added noise, the discrepancy between the three methods is minimal, demonstrating a high level of consistency, especially in the Gaussian, Shot, Impulse, and Jpeg corruption types. However, as the model continues to iterate and more noise is introduced into the data, the difference between TENT-G, TENT-U, and the TENT method progressively widens. Due to its lack of consideration for noise during training, TENT struggles to combat the impact of noise as the corruption level increases, leading to a gradual increase in classification error rates from under 17.5% to over 20% across most corruption types. In contrast, TENT-G and TENT-U exhibit resilience to injected noise, with the classification failure rates of TENT(5)-G and TENT(5)-U resembling those of TENT(1)-G and TENT(1)-U, showing no significant increase. Upon analyzing Figure 4, it can be observed that similar to the observations made on the WideResNet-28-10 and WideResNet-40-2 network architectures, the optimization

effectiveness of TENT-U and TENT-G, compared to the TENT results, progressively enhances as the corruption severity diminishes.

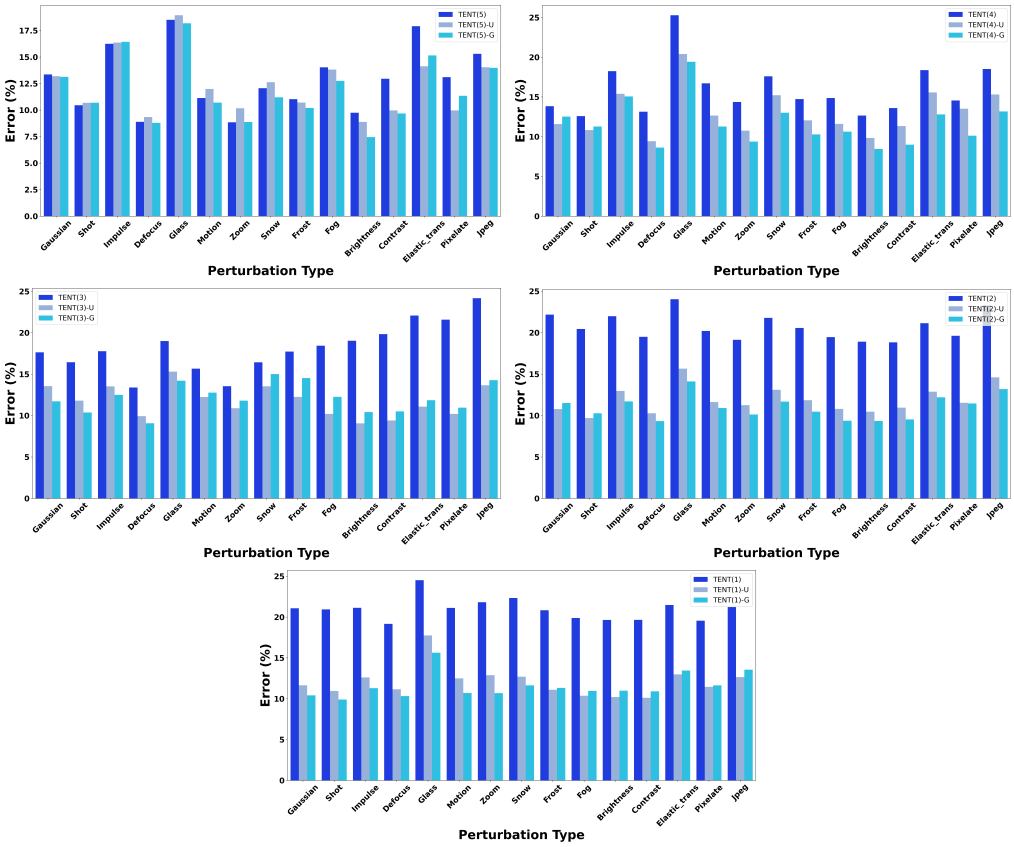

Figure 4. Results for five continuous perturbation levels on PreActResNet-18.

4.3. Results on CIFAR100-C

In this section, we conducted experiments on the CIFAR100-C dataset to assess the generalization capability of the proposed method, utilizing the WideResNet-40-2 network architecture for the performance evaluation. The model's robustness under diverse corruption types was also appraised using three distinct adaptation techniques: TENT, TENT-U, and TENT-G. The experimental findings are tabulated in Table 2, with bold values denoting the lowest error rates achieved by the WideResNet-40-2 network.

Table 2. The average classification error rate (%) for 5 levels of corruption on CIFAR100-C TTA tasks. Results are evaluated on TENT, TENT-U, and TENT-G methods using WideResNet-40-2 architecture. The **bold** ones are the best results under this noise.

Method	*Gaussian*	*Shot*	*Impulse*	*Defocus*	*Glass*	*Motion*	*Zoom*	*Snow*	*Frost*	*Fog*	*Brightness*	*Contrast*	*Elastic_trans*	*Pixelate*	*Jpeg*	**Mean**
TENT	59.2	58.0	59.4	55.7	63.7	58.7	58.3	62.7	62.6	63.0	60.6	62.9	65.5	63.9	69.3	61.6
TENT-U	56.9	56.0	57.4	53.6	61.7	56.5	56.2	60.1	59.8	60.0	57.7	59.4	62.2	60.0	**65.9**	58.9
TENT-G	**56.8**	**55.3**	**56.7**	**52.9**	**61.4**	**55.7**	**55.0**	**59.2**	**59.0**	**59.5**	**56.8**	**58.8**	**61.7**	**59.8**	66.0	**58.3**

As per the data analysis in Table 2, TENT-G consistently outperforms TENT and TENT-U across all remaining corruption types, except for Jpeg, as evidenced by the mean results. Specifically, under Jpeg corruption, the standard TENT method exhibits an error rate of 69.3%, while the TENT-G method yields a reduced error rate of 66%, marking a 4.76% improvement.

Meanwhile, the TENT-U method, incorporating homogeneous noise, achieves an error rate of 65.9%, representing a 4.9% reduction compared to TENT. When using WideResNet-40-2, the performance of TENT-G on the CIFAR100-C dataset demonstrates remarkable consistency, ensuring minimal impact on the model's predictive accuracy under visual noise conditions. Notably, TENT-G showcases an average error rate of 58.3%, outperforming TENT-U at 58.9% and TENT at 61.6%, thereby affirming its overall efficacy.

5. Parameter Sensitivity Analysis

In this section, we investigate the impact of noise intensity on test-time domain adaptation. We take the performance of the WideResNet-40-2 network under the CIFAR10-C task as an example. We focused on three different noise intensities, $lambda = 0.05$, 0.1, and 0.15, to explore the model's sensitivity to these noise parameters.

Table 3 presents the mean error rates for five corruption levels across all corruption types, providing insights into the impact of varying noise intensities under both uniform and Gaussian noise scenarios. The results demonstrate that in the context of CIFAR10-C with the WideResNet-40-2 network, both Gaussian and uniform noise effectively enhance model generalization during test-time adaptation. For instance, when applying uniform noise, a noise intensity of 0.15 proves most effective, with an average accuracy rate of 13.7%. Conversely, Gaussian noise yields optimal results at an intensity of 0.05, achieving an average accuracy rate of 12.9%. These findings indicate that while introducing random noise is beneficial, the ideal noise strength varies depending on the noise distribution. Notably, since our approach revolves around batch-normalization parameters at the matrix level, the noise intensity correlates positively with the variance of these parameters. In other words, under constant conditions, the parameter variance is directly proportional to the added noise intensity, thereby ensuring robustness against parameter sensitivity.

Table 3. The average classification error rate (%) for 5 levels of corruption on CIFAR10-C tasks. Results are evaluated on TENT, TENT-U, and TENT-G methods using WideResNet-40-2 architectures with λ = 0.05, 0.1, and 0.15. The **bold** ones are the best results under this noise.

Corruptions	TENT	TENT-U (0.05)	TENT-U (0.1)	TENT-U (0.15)	TENT-G (0.05)	TENT-G (0.1)	TENT-G (0.15)
Gaussian	22.6	16.2	16.1	**15.7**	**14.5**	17.2	16.0
Shot	21.1	14.5	14.3	**14.0**	**12.9**	15.3	14.5
Impulse	24.0	17.2	16.6	**15.6**	**15.4**	17.7	16.7
Defocus	18.9	12.1	12.3	**11.0**	**10.3**	12.6	11.7
Glass	26.8	19.6	19.4	**18.5**	**17.7**	20.3	19.5
Motion	21.2	14.2	14.1	**12.9**	**11.8**	14.9	14.3
Zoom	20.1	13.3	13.2	**11.9**	**11.0**	13.8	13.4
Snow	22.2	15.3	15.2	**13.8**	**13.1**	15.6	15.3
Frost	21.3	14.3	14.1	**12.7**	**12.5**	14.8	14.1
Fog	21.6	14.2	13.9	**12.5**	**11.8**	15.1	13.7
Brightness	20.1	12.5	12.6	**10.8**	**10.2**	13.3	12.3
Contrast	21.2	13.7	13.7	**11.6**	**11.4**	14.1	13.1
Elastic_trans	24.0	15.8	16.4	**14.1**	**13.6**	17.0	15.3
Pixelate	22.4	13.7	14.4	**12.3**	**11.5**	14.7	13.9
Jpeg	28.2	18.3	19.1	**17.5**	**16.4**	19.9	18.5
Mean	22.4	15.0	15.0	**13.7**	**12.9**	15.7	14.8

6. Conclusions

In this paper, we focused on addressing the challenge of continual test-time adaptation in non-stationary environments, where the distribution of the target domain can dynamically evolve over time. To address the progressive build-up of errors in this setup, we proposed CycleTTA, a lightweight yet effective method that adds matrix-wise perturbation noise into batch-normalization statistics. The proposed method is computationally efficient because of no need for additional parameters and may have wide application in real-world industry systems due to its simplicity and ease of use. Our extensive experiments demonstrated the effectiveness of the proposed method, with significant reductions in the average error rates: 39.8% on CIFAR10-C using WideResNet-28-10, 38.8% using WideResNet-40-2, 33.8% using PreActResNet-

18, and 5.3% on CIFAR100-C using WideResNet-40-2. A parameter sensitivity analysis further demonstrates that the proposed method is robust against parameter perturbation.

Author Contributions: Writing—original draft, J.J.; Writing—review & editing, H.Y. and L.Y.; Project administration, Y.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Natural Science Foundation of China (grant no. 62276262); the Science and Technology Innovation Program of Hunan Province (grant no. 2021RC3076); the Training Program for Excellent Young Innovators of Changsha (grant no. KQ2009009); and Research on Emergency Search and Rescue of the South China Sea Based on Aerospace, Land, and Sea Collaborative Perception and Intelligent Information Processing, 2023 Hainan Provincial Natural Science Foundation High-Level Talent Project, no. 623RC515.

Data Availability Statement: The data presented in this study are available in github link https://robustbench.github.io/.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Jacobs, K.; Jacobs, K. Independent identically distributed (iid) random variables. In *Discrete Stochastics*; Springer: Birkhäuser, Basel, 1992; pp. 65–101.
2. Yu, E.; Ye, Z.; Zhang, Z.; Qian, L.; Xie, M. A federated recommendation algorithm based on user clustering and meta-learning. *Appl. Soft Comput.* **2024**, *158*, 111483. [CrossRef]
3. Wang, D.; Shelhamer, E.; Liu, S.; Olshausen, B.; Darrell, T. Tent: Fully Test-Time Adaptation by Entropy Minimization. In Proceedings of the International Conference on Learning Representations, Vienna, Austria, 4 May 2021.
4. Wang, Q.; Fink, O.; Van Gool, L.; Dai, D. Continual Test-Time Domain Adaptation. In Proceedings of the Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022.
5. Garbin, C.; Zhu, X.; Marques, O. Dropout vs. batch normalization: An empirical study of their impact to deep learning. *Multimed. Tools Appl.* **2020**, *79*, 12777–12815. [CrossRef]
6. Ioffe, S.; Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the International Conference on Machine Learning, Lile, France, 6–11 July 2015; PMLR: Seattle, WA, USA, 2015; pp. 448–456.
7. Kohler, J.; Daneshmand, H.; Lucchi, A.; Zhou, M.; Neymeyr, K.; Hofmann, T. Towards a theoretical understanding of batch normalization. *Stat* **2018**, *1050*, 27.
8. Luo, P.; Wang, X.; Shao, W.; Peng, Z. Towards Understanding Regularization in Batch Normalization. In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.
9. Shimodaira, H. Improving predictive inference under covariate shift by weighting the log-likelihood function. *J. Stat. Plan. Inference* **2000**, *90*, 227–244. [CrossRef]
10. Hu, X.; Chu, L.; Pei, J.; Liu, W.; Bian, J. Model Complexity of Deep Learning: A Survey. *arXiv* **2021**, arXiv:2103.05127. [CrossRef]
11. Salimans, T.; Kingma, D.P. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *Adv. Neural Inf. Process. Syst.* **2016**, *29*.
12. Xu, J.; Sun, X.; Zhang, Z.; Zhao, G.; Lin, J. Understanding and improving layer normalization. *Adv. Neural Inf. Process. Syst.* **2019**, *32*.
13. Bjorck, N.; Gomes, C.P.; Selman, B.; Weinberger, K.Q. Understanding batch normalization. *Adv. Neural Inf. Process. Syst.* **2018**, *31*.
14. Yong, H.; Huang, J.; Meng, D.; Hua, X.; Zhang, L. Momentum batch normalization for deep learning with small batch size. In Proceedings of the Computer Vision—ECCV 2020: 16th European Conference, Glasgow, UK, 23–28 August 2020; Proceedings, Part XII 16; Springer: Berlin/Heidelberg, Germany, 2020; pp. 224–240.
15. Kaur, S.; Cohen, J.; Lipton, Z.C. On the maximum hessian eigenvalue and generalization. In *Proceedings on "I Can't Believe It's Not Better! - Understanding Deep Learning Through Empirical Falsification" at NeurIPS 2022 Workshops*; PMLR: Seattle, WA, USA, 2023; pp. 51–65.
16. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
17. Hinton, G.E.; Salakhutdinov, R.R. Reducing the dimensionality of data with neural networks. *Science* **2006**, *313*, 504–507. [CrossRef]
18. Xiong, R.; Yang, Y.; He, D.; Zheng, K.; Zheng, S.; Xing, C.; Zhang, H.; Lan, Y.; Wang, L.; Liu, T. On layer normalization in the transformer architecture. In Proceedings of the International Conference on Machine Learning, Online, 13–18 July 2020; PMLR: Seattle, WA, USA, 2020; pp. 10524–10533.
19. Kurach, K.; Lučić, M.; Zhai, X.; Michalski, M.; Gelly, S. A large-scale study on regularization and normalization in GANs. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; PMLR: Seattle, WA, USA, 2019; pp. 3581–3590.
20. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.

21. Zagoruyko, S.; Komodakis, N. Wide Residual Networks. In Proceedings of the British Machine Vision Conference, York, UK, 19–22 September 2016; British Machine Vision Association: Glasgow, UK, 2016.
22. Li, Y.; Wang, N.; Shi, J.; Liu, J.; Hou, X. Revisiting batch normalization for practical domain adaptation. *arXiv* **2016**, arXiv:1603.04779.
23. Maria Carlucci, F.; Porzi, L.; Caputo, B.; Ricci, E.; Rota Bulo, S. Autodial: Automatic domain alignment layers. In Proceedings of the IEEE international Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 5067–5075.
24. Romijnders, R.; Meletis, P.; Dubbelman, G. A domain agnostic normalization layer for unsupervised adversarial domain adaptation. In Proceedings of the 2019 IEEE Winter Conference on Applications of Computer Vision (WACV), Waikoloa Village, HI, USA, 7–11 January 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 1866–1875.
25. Wang, X.; Jin, Y.; Long, M.; Wang, J.; Jordan, M.I. Transferable normalization: Towards improving transferability of deep neural networks. *Adv. Neural Inf. Process. Syst.* **2019**, *32*.
26. Pan, X.; Luo, P.; Shi, J.; Tang, X. Two at once: Enhancing learning and generalization capacities via ibn-net. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 464–479.
27. Schneider, S.; Rusak, E.; Eck, L.; Bringmann, O.; Brendel, W.; Bethge, M. Improving robustness against common corruptions by covariate shift adaptation. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 11539–11551.
28. Benz, P.; Zhang, C.; Karjauv, A.; Kweon, I.S. Revisiting batch normalization for improving corruption robustness. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, Virtual, 5–9 January 2021; pp. 494–503.
29. Ishii, M.; Sugiyama, M. Source-free domain adaptation via distributional alignment by matching batch normalization statistics. *arXiv* **2021**, arXiv:2101.10842.
30. Liu, A.; Tang, S.; Liu, X.; Chen, X.; Huang, L.; Tu, Z.; Song, D.; Tao, D. Towards defending multiple adversarial perturbations via gated batch normalization. *arXiv* **2020**, arXiv:2012.01654.
31. Choi, S.; Jung, S.; Yun, H.; Kim, J.T.; Kim, S.; Choo, J. Robustnet: Improving domain generalization in urban-scene segmentation via instance selective whitening. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Virtual, 19–25 June 2021; pp. 11580–11590.
32. Li, W.; Qi, K.; Chen, W.; Zhou, Y. Bridging the distribution gap of visible-infrared person re-identification with modality batch normalization. In Proceedings of the 2021 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA), Dalian, China, 28–30 June 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 23–28.
33. Han, K.; Si, C.; Huang, Y.; Wang, L.; Tan, T. Generalizable person re-identification via self-supervised batch norm test-time adaption. In Proceedings of the AAAI Conference on Artificial Intelligence, Online, 22 February–1 March 2022; Volume 36, pp. 817–825.
34. Roy, S.; Mitra, S.; Biswas, S.; Soundararajan, R. Test Time Adaptation for Blind Image Quality Assessment. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Paris, France, 1–6 October 2023; pp. 16742–16751.
35. Su, Y.; Xu, X.; Jia, K. Towards Real-World Test-Time Adaptation: Tri-Net Self-Training with Balanced Normalization. *arXiv* **2023**, arXiv:2309.14949. [CrossRef]
36. Wu, Q.; Yue, X.; Sangiovanni-Vincentelli, A. Domain-agnostic test-time adaptation by prototypical training with auxiliary data. In Proceedings of the NeurIPS 2021 Workshop on Distribution Shifts: Connecting Methods and Applications, Virtual, 6–14 December 2021.
37. Jin, W.; Zhao, T.; Ding, J.; Liu, Y.; Tang, J.; Shah, N. Empowering graph representation learning with test-time graph transformation. *arXiv* **2022**, arXiv:2210.03561.
38. Su, Y.; Xu, X.; Jia, K. Revisiting realistic test-time training: Sequential inference and adaptation by anchored clustering. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 17543–17555.
39. Döbler, M.; Marsden, R.A.; Yang, B. Robust mean teacher for continual and gradual test-time adaptation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 18–22 June 2023; pp. 7704–7714.
40. Park, S.; Yang, S.; Choo, J.; Yun, S. Label shift adapter for test-time adaptation under covariate and label shifts. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Paris, France, 1–6 October 2023; pp. 16421–16431.
41. Tang, Y.; Zhang, C.; Xu, H.; Chen, S.; Cheng, J.; Leng, L.; Guo, Q.; He, Z. Neuro-Modulated Hebbian Learning for Fully Test-Time Adaptation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 18–22 June 2023; pp. 3728–3738.
42. Gao, J.; Zhang, J.; Liu, X.; Darrell, T.; Shelhamer, E.; Wang, D. Back to the source: Diffusion-driven adaptation to test-time corruption. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 18–22 June 2023; pp. 11786–11796.
43. Lee, J.; Das, D.; Choo, J.; Choi, S. Towards open-set test-time adaptation utilizing the wisdom of crowds in entropy minimization. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Paris, France, 1–6 October 2023; pp. 16380–16389.
44. Jing, M.; Zhen, X.; Li, J.; Snoek, C. Variational model perturbation for source-free domain adaptation. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 17173–17187.
45. Kundu, J.N.; Venkat, N.; Rahul, M.V.; Babu, R.V. Universal source-free domain adaptation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WT, USA, 14–19 June 2020; pp. 4544–4553.
46. Yang, S.; Wang, Y.; Van De Weijer, J.; Herranz, L.; Jui, S. Generalized source-free domain adaptation. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Virtual, 11–17 October 2021; pp. 8978–8987.

47. Mutlu, O.C.; Honarmand, M.; Surabhi, S.; Wall, D.P. TempT: Temporal consistency for Test-time adaptation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 18–22 June 2023; pp. 5916–5922.
48. Mirza, M.J.; Micorek, J.; Possegger, H.; Bischof, H. The norm must go on: Dynamic unsupervised domain adaptation by normalization. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 14765–14775.
49. Lim, H.; Kim, B.; Choo, J.; Choi, S. TTN: A domain-shift aware batch normalization in test-time adaptation. *arXiv* **2023**, arXiv:2302.05155.
50. Wang, S.; Zhang, D.; Yan, Z.; Zhang, J.; Li, R. Feature alignment and uniformity for test time adaptation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 18–22 June 2023; pp. 20050–20060.
51. Chen, D.; Wang, D.; Darrell, T.; Ebrahimi, S. Contrastive test-time adaptation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 295–305.
52. Gandelsman, Y.; Sun, Y.; Chen, X.; Efros, A. Test-time training with masked autoencoders. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 29374–29385.
53. Niu, S.; Wu, J.; Zhang, Y.; Chen, Y.; Zheng, S.; Zhao, P.; Tan, M. Efficient test-time model adaptation without forgetting. In Proceedings of the International Conference on Machine Learning, Baltimore, MD, USA, 17–23 July 2022; PMLR: Seattle, WA, USA, 2022; pp. 16888–16905.
54. Goyal, S.; Sun, M.; Raghunathan, A.; Kolter, J.Z. Test time adaptation via conjugate pseudo-labels. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 6204–6218.
55. Gong, T.; Jeong, J.; Kim, T.; Kim, Y.; Shin, J.; Lee, S.J. NOTE: Robust continual test-time adaptation against temporal correlation. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 27253–27266.
56. Ganin, Y.; Ustinova, E.; Ajakan, H.; Germain, P.; Larochelle, H.; Laviolette, F.; Marchand, M.; Lempitsky, V. Domain-adversarial training of neural networks. *J. Mach. Learn. Res.* **2016**, *17*, 2096–2130.
57. Kojima, T.; Matsuo, Y.; Iwasawa, Y. Robustifying Vision Transformer without Retraining from Scratch by Test-Time Class-Conditional Feature Alignment. In Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22, Vienna, Austria, 23–29 July 2022; Raedt, L.D., Ed.; International Joint Conferences on Artificial Intelligence Organization: Montréal QC, Canada, 2022; pp. 1009–1016. [CrossRef]
58. Chu, B.; Madhavan, V.; Beijbom, O.; Hoffman, J.; Darrell, T. Best practices for fine-tuning visual classifiers to new domains. In Proceedings of the Computer Vision—ECCV 2016 Workshops: Amsterdam, The Netherlands, 8–10, 15–16 October 2016; Proceedings, Part III 14; Springer: Berlin/Heidelberg, Germany, 2016; pp. 435–442.
59. Hoffman, J.; Guadarrama, S.; Tzeng, E.S.; Hu, R.; Donahue, J.; Girshick, R.; Darrell, T.; Saenko, K. LSDA: Large scale detection through adaptation. *Adv. Neural Inf. Process. Syst.* **2014**, *27*.
60. Sun, T.; Segu, M.; Postels, J.; Wang, Y.; Van Gool, L.; Schiele, B.; Tombari, F.; Yu, F. SHIFT: A synthetic driving dataset for continuous multi-task domain adaptation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 21371–21382.
61. Boudiaf, M.; Mueller, R.; Ben Ayed, I.; Bertinetto, L. Parameter-free online test-time adaptation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 8344–8353.
62. Wang, F.; Han, Z.; Gong, Y.; Yin, Y. Exploring domain-invariant parameters for source free domain adaptation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 7151–7160.
63. Colomer, M.B.; Dovesi, P.L.; Panagiotakopoulos, T.; Carvalho, J.F.; Härenstam-Nielsen, L.; Azizpour, H.; Kjellström, H.; Cremers, D.; Poggi, M. To Adapt or Not to Adapt? Real-Time Adaptation for Semantic Segmentation. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Paris, France, 1–6 October 2023; pp. 16548–16559.
64. Sun, Y.; Wang, X.; Liu, Z.; Miller, J.; Efros, A.; Hardt, M. Test-time training with self-supervision for generalization under distribution shifts. In Proceedings of the International Conference on Machine Learning, Virtual, 13–18 July 2020; PMLR: Seattle, WA, USA, 2020; pp. 9229–9248.
65. Niu, S.; Wu, J.; Zhang, Y.; Wen, Z.; Chen, Y.; Zhao, P.; Tan, M. Towards stable test-time adaptation in dynamic wild world. *arXiv* **2023**, arXiv:2302.12400.
66. Jiang, L.; Lin, T. Test-time robust personalization for federated learning. *arXiv* **2022**, arXiv:2205.10920.
67. Wang, Z.; Luo, Y.; Zheng, L.; Chen, Z.; Wang, S.; Huang, Z. In Search of Lost Online Test-time Adaptation: A Survey. *arXiv* **2023**, arXiv:2310.20199.
68. Song, J.; Lee, J.; Kweon, I.S.; Choi, S. EcoTTA: Memory-Efficient Continual Test-time Adaptation via Self-distilled Regularization. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 18–22 June 2023; pp. 11920–11929.
69. Li, S.; Yuan, L.; Xie, B.; Yang, T. Generalized Robust Test-Time Adaptation in Continuous Dynamic Scenarios. *arXiv* **2023**, arXiv:2310.04714.
70. Yuan, L.; Xie, B.; Li, S. Robust test-time adaptation in dynamic scenarios. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 18–22 June 2023; pp. 15922–15932.

71. Brahma, D.; Rai, P. A Probabilistic Framework for Lifelong Test-Time Adaptation. In Proceedings of the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 18–22 June 2023; pp. 3582–3591.
72. Hendrycks, D.; Dietterich, T.G. Benchmarking neural network robustness to common corruptions and surface variations. *arXiv* **2018**, arXiv:1807.01697.