NC-TTT: A Noise Constrastive Approach for Test-Time Training

David Osowiechi* Gustavo A. Vargas Hakim*
Mehrdad Noori Milad Cheraghalikhani Ali Bahri Moslem Yazdanpanah
Ismail Ben Ayed Christian Desrosiers

ÉTS Montréal, Canada

Abstract

Despite their exceptional performance in vision tasks, deep learning models often struggle when faced with domain shifts during testing. Test-Time Training (TTT) methods have recently gained popularity by their ability to enhance the robustness of models through the addition of an auxiliary objective that is jointly optimized with the main task. Being strictly unsupervised, this auxiliary objective is used at test time to adapt the model without any access to labels. In this work, we propose Noise-Contrastive Test-Time Training (NC-TTT), a novel unsupervised TTT technique based on the discrimination of noisy feature maps. By learning to classify noisy views of projected feature maps, and then adapting the model accordingly on new domains, classification performance can be recovered by an important margin. Experiments on several popular test-time adaptation baselines demonstrate the advantages of our method compared to recent approaches for this task. The code can be found at: `https://github.com/GustavoVargasHakim/NCTTT.git`

1. Introduction

A crucial requirement for the success of traditional deep learning methods is that training and testing data should be sampled from the same distribution. As widely shown in the literature [20, 22], this assumption rarely holds in practice and a model's performance can drop dramatically in the presence of domain shifts. The field of Domain Adaptation (DA) has emerged to address this important issue, proposing various mechanisms that adapt learning algorithms to new domains.

In the realm of domain adaptation, two notable directions of research have surfaced: Domain Generalization and Test-Time Adaptation. Domain Generalization (DG) approaches [12, 21, 24, 26, 27] typically train a model with an extensive source dataset encompassing diverse domains and augmentations, so that it can achieve a good performance on test examples from unseen domains, without retraining. Conversely, Test-Time Adaptation (TTA) [2, 11, 25] entails the dynamic adjustment of the model to test data in real-time, typically adapting to subsets of the new domain, such as mini-batches. TTA presents a challenging, yet practical problem as it functions without supervision for test samples or access to the source domain data. While they do not require training data from diverse domains as DG approaches, TTA methods are often susceptible to the choice of unsupervised loss used at test time, a factor that can substantially influence their overall performance. Test-Time Training (TTT), as presented in [5, 7, 15, 19, 23], offers a compelling alternative to TTA. In TTT, an auxiliary task is learned from the training data (source domain) and subsequently applied during test-time to refine the model. Generally, unsupervised and self-supervised tasks are selected for their capacity to support an adaptable process, without relying on labeled data. Finally, employing a dual-task training approach in the source domain allows the model to be more confident at test time, as it is already familiar with the auxiliary loss.

Motivated by recent developments in machine learning using Noise-Contrastive Estimation (NCE) [1, 16, 18], we introduce a Noise-Contrastive Test-Time-Training (NC-TTT) method that efficiently learns the distribution of sources samples by contrasting it with a noisy distribution. This is achieved by training a discriminator that learns to distinguish noisy out-of-distribution (OOD) features from in-distribution ones. At test time, the output of the discriminator is used to guide the adaptation process, modifying the parameters of the network encoder so that it produces features that match in-distribution ones. Our contributions can be summarized as follows:

- We present an innovative Test-Time Training approach inspired by the paradigm of Noise-Constrastive Estimation (NCE). While NCE was initially proposed for generative

[*] Equal contribution. Correspondence to david.osowiechi.1@ens.etsmtl.ca, gustavo-adolfo.vargas-hakim.1@ens.etsmtl.ca

models as a way to learn a data distribution without having to explicitly compute the partition function [6, 16], and later employed for unsupervised representation learning [1, 18], our work is the first to show the usefulness of this paradigm for test-time training.
- We motivate our method with a principled and efficient framework deriving from density estimation, and use this framework to guide the selection of important hyperparameters.
- In a comprehensive set of experiments, we expose our NC-TTT method to a variety of challenging TTA scenarios, each featuring unique types of domain shifts. Results of these experiments demonstrate the superior performance of our method compared to recent approaches for this problem.

The subsequent sections of this paper are structured as follows. Section 2 reviews prior research on TTA, TTT, and NCE. Section 3 presents our NC-TTT method along with the experimental framework for its evaluation, detailed in Section 4. Section 5 offers experimental results and discussions, while Section 6 concludes the paper with final remarks.

2. Related work

Test-Time Adaptation. TTA is the challenging problem of adapting a pre-trained model from a source domain to an unlabeled target domain in an online manner (i.e., on a batch-wise basis). In this problem, it is assumed that the model no longer has access to source samples, making the setting more realistic and applicable as an *off-the-shelf* tool. Finally, the online nature of TTA also limits the possibility of computing accurate target data distributions, specially when the number of samples is low.

Two classic TTA methods have prevailed in the literature, Prediction Time Batch Normalization (PTBN) [17] and Test-Time Adaptation by Entropy Minimization (TENT) [25]. The former consists in simply recomputing the statistics from each batch of data inside the batch norm layers, instead of using the frozen source statistics. The later goes one step further by minimizing the entropy loss on the model's predictions and updating only the affine parameters of the batch norm layers. Recently, LAME [2] introduced a closed-form optimization mechanism that acts on the model's predictions for target images. This method is based on the Laplacian of the feature maps, which enforces their clustering based on similarity. A more detailed presentation of TTA approaches can be found in [14].

Test-Time Training. TTA methods assume the existence of an implicit property in the model that can be linked to accuracy and can be used for adaptation at test time (e.g., entropy [25]). In contrast, TTT techniques explicitly introduce a given property by learning a secondary task alongside the main classification task at training. As seminal work in the field, TTT [23] introduced a Y-shaped architecture allowing for a self-supervised rotation prediction task. This subnetwork can be attached to any layer of a CNN. Formally, the overall TTT objective is composed of a supervised loss $\mathcal{L}_{sup}$ (e.g., cross-entropy) and an auxiliary, task-dependent loss $\mathcal{L}_{aux}$, as follows:

$$\mathcal{L}_{TTT} = \mathcal{L}_{sup} + \lambda \mathcal{L}_{aux} \tag{1}$$

The auxiliary loss is used at test time to update the model's encoder, reconditioning the features into being more similar to those from the source domain. TTT++ [15] proposed using contrastive learning as the secondary task, while also preserving statistical information from the source domain's feature maps to align the test-time features. Similarly, TTT-MAE [9] used Masked Autoencoder (MAE) [5] image reconstruction as the auxiliary task. Normalizing Flows (NF) [4, 13] have also been employed in TTTFlow [19], adapting the feature encoder at test time by approximating a likelihood-based domain shift detector. Unlike previous approaches, TTTFlow requires two separate training procedures for the original model and the NF network, which makes source training more complex. Recently, ClusT3 [7] introduced an unsupervised secondary task where the projected features of a given layer are clustered using a mutual information maximization objective. Although ClusT3 achieves competitive results, the hyperparameters of this method (e.g., number clusters) are dataset dependent, which limits its generalization capabilities.

Noise-contrastive estimation (NCE). Our work is also related to NCE, a useful tool to model unknown distibutions by *comparison* [6]. In NCE, a dataset is contrasted against a set of noisy points drawn by an arbitrary distribution. A discriminator is then trained to distinguish between both sets, thereby learning the original dataset's properties. This approach has been employed to learn word embeddings [16], training Variational Autoencoders [1], and self-supervised learning (InfoNCE) [18], among others. To our knowledge, this work is the first to investigate the potential of NCE for test-time training. We hypothesize that NCE is well suited to estimate the source domain distribution at training time, and that this estimation can be used in an unsupervised manner at test time to adapt a model to target domain samples.

3. Methodology

We begin by presenting an overview of our NC-TTT method for Test-Time Training. We then proceed to detail the Noise-Contrastive Estimation framework on which it is grounded.

3.1. The proposed method

The problem of Test-Time Training can be formally defined as follows. Let the source domain be represented

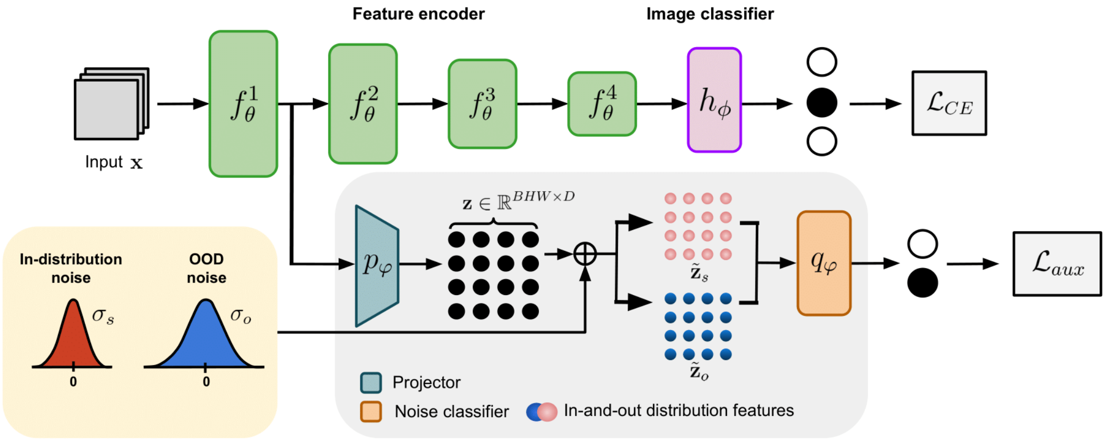


Figure 1. Overview of our Noise-Contrastive Test-Time-Training (NC-TTT) method. The auxiliary module comprises a linear projector p_φ that reduces the scale of features, and a classifier q_φ to discriminate between two different noisy views of the reduced features.

by a joint distribution $\mathcal{P}(\mathcal{X}_s, \mathcal{Y}_s)$, where $\mathcal{X}_s$ and $\mathcal{Y}_s$ correspond to the image and labels spaces, respectively. Likewise, denote as $\mathcal{P}(\mathcal{X}_t, \mathcal{Y}_t)$ the target domain distribution, with $\mathcal{X}_t$ and $\mathcal{Y}_t$ as the respective target images and labels. Following previous research, we consider the likelihood shift [2] between source and target datasets, expressed as $\mathcal{P}(\mathcal{X}_s|\mathcal{Y}_s) \neq \mathcal{P}(\mathcal{X}_t|\mathcal{Y}_t)$, and assume the label space to be the same between domains ($\mathcal{Y}_s = \mathcal{Y}_t$). Given a model $F : \mathcal{X} \rightarrow \mathcal{Y}$ trained on source data $(\mathbf{x}, y) \in \mathcal{X}_s \times \mathcal{Y}_s$, the goal of TTT is to adapt this model to target domain examples from $\mathcal{X}_t$ at test time, without having access to source samples or target labels.

As shown in Fig. 1, our NC-TTT model follows the same Y-shaped architecture as in previous works, with the first branch corresponding to the main classification task and the second one to the auxiliary TTT task. The classification branch can be defined as $F_{\theta,\phi} = (h_\phi \circ f_\theta)$ where $f_\theta = (f_\theta^L \circ \ldots \circ f_\theta^1)$ is an encoder that transforms images into feature maps via L convolutional layers (blocks) and h_ϕ is a classification head that takes features from the last encoder layer and outputs the class probabilities. This branch is trained with a standard cross-entropy loss $\mathcal{L}_{CE}$

Following recent TTT approaches [7, 19], our auxiliary task operates on the features of the encoder. Without loss of generality, we suppose that the features come from layer ℓ of the encoder and denote as $f_\theta^\ell(\mathbf{x}) \in \mathbb{R}^{B \times W \times H \times D}$ the D feature maps of size $W \times H$ for a batch of B images. We first reshape these feature maps to a $(BWH) \times D$ feature matrix and then use a linear projector to reduce its dimensionality, giving projected features $\mathbf{z} = p_\varphi(f_\theta^\ell(\mathbf{x})) \in \mathbb{R}^{BWH \times d}$ with $d \ll D$. Next, we generate two noisy versions of $\mathbf{z}$, an in-distribution version $\widetilde{\mathbf{z}}_s = \mathbf{z} + \boldsymbol{\epsilon}_s$, $\boldsymbol{\epsilon}_s \sim \mathcal{N}(\mathbf{0}, \sigma_s^2 I)$, and an out-of-distribution (OOD) version $\widetilde{\mathbf{z}}_o = \mathbf{z} + \boldsymbol{\epsilon}_o$, $\boldsymbol{\epsilon}_o \sim \mathcal{N}(\mathbf{0}, \sigma_o^2 I)$ where $\sigma_o > \sigma_s$. These noisy features are fed into a discriminator q_φ which predicts in-distribution probabilities $[0, 1]^{BWH}$. This discriminator, which is built using two linear layers with ReLU in between, is trained by minimizing loss $\mathcal{L}_{aux}$ computing the binary cross-entropy between the predicted probabilities and *soft-labels* which will be described in the next section. To update the encoder parameters at test-time, as we do not have class labels, we only compute gradients from $\mathcal{L}_{aux}$.

3.2. Noise-contrastive Test-time Training

We now present our noise-contrastive strategy for test-time training. Let us denote as $p_s(\mathbf{z})$ the probability of features from the source domain. Our method employs a density estimation strategy to learn $p_s(\mathbf{z})$ from training source examples $\mathcal{D}_s = \{\mathbf{z}_i\}_{i=1}^{N_s}$, where $N_s = BWH$. Afterwards, it uses the estimated distribution $\widehat{p}_s(\mathbf{z})$ to adapt the model to distribution shifts at test time.

Estimating the source distribution. We consider the well-known kernel density estimation approach to model $p_s(\mathbf{z})$. This approach puts a small probability mass around each training example $\mathbf{x}_i \in \mathcal{D}_s$, in the shape of a D-dimensional Gaussian with isotropic variance $\Sigma_s = \sigma_s^2 I$, and then estimates the distribution as

$$\widehat{p}_s(\mathbf{z}) = \frac{1}{N_s (2\pi)^{D/2} \sigma_s^D} \sum_{i=1}^{N_s} \exp\left(-\frac{1}{2\sigma_s^2} \|\mathbf{z} - \mathbf{z}_i\|^2\right) \quad (2)$$

At test-time, one could use this probability estimation to define an adaption objective $\mathcal{L}_{aux}$ that minimizes the negative log-likelihood of test examples $\mathcal{D}_t = \{\mathbf{z}_j\}_{j=1}^{N_t}$:

$$\mathcal{L}_{aux} = -\frac{1}{N_t} \sum_{j=1}^{N_t} \log\, \widehat{p}_s(\mathbf{z}_j). \quad (3)$$

However, this simple approach faces two important issues. First, estimating the density in high-dimensional space is problematic since moving away from a training example

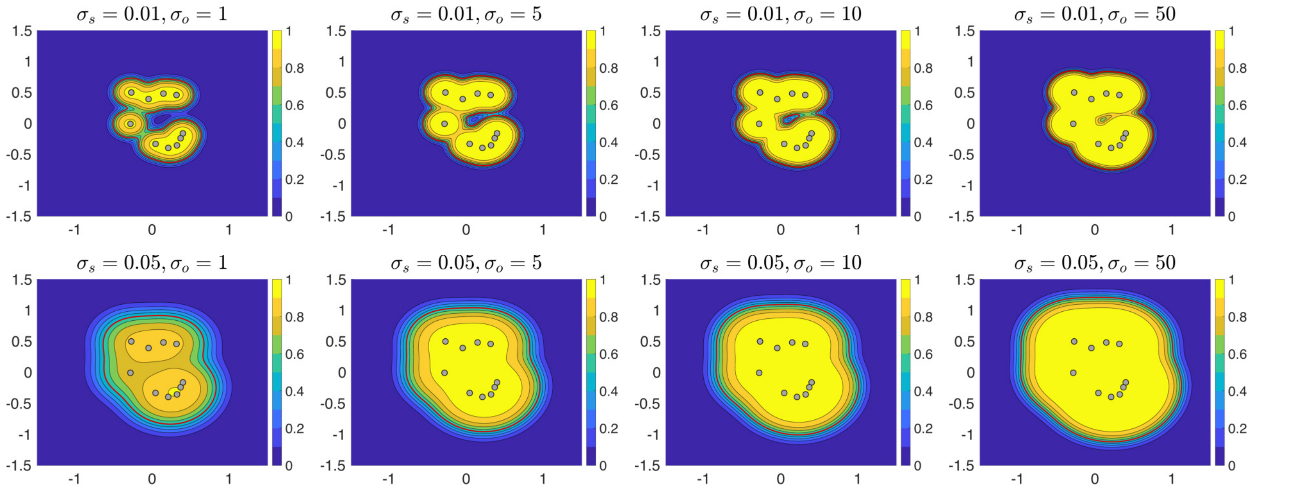


Figure 2. Posterior probability $p(y_s = 1|\mathbf{z})$ of 2D points with different pairs (σ_s, σ_o). The in-domain *influence* expands by increasing σ_o for a fixed σ_s (see difference row-wise). Furthermore, this region is more regular when σ_s increases when σ_o is fixed (see difference column-wise).

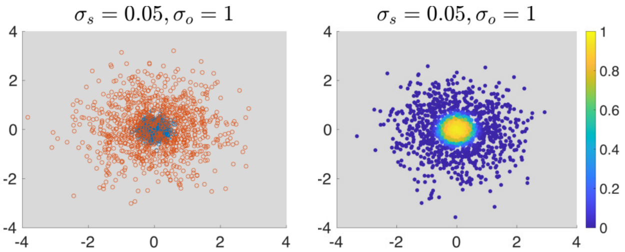


Figure 3. Noise 2D vectors sampled with $\sigma_s = 0.05$ and $\sigma_o = 1$ (*left*). The overlapping of both distributions can be overcome by assigning a probability to each point based on our threshold method.

quickly reduces the probability to zero. Second, the training examples from the source domain are no longer available at test time, hence the density of samples in Eq. (2) cannot be evaluated.

To overcome these issues, we propose a *noise contrastive* approach, which uses a discriminator to learn feature distribution $p_s(\mathbf{z})$. Toward this goal, we contrast $p_s(\mathbf{z})$ with an out-of-domain distribution $p_o(\mathbf{z})$ which is also estimated using Eq. (2) but replacing the variance with σ_o^2, where $\sigma_o > \sigma_s$. Let y_s be a domain indicator variable such that $y_s = 1$ if an example is from the source domain, else $y_s = 0$. Assuming equal priors $p(y_s = 1) = p(y_s = 0)$, we can use Bayes' theorem to get the posterior

$$p(y_s = 1\,|\,\mathbf{z}) \;=\; \frac{\widehat{p}_s(\mathbf{z})}{\widehat{p}_s(\mathbf{z}) \,+\, \widehat{p}_o(\mathbf{z})}. \tag{4}$$

To illustrate this model, we show in Figure 2 the probability $p(y_s = 1\,|\,\mathbf{z})$ obtained for different values of σ_s and σ_o, when training with randomly-sampled 2D points. For a fixed σ_s, increasing σ_o expands the in-domain region around the training samples. Likewise, for the same σ_o, using a greater σ_s gives a larger and more regular (less determined by individual points) in-domain region.

Training the discriminator. To train the discriminator $q_\varphi(\cdot)$, for each training example $\mathbf{z}_i \in \mathcal{D}_s$, we generate $2M$ samples $\widetilde{\mathbf{z}}_{i,m} = \mathbf{z}_i + \boldsymbol{\epsilon}_{i,m}$, the first M from the in-domain distribution, i.e. $\boldsymbol{\epsilon}_{i,m} \sim \mathcal{N}(\mathbf{0}, \sigma_s^2 I)$, and the other M ones from the noisier out-of-domain distribution, i.e. $\boldsymbol{\epsilon}_{i,m} \sim \mathcal{N}(\mathbf{0}, \sigma_o^2 I)$. For these samples, we assume that $\exp(-\|\widetilde{\mathbf{z}}_{i,m} - \mathbf{z}_j\|_2^2/2\sigma_s^2) \approx 0$, for $j \neq i$, hence the posterior simplifies to

$$p(y_s = 1\,|\,\widetilde{\mathbf{z}}_{i,m}) \;=\; \frac{\sigma_s^{-D}\exp\!\Big(-\frac{1}{2\sigma_s^2}\|\boldsymbol{\epsilon}_{i,m}\|^2\Big)}{\sigma_s^{-D}\exp\!\Big(-\frac{1}{2\sigma_s^2}\|\boldsymbol{\epsilon}_{i,m}\|^2\Big) + \sigma_o^{-D}\exp\!\Big(-\frac{1}{2\sigma_o^2}\|\boldsymbol{\epsilon}_{i,m}\|^2\Big)} \tag{5}$$

where $\boldsymbol{\epsilon}_{i,m} = \widetilde{\mathbf{z}}_{i,m} - \mathbf{z}_i$. For large values of D, this formulation is numerically unstable it leads to *division by zero* errors. Instead, we use an equivalent formulation $p(y_s = 1\,|\,\widetilde{\mathbf{z}}) = \mathrm{sigmoid}(u)$, where pre-activation "logit" u is given by

$$u \;=\; \frac{1}{2}\left(\frac{1}{\sigma_o^2} - \frac{1}{\sigma_s^2}\right)\|\boldsymbol{\epsilon}_{i,m}\|^2 \,+\, D\log\left(\frac{\sigma_o}{\sigma_s}\right) \tag{6}$$

See Appendix A in the supplementary material for a proof. The in-domain region, $p(y_s = 1\,|\,\widetilde{\mathbf{z}}) \geq 0.5$, which corresponds to the case where $u \geq 0$, is thus defined by the following condition:

$$\|\boldsymbol{\epsilon}_{i,m}\| \;\leq\; \sigma_s\sigma_o\sqrt{\frac{2D}{(\sigma_s^2 - \sigma_o^2)}\log\left(\frac{\sigma_s}{\sigma_o}\right)} \tag{7}$$

Figure 3 shows examples of noise vectors $\boldsymbol{\epsilon}$ sampled with $\sigma_s = 0.05$ and $\sigma_o = 1$ (*left*), and their corresponding posterior probability (*right*). As can be seen, the posterior

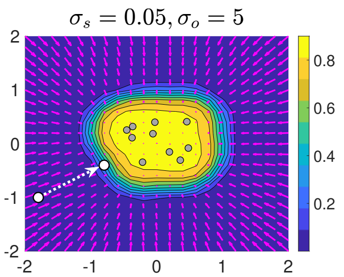


Figure 4. Heatmap of in-distribution probabilities, i.e., $p(y_s = 1 \mid \mathbf{z})$ approximated by $q_\varphi(\mathbf{z})$ in our model, and spatial gradient of log-likelihood function, i.e. $\nabla \log q_\varphi(\mathbf{z})$, which is used as test-time adaptation objective. The arrow shows how an OOD test sample (white point) is adapted toward the source distribution.

probability correctly separates in-distribution samples from OOD ones. Doing so, it overcomes the problem of having OOD samples that are similar to in-distribution ones (red circles near the center), which would confuse the discriminator during training.

Using these samples $\widetilde{\mathbf{z}}_{i,m}$, we train the discriminator $q_\varphi(\cdot)$ by minimizing the cross-entropy between its prediction and the soft-label $\widetilde{p}_{i,m} = p(y_s = 1 \mid \widetilde{\mathbf{z}}_{i,m})$:

$$\begin{aligned}\mathcal{L}_{aux} \,=\, -\,\frac{1}{2MN_s}\sum_{i=1}^{N_s}\sum_{m=1}^{2M} \widetilde{p}_{i,m} \log q_\varphi(\widetilde{\mathbf{z}}_{i,m}) \\ +\, \big(1-\widetilde{p}_{i,m}\big)\log\big(1-q_\varphi(\widetilde{\mathbf{z}}_{i,m})\big)\end{aligned} \tag{8}$$

Adapting the model at test time. During inference, we adapt the parameters of the encoder in layers where the auxiliary loss is computed, as well as those of preceding layers. The adaptation modifies the encoder so that the trained discriminator $q_\varphi(\cdot)$ perceives the encoded features $\{\mathbf{z}_j\}_{j=1}^{N_t}$ of test examples as being in-distribution. This is achieved by minimizing the following test-time loss:

$$\mathcal{L}_{aux}^{test} \,=\, -\frac{1}{N_t}\sum_{j=1}^{N_t} \log q_\varphi(\mathbf{z}_j) \tag{9}$$

As illustrated in Fig. 4, our method models the in-distribution probability $p(y_s = 1 \mid \mathbf{z})$ using NCE and then approximates this distribution with discriminator $q_\varphi(\cdot)$. At test time, the encoder is updated to move OOD features (white point) toward the source distribution, making them more suitable for the source-trained classifier. Thanks to the non-zero in-distribution noise ($\sigma_s > 0$), we avoid over-adapting the encoder (the white point stops at the border of the in-distribution region and not at a training sample), a problem often found in other TTT approaches.

3.3. Selecting the distribution variances

Our model requires to specify the in-distribution variance σ_s^2 and the OOD variance σ_o^2. In this section, we present how these can be chosen. The OOD variance should be greater than the in-distribution, hence we can write $\sigma_o = \beta\sigma_s$, with $\beta = \sigma_o/\sigma_s > 1$. Hence, β is a measure of noise ratio for the in-distribution and OOD samples. Using this relationship, Eq. (6) simplifies to

$$u \,=\, -\frac{1}{2\sigma_s^2}\left(\frac{\beta^2-1}{\beta^2}\right)\|\boldsymbol{\epsilon}\|^2 \,+\, D\log\beta \tag{10}$$

For OOD samples, the expected value of "logit" u is then given by

$$\begin{aligned}\overline{u}_\beta \,&=\, \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\sigma_o^2 I)}\left[-\frac{1}{2\sigma_s^2}\left(\frac{\beta^2-1}{\beta^2}\right)\|\boldsymbol{\epsilon}\|^2 \,+\, D\log\beta\right] \\ &= -\frac{1}{2\sigma_s^2}\left(\frac{\beta^2-1}{\beta^2}\right)\underbrace{\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\sigma_o^2 I)}\left[\|\boldsymbol{\epsilon}\|^2\right]}_{\sigma_o^2=\beta^2\sigma_s^2} + D\log\beta \\ &= -\frac{1}{2}\left(\beta^2-1\right) \,+\, D\log\beta\end{aligned} \tag{11}$$

Figure 5 show how the expected in-distribution prediction $\mathbb{E}[y_s \mid \mathbf{z}] \,=\, \mathrm{sigmoid}(\overline{u}_\beta)$ varies as function of β, for $D = 16$ (the dimension used in our experiments). In this case, to have near-zero probability for OOD samples, one can choose any $\beta > 1.5$. In our experiments, we selected $\beta = 2$.

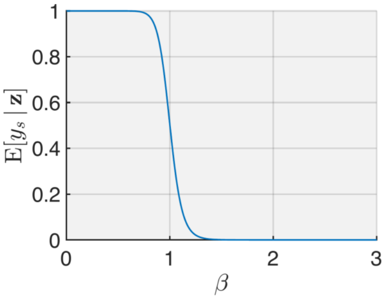


Figure 5. Expected in-distribution label as a function of noise ratio $\beta = \sigma_o/\sigma_s$.

4. Experimental Settings

We evaluate NC-TTT on several TTT datasets, following the protocol of previous works. These benchmarks emulate different challenging domain shift scenarios, which help evaluating the effectiveness of our approach. As in [7, 23], these benchmarks are categorized as *common corruptions*, and *sim-to-real* domain shift.

For *common corruptions*, we evaluate our method on CIFAR-10-C and CIFAR-100-C [10]. This family of domain shifts include 15 different corruptions such as Gaussian noise, JPEG compression, among others. Each corruption has 5 different levels of severity with 10,000 images, which amounts to 75 different testing scenarios. For each

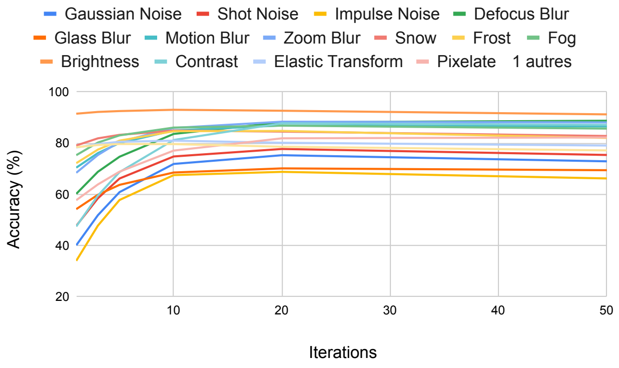


Figure 6. Evolution of accuracy on all corruptions in CIFAR-10-C.

of the aforementioned datasets, CIFAR-10 and CIFAR-100 are used as source domains, with 10 and 100 classes respectively. Finally, the challenging large-scale VisDA-C [20] dataset corresponds to the *sim-to-real domain shift*. The source domain comprises a training set of 152,397 images of 3D renderings from 12 different classes, while the test set consists in 72,372 video frames of the same categories.

Source training. The cross-entropy and auxiliary losses are jointly trained on the source dataset. We explored different architectural choices for each setting. For common corruptions (i.e. CIFAR-10/100-C), we define the projector as a 1×1 convolutional layer that reduces the number of channels to $D = 96$ to later be flattened for classification. We utilize a discriminator composed of two linear layers with a Batch Norm layer and Leaky ReLU in between, and a hidden dimension of 1024 in the intermediate layer. For this particular case, we use the tuple $(\sigma_s = 0, \sigma_o = 0.015)$, which was experimentally determined as it produced the best performance. The model is trained using 128 images per batch for 350 epochs using SGD, an initial learning rate of 0.1, and a multi-step scheduler with a decreasing factor of 10 at epochs 150 and 250. Due to the challenging nature of the *sim-to-real* domain shift from VisDA-C, we escalate the architecture to make it able to learn more source domain information. We utilize a 1×1 convolutional projector with an output number of channels of $D = 16$. As opposed to flattening the features, we also employ two 1×1 convolutional layers for the discriminator, with an intermediate number of channels of 1024. The noise values are sampled with $(\sigma_s = 0.025, \sigma_o = 0.05)$ and added *pixel-wise* to the projected feature maps. Following related works' protocol for VisDA-C, we use an ImageNet-pre-trained model [3] as a warm start, to then perform the source training with a batch size of 50 for 100 epochs with SGD and a learning rate of 0.01. ResNet50 [8] is the chosen architecture for all datasets.

Test-time adaptation. Adaptation is performed on the encoder's blocks (including BatchNorm layers). If the auxiliary task is plugged to the third layer block, for instance, the weights of all the previous blocks will be optimized. The source training on CIFAR-10 is used to adapt for CIFAR-10-C. In an analog way, CIFAR-100 is utilized to adapt for CIFAR-100-C. For all this cases, the ADAM optimizer with a learning rate of 10^{-5} is used in batches of 128 images. As for VisDA-C, a batch size of 50 is employed with a learning rate of 10^{-4}. The weights of the source model are restored after each batch.

Benchmarking. We compare the performance of NC-TTT with previous works from the *state-of-the-art* in TTT and TTA. Chosen works in TTA include PTBN [17], TENT [25], and LAME [2], whereas for TTT we consider TTT [23], TTT++ [15], and ClusT3 [7]. We utilize the source model (named ResNet50 in our results) without adaptation to measure accuracy gains.

5. Results

In this section, we present the experimental results obtained from NC-TTT and compare them against the *state-of-the-art*. In accordance with previous TTT research, we also offer insights on the working mechanisms that take part in the success of our technique.

5.1. Image classification on common corruptions

We assess the performance of ClusT3 using the CIFAR-10/100-C dataset, considering 15 distinct corruptions. Subsequently, our experiments concentrate exclusively on Level 5, recognized as the most demanding adaptation scenario. Comprehensive results for all severity levels are provided in the Supplementary material.

The data presented in Fig 6 reveals that peak accuracy is typically reached around 20 iterations, depending on the specific corruption type. Remarkably, accuracy remains stable even beyond the 20th iteration. In the case of certain corruptions, specifically the ones with noise such as Impulse Noise which significantly degrade the image quality, we observe a decline in performance with an increase in the number of adaptation iterations.

As shown in Table 1, NC-TTT achieves an average improvement of 30.61% with respect to the baseline (i.e. ResNet50), and obtains a considerable advantage in all the different corruptions. Moreover, our method achieves to outperform ClusT3 in most corruptions and in average for the whole dataset. It is worth noticing that, besides the strong relation of NC-TTT to Gaussian-like noise, the performance on the *Gaussian Noise* corruption is not necessarily the highest, which could be due to the fact that the auxiliary task does not bias the model towards any type of domain shift. Table 2 shows a more surprising trend

	ResNet50	LAME [2]	PTBN [17]	TENT [25]	TTT [23]	TTT++ [15]	ClusT3 [7]	NC-TTT (ours)
Gaussian Noise	21.01	22.90 ±0.07	57.23 ±0.13	57.15 ±0.19	66.14 ±0.12	75.87 ±5.05	**76.01 ±0.19**	75.30 ±0.04
Shot noise	25.77	27.11 ±0.13	61.18 ±0.03	61.08 ±0.18	68.93 ±0.06	77.18 ±1.36	77.67 ±0.17	**77.74 ±0.05**
Impulse Noise	14.02	30.99 ±0.15	54.74 ±0.13	54.63 ±0.15	56.65 ±0.03	**70.47 ±2.18**	69.76 ±0.15	68.80 ±0.11
Defocus blur	51.59	45.16 ±0.13	81.61 ±0.07	81.39 ±0.22	88.11 ±0.08	86.02 ±1.35	87.85 ±0.11	**88.77 ±0.09**
Glass blur	47.96	36.58 ±0.06	53.43 ±0.11	53.36 ±0.14	60.67 ±0.06	69.98 ±1.62	**71.34 ±0.15**	70.15 ±0.16
Motion blur	62.30	55.41 ±0.15	78.20 ±0.28	78.04 ±0.17	83.52 ±0.03	85.93 ±0.24	86.10 ±0.11	**86.93 ±0.05**
Zoom blur	59.49	51.48 ±0.20	80.29 ±0.13	80.26 ±0.22	87.25 ±0.03	**88.88 ±0.95**	86.68 ±0.05	88.40 ±0.06
Snow	75.41	66.14 ±0.12	71.59 ±0.21	71.59 ±0.04	79.29 ±0.05	82.24 ±1.69	83.71 ±0.09	**84.92 ±0.08**
Frost	63.14	50.03 ±0.22	68.77 ±0.25	68.52 ±0.20	79.84 ±0.11	82.74 ±1.63	83.69 ±0.03	**84.79 ±0.05**
Fog	69.63	64.56 ±0.19	75.79 ±0.05	75.73 ±0.10	84.46 ±0.09	84.16 ±0.28	85.12 ±0.13	**86.85 ±0.10**
Brightness	90.53	84.27 ±0.10	84.97 ±0.05	84.77 ±0.13	91.23 ±0.08	89.07 ±1.20	91.52 ±0.02	**93.05 ±0.03**
Contrast	33.88	31.46 ±0.23	80.81 ±0.15	80.70 ±0.15	**88.58 ±0.09**	86.60 ±1.39	84.40 ±0.11	87.78 ±0.15
Elastic transform	74.51	64.23 ±0.10	67.14 ±0.17	67.13 ±0.10	75.69 ±0.10	78.46 ±1.83	**82.04 ±0.17**	80.99 ±0.11
Pixelate	44.43	39.32 ±0.08	69.17 ±0.31	68.70 ±0.29	76.35 ±0.19	**82.53 ±2.01**	82.03 ±0.09	82.26 ±0.11
JPEG compression	73.61	66.19 ±0.02	65.86 ±0.05	65.83 ±0.07	73.10 ±0.19	81.76 ±1.58	**83.24 ±0.10**	79.66 ±0.06
Average	53.82	49.06	70.05	69.93	77.32	81.46	82.08	**82.43**

Table 1. Accuracy (%) on CIFAR-10-C dataset with Level 5 corruption for NC-TTT compared to previous TTA and TTT methods.

	ResNet50	LAME [2]	PTBN [17]	TENT [25]	TTT [23]	ClusT3 [7]	NC-TTT (ours)
Gaussian Noise	12.67	10.55 ±0.08	43.00 ±0.16	43.17 ±0.24	33.99 ±0.11	**49.77 ±0.18**	46.03 ±0.12
Shot noise	14.79	12.58 ±0.04	44.57 ±0.16	44.47 ±0.23	36.55 ±0.08	**50.54 ±0.16**	47.04 ±0.14
Impulse Noise	6.47	5.83 ±0.07	36.76 ±0.11	36.64 ±0.28	26.87 ±0.08	**44.35 ±0.31**	41.53 ±0.11
Defocus blur	29.97	29.07 ±0.11	66.68 ±0.06	66.74 ±0.06	65.96 ±0.14	64.40 ±0.12	**67.00 ±0.09**
Glass blur	21.36	19.58 ±0.02	45.17 ±0.08	45.09 ±0.06	34.90 ±0.01	**50.78 ±0.24**	48.08 ±0.07
Motion blur	39.60	41.26 ±0.09	62.61 ±0.17	62.54 ±0.23	57.10 ±0.10	62.62 ±0.15	**64.31 ±0.02**
Zoom blur	35.75	34.93 ±0.02	65.36 ±0.03	65.29 ±0.05	62.90 ±0.07	63.81 ±0.08	**66.24 ±0.25**
Snow	42.05	43.58 ±0.20	52.82 ±0.27	52.31 ±0.16	54.97 ±0.03	55.84 ±0.12	**58.70 ±0.10**
Frost	31.44	32.67 ±0.12	51.92 ±0.09	51.79 ±0.23	54.60 ±0.16	55.46 ±0.06	**58.55 ±0.11**
Fog	30.96	35.95 ±0.12	55.78 ±0.05	55.91 ±0.28	55.80 ±0.09	51.39 ±0.07	**57.73 ±0.17**
Brightness	61.80	64.84 ±0.03	66.20 ±0.06	66.47 ±0.06	73.25 ±0.06	66.71 ±0.11	**71.36 ±0.10**
Contrast	12.31	15.50 ±0.04	60.84 ±0.15	60.91 ±0.19	60.97 ±0.09	54.67 ±0.05	**61.53 ±0.20**
Elastic transform	53.06	51.32 ±0.13	56.38 ±0.04	56.43 ±0.33	53.51 ±0.04	59.44 ±0.27	**60.25 ±0.04**
Pixelate	26.08	27.65 ±0.02	58.21 ±0.14	58.19 ±0.22	50.39 ±0.05	60.75 ±0.09	**61.17 ±0.33**
JPEG compression	52.19	49.95 ±0.07	51.65 ±0.16	51.30 ±0.16	49.62 ±0.09	**59.94 ±0.12**	55.69 ±0.09
Average	31.37	31.68	54.53	54.48	51.43	56.70	**57.68**

Table 2. Accuracy (%) on CIFAR-100-C dataset with Level 5 corruption for NC-TTT and the works from the *state-of-the-art*.

on CIFAR-100-C, as our technique outperforms the closest competitor on the majority of the corruptions, and obtains an average improvement of 26.31% with respect to ResNet50. Based on the above, NC-TTT can approximate the source information even when: a) the number of classes increases, and b) the auxiliary task works at a smaller scale as the main classification task.

Figure 7 demonstrates the impact of NC-TTT during adaptation through t-SNE plots showcasing the target feature maps before and after adaptation, along with the associated model predictions. The challenging corruption of shot noise becomes more manageable with the assistance of NCE, contributing to improved predictions by refining the clustering of diverse class samples within the target dataset.

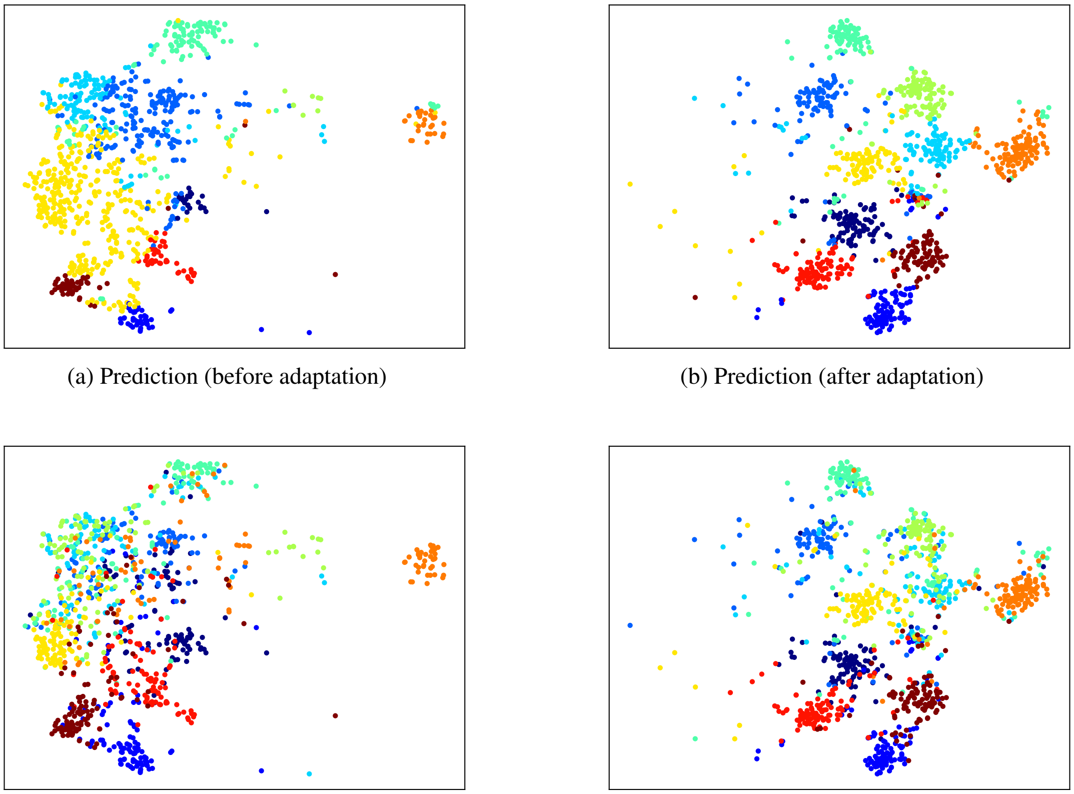
5.2. Image classification on sim-to-real domain shift

For adaptation on VisDA-C, the first encoder's layer block is chosen for the auxiliary task. The obtained results concur with previous works [7, 23], in that the first layers of the network's encoder are sufficient for adaptation.

As shown in Table 3, NC-TTT obtains a competitive performance with respect to previous works on VisDA-C. The severe domain shift in this dataset makes it a very challenging scenario, as can be seen when testing the source model. NC-TTT obtains a gain of 16.19% in accuracy, and surpasses previous methods by an important margin.

6. Conclusions

We proposed NC-TTT, a Test-Time Training method based on the popular theory of Noise-Contrastive Estimation. Our



(a) Prediction (before adaptation)

(b) Prediction (after adaptation)

(c) Ground truth (before adaptation)

(d) Ground truth (after adaptation)

Figure 7. t-SNE visualizations depict shot noise characteristics in the features extracted from NC-TTT. Panels (a) and (b) illustrate the model predictions without and with 20 iterations of adaptation, respectively. Panels (c) and (d) showcase the ground truth labels in the absence of adaptation and for the adapted representations, respectively.

Method	**Acc. (%)**
ResNet50	46.31
LAME-L [2]	22.02 ±0.23
LAME-K [2]	42.89 ±0.14
LAME-R [2]	19.33 ±0.11
PTBN [17]	60.33 ±0.04
TENT [25]	60.34 ±0.05
TTT [23]	40.57 ±0.02
ClusT3 [7]	61.91 ±0.02
NC-TTT (ours)	**62.71** ±**0.09**

Table 3. Results on VisDA-C.

method learns a proximal representation of the source domain by discriminating between noisy views of feature maps. The entire model can be added on top of any given layer of a CNN's encoder, and comprises only a linear projector and a classifier.

The proposed experiments support already established hypothesis of TTT, which states that adaptation in the first encoder's layer blocks (e.g. first or second) is often sufficient to recover the model's performance on a new domain. NC-TTT is evaluated on different challenging benchmarks, and its performance is compared against recent *state-of-the-art* methods in the field.

This work leads to interesting questions that can be addressed as future work. First, different types of added noise could be explored to analyze their impact in the learning of the auxiliary task. A similar framework can eventually be derived for different distributions. Moreover, and as an open question partaking all the existent TTT methods, the exact mechanisms that allow auxiliary tasks to learn domain-related information are unclear. This is especially intriguing considering that the scale of such tasks is small compared to the classification task. Their properties and their relation with the models' performance a suitable research direction.

References

[1] Jyoti Aneja, Alex Schwing, Jan Kautz, and Arash Vahdat. A contrastive learning approach for training variational autoencoder priors. *Advances in neural information processing systems*, 34:480–493, 2021. 1, 2

[2] Malik Boudiaf, Romain Mueller, Ismail Ben Ayed, and

Luca Bertinetto. Parameter-free online test-time adaptation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8344–8353, 2022. 1, 2, 3, 6, 7, 8

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 6

[4] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016. 2

[5] Yossi Gandelsman, Yu Sun, Xinlei Chen, and Alexei A Efros. Test-time training with masked autoencoders. In *Advances in Neural Information Processing Systems*, 2022. 1, 2

[6] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 297–304. JMLR Workshop and Conference Proceedings, 2010. 2

[7] Gustavo A. Vargas Hakim, David Osowiechi, Mehrdad Noori, Milad Cheraghalikhani, Ali Bahri, Ismail Ben Ayed, and Christian Desrosiers. ClusT3: Information Invariant Test-Time Training. pages 6136–6145, 2023. 1, 2, 3, 5, 6, 7, 8

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 6

[9] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16000–16009, 2022. 2

[10] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. 2019. 5

[11] Ansh Khurana, Sujoy Paul, Piyush Rai, Soma Biswas, and Gaurav Aggarwal. Sita: single image test-time adaptation. *arXiv:2112.02355 [cs]*, 2021. arXiv: 2112.02355. 1

[12] Donghyun Kim, Kaihong Wang, Stan Sclaroff, and Kate Saenko. A broad study of pre-training for domain generalization and adaptation. In *Computer Vision – ECCV 2022*, pages 621–638, Cham, 2022. Springer Nature Switzerland. 1

[13] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2018. 2

[14] Jian Liang, Ran He, and Tieniu Tan. A comprehensive survey on test-time adaptation under distribution shifts. *arXiv preprint arXiv:2303.15361*, 2023. 2

[15] 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? *Neural Information Processing Systems (NeurIPS)*, 2021. 1, 2, 6, 7

[16] Andriy Mnih and Koray Kavukcuoglu. Learning word embeddings efficiently with noise-contrastive estimation. *Advances in neural information processing systems*, 26, 2013. 1, 2

[17] 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:2006.10963 [cs, stat]*, 2021. arXiv: 2006.10963. 2, 6, 7, 8

[18] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. 1, 2

[19] David Osowiechi, Gustavo A. Vargas Hakim, Mehrdad Noori, Milad Cheraghalikhani, Ismail Ayed, and Christian Desrosiers. Tttflow: Unsupervised test-time training with normalizing flow. In *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 2125–2126, Los Alamitos, CA, USA, 2023. IEEE Computer Society. 1, 2, 3

[20] Xingchao Peng, Ben Usman, Neela Kaushik, Dequan Wang, Judy Hoffman, and Kate Saenko. Visda: A synthetic-to-real benchmark for visual domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2018. 1, 6

[21] 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. IEEE, 2019. 1

[22] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do CIFAR-10 classifiers generalize to cifar-10? *CoRR*, abs/1806.00451, 2018. 1

[23] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A. Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *International Conference on Machine Learning (ICML)*, 2020. 1, 2, 5, 6, 7, 8

[24] Riccardo Volpi, Hongseok Namkoong, Ozan Sener, John C Duchi, Vittorio Murino, and Silvio Savarese. Generalizing to unseen domains via adversarial data augmentation. *Advances in neural information processing systems*, 31, 2018. 1

[25] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. 2021. 1, 2, 6, 7, 8

[26] Jindong Wang, Cuiling Lan, Chang Liu, Yidong Ouyang, Tao Qin, Wang Lu, Yiqiang Chen, Wenjun Zeng, and Philip Yu. Generalizing to unseen domains: A survey on domain generalization. *IEEE Transactions on Knowledge and Data Engineering*, 2022. 1

[27] Kaiyang Zhou, Yongxin Yang, Yu Qiao, and Tao Xiang. Domain generalization with mixstyle. In *International Conference on Learning Representations*, 2020. 1