TTT++: Improved Test-Time Training

Anonymous Author(s)
Affiliation
Address
email

Abstract

Test-Time Adaptation (TTA) aims at updating trained models on test data. It has garnered great interest to tackle generalization in the presence of distributional shifts, i.e., when training and test-time data distributions differ. Test-Time Training (TTT) is a promising paradigm that leverages an auxiliary Self-Supervised Learning (SSL) task at test-time to improve generalization. In this work, we first shed light on some limitations of TTT, and show that it can possibly deteriorate, instead of improving, the test-time performance. To address this issue, we introduce an online feature alignment strategy by utilizing offline feature summarization. Specifically, we compute feature moments after training, and align the test-time feature distribution to the training one through moment matching. Given perfectly aligned feature distributions, we derive expressions for the test-time performance, and show that a careful design of SSL tasks should yield better TTT adaptation. Therefore, we propose to incorporate a suitable SSL task in the form of contrastive learning into the TTT framework. We empirically demonstrate that our proposed TTA strategy, termed *TTT++*, outperforms state-of-the-art methods by a significant margin on multiple popular vision benchmarks.

1 Introduction

Machine learning models often struggle to generalize under distribution shifts. When the test data do not come from the same distribution as the training one, even mild difference, e.g., JPEG compression, may cause severe prediction errors [1]. In practice, this shift can easily happen in uncontrolled environments [2], and is a real challenge for effective deployment. One popular solution to this issue is to learn a representation invariant to distributional shifts [3]. For this, most previous methods rely on the simultaneous use of training and test data [4–7], which can be impractical due to increasing privacy concerns, inflating sizes of datasets, and other constraints. This issue has shifted the focus of the research community toward the *Test-Time Adaptation* (TTA) problem: given a trained model, how to effectively adapt it on test data, without any training data or human annotations?

One promising TTA approach to tackle distributional shifts is Test-Time Training (TTT) [8]. In this framework, an additional self-supervised learning (SSL) task is considered, which does not require manual annotations. The model is learned from both the main task at hand and this auxiliary SSL one during training, and only from the SSL task at test-time. In the context of image classification, TTT consistently improves the robustness to various distributional shifts when using the rotation prediction task as SSL [8]. However, the empirical performance of this method is often inferior to other families of TTA algorithms [9, 10].

In this paper, we first give an in-depth look at TTT, and shed light on some of its weaknesses. Our analysis starts with a simple question: can TTT always reduce the effects of distributional shifts? Through an illustrative toy problem, we show that the vanilla TTT framework may lead to surprising failures, severely deteriorating, rather than improving, the test-time performance. This problem is largely attributed to unconstrained updates from the SSL task, interfering with the main one. To

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

address it, we introduce an online feature alignment loss term by making use of offline feature summarization, not requiring further access to the training data at test-time. In particular, we collect the mean and covariance matrix of training features to represent the training feature distribution once learning is done. At test-time, we maintain running estimates of these moments while the model is adapted, and enforce distribution alignment by matching the corresponding pre-computed and estimated moments.

Secondly, given perfectly aligned feature distributions, we derive a lower bound on the test performance, and further show that this bound increases when the SSL tasks are more related to the main one. A right choice of SSL task should therefore improve generalization guarantees. Based on these theoretical findings, we motivate the use of more informative SSL tasks for TTT. Contrastive SSL [11–14] is a well known approach to effectively learn semantics from unlabeled data, so that learned features carry useful information for various downstream tasks. We integrate it into TTT as a strong instance of a SSL task that benefits from good theoretical guaranties.

We introduce our model TTT++, combining online feature alignment and contrastive Self-Supervised Learning into TTT. Through extensive experiments, we demonstrate that TTT++ significantly boosts the test-time performance on multiple vision benchmarks. It outperforms state-of-the-art methods, including TENT [10] and SHOT [9], by significant margins on CIFAR10-C/CIFAR100-C [1], CIFAR10.1 [15], and the large-scale VisDA [16] benchmark.

2 Related Work

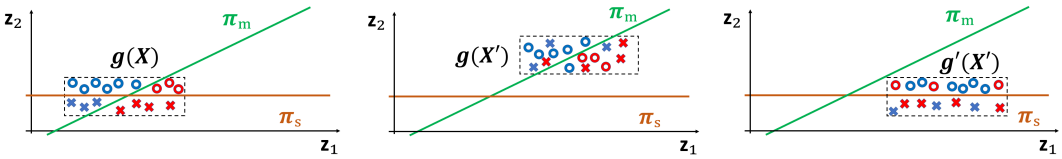
Test-time Adaptation. The idea of adapting trained machine learning models based on test samples has been explored in several contexts. [17] introduced a super-resolution method that can learn from each test image without relying on any prior training example. Similarly, [18] proposed to adapt the learned image prior based on test instances for semantic image manipulation. Some other works [19, 20] attempt to improve generative adversarial networks (GANs) at test time by updating feature activations in the GAN generator using the feedback from the GAN discriminator. Different from these generative problems, our work is focused on classification problems in the presence of distributional shifts.

Early work from [21] advocated model adaptation with limited reuse of source data, e.g., a few representative examples. More recently, several methods [22] have been designed for the source-free setting where no source data is accessible at test-time. [23] showed that adapting the activation statistics estimated by batch normalization can be an effective way to improve the robustness of deep models against common corruptions. Another line of work proposed to adapt the model parameters by exploiting the outputs on test samples, such as entropy minimization [10] or pseudo-labeling [9]. While these methods yield strong results on some benchmarks, they are often restricted to classification problems and vulnerable under large distribution shifts [24].

More closely related to ours, [8] proposed to adapt model parameters at test-time by learning from a self-supervised task, *e.g.*, predicting image rotation. This approach does not involve any assumptions over the output of the main task and has been applied to several problems, such as instance tracking [25] and Reinforcement Learning [26]. Nevertheless, the results of this approach were shown to be inferior to other families of methods [10]. In this work, we theoretically analysis its limitations and introduce simple yet critical modifications that outperform state-of-the-art methods.

Feature Alignment. Matching the distribution of feature activations between the training and test samples is a widely used technique in domain adaptation. It is often achieved by either explicitly minimizing a divergence measure, such as MMD [27], Coral [28] and CMD [29], or by employing an adversarial loss function to encourage domain confusion [7, 30]. However, most of these methods rely on the co-existence of source and target data, and thus cannot be readily applied to the test-time adaptation setting where source data is not available. Our work revisits the critical role of feature alignment for self-supervised adaptation at test-time and proposes a simple and practical strategy to summarize the training features for online feature alignment.

Self-supervised Learning. Self-supervised learning is a powerful paradigm to learn rich visual representations from unlabeled images. Stunning progress has been made in recent years by designing informative self-supervised tasks [12, 13, 31, 32] and stabilizing the training process [14, 33]. Yet,



(a) Training Time **(b)** Test-Time before TTA **(c)** Test-Time after TTA

Figure 1: Qualitative illustration of a failure case of Test-Time Training. Even if the classifiers π_s and π_m perfectly fit the training data X in the encoder space (a), the classifiers no longer perform optimally on test data X' without adaptation, due to a distributional shift (b). After adapting the encoder g into g' on the test data X' based solely on self-supervision, representations in the encoder space change, and the main task classifier π_m performs worse (c).

previous works are largely focused on pre-training without labels. Our work sheds light on the importance of incorporating strong self-supervised learning for test-time adaptation.

3 When Does Test-Time Training Fail?

Test-Time Training (TTT) [8] is a general framework for TTA based on Self-Supervised Learning (SSL) with unlabeled test samples, and is illustrated in Figure 2. We consider two tasks here: a main one defined by the problem addressed, and a SSL one used for TTT. To solve them, we consider a multi-task network composed of a shared encoder g and two sibling decoder heads π_m and π_s for the main and SSL tasks. Given a training labeled dataset $D = \{(x_i, y_i)\}_{i \in \{1,\dots,N\}}$, the training loss function $\mathcal{L}_{train}$ is the sum of a main task loss function $\mathcal{L}_m$ and a self-supervised loss function $\mathcal{L}_s$, both adapted to their respective tasks:

$$\mathcal{L}_{train}(D; g, \pi_m, \pi_s) = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}_m(x_i, y_i; g, \pi_m) + \lambda \mathcal{L}_s(x_i; g, \pi_s), \tag{1}$$

where λ is a hyper-parameter to balance the two tasks during training.

In the presence of a distributional shift, the learned model may not generalize well to test samples $D' = \{x'_i\}_{i \in \{1,\dots,N'\}}$. The goal of TTT is to update the shared encoder g into g' at test-time, in hope that the updated model $\pi_m \circ g'$ generalizes better to the test distribution. The test-time loss function $\mathcal{L}_{test}$ is then only based on the SSL task, whose labels are always available:

$$\mathcal{L}_{test}(D'; g') = \frac{1}{N'} \sum_{i=1}^{N'} \mathcal{L}_s(x'_i; g'). \tag{2}$$

Despite its simplicity, it has been demonstrated that updating the encoder through image rotation prediction can consistently improve the robustness of image classifiers on several benchmarks [8]. Nevertheless, it was shown less effective than other TTA methods [10]. In this section, we analyze some limitations of TTT, and then propose a practical solution to address them.

3.1 Illustrative Example of a Failure Case

In order to analyze the behavior of TTT, we now introduce an illustrative example revealing some of its limitations. Despite how apparently simple the following example may seem, in particular for a neural network, we show that TTT may surprisingly lead to failures under a significant distributional shift. The problem, illustrated in Figure 1, consists in separating blue and red points, with the SSL task of separating circles and crosses. We assume having trained classifiers π_m and π_s for these two tasks, with perfect performances on the training data as shown in Figure 1 (a).

In presence of a distributional shift, displayed in Figure 1 (b), the feature distribution changes and the trained classifiers π_m and π_s are no longer perfect, even though they may still perform decently. After updating the encoder g through the SSL task, see Figure 1 (c), the SSL classifier π_s now performs well. However, this may lead to a worse performance for the main classifier π_m.

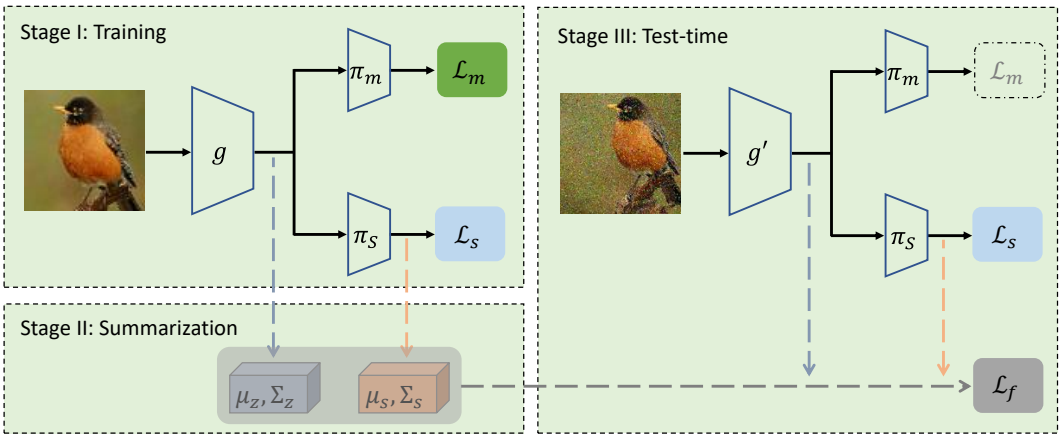


Figure 2: Our proposed Test-Time Adaptation strategy (TTT++). Our method consists of three stages: model training, offline feature summarization, test-time adaptation. (i) During training, the model is jointly optimized with respect to the main task and contrastive self-supervised task; (ii) For offline feature summarization, we collect and save the empirical mean and covariance matrix of the training feature distributions at the outputs of the encoder and of the self-supervised head; (iii) At test-time, we adapt the encoder g based on self-supervised learning and online feature alignment, which allows our model $\pi_m \circ g'$ to generalize better on the main task under distributional shifts.

In addition to this illustrative example, we provide empirical evidence in the supplementary material, Section A, that this phenomenon can happen in practice, with simulation on a synthetic problem inspired from the problem of confounding variables.

3.2 Online Feature Alignment

As illustrated previously, simply applying self-supervised adaptation based on $\mathcal{L}_s$ can result in arbitrarily poor accuracy. We now introduce an online feature alignment strategy to promote more robust adaptation at test-time. At the core of our solution is a compact feature summarization that can be computed offline without revisiting the source data or consuming large storage. Unlike sampling from training and test sets concurrently, which popular MMD-based [4] or adversarial-based [30] methods are dependent upon, we turn to standard divergence based on statistics that can be estimated independently for each distribution. Specifically, we draw inspirations from DDC [34] and Coral[28], and use the square distance of the first and second moments between the training and test-time feature distributions to approximate the domain discrepancy.

Given this discrepancy measure, one simple way to perform online feature alignment is to minimize it for the feature activations produced by the encoder. However, this choice may suffer from two shortcomings: (i) the dimension of this layer can be quite high, *e.g.*, 2048 for the standard ResNet-50; (ii) the low-order statistics may not be sufficient to characterize complex distributions. To alleviate these issues, we propose to align the distributions at two layers: (i) the output of the encoder; and (ii) the output of the self-supervised head, of lower dimension.

Concretely, once training completes, we summarize the whole feature set encoded from the training data. Regarding the features $Z = \{z_1^T, \dots, z_N^T\}$ from the encoder, we compute the empirical mean $\mu_z = \frac{1}{N}\sum_i^N z_i$ and the empirical covariance matrix $\Sigma_Z = \frac{1}{N-1}(Z^T Z - (I^T Z)^T (I^T Z))$. The former one is essentially the same as channel-wise batch normalization statistics; the latter one is more informative yet remains light-weight to be saved as part of the model. The summarization of the features at the self-supervised head, denoted by S, follows the same procedure. At test-time, we introduce an additional online feature alignment loss function $\mathcal{L}_{f,z}$. Its goal is to minimize the distance between the feature statistics estimated from a mini-batch of test samples, i.e., μ'_z and Σ'_z, and that from the summarization of training features μ_z and Σ_z,

$$\mathcal{L}_{f,z} = \|\mu_z - \mu'_z\|_2^2 + \|\Sigma_z - \Sigma'_z\|_F^2 \,, \tag{3}$$

where $\|\cdot\|_2$ is the Euclidean norm and $\|\cdot\|_F$ is the Frobenius norm. We have a similar process for the features from the SSL head.

Our final objective at test-time is a weighted combination of the self-supervised loss $\mathcal{L}_s$, the feature alignment loss at the encoder $\mathcal{L}_{f,z}$ and the one at the self-supervised head $\mathcal{L}_{f,s}$:

$$\mathcal{L}_{test} = \mathcal{L}_s + \lambda_z \mathcal{L}_{f,z} + \lambda_s \mathcal{L}_{f,s}, \tag{4}$$

where λ_z and λ_s are hyper-parameters controlling the emphasis on feature alignment at different layers. This test-time adaptation procedure takes into account both the discriminative power of the updated representation and the domain discrepancy under distributional shift.

Note that in this work, we only adopt a simple instance of feature summarization to demonstrate the effects of online feature alignment. The method can be naturally extended to incorporate higher-order statistics [29, 35], which are expected to bring additional benefits at a mild cost of model size and adaptation efficiency.

4 When Does Test-Time Training Work Well?

Given feature alignment, we now explore the relationship between the self-supervised task and the main task. We first derive theoretical guarantees of performance in an ideal setup, then introduce a suitable self-supervised learning task into our framework.

4.1 Theoretical Results

We assume training samples and labels are drawn from probability distributions represented by random variables X, Y_m, and Y_s, respectively the inputs, the main task labels, and the SSL task labels. Similarly, we assume the existence of X', Y'_m, and Y'_s from which test samples and labels are drawn.

Assumptions. We assume there exist perfect classifiers π_m and π_s on training data, i.e., $\mathbb{P}(\pi_m(g(X)) = Y_m) = \mathbb{P}(\pi_s(g(X)) = Y_s) = 1$. Moreover, we assume that the priors of the labels and their joint distributions do not change between training and test-time, i.e.,

$$Y_m \sim Y'_m, \quad Y_s \sim Y'_s, \quad (Y_m, Y_s) \sim (Y'_m, Y'_s). \tag{5}$$

Following the TTT procedure [36], the head classifiers are kept fixed at test-time, and only the shared encoder g is updated to g' such that the self-supervised head π_s fits the test data. We call $Z' = g'(X')$ the feature distribution of the test data. Thus, $\mathbb{P}(\pi_m(Z') = Y'_m)$ is the accuracy of the model on the main task at test time. When we consider conditional distributions we always imply conditioned on the self-supervised labels as those are available at both train and test times.

Theorem 1. *If we assume that:*

- $Z \sim Z'$ *(feature distributions are aligned);*
- $Z \mid Y_s \sim Z' \mid Y'_s$ *(conditional distributions of the features are aligned);*
- $Y'_m \mid Z', Y'_s \sim Y'_m \mid Y'_s$ *(encoder captures information about the SSL task only);*

then the accuracy of the model is:

$$\mathbb{P}(\pi_m(Z') = Y'_m) = \sum_{y_s} \left[\mathbb{P}(Y_s = y_s) \sum_{y_m} \mathbb{P}(Y_m = y_m \mid Y_s = y_s)^2 \right]. \tag{6}$$

Proof. In the supplementary material, Section B.1. □

According to Theorem 1, the performance of the main classifier π_m at test-time depends on its amount of shared information with the SSL task. The relationship involves squared probabilities, therefore the performance drops quickly as the level of shared information decreases. The assumption $Y'_m \mid Z', Y'_s \sim Y'_m \mid Y'_s$ corresponds to the case when the encoder is minimal on top of being optimal, i.e., it only captures the information relative to the SSL task. When this assumption does not hold, the worst case scenario is described by the following theorem.

Theorem 2. *If we assume that:*

- *$Z \sim Z'$ (feature distributions are aligned);*
- *$Z \mid Y_s \sim Z' \mid Y'_s$ (conditional distributions of the features are aligned);*

then the accuracy of the main task classifier is lower-bounded:

$$\mathbb{P}(\pi_m(Z') = Y'_m) \geq \sum_{y_s} \mathbb{P}(Y_s = y_s) \max\left\{0, 2\left(\max_{y_m} \mathbb{P}(Y_m = y_m \mid Y_s = y_s) - \frac{1}{2}\right)\right\}. \quad (7)$$

Proof. In the supplementary material, Section B.2. □

Theorem 2 gives a lower bound on the performance that offers guarantees independently of the training process at test-time. Moreover, it provides insights on what kind of SSL task would be beneficial for TTT. Indeed, knowing the proportion of Y_m given Y_s guarantees a minimum accuracy at test time. Thus, a strong SSL task is one for which such measure is high. This would lead to a greater lower bound, favoring better generalization.

4.2 Contrastive Learning at Test-Time

Motivated by Theorem 1 and Theorem 2, we now describe an improved version of test-time training, named TTT++, that incorporates both the online feature alignment described in Section 3.2 and a strong self-supervised learning for the adaptation of image recognition models.

One way to quantify the relation between the main task and self-supervised task is to measure the transferability of the representation learned from one to another [37]. In fact, recent works [13, 14, 38, 39] have shown that the standard ResNet-50 pre-trained by contrastive method can achieve remarkable accuracy on the downstream classification problems, not only surpassing other self-supervised approaches including image rotation prediction but also matching the performance of their supervised counterparts. This stunning transfer learning result suggests a high correlation between the visual contrastive learning task and the image classification task.

We therefore replace the rotation prediction task with SimCLR [13] in our framework. Specifically, given a mini-batch of B images, we first augment them to $2B$ data points. For each augmented image, we consider its counterpart augmented from the same source instance as a positive sample and treat the other $2(B-1)$ augmented samples as negative ones. The representation of each image $z_i = g(x_i)$ is transformed by a non-linear self-supervised head (a MLP projection head) to a lower dimensional space $h_i = \pi_s(z_i)$. We encourage the projected hidden embeddings from a positive pair (h_i, h_j) to be closer in the embedding space while pushing apart the negative ones that are semantically different through the contrastive loss

$$\mathcal{L}_s = -\log \frac{\exp(\text{sim}(h_i, h_j)/\tau)}{\sum_{k=1}^{2B} \mathbb{1}_{k \neq i} \exp(\text{sim}(h_i, h_k)/\tau)}, \quad (8)$$

where τ is a temperature scaling parameter. The distance between feature vectors is measured by cosine similarity $\text{sim}(u, v) = u^T v/(\|u\|\|v\|)$.

For data augmentation, we use random crop and resize (with random flip), color distortions, and Gaussian blur (please refer to Appendix for details). We use the ResNet-50 as the backbone and a 2-layer MLP with output dimension of 128 as the self-supervised projection head. As introduced in Sec 3.2, the lower dimensional output from the self-supervised task allows us to estimate the discrepancy in the z space with limited batch size efficiently.

5 Experiments

To validate the versatility of our proposed method, we evaluate TTT++ in diverse TTA scenarios, including several popular visual benchmarks.

Datasets. We verify the efficacy of our proposed method on image classification datasets. On a small scale, we perform our experiments on CIFAR10/CIFAR100 [40], with 10/100 classes, a training set of 50,000 examples, and a test set of 10,000. For evaluating robustness against corruption, we

utilize the CIFAR10-C/CIFAR100-C [1] datasets which comprise corrupted versions of the 10,000 images in the test set. They offer five severities per corruption type, for a total of 15 test corruptions. We also test our model on CIFAR10.1 [15], containing roughly 2,000 new test images, designed to minimize distribution shift relative to the original dataset. CIFAR10.1 has been shown to present a large drop in accuracy (4% to 10%) for a wide range of deep learning models.

For large-scale experiments, we choose the VisDA-C dataset [16], a challenging benchmark that involves synthetic-to-real object classification task. With 12 overall classes, the training set (source domain) comprises 152k synthetic images, while the testing set (target domain) has 55k real images sampled from Microsoft COCO.

We consider the following baselines:

- Test: The trained network evaluated on the test data without any test-time adaptation;
- Test-time normalization (BN) updates batch normalization [41] statistics of the trained network according to the target data during testing [42];
- Test entropy minimization (Tent) [43] updates batch normalization statistics of the trained network by minimizing the entropy of the model predictions on target data during testing;
- Source Hypothesis Transfer (SHOT) [9] framework freezes the classifier module and updates only the feature extraction module by exploiting the concepts of information maximization and self-supervised pseudo-labeling during testing;
- Test-time training (TTT-R) [36] trains the network jointly on the main task and a rotation-based self-supervised task in the source domain; during test time, TTT continues to train on the rotation-based task in the target domain.

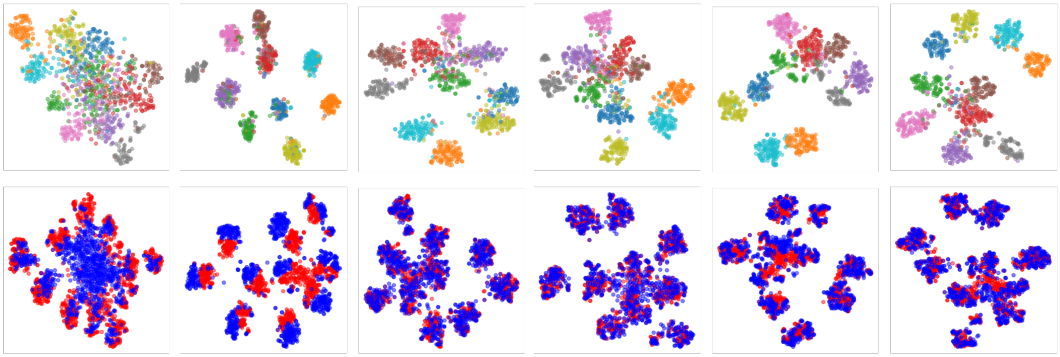
We evaluate the full version as well as the ablation versions of our method:

- Test-Time Feature Alignment (TFA) aligns the first-order and second-order statistics of source and target distributions during testing (Section 3.2);
- Test-Time Contrastive Learning (TTT-C) trains the network jointly on the main task and a contrastive learning self-supervised task in the source domain; during test time, TTT continues to train on the contrastive learning in the target domain (Section 4.2);
- Test-Time Training++ (TTT++) trains the network jointly on the main task and a contrastive learning self-supervised task in the source domain; during test time, TTT++ continues to train on the contrastive learning in the target domain along with performing feature alignment.

Implementation Details. We use ResNet-50 [44] as the backbone architecture throughout our experiments. We follow the Pytorch official tutorial implementation to preprocess the Cifar-10 and Cifar-100 series of datasets. For the VisDA dataset, we resize all images to 256 × 256 px and take the center 224 × 224 px crop. We re-scale the color values with $\mu = [0.485, 0.456, 0.406]$ and $\sigma = [0.229, 0.224, 0.225]$. Our implementation is based on PyTorch [45]. Our experiments were conducted on 2 V100 GPUs. Training models for 1000 epochs jointly on the classification and contrastive tasks took 20 hours for the Cifar-10 and Cifar-100 datasets. On VisDA dataset, we finetune the Pytorch pre-trained ResNet for 50 epochs. All test-time adaptation algorithms took less than 1 hour. For Tent and SHOT baselines, we follow their official code implementation for our experiments. Please refer to the supplementary materials for more details about hyper-parameters.

5.1 Results on Image Corruption

To highlight the efficacy of our proposed strategy TTT++ against test-time image corruption, we compare it against competitive baselines on the standard CIFAR10-C and CIFAR100-C corruption benchmark. Fig 3 illustrates quantitative comparison against state-of-the-art methods Tent and SHOT on different corruptions of highest severity on CIFAR10. The average results on CIFAR10-C and CIFAR100-C are reported in Table 1. It is apparent that TTT++ provides a $\sim 23\%$ relative improvement over the state-of-the-art baseline, Tent on CIFAR10-C, while remaining competitive against Tent on CIFAR100-C. It is worth noting that having a strong self-supervision task only (TTT-C) suffices to beat Tent on CIFAR10-C. Adding test-time feature alignment (TFA) on top of strong self-supervision results in an additional $\sim 10\%$ relative improvement.



(a) Test **(b)** Tent **(c)** SHOT **(d)** TFA **(e)** TTT-C **(f)** TTT++

Figure 4: Qualitative illustration of the encoder feature distribution obtained by performing different test-time adaptation strategies on a fixed trained model. We show the t-SNE plot of the feature distribution in the use-case of 'fog' corruption on CIFAR10-C. The top row indicates the per-class feature distribution after performing the corresponding test-time adaptation strategy. The bottom row highlights the overall feature alignment between the original test set (in red) and corrupted test set post adaptation (in blue).

Qualitative Analysis of Test-time Adaptation Strategies. We qualitatively demonstrate on CIFAR10 *fog* corruption, the improvement in feature distributions obtained by TTT++ at test-time. Figure 4 demonstrates the t-SNE [46] plot for the feature distributions output by the shared encoder $g(\cdot)$ for the original testing set (red) and corrupted testing set (blue) under various strategies. We can clearly observe the shift in the feature distribution on corruption of the test set (Fig 4a). Tent minimizes the entropy of the features but fails to perform feature alignment (bottom row). SHOT and TTT++ both improve with respect to aligning the feature distribution of original test data and corrupted test data. However, thanks to test-time feature alignment module of TTT++, our method displays an improved per-class separation (top row) in comparison to SHOT.

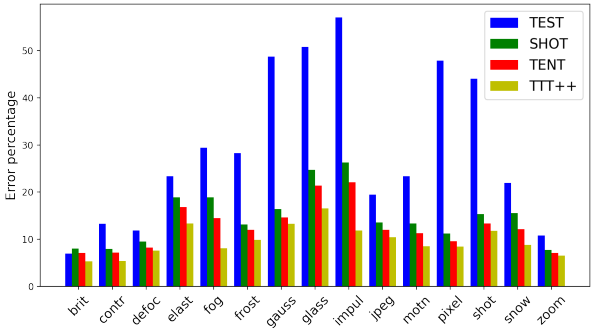


Figure 3: Classification error on CIFAR10-C.

Table 1: Average classification error (%) on CIFAR10-C, CIFAR100-C and CIFAR10.1.

Method	C10-C	C100-C	C10.1
Test	29.1	61.2	12.1
BN [41]	15.7	43.3	14.1
TTT-R [36]	14.3	40.4	11.0
SHOT [9]	14.7	38.1	11.1
TENT [43]	12.6	36.3	13.4
TFA (Ours)	12.4	42.5	12.1
TTT-C (Ours)	10.7	36.9	9.7
TTT++ (Ours)	**9.8**	**36.1**	**9.5**

5.2 Results on Natural Domain Shift

We now demonstrate the efficacy of our method TTT++ to tackle natural distribution shifts. Adapting batch normalization statistics according to the available test data samples relies on the implicit assumption that different samples and spatial locations are shifted in similar manner. However, this assumption is not always true in practise [24] and CIFAR10.1 presents one such scenario. The results of various test-time strategies on CIFAR10.1 is provided in Table 1. As expected, batch normalization based methods like Tent perform poorly and even end up degrading model accuracy. In contrast, TTT++ is able to tackle these natural distribution shifts and improves accuracy by 2.5%.

5.3 Results on Large-scale VisDA-C benchmark

In this experiment, we show that TTT++ can scale to large-scale benchmarks. We choose the VisDA-C benchmark as it has been designed to explicitly tackle the problem of domain adaptation: the training dataset consists of synthetic images generated by rendering 3D models while testing is performed on

Table 2: Classification error rates (%) on large-scale VisDA-C dataset

Method	plane	bcycl	bus	car	horse	knife	mcycl	person	plant	sktbrd	train	truck	Per-class
Test	56.52	88.71	62.77	**30.56**	81.88	99.03	17.53	95.85	51.66	77.86	20.44	99.51	58.72
BN [41]	44.38	56.98	33.24	55.28	37.45	66.60	16.55	59.02	43.55	60.72	31.07	82.98	48.12
TENT [43]	13.43	77.98	**20.17**	48.15	21.72	82.45	12.37	35.78	21.06	76.41	34.11	98.93	42.73
SHOT [9]	5.73	**13.64**	23.33	42.69	7.93	86.99	19.17	**19.97**	11.63	11.09	15.06	**43.26**	25.04
TFA (Ours)	28.25	32.03	33.67	64.77	20.49	**56.63**	22.52	36.30	24.84	35.20	25.31	64.24	39.58
TTT-C (Ours)	5.46	32.23	25.42	37.03	7.84	85.20	9.14	23.80	11.72	11.00	7.74	56.87	25.72
TTT++ (Ours)	**4.13**	26.20	21.60	31.70	**7.43**	83.30	**7.83**	21.10	**7.03**	**7.73**	**6.91**	51.40	**22.46**

real-world images. Indeed, TTT++ outperforms all the baseline methods significantly on VisDA-C. We beat the state-of-the-art method on VisDA-C, SHOT by reducing the error by more than 2.5% in average. Furthermore, we can see that test-time feature alignment plays a crucial role in this synthetic-to-real domain adaptation setup as it provides a performance boost of $\sim 15\%$.

6 Discussions

6.1 Limitations

For online feature alignment in Sec 3.2, we restrain the feature summarization to the second-order statistics. This choice is subject to two limitations in the case of high-dimensional feature space: (i) the low-order statistics may not be insufficient to characterize the complex distribution of features; (ii) the estimation of empirical covariance can be noisy due to the relatively limited batch size. Developing advanced feature summarization methods tailored for test-time adaptation is an interesting avenue for future work. Our theoretical analysis in Sec 4.1 gives insight regarding the performance of main task post test-time training conditioned on the self-supervision objective. However, there can exist a gap between the theoretical and the practical estimates when the stated assumptions do not hold. For instance, the classifiers are rarely perfect in practice.

6.2 Open Questions

In our experiments, we only consider the standard ResNet-50 as the backbone architecture and share the whole feature extractor between the main task and self-supervised task. Yet, recent literature [34, 47] has shown that different layers capture different level of semantic granularity. Determining the best architectural design for our TTT++ framework remains an open question. To demonstrate the efficacy of our method, we follow the common evaluation protocol [10] and conduct an empirical objective comparison with other TTA algorithms. These algorithms exploit different supervisory signals from unlabeled data, which can be not only mutually exclusive but also complementary to each other. Whether these techniques can be blended into a unified framework in a principled manner is another interesting direction to explore in the future.

6.3 Societal Impact

Our improved test-time training strategy aims at expanding the current horizon of machine learning algorithms for test-time adaptation. For applications where humans' lives are at risk, such as autonomous driving, trust, safety, robustness are all mandatory keywords. The field has made amazing progresses when distribution shift between the training and testing scenarios is not significant. What if a deployed machine encounters such shift in safety-critical scenarios? We, humans, have an innate capability for handling distributional shifts. We believe that machines should have the same capability to be used in a safe, trustworthy, and robust manner alongside humans. Indeed, there is a long way to go. Nevertheless, we hope that our work will pave the way to foster more research in analyzing the role of feature alignment and self-supervision in test-time adaptation scenarios.

References

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

[2] Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Sara Beery, Jure Leskovec, Anshul Kundaje, Emma Pierson, Sergey Levine, Chelsea Finn, and Percy Liang. WILDS: A Benchmark of in-the-Wild Distribution Shifts. *arXiv:2012.07421 [cs]*, December 2020. arXiv: 2012.07421.

[3] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant Risk Minimization. *arXiv:1907.02893 [cs, stat]*, September 2019. arXiv: 1907.02893.

[4] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning Transferable Features with Deep Adaptation Networks. In *International Conference on Machine Learning*, pages 97–105. PMLR, June 2015. ISSN: 1938-7228.

[5] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I. Jordan. Deep Transfer Learning with Joint Adaptation Networks. *arXiv:1605.06636 [cs, stat]*, August 2017. arXiv: 1605.06636.

[6] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, January 2016.

[7] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial Discriminative Domain Adaptation. pages 7167–7176, 2017.

[8] 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. *arXiv:1909.13231 [cs, stat]*, July 2020. arXiv: 1909.13231.

[9] Jian Liang, Dapeng Hu, and Jiashi Feng. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *International Conference on Machine Learning (ICML)*, pages 6028–6039, July 13–18 2020.

[10] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully Test-Time Adaptation by Entropy Minimization. September 2020.

[11] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality Reduction by Learning an Invariant Mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2 (CVPR'06)*, volume 2, pages 1735–1742, New York, NY, USA, 2006. IEEE.

[12] Zhirong Wu, Yuanjun Xiong, Stella Yu, and Dahua Lin. Unsupervised Feature Learning via Non-parametric Instance Discrimination. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3733–3742, June 2018. ISSN: 2575-7075.

[13] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A Simple Framework for Contrastive Learning of Visual Representations. *arXiv:2002.05709 [cs, stat]*, March 2020. arXiv: 2002.05709.

[14] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum Contrast for Unsupervised Visual Representation Learning. *arXiv:1911.05722 [cs]*, March 2020. arXiv: 1911.05722.

[15] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do cifar-10 classifiers generalize to cifar-10? 2018. `https://arxiv.org/abs/1806.00451`.

[16] Xingchao Peng, Ben Usman, Neela Kaushik, Judy Hoffman, Dequan Wang, and Kate Saenko. Visda: The visual domain adaptation challenge. *ArXiv*, abs/1710.06924, 2017.

[17] Assaf Shocher, Nadav Cohen, and Michal Irani. Zero-Shot Super-Resolution Using Deep Internal Learning. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3118–3126, June 2018. ISSN: 2575-7075.

[18] David Bau, Hendrik Strobelt, William Peebles, Jonas Wulff, Bolei Zhou, Jun-Yan Zhu, and Antonio Torralba. Semantic photo manipulation with a generative image prior. *ACM Transactions on Graphics*, 38(4):59:1–59:11, July 2019.

[19] Firas Shama, Roey Mechrez, Alon Shoshan, and Lihi Zelnik-Manor. Adversarial Feedback Loop. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3204–3213, October 2019. ISSN: 2380-7504.

[20] Yuejiang Liu, Parth Kothari, and Alexandre Alahi. Collaborative Sampling in Generative Adversarial Networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):4948–4956, April 2020. Number: 04.

[21] Boris Chidlovskii, Stephane Clinchant, and Gabriela Csurka. Domain Adaptation in the Absence of Source Domain Data. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 451–460, San Francisco California USA, August 2016. ACM.

[22] Jogendra Nath Kundu, Naveen Venkat, Rahul M. V, and R. Venkatesh Babu. Universal Source-Free Domain Adaptation. pages 4544–4553, 2020.

[23] Steffen Schneider, Evgenia Rusak, Luisa Eck, Oliver Bringmann, Wieland Brendel, and Matthias Bethge. Improving robustness against common corruptions by covariate shift adaptation. *Advances in Neural Information Processing Systems*, 33:11539–11551, 2020.

[24] Collin Burns and Jacob Steinhardt. Limitations of Post-Hoc Feature Alignment for Robustness. *arXiv:2103.05898 [cs]*, March 2021. arXiv: 2103.05898.

[25] Yang Fu, Sifei Liu, Umar Iqbal, Shalini De Mello, Humphrey Shi, and Jan Kautz. Learning to Track Instances without Video Annotations. *arXiv:2104.00287 [cs]*, April 2021. arXiv: 2104.00287.

[26] Nicklas Hansen, Rishabh Jangir, Yu Sun, Guillem Alenyà, Pieter Abbeel, Alexei A. Efros, Lerrel Pinto, and Xiaolong Wang. Self-Supervised Policy Adaptation during Deployment. September 2020.

[27] Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A Kernel Two-Sample Test. *Journal of Machine Learning Research*, 13(25):723–773, 2012.

[28] Baochen Sun, Jiashi Feng, and Kate Saenko. Correlation Alignment for Unsupervised Domain Adaptation. *arXiv:1612.01939 [cs]*, December 2016. arXiv: 1612.01939.

[29] Werner Zellinger, Thomas Grubinger, Edwin Lughofer, Thomas Natschläger, and Susanne Saminger-Platz. Central Moment Discrepancy (CMD) for Domain-Invariant Representation Learning. *arXiv:1702.08811 [cs, stat]*, May 2019. arXiv: 1702.08811.

[30] Yaroslav Ganin and Victor Lempitsky. Unsupervised Domain Adaptation by Backpropagation. *arXiv:1409.7495 [cs, stat]*, February 2015. arXiv: 1409.7495.

[31] Mehdi Noroozi and Paolo Favaro. Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles. *arXiv:1603.09246 [cs]*, August 2017. arXiv: 1603.09246.

[32] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep Clustering for Unsupervised Learning of Visual Features. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision – ECCV 2018*, Lecture Notes in Computer Science, pages 139–156, Cham, 2018. Springer International Publishing.

[33] Xinlei Chen and Kaiming He. Exploring Simple Siamese Representation Learning. *arXiv:2011.10566 [cs]*, November 2020. arXiv: 2011.10566.

[34] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep Domain Confusion: Maximizing for Domain Invariance. *arXiv:1412.3474 [cs]*, December 2014. arXiv: 1412.3474.

[35] Chao Chen, Zhihang Fu, Zhihong Chen, Sheng Jin, Zhaowei Cheng, Xinyu Jin, and Xian-Sheng Hua. HoMM: Higher-order Moment Matching for Unsupervised Domain Adaptation. *arXiv:1912.11976 [cs]*, December 2019. arXiv: 1912.11976.

[36] Y. Sun, X. Wang, Zhuang Liu, J. Miller, Alexei A. Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *ICML*, 2020.

[37] Amir Zamir, Alexander Sax, William Shen, Leonidas Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling Task Transfer Learning. *arXiv:1804.08328 [cs]*, April 2018. arXiv: 1804.08328.

[38] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised Learning. *arXiv:2006.07733 [cs, stat]*, September 2020. arXiv: 2006.07733.

[39] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised Learning of Visual Features by Contrasting Cluster Assignments. *Advances in Neural Information Processing Systems*, 33:9912–9924, 2020.

[40] A. Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[41] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis R. Bach and David M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 448–456. JMLR.org, 2015.

[42] Steffen Schneider, Evgenia Rusak, Luisa Eck, Oliver Bringmann, Wieland Brendel, and Matthias Bethge. Removing covariate shift improves robustness against common corruptions. *CoRR*, abs/2006.16971, 2020.

[43] Dequan Wang, Evan Shelhamer, Shaoteng Liu, B. Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. 2021.

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

[45] Adam Paszke, S. Gross, Francisco Massa, A. Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Z. Lin, N. Gimelshein, L. Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

[46] L. V. D. Maaten and Geoffrey E. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[47] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *arXiv:1411.1792 [cs]*, November 2014. arXiv: 1411.1792.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]

(b) Did you describe the limitations of your work? [Yes] The limitations of our work are described in Sec 6.1

(c) Did you discuss any potential negative societal impacts of your work? [Yes] The societal impacts of this work is described in Sec 6.3

(d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...

(a) Did you state the full set of assumptions of all theoretical results? [Yes]

(b) Did you include complete proofs of all theoretical results? [Yes] It is provided in the supplementary material.

3. If you ran experiments...

(a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Details are provided in the supplemental material.

(b) Did you specify all the training details (*e.g.*, data splits, hyperparameters, how they were chosen)? [Yes] The implementation details have been provided in Sec 5.

(c) Did you report error bars (*e.g.*, with respect to the random seed after running experiments multiple times)? [Yes] We follow the common evaluation protocol in the test-time adaptation literature. Results of multiple runs are provided in the supplementary material.

(d) Did you include the total amount of compute and the type of resources used (*e.g.*, type of GPUs, internal cluster, or cloud provider)? [Yes] The details regarding resources and compute has been provided in Sec 5.

4. If you are using existing assets (*e.g.*, code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes]

(b) Did you mention the license of the assets? [Yes] We mention the license of other open-sourced code.

(c) Did you include any new assets either in the supplemental material or as a URL? [Yes] The URL is provided in the supplementary material.

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]