TTT++: When Does Self-Supervised Test-Time Training Fail or Thrive?

Yuejiang Liu **Parth Kothari** **Bastien van Delft**

Baptiste Bellot-Gurlet **Taylor Mordan** **Alexandre Alahi**

École Polytechnique Fédérale de Lausanne (EPFL)

{firstname.lastname}@epfl.ch

Abstract

Test-time training (TTT) through self-supervised learning (SSL) is an emerging paradigm to tackle distributional shifts. Despite encouraging results, it remains unclear when this approach thrives or fails. In this work, we first provide an in-depth look at its limitations and show that TTT can possibly deteriorate, instead of improving, the test-time performance in the presence of severe distribution shifts. To address this issue, we introduce a test-time feature alignment strategy utilizing offline feature summarization and online moment matching, which regularizes adaptation without revisiting training data. We further scale this strategy in the online setting through batch-queue decoupling to enable robust moment estimates even with limited batch size. Given aligned marginal distributions of encoded features, we shed light on the strong potential of TTT by theoretically analyzing the performance post adaptation. This analysis motivates our use of more informative self-supervision in the form of contrastive learning. We empirically demonstrate that our modified version of test-time training, termed *TTT++*, outperforms state-of-the-art methods by a significant margin on multiple vision benchmarks. Our result indicates that exploiting extra information stored in a compact form, such as related SSL tasks and feature distribution moments, can be critical to the design of test-time algorithms. Our code is available at `https://github.com/vita-epfl/ttt-plus-plus`.

1 Introduction

Machine learning models often struggle to generalize under distribution shifts. Even a perceptually mild shift between training and test data, *e.g.*, JPEG compression, may cause severe prediction errors [1]. One popular family of methods to address this challenge is to learn an invariant representation across domains by making use of labelled training data and unlabelled test data simultaneously [2–5]. However, revisiting training data at test time can be impractical due to increasing privacy concerns, inflating sizes of datasets as well as many other real-world constraints. This shortcoming prompts a more challenging yet appealing *test-time adaptation* paradigm: given a trained model, how can we effectively adapt it from one domain to another on the fly, without access to training data and human annotations?

One promising approach towards this goal is test-time training (TTT) through self-supervision [6]. The key idea of TTT is simple and straightforward: train the model on two tasks, the main task and a self-supervised learning (SSL) task, and update the model based only on the SSL task at test time. This technique implemented with rotation prediction as the SSL task has shown encouraging results

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

for improving the robustness of image classifiers under a variety of distributional shifts. Yet, its empirical performance is still inferior to other variants of test-time algorithms [7, 8].

In this paper, we first take an in-depth look at TTT with emphasis on its limitations. Our analysis starts with a basic question: can TTT always mitigate the effects of distributional shifts? Through an illustrative problem, we show that the TTT framework can lead to surprising failures, deteriorating the test accuracy rather than improving it. This problem is largely attributed to unconstrained updates from the SSL task that interfere with the main task. To address this issue, we introduce a test-time feature alignment strategy by means of offline feature summarization and online moment matching: once training completes, we compute the mean and covariance matrix of training features and store them as part of the model, referred to as *offline feature summarization*; at test time, we encourage the test feature distribution to be close to the training one by matching the moments estimated online with those pre-computed offline, a process referred to as *online moment matching*.

One practical challenge for online feature alignment lies in scaling the strategy to problems with a large number of classes, as obtaining a robust estimate of moments often requires at least a handful of samples per class. To mitigate this issue, we draw inspiration from recent literature [9] and decouple the sample size from the batch size for moment estimates. Specifically, we maintain a large dynamic queue of encoded features and progressively update it in a mini-batch manner. This modification enables effective feature alignment even with limited batch size, greatly improving its viability in the online test-time setting.

Finally, we shed light on the strong potential of TTT through a theoretical analysis of the test accuracy after adaptation. In particular, we derive a lower bound of the test accuracy on the main task and show that it is expected to grow rapidly when the SSL task gets closer to the main task. These findings motivate our integration of contrastive representation learning [9–12], as a strong instance of SSL, into the TTT framework.

By combining the three proposed components, we devise an improved version of test-time training, termed *TTT++*. Experimental results show that TTT++ significantly outperforms other recent methods by significant margins on various robustness benchmarks. Our results suggest that exploiting extra information, including both task-specific information in the form of strong self-supervision and model-specific information in the form of feature summarization, can be a promising direction to enhance the effectiveness of test-time adaptation.

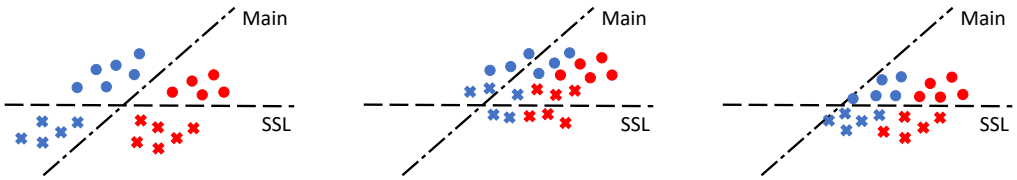
2 Background

2.1 Related Work

Test-time Adaptation. Adapting machine learning models based on test samples has garnered growing interests in both generative problems such as super-resolution [13], image synthesis [14, 15] and image manipulation [16], and discriminative problems like image classification [17]. Our work is focused on the latter one in the presence of distributional shifts. Several recent methods [17, 18] have shown the potential of adapting the learned models to a new domain at test time without access to the training data. One simple yet effective approach is to replace the batch-norm statistics estimated on the training set with the test examples [18]. Another line of work proposed to adapt the model parameters by exploiting the outputs on test samples, such as entropy minimization [8] or pseudo-labeling [7]. While these methods yield promising results on some benchmarks, they are largely restricted to classification problems and vulnerable under large distribution shifts [19].

More closely related to ours, [6] proposed test-time training through self-supervised learning, *e.g.*, predicting image rotation. This approach does not involve any assumptions over the output of the main task and is hence more generic. It has been successfully applied to a variety of problems, such as instance tracking [20] and reinforcement learning [21]. Nevertheless, it was shown empirically inferior to other families of test-time algorithms [8]. Our work provides an in-depth analysis of its limitations and introduces simple yet effective remedies with more theoretical grounds.

Feature Alignment. Matching the distributions of feature activations between the training and test samples is commonly used for domain adaptation. Previous works often promote feature alignment in two ways: either explicitly minimizing a divergence measure, such as MMD [22], Coral [23] and CMD [24], or employing an adversarial loss function that encourages domain confusion [5, 25].



(a) training features (b) test features before TTT (c) test features after TTT

Figure 1: Illustration of a failure case where test-time training (TTT) hurts generalization. (a) The predictive model reaches high accuracy on both the main classification task, *i.e.*, separating red and blue data points, and the auxiliary self-supervised task, *i.e.*, separating circles and crosses, in the training domain. (b) Under significant distributional shift, test samples are encoded into a new subspace, resulting in limited accuracy on both tasks. (c) Without any constraints over the feature distribution, TTT may yield an updated encoder that overfits the self-supervised task, which deteriorates, instead of improving, the performance on the main task.

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 setting where source data is not available. Our work revisits the critical role of feature alignment for test-time training and proposes a simple and practical strategy that enables online feature alignment even with a limited batch size.

Self-Supervised Learning. Self-supervised learning is a powerful paradigm to learn rich visual representations from unlabeled samples. Stunning progress has been made in recent years by designing informative self-supervised tasks [11, 12, 26–29] and stabilizing the training process [9, 30]. While previous works focused on unsupervised pre-training tasks, our work sheds light on the importance of incorporating strong self-supervised learning for test-time training.

2.2 Preliminary: Test-Time Training

Test-time training (TTT) [6] is a general framework for adapting neural network models to a new test distribution based on unlabeled samples. Different from the conventional approach that trains the model only on the task of interest, TTT considers two tasks: a main task and an auxiliary SSL task. The model is trained on both tasks simultaneously with a multi-task architecture composed of one shared encoder g and two separate heads π_m and π_s respectively for each task. Given a labeled training dataset $D = \{(x_i, y_i)\}_{i\in\{1,\dots,N\}}$, the model is trained to jointly optimize two losses:

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

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

At test time, in the presence of a distributional shift, the learned model often struggles to directly generalize to test samples $D' = \{x'_i\}_{i\in\{1,\dots,N'\}}$. The core idea of TTT is to fine-tune the encoder g based on the self-supervised task, with the hope that the updated model $\pi_m \circ g'$ achieves improved results on the main task:

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

TTT instantiated with rotation prediction as the SSL task consistently improves the robustness of image classifiers on several benchmarks [6]. However, it was shown less effective than other test-time adaptation methods [7, 8].

3 When Does Test-Time Training Fail?

In this section, we first throw light on a caveat of test-time training under large distributional shifts, and subsequently propose practical solutions tailored for the test-time setting.

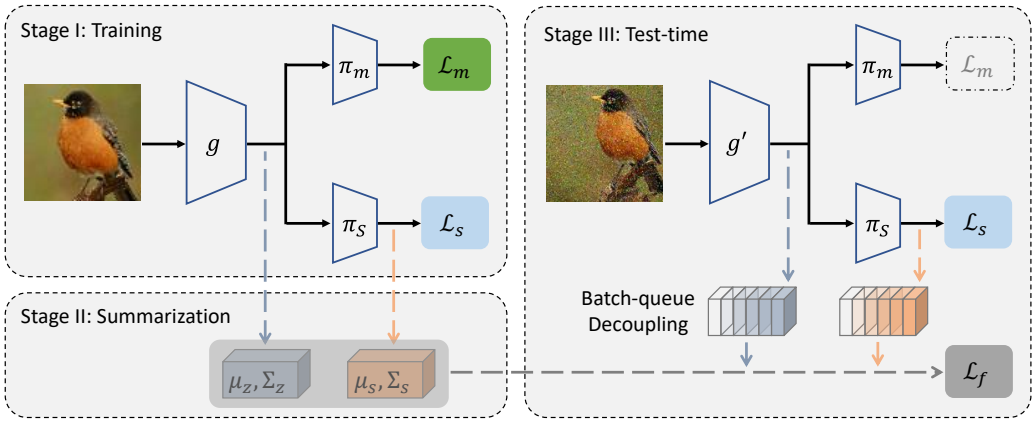


Figure 2: Our modified version of test-time training (TTT++). Our method consists of three stages: model training, offline feature summarization, and online test-time adaptation. (i) During training, the model is jointly optimized for the main task and contrastive self-supervised task. (ii) Once training completes, we summarize the feature distributions after the encoder and the self-supervised head in the form of first and second-order moments. (iii) At test time, we adapt the encoder through self-supervised learning and online feature alignment with a large dynamic queue for robust moment estimates.

3.1 Illustrative Example of Failures

We first introduce a simple toy problem to illustrate the limitation of TTT. Consider the low-dimensional problem shown in Figure 1, where the main task and the SSL task are defined as classifying colors and symbols of encoded features in the 2-dimensional latent space. Thanks to the strong correlation between these two tasks, the model can attain high accuracy on both of them in the training domain. However, at test time, the encoded features undergo a significant distributional shift, resulting in substantial prediction errors on both tasks.

While the objective of test-time training is to restore the discriminative power of the learned representation for the main task, updating the encoder solely based on the self-supervised task may yield severe overfitting to the auxiliary task. As a consequence, the performance on the main task can severely deteriorate as opposed to improving. This phenomenon is not restricted to this illustration and also occurs in practice, as evidenced in our simulation in Section 5.1.

3.2 Online Feature Alignment

As illustrated in the toy example above, simply applying self-supervised adaptation at test time can lead to arbitrarily poor results. To address this issue, we introduce an online feature alignment strategy to promote robust adaptation at test time. The core idea of our strategy is to impose a constraint over the feature space during TTT such that the feature distribution of test examples remains close to that in the training domain. While feature alignment techniques based on MMD [2] or adversarial training [25] have been widely used for domain adaptation, they often rely on sampling from training and test domains concurrently, which is impractical in the test-time setting. We, therefore, turn to classical divergence measures that can be estimated independently for each distribution. More specifically, we use the square distance of the first and second moments between two feature distributions, inspired by DDC [31] and Coral [23], to approximate the domain discrepancy. This design choice allows us to summarize the distribution of training features in a compact format and store it as part of the model, eliminating the need to revisit the training data during test-time adaptation.

Concretely, once training completes, we perform an offline feature summarization step that characterizes the entire set of feature vectors encoded from the training data $Z = \{z_1^T, \dots, z_N^T\}$, in the forms of empirical mean $\mu_z = \frac{1}{N}\sum_i^N z_i$ and covariance matrix $\Sigma_Z = \frac{1}{N-1}(Z^T Z - (I^T Z)^T (I^T Z))$. The former is essentially equivalent to the channel-wise batch normalization statistics while the latter is more informative while remaining light-weight for efficient storage. At test time, we constrain the self-supervised adaptation by minimizing 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 stored summarization of training

features:

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

where $\|\cdot\|_2$ is the Euclidean norm and $\|\cdot\|_F$ is the Frobenius norm.

One limitation of online moment matching in its naive form is that the low-order statistics may be insufficient to fully capture complex distributions in high dimensions, *e.g.*, 2048 for the standard ResNet-50. To alleviate this issue, we align the feature distributions at both the output of the encoder and the output of the self-supervised head, which are of lower dimensions, *e.g.*, 128 in the case of contrastive projection head in Sec. 4. Thus, 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 $\mathcal{L}_{f,s}$:

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

where λ_z and λ_s are hyper-parameters controlling the emphasis on feature alignment at different layers.

This proposed test-time algorithm takes into account both the discriminative power and the domain discrepancy of the updated representations and hence enables the model to adapt more robustly under various settings. In fact, the current moment matching strategy is just one instance of the online feature alignment scheme. It can be naturally extended to incorporate higher-order statistics [24, 32], to bring further performance gain at the cost of model size and computational efficiency.

3.3 Online Dynamic Queue

One practical challenge for online feature alignment lies in scaling the strategy to problems having large numbers of classes. Intuitively, a good estimate of moments of the entire distribution needs at least a handful of samples per class. As a consequence, the demand for sample size grows linearly with the number of classes, for instance, over $\sim$1000 samples are required in the case of CIFAR-100. However, the computational resources during deployment are often limited to accommodate such a large batch size in the test-time setting.

To overcome this challenge, we draw inspiration from recent literature [9] and maintain a dynamic queue of encoded features to decouple the batch size at test time from the sample size for moment estimates. The dynamic queue contains a few batches of feature vectors encoded at test time. We progressively update it by appending the latest mini-batch and popping out the oldest one, as illustrated in Figure 2. This decoupling allows us to collect a large and consistent pool of samples for online moment matching, even with a very limited batch size.

4 When Does Test-Time Training Thrive?

Given properly aligned feature distribution, we next look into the potential of test-time training given strong SSL tasks. We first derive a lower bound of the test accuracy in general scenarios and then analyze it in a specific setup where the performance on the main task can be directly estimated. These analyses motivate our use of more informative self-supervised learning for test-time training.

4.1 Theoretical Results

We consider a training set comprised of samples drawn from the joint distribution $\mathbb{P}_{X,Y_m,Y_s}$, where X, Y_m and Y_s are random variables corresponding to the training samples, the main task labels and the self-supervised labels respectively. Similarly, the test set consists of samples drawn from the joint distribution $\mathbb{P}_{X',Y_m',Y_s'}$. In the presence of distribution shift, $\mathbb{P}_{X,Y_m,Y_s}$ and $\mathbb{P}_{X',Y_m',Y_s'}$ are not identical. However, we make the following assumption about label distribution in our analysis.

Assumption 1. *The training and test labels are equal in distribution,* $Y_m \overset{d}{=} Y_m'$, $Y_s \overset{d}{=} Y_s'$ *and* $(Y_m, Y_s) \overset{d}{=} (Y_m', Y_s')$.

In addition, we restrict our analysis to the scenarios where both two tasks can be solved perfectly during training.

Assumption 2. *There exist an encoder* g *and classifiers* π_m *and* π_s *such that* $\mathbb{P}(\pi_m(g(X)) = Y_m) = 1$ *and* $\mathbb{P}(\pi_s(g(X)) = Y_s) = 1$.

During TTT, the shared encoder g is updated to g' such that the self-supervised head π_s fits the test data. Given our proposed online feature alignment in Equation 4, we assume the encoded features in the training and test domains, *i.e.*, Z and Z', have the following property.

Assumption 3. *The marginal feature distribution and the conditional feature distributions at test time are aligned with their counterparts during training, that is,* $Z \overset{d}{=} Z'$ *and* $(Z \mid Y_s = k) \overset{d}{=} (Z' \mid Y'_s = k)$ *for all classes* k *in the SSL task.*

Under these assumptions, the test accuracy on the main task is lower bounded as follows.

Theorem 1. *If assumptions 1-3 hold, then the prediction accuracy on the main task is lower bounded:*

$$\mathbb{P}(\pi_m(Z') = Y'_m) \geq \sum_{y_s} \mathbb{P}(Y_s = y_s) \max\left\{0, 2\left(\max_{y_m} \mathbb{P}(Y_m = y_m \mid Y_s = y_s) - 0.5\right)\right\}. \quad (5)$$

Proof. Please refer to Section A.1 in the supplementary material. □

Theorem 1 highlights the importance of the relation between the main task and the SSL task for the test accuracy after adaptation. In fact, it can be difficult to ensure that there always exists a class in the main task for which $\mathbb{P}(Y_m = y_m \mid Y_s = y_s)$ is sufficiently large in the worst-case scenario.

To further understand the impact of the task relation on test-time training, we next consider a particular setting, where the encoded features fully overfit to the SSL task (*e.g.*, lengthy test-time training) and become independent of the main task label given the SSL label. Under this condition, the prediction accuracy of the main task can be directly estimated as follows.

Theorem 2. *If assumptions 1-3 hold and* $Z' \perp\!\!\!\perp Y'_m \mid Y'_s$*, then the prediction accuracy on the main task is given by*

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

Proof. Please refer to Section A.2 in the supplementary material. □

Intuitively, if the encoded features do not contain more information than the SSL labels about the main task, the accuracy of the main classifier π_m only depends on the property of the SSL task. In particular, the square term on the right-hand side of Equation 6 emphasizes the paramount importance of designing a well-suited SSL task for the prediction accuracy on the main task. When the two tasks diverge, the test accuracy drops quadratically fast, leading to ineffective adaptation.

4.2 Test-Time Training through Contrastive Learning

Our theoretical analysis above reveals the importance of incorporating an SSL task highly correlated with the main task for test-time training. One practical way to quantify the relation between two tasks is to measure the transferability of the representation learned from one task to another [33]. Given the remarkable results of contrastive methods for pre-training visual representations [9, 12, 34, 35], we hypothesize that they would also be suitable choices for test-time training. Thus, we replace the rotation prediction task with SimCLR [12]. For a mini-batch of B images, we augment each image to two views. We consider the two augmented views from the same original instance as a positive pair and treat the other pairs as negative ones. The feature vector $z_i = g(x_i)$ of each image x_i is projected to a lower-dimensional space $h_i = \pi_s(z_i)$ through our self-supervised head. 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:

$$\mathcal{L}_s = -\log \frac{\exp(\text{sim}(h_i, h_j)/\tau)}{\sum_{k=1}^{2B} \mathbb{1}_{k \neq i} \exp(\text{sim}(h_i, h_k)/\tau)}, \quad (7)$$

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

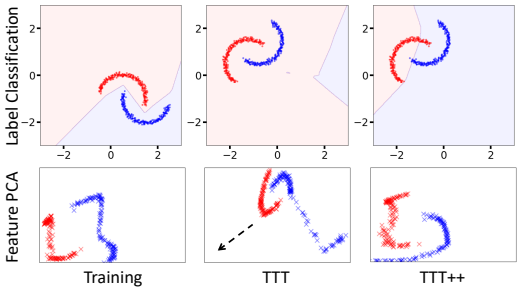


Figure 3: Qualitative comparison of TTT [6] and our TTT++ on the inter-twinning moons problem. Our method effectively adapts the decision boundary to the test domain, whereas the vanilla TTT suffers from a severe feature misalignment, marked by the dashed arrow in the PCA visualization.

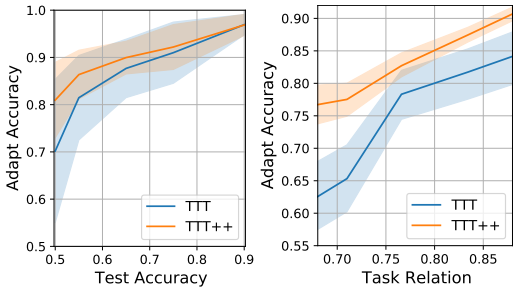


Figure 4: Quantitative comparison of TTT [6] and our TTT++ on the inter-twinning moons problem under various setups. Our method is particularly advantageous under large shifts (low test accuracy before adaptation) and substantially benefits from SSL tasks closely related to the main one (higher label agreement).

5 Experiments

We validate our proposed method in four scenarios: synthetic toy problem, common image corruptions, natural domain shifts, and sim-to-real transfer.

We consider the following baselines throughout our experiments:

- **Test** - the trained network is evaluated on the test data without any test-time adaptation;
- Test-time normalization (**BN**) [36] - we update the batch normalization statistics of the trained network according to the target data during testing;
- Test entropy minimization (**TENT**) [8] - we update the batch normalization statistics of the trained network by minimizing the entropy of the model predictions on target data during testing;
- Source Hypothesis Transfer (**SHOT**) [37] - we freeze the classifier module and update only the feature extraction module by exploiting the concepts of information maximization and self-supervised pseudo-labeling during testing;
- Test-time training (**TTT-R**) [6] - we train the network jointly on the main task and a rotation-based SSL task in the source domain; during test time, TTT-R continues to train on the rotation-based task in the target domain.

We also evaluate the following ablated 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 (SSL) task in the source domain; during test time, TTT-C continues to update the encoder based on contrastive learning in the target domain (Section 4.2).

Our proposed improved 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.

5.1 Synthetic Toy Problem

We first evaluate our method on the inter-twinning moons problem [4, 38], where the main task is to predict the moon class of a given data point and the SSL task is to predict on which side of the hyperplane (*i.e.*, linear separator between the two moons) the data point lies on. The relation (label agreement) between these two tasks depends on the separation distance between the moons. To solve both tasks simultaneously, we build a small neural network that consists of a 2-layer MLP as the shared encoder and two 2-layer MLPs as separate task heads. Each hidden layer contains 8 neurons. The learned model attains over 99% accuracy on both the main and SSL tasks in the training domain. We simulate a variety of distributional shifts through translation and rotation of all data points.

Figure 3 shows the decision boundaries and encoded features from the vanilla TTT and our proposed TTT++ in a particular test case of large distributional shift. TTT fails to improve the classification

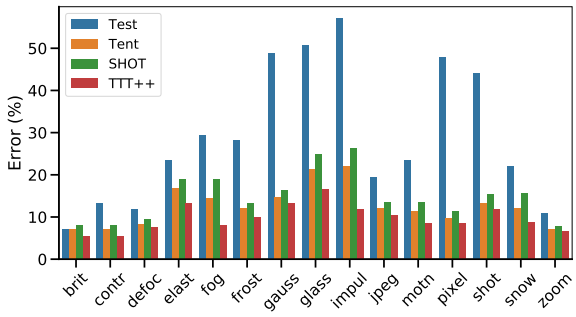


Figure 5: Classification error (%) on CIFAR10-C [1].

Table 1: Average classification error (%) on CIFAR10-C/100-C [1] and CIFAR10.1 [41]

Method	C10-C	C100-C	C10.1
Test	29.1	61.2	12.1
BN [42]	15.7	43.3	14.1
TTT-R [6]	14.3	40.4	11.0
SHOT [37]	14.7	38.1	11.1
TENT [8]	12.6	36.3	13.4
TFA (Ours)	11.9	35.8	12.1
TTT-C (Ours)	10.7	36.9	9.7
TTT++ (Ours)	**9.8**	**34.1**	**9.5**

accuracy. In fact, the resulting decision boundary goes even further from a desired one due to the severe distribution mismatch, which we can observe in the PCA visualization of encoded features. In comparison, our method TTT++ yields substantial performance gain, boosting the test accuracy from 50% to 93%, thanks to the reduced feature distributional shift enforced by the proposed online moment matching.

Figure 4 summarizes the quantitative results of our methods as well as the vanilla counterpart under 150 different simulated setups. As shown on the left, when the *domain shift* only causes mild test errors on the main task, both the original TTT and our TTT++ yield strong adaptation results. However, the effectiveness of TTT deteriorates quickly along with the growth of domain shift (reflected on the lower test accuracy). In comparison, our proposed TTT++ demonstrates clear advantages under large shifts, *e.g.*, when the test accuracy before adaptation is around 0.5. In the figure on the right side, we vary the separation distance between the two moons in order to examine the impact of the *task relation* on the prediction accuracy after adaptation. The simulation results confirm the high potential of test-time training given improved SSL tasks, as analyzed in Section 4.1.

5.2 Common Image Corruption

We further assess the robustness of our method against common image corruptions. Following the evaluation protocol of previous work [8], we train ResNet-50 [39] on CIFAR10/CIFAR100 [40] and test it on the CIFAR10-C/CIFAR100-C [1] datasets, which contain 15 types of *algorithmically generated* corruptions, such as noise, blur and snow effects. We use a batch size of 256 for test-time training. In addition, we use a dynamic queue containing 16 batches of feature vectors for online feature alignment on CIFAR100-C.

Figure 5 shows the quantitative results under each type of image corruption. The average results on CIFAR10-C and CIFAR100-C are reported in Table 1. Our TTT++ clearly outperforms the prior state-of-the-art test-time methods on CIFAR10-C and CIFAR100-C. In particular, incorporating a strong self-supervision task (TTT-C) already suffices to perform on par or better than TENT. Adding test-time feature alignment (TFA) on top of that yields an additional $\sim 8\%$ relative reduction in terms of the test error.

5.3 Natural Domain Shift

We next demonstrate the efficacy of our method TTT++ to tackle *natural* distribution shifts. We again use the pre-trained ResNet-50 and test it on CIFAR10.1 [41], a recently collected test set subject to natural distributional shift. Despite its high perceptual similarity with the CIFAR10 dataset, the CIFAR10.1 typically leads to a drop of accuracy (4% to 10%) for a wide range of deep learning models [41].

Table 1 summarizes the results of various test-time algorithms on CIFAR10.1. Previous batch-norm-based methods perform poorly and even degrade model accuracy. This phenomenon is tied to their implicit assumption that different samples and spatial locations are shifted in a similar manner, which is true for algorithmically generated image corruptions but does not hold on CIFAR-10.1 [19]. In contrast, TTT++ is more generic and yields stronger performance under the natural distribution shift.

Table 2: Classification error (%) on the large-scale VisDA-C dataset [43].

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 [42]	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 [8]	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 [37]	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**

Table 3: Classification error (%) results from online feature alignment with or without a dynamic queue of feature vectors on the CIFAR100-C under level-5 fog corruption. Sample size = Batch size $\times$ # Batches. Given a fixed batch size, enlarging the queue size leads to similar results as having a larger batch size.

	w/o queue			w/ queue			
Sample Size	64	128	256	64×2	64×4	64×8	64×16
Test Error	40.31	38.67	37.01	39.84	37.37	36.18	36.02

5.4 Sim-to-Real Transfer

We finally validate the effectiveness of our method on the VisDA-C dataset [43], a challenging large-scale benchmark of synthetic-to-real object classification. As shown in Table 2, prior methods that are fairly competitive under image corruptions, such as BN [42] and TENT [8], are not effective on VisDA-C. We conjecture that this is attributed to their strong restrictions over the adaptable parameters at test time. In contrast, our proposed method is more flexible, allows the model to update the entire encoder, and thus achieves compelling results on VisDA-C. Furthermore, the test-time feature alignment plays a crucial role in this synthetic-to-real domain adaptation problem, providing $\sim 13\%$ performance boost on top of the TTT-C.

5.5 Effect of Batch-Queue Decoupling

To verify the effects of batch-queue decoupling, we compare the results of online feature alignment with different sample sizes. Table 3 summarizes the test errors on CIFAR100 under the level-5 fog corruption. As expected, larger sample sizes generally lead to lower classification errors. Interestingly, while the performance of using a dynamic queue is slightly worse than its counterpart of having the same sample size using a single large batch, enlarging the queue size always yields better results. For instance, given a small batch size of 64, using a dynamic queue maintaining 512 or 1024 feature vectors from 8 or 16 consecutive batches respectively is more advantageous compared to the vanilla moment matching based on a batch size of 256 samples. This result corroborates the benefit of integrating a dynamic queue into our proposed feature alignment framework, for enhancing the scalability in the online setting.

5.6 Design Choice for Moment Matching

As discussed in Section 3, our proposed online feature alignment can be instantiated with different orders of moments and applied at various layers. To understand the effects of these different design choices, we empirically compare our proposed version against several ablated variants in Table 4. Irrespective of the layer choice, our proposed online feature alignment consistently results in reduced classification error. Nevertheless, moment matching applied to the self-supervised head alone leads to lower test error compared to applying it to the feature extractor output in 11 out of 15 types of corruption. The strong performance of the former can be attributed to the lower dimensionality of the feature vector, which allows for a more accurate estimate of statistics given the limited batch size. The best result comes from the online feature alignment at the outputs of both the encoder and the projection head simultaneously, which validates our design choice in Section 3.2.

We further verify the choice of divergence measure through an ablation study. The online feature alignment using the second-order moment (covariance) leads to clearly better results than the one

Table 4: Classification error (%) on CIFAR10-C [1] with different versions of online feature alignment. Taking into account both the first and second-order moments at two different layers is better than the other counterparts under most types of image corruptions.

TFA	brit	contr	defoc	elast	fog	frost	gauss	glass	impul	jpeg	motn	pixel	shot	snow	zoom
w/o $\mathcal{L}_{f,s}$	7.96	7.57	9.4	17.24	14.38	12.54	14.62	21.05	21.4	12.68	11.92	10.7	13.7	12.74	7.32
w/o $\mathcal{L}_{f,z}$	7.85	7.84	9.18	16.51	14.33	11.99	13.79	20.08	20.17	12.42	12.02	10.5	12.78	13.28	7.44
w/o Σ	7.49	7.56	9.62	18.62	19.22	12.72	16.02	25.07	25.17	13.43	13.63	11.22	15.04	15.11	7.77
w/o μ	**7.43**	**7.37**	8.90	15.92	12.98	11.57	13.46	19.27	**18.95**	11.87	11.11	9.97	12.81	11.76	7.04
Full	7.44	7.40	**8.89**	**15.73**	**12.82**	**11.49**	**12.94**	**18.46**	19.13	**11.66**	**10.77**	**9.93**	**12.67**	**11.73**	**7.03**

using the first-order moment (mean). It is also evident that the online feature alignment is most effective when both the mean and the covariance are taken into account.

6 Conclusion and Discussions

In this work, we conduct an in-depth analysis of the limitations and potential of the test-time training through self-supervised learning. We draw attention to the risk of feature distribution mismatch, which is critical but largely overlooked in recent test-time algorithms. We shed light on the strong potential of this approach by analyzing the growth of test accuracy given improved SSL tasks. These analyses inspire three proposed modifications, namely online feature alignment, batch-queue decoupling and contrastive test-time training, which yield state-of-the-art results on multiple robustness benchmarks. Our results suggest the advantages of bringing additional task-specific and model-specific information in a compact format for test-time adaptation. We hope these findings will motivate researchers and practitioners to rethink what should be stored, in addition to weight parameters, for the robust deployment of machine learning models.

Limitations. In this work, we restrain feature summarization to first and second-order moments. Yet, the low-order statistics may be insufficient to characterize the complex distribution of high-dimensional features. Developing more advanced summarization methods tailored for test-time adaptation is an interesting avenue for future work. In addition, there may exist a considerable gap between our theoretical analysis and the empirical results, when the stated assumptions do not hold. For instance, neither the classifier nor the feature alignment is perfect in practice. More theoretical guarantees can be valuable for practical use of test-time adaptation.

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 [31, 44] has shown that different layers capture different levels of semantic granularity. The impact of architectural design on test-time training remains an open question. Furthermore, while we empirically compare our proposed method against other families of test-time adaptation algorithms, these techniques exploit different supervisory signals extracted from unlabeled data, which can be complementary to each other. Blending these techniques into a unified framework is another interesting direction to explore in the future.

Societal Impact. Our work 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 progress when the training and testing environments are highly similar. What if a machine encounters deployed to new environments? We, humans, have an innate capability for handling such shifts. We believe that machines should have the same capability as humans. Indeed, there is a long way to go. Nevertheless, we hope that our work will foster more research in analyzing and devising algorithms for robust test-time adaptation.

Acknowledgements

This work was supported by the Swiss National Science Foundation under the Grant 2OOO21-L92326, Honda R&D Co. Ltd, EPFL Open Science fund and Valeo. We thank Sudeep Salgia, Tao Lin, Lingjun Meng for helpful inputs to our early drafts and reviewers for valuable comments.

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] 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.

[3] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I. Jordan. Deep Transfer Learning with Joint Adaptation Networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 2208–2217. PMLR, July 2017. ISSN: 2640-3498.

[4] 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.

[5] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial Discriminative Domain Adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7167–7176, 2017.

[6] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-Time Training with Self-Supervision for Generalization under Distribution Shifts. In *Proceedings of the 37th International Conference on Machine Learning*, pages 9229–9248. PMLR, November 2020. ISSN: 2640-3498.

[7] 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.

[8] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully Test-Time Adaptation by Entropy Minimization. In *International Conference on Learning Representations*, September 2020.

[9] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum Contrast for Unsupervised Visual Representation Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020.

[10] 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.

[11] 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.

[12] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A Simple Framework for Contrastive Learning of Visual Representations. In *Proceedings of the 37th International Conference on Machine Learning*, pages 1597–1607. PMLR, November 2020. ISSN: 2640-3498.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] Jogendra Nath Kundu, Naveen Venkat, Rahul M. V, and R. Venkatesh Babu. Universal Source-Free Domain Adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4544–4553, 2020.

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

[19] Collin Burns and Jacob Steinhardt. Limitations of Post-Hoc Feature Alignment for Robustness. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2525–2533, 2021.

[20] Yang Fu, Sifei Liu, Umar Iqbal, Shalini De Mello, Humphrey Shi, and Jan Kautz. Learning to Track Instances without Video Annotations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8680–8689, 2021.

[21] Nicklas Hansen, Rishabh Jangir, Yu Sun, Guillem Alenyà, Pieter Abbeel, Alexei A. Efros, Lerrel Pinto, and Xiaolong Wang. Self-Supervised Policy Adaptation during Deployment. In *International Conference on Learning Representations*, September 2020.

[22] 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.

[23] Baochen Sun, Jiashi Feng, and Kate Saenko. Correlation Alignment for Unsupervised Domain Adaptation. In Gabriela Csurka, editor, *Domain Adaptation in Computer Vision Applications*, Advances in Computer Vision and Pattern Recognition, pages 153–171. Springer International Publishing, Cham, 2017.

[24] Werner Zellinger, Thomas Grubinger, Edwin Lughofer, Thomas Natschläger, and Susanne Saminger-Platz. Central Moment Discrepancy (CMD) for Domain-Invariant Representation Learning. *International Conference on Learning Representations*, International Conference on Learning Representations, November 2016. 00283.

[25] Yaroslav Ganin and Victor Lempitsky. Unsupervised Domain Adaptation by Backpropagation. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 1180–1189. PMLR, June 2015. ISSN: 1938-7228.

[26] Mehdi Noroozi and Paolo Favaro. Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *European Conference on Computer Vision*, pages 69–84, Cham, 2016. Springer International Publishing.

[27] 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.

[28] Yuejiang Liu, Qi Yan, and Alexandre Alahi. Social NCE: Contrastive Learning of Socially-Aware Motion Representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15118–15129, 2021.

[29] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging Properties in Self-Supervised Vision Transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9650–9660, 2021.

[30] Xinlei Chen and Kaiming He. Exploring Simple Siamese Representation Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15750–15758, 2021.

[31] 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.

[32] Chao Chen, Zhihang Fu, Zhihong Chen, Sheng Jin, Zhaowei Cheng, Xinyu Jin, and Xiansheng Hua. HoMM: Higher-Order Moment Matching for Unsupervised Domain Adaptation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):3422–3429, April 2020. Number: 04.

[33] Amir R. Zamir, Alexander Sax, William Shen, Leonidas J. Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling Task Transfer Learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3712–3722, 2018.

[34] 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. 00623 arXiv: 2006.07733.

[35] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised Learning of Visual Features by Contrasting Cluster Assignments. *arXiv:2006.09882 [cs]*, January 2021. 00403 arXiv: 2006.09882.

[36] 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.

[37] 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*, pages 6028–6039. PMLR, November 2020. ISSN: 2640-3498.

[38] Pascal Germain, Amaury Habrard, François Laviolette, and Emilie Morvant. A PAC-Bayesian Approach for Domain Adaptation with Specialization to Linear Classifiers. In *Proceedings of the 30th International Conference on Machine Learning*, pages 738–746. PMLR, May 2013. ISSN: 1938-7228.

[39] 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.

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

[41] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do ImageNet Classifiers Generalize to ImageNet? In *Proceedings of the 36th International Conference on Machine Learning*, pages 5389–5400. PMLR, May 2019. ISSN: 2640-3498.

[42] 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.

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

[44] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How Transferable Are Features in Deep Neural Networks? In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 3320–3328, Cambridge, MA, USA, 2014. MIT Press.

[45] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised Contrastive Learning. *arXiv:2004.11362 [cs, stat]*, April 2020. arXiv: 2004.11362.

A Proofs of Theorems

In this section, we prove the theoretical results in the Section 4.1.

A.1 Proof of Theorem 1

Lemma 1. *Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space and let $(A_i)_{i\in\{1...n\}}$ be a partition of Ω. Let C be the set of partitions of Ω whose elements have the same probabilities as $(A_i)_{i\in\{1...n\}}$, that is :*

$$C = \{(U_i)_{i\in\{1...n\}} \; / \quad \bigcup_i U_i = \Omega; \quad \forall (i,j), i \neq j, U_i \cap U_j = \varnothing; \quad \forall i, \mathbb{P}(U_i) = \mathbb{P}(A_i)\}. \tag{8}$$

If $n = 2$ or $\mathbb{P}(A_1) \geq 1/2$ then :

$$\min_{(B_i)_{i\in\{1...n\}}\in C} \sum_i \mathbb{P}(B_i \cap A_i) \geq 2 \times P(A_1) - 1. \tag{9}$$

Proof. If $n > 2$ and $\mathbb{P}(A_1) \geq 1/2$, then we can write $A'_1 = A_1$ and $A'_2 = (\bigcup_{i=2...,n} A_i)$ and reason similarly as in the case where $n = 2$ with (A'_1, A'_2) and (B'_1, B'_2).

In the case n = 2, we have:

$$\begin{aligned}
\mathbb{P}(A_1 \cap B_1) + \mathbb{P}(A_2 \cap B_2) &= 1 - \mathbb{P}(A_1 \cap B_2) - \mathbb{P}(A_2 \cap B_1) \\
&\geq 1 - \mathbb{P}(B_2) - \mathbb{P}(A_2) \\
&\geq 1 - 2 \times (1 - P(A_1)) \\
&\geq 2 \times \mathbb{P}(A_1) - 1
\end{aligned}$$

The intuition is that no matter how the partitions are built, if $P(A_1) > 1/2$ and $P(B_1) > 1/2$, there is necessarily an overlap between the two subsets such that $A_1 \cap B_1 \neq \varnothing$. □

Theorem 1. *If we assume that:*

- $Z \overset{d}{=} Z'$;
- $(Z \mid Y_s = k) \overset{d}{=} (Z' \mid Y'_s = k), \forall k \in \{1, ..., K\}$;

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

$$\mathbb{P}(\pi_m(Z') = Y'_m) \geq \sum_{y_s} \mathbb{P}(Y_s = y_s) \max\left\{0, 2\left(\max_{y_m} \mathbb{P}(Y_m = y_m \mid Y_s = y_s) - \frac{1}{2}\right)\right\}. \tag{10}$$

Proof. Using that $\bigcup_{y_s}\{Y'_s = y_s\} = \Omega$, we obtain using the law of total probability that:

$$\begin{aligned}
\overbrace{\mathbb{P}(\pi_m(Z') = {Y_m}')}^{\text{Accuracy}} &= \sum_{y_s} \mathbb{P}(\pi_m(Z') = {Y_m}' \mid {Y_s}' = y_s)\mathbb{P}({Y_s}' = y_s) \\
&= \sum_{y_s} \mathbb{P}({Y_s}' = y_s) \sum_{y_m} \mathbb{P}(\{\pi_m(Z') = y_m\} \cap \{{Y_m}' = y_m\} \mid {Y_s}' = y_s).
\end{aligned} \tag{11}$$

Let us introduce $A^{(y_s)}_{y_m} = \{Y'_m = y_m \mid Y'_s = y_s\}$ and $B^{(y_s)}_{y_m} = \{\pi_m(Z') = y_m \mid Y'_s = y_s\}$. It is important to note that $\mathbb{P}(A^{(y_s)}_{y_m}) = \mathbb{P}(B^{(y_s)}_{y_m})$. Indeed, we know that:

$$\mathbb{P}(B^{(y_s)}_{y_m}) = \mathbb{P}(\pi_m(Z') = y_m \mid Y'_s = y_s). \tag{12}$$

Moreover, we assume conditional distribution to be aligned, and π_m not to be retrained, as a result equation (12) can be written as:

$$\begin{aligned}\mathbb{P}(B_{y_m}^{(y_s)}) &= \mathbb{P}(\pi_m(Z) = y_m \mid Y_s = y_s) \\ &= \mathbb{P}(Y_m = y_m \mid Y_s = y_s) \\ &= \mathbb{P}(Y_m' = y_m' \mid Y_s' = y_s') \\ &= \mathbb{P}(A_{y_m}^{(y_s)}).\end{aligned} \tag{13}$$

Then, we can rewrite equation (11), namely the accuracy, as:

$$\overbrace{\mathbb{P}(\pi_m(Z') = {Y_m}')}^{\text{Accuracy}} = \sum_{y_s} \mathbb{P}({Y_s}' = y_s) \sum_{y_m} \mathbb{P}(A_{y_m}^{(y_s)} \cap B_{y_m}^{(y_s)}). \tag{14}$$

Finally, without loss of generality, we can assume that the indexes of $(A_{y_m}^{(y_s)})_{y_m}$ and $(B_{y_m}^{(y_s)})_{y_m}$ are ordered such that:

1. $\forall y_s \in \{1...K_s\}, \quad \mathbb{P}(A_1^{(y_s)}) \geq \mathbb{P}(A_2^{(y_s)}) \geq ... \geq \mathbb{P}(A_{K_m}^{(y_s)})$;
2. $\forall y_s \in \{1...K_s\}, \forall i \in \{1...K_m\}, \quad \mathbb{P}(A_i^{(y_s)}) = \mathbb{P}(B_i^{(y_s)})$.

Let us now define $C^{(y_s)}$, the set of partitions of Ω whose elements have the same probabilities as $(A_i^{(y_s)})_{i \in \{1...K_m\}}$. That is,

$$C^{(y_s)} = \{(U_i)_{i \in \{1...K_m\}} \; / \quad \bigcup_i U_i = \Omega; \quad \forall (i,j), i \neq j, U_i \cap U_j = \varnothing; \quad \forall i, \mathbb{P}(U_i) = \mathbb{P}(A_i^{(y_s)})\}. \tag{15}$$

It is clear that: $(B_i^{(y_s)})_{i \in \{1...K_m\}} \in C^{(y_s)}$.

Hence, it is also clear that:

$$\sum_{y_s} \mathbb{P}({Y_s}' = y_s) \sum_i \mathbb{P}(A_i^{(y_s)} \cap B_i^{(y_s)}) \geq \sum_{y_s} \mathbb{P}({Y_s}' = y_s) \min_{(U_i^{(y_s)}) \in C^{(y_s)}} \sum_i \mathbb{P}(A_i^{(y_s)} \cap U_i^{(y_s)}) \tag{16}$$

Therefore, we can lower bound the accuracy in equation (14) using the inequality (16) above, such that:

$$\overbrace{\mathbb{P}(\pi_m(Z') = {Y_m}')}^{\text{Accuracy}} \geq \sum_{y_s} \mathbb{P}({Y_s}' = y_s) \min_{(U_i^{(y_s)}) \in C^{(y_s)}} \sum_i \mathbb{P}(A_i^{(y_s)} \cap U_i^{(y_s)}) \tag{17}$$

Let us now separate two cases:

1. $K_m > 2$ and $\mathbb{P}(A_1^{(y_s)}) < 1/2$;
2. $K_m \leq 2$ or $\mathbb{P}(A_1^{(y_s)}) \geq 1/2$.

We shall henceforth ignore the index (y_s) for better clarity.

In case 1, we simply use that:

$$\forall (U_i^{(y_s)})_{i \in \{1...K_m\}} \in C^{(y_s)}, \sum_i \mathbb{P}(A_i^{(y_s)} \cap U_i^{(y_s)}) \geq 0. \tag{18}$$

In case 2 , we show that (cf. Lemma 1):

$$\min_{(B_i)_{i \in \{1...K_m\}} \in C} \sum_i \mathbb{P}(B_i \cap A_i) \geq 2 \times P(A_1) - 1. \tag{19}$$

The final result comes from the fact that:

$$\mathbb{P}(A_1) < 1/2 \implies P(A_1) - 1/2 < 0$$

Hence the two cases are summarized by the formula:

$$\min_{(B_i)_{i\in\{1\dots K_m\}}\in C} \sum_i \mathbb{P}(B_i \cap A_i) \geq \max\left\{0, 2\left(\max_i \mathbb{P}(A_i) - \frac{1}{2}\right)\right\}.$$

Finally, as the joint laws are assumed equal, namely $(Y_m, Y_s) \sim (Y'_m, Y'_s)$, it comes that:

$$\begin{aligned} P(A_1^{(y_s)}) &= \max_{y_m} \mathbb{P}(Y'_m = y_m \mid Y'_s = y_s) \\ &= \max_{y_m} \mathbb{P}(Y_m = y_m \mid Y_s = y_s). \end{aligned} \tag{20}$$

□

A.2 Proof of Theorem 2

Lemma 2. *If $Y \perp\!\!\!\perp X \mid Z$ and X, Y, Z are discrete random variables then,*

$$\forall (x, y, z) \in (X(\Omega) \times Y(\Omega) \times Z(\Omega)) \quad \textit{with} \quad \mathbb{P}(X = x \cap Z = z) > 0,$$
$$\mathbb{P}(Y = y \mid X = x, Z = z) = \mathbb{P}(Y = y \mid Z = z)$$

Proof. $\forall (x, y, z) \in (X(\Omega) \times Y(\Omega) \times Z(\Omega))$ such that $\mathbb{P}(X = x \cap Z = z) > 0$:

$$\begin{aligned} \mathbb{P}(Y = y \mid X = x, Z = z) &= \frac{\mathbb{P}(X = x, Y = y, Z = z)}{\mathbb{P}(X = x, Z = z)} \\ &= \frac{\overbrace{\mathbb{P}(X = x \mid Y = y, Z = z)}^{\text{cf. Assumption}} \mathbb{P}(Y = y, Z = z)}{\mathbb{P}(X = x, Z = z)} \\ &= \frac{\mathbb{P}(X = x \mid Z = z)\mathbb{P}(Y = y, Z = z)}{\mathbb{P}(X = x, Z = z)} \\ &= \frac{\mathbb{P}(X = x \mid Z = z)\mathbb{P}(Y = y \mid Z = z)}{\mathbb{P}(X = x \mid Z = z)} \\ \mathbb{P}(Y = y \mid X = x, Z = z) &= \mathbb{P}(Y = y \mid Z = z) \end{aligned}$$

□

Theorem 2. *If we assume that:*

- $Z \overset{d}{=} Z'$;
- $(Z \mid Y_s = k) \overset{d}{=} (Z' \mid Y'_s = k), \forall k \in \{1, ..., K\}$;
- $Z' \perp\!\!\!\perp Y'_m \mid Y'_s$;

then the accuracy of the model is:

$$\mathbb{P}(\pi_m(Z') = Y'_m) = \sum_{y_s} \left[\mathbb{P}(Y_s = y_s) \sum_{y_m} \mathbb{P}(Y_m = y_m \mid Y_s = y_s)^2\right]. \tag{21}$$

Proof. Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space. Let $Y_m(\Omega) = Y'_m(\Omega) = \{1, ..., K_m\}$ and $Y_s(\Omega) = Y'_s(\Omega) = \{1, ..., K_s\}$. In the following, for the sake of clarity we shall try to omit writing $Y_s(\Omega)$ and $Y_m(\Omega)$. Thus,when no confusion is possible we shall write $\bigcup_{y_s}$ instead of $\bigcup_{y_s \in Y(\Omega)}$.

Using the law of total probability, with $\bigcup_{y_s}\{Y'_s = y_s\} = \Omega$, it comes that:

$$\overbrace{\mathbb{P}(\pi_m(Z') = Y'_m)}^{\text{Accuracy}} = \sum_{y_s} \underbrace{\mathbb{P}(\pi_m(Z') = Y'_m \mid Y'_s = y_s)}_{A^{(y_s)}} \mathbb{P}(Y'_s = y_s). \tag{22}$$

Similarly, we reformulate $A^{(y_s)}$ with the law of total probability, using that $\bigcup_{y_m}\{Y'_m = y_m \mid Y'_s = y_s\} = \Omega$,and it comes that:

$$A^{(y_s)} = \sum_{y_m} \mathbb{P}(\pi_m(Z') = y_m \mid {Y_s}' = y_s, {Y_m}' = y_m)\mathbb{P}({Y_m}' = y_m \mid {Y_s}' = y_s).$$

We now replace $A^{(y_s)}$ in equation 22, which is the accuracy, and it comes that :

$$\overbrace{\mathbb{P}(\pi_m(Z') = Y'_m)}^{\text{Accuracy}} = \sum_{y_s} \mathbb{P}({Y_s}' = y_s) \sum_{y_m} \mathbb{P}(\pi_m(Z') = y_m \mid {Y_s}' = y_s, {Y_m}' = y_m)\mathbb{P}({Y_m}' = y_m \mid {Y_s}' = y_s). \tag{23}$$

If $Y'_m \perp\!\!\!\perp Z' \mid Y'_s$ (assumption 3), it can easily be shown (cf. Lemma 2) for all (y_s, y_m) such that $\mathbb{P}(Y'_s = y_s, Y'_m = y_m) > 0$, we have:

$$\mathbb{P}(\pi_m(Z') = y_m \mid {Y_s}' = y_s, {Y_m}' = y_m) = \mathbb{P}(\pi_m(Z') = y_m \mid Y'_s = y_s)$$

Furthermore, it is clear that

$$\mathbb{P}(Y'_s = y_s, Y'_m = y_m) = 0 \implies \mathbb{P}(Y'_s = y_s \mid Y'_m = y_m) = 0.$$

Hence, we can rewrite equation 23, namely the accuracy, using equation A.2 for all (y_s, y_m), and it comes that:

$$\overbrace{\mathbb{P}(\pi_m(Z') = {Y_m}')}^{\text{Accuracy}} = \sum_{y_s} \mathbb{P}({Y_s}' = y_s) \sum_{y_m} \mathbb{P}(\pi_m(Z') = y_m \mid {Y_s}' = y_s)\mathbb{P}({Y_m}' = y_m \mid {Y_s}' = y_s). \tag{24}$$

From assumption 2 on conditional features alignment, namely $\forall k \in Y_s(\Omega), (Z \mid Y_s = k) \overset{d}{=} (Z' \mid Y'_s = k)$, and given that the classifier π_m is fixed, it comes that :

$$\forall (y_m, y_s), \quad \mathbb{P}(\pi_m(Z') = y_m \mid Y'_s = y_s) = \mathbb{P}(\pi_m(Z) = y_m \mid Y_s = y_s). \tag{25}$$

We assumed that the classifier π_m is perfect on the training set, such that:

$$\forall (y_m, y_s), \quad \mathbb{P}(\pi_m(Z) = y_m \mid Y_s = y_s) = \mathbb{P}(Y_m = y_m \mid Y_s = y_s). \tag{26}$$

Hence, combining equality 26 and equality 25, it comes that:

$$\forall (y_m, y_s), \quad \mathbb{P}(\pi_m(Z') = y_m \mid Y'_s = y_s) = \mathbb{P}(Y_m = y_m \mid Y_s = y_s). \tag{27}$$

We now rewrite the accuracy, that is equation 24, using the equality 27 above, and it comes that:

$$\overbrace{\mathbb{P}(\pi_m(Z') = {Y_m}')}^{\text{Accuracy}} = \sum_{y_s} \mathbb{P}({Y_s}' = y_s) \sum_{y_m} \mathbb{P}(Y_m = y_m \mid Y_s = y_s) \underbrace{\mathbb{P}({Y_m}' = y_m \mid {Y_s}' = y_s)}_{\text{Joint distributions of labels}}. \tag{28}$$

We assumed that the joint distributions of the labels were constant over time, i.e., $(Y_m \cap Y_s) \stackrel{d}{=} (Y_m' \cap Y_s')$. Consequently, we replace the test time joint distribution by their training counterpart in equation 28, such that:

$$\overbrace{\mathbb{P}(\pi_m(Z') = {Y_m}')}^{\text{Accuracy}} = \sum_{y_s} \underbrace{\mathbb{P}({Y_s}' = y_s)}_{\text{Prior distribution}} \sum_{y_m} \mathbb{P}(Y_m = y_m \mid Y_s = y_s)\mathbb{P}(Y_m = y_m \mid Y_s = y_s). \quad (29)$$

Finally, we assumed that the prior distributions of the labels are constant over time, i.e., $Y_s \stackrel{d}{=} Y_s'$ Therefore, we replace the test time prior by the training time prior in equation 29 and it gives:

$$\mathbb{P}(\pi_m(Z') = {Y_m}') = \sum_{y_s} \mathbb{P}(Y_s = y_s) \sum_{y_m} \mathbb{P}(Y_m = y_m \mid Y_s = y_s)^2. \quad (30)$$

□

B Implementation Details

Joint Training. We use the same hyper-parameters as [45] to train the ResNet-50 on the classification and contrastive tasks jointly. We set the batch size to 256 and the weight of the self-supervised task λ to 0.1 in all experiments. We train the model for 1,000 epochs on CIFAR-10 and CIFAR-100 from scratch. On VisDA, we reduce the number of epochs to 100 and warm start the training from a pre-trained ResNet-50 due to limited training data.

Test-Time Adaptation. At test-time, we adapt the encoder using stochastic gradient descent with a learning rate of 0.001 and momentum of 0.9. We use a batch size of 256 for the self-supervised task and online feature alignment.

Contrastive Task. We use the same data augmentation strategy as [12]. For random cropping, we first create crops of random size and aspect ratio from raw images and subsequently resize them to the original size. For color distortion, we set the strength of color jitter to 0.5. We set the temperature parameter to 0.5 for CIFAR-10, CIFAR10-C, CIFAR-100 and CIFAR100-C, and 0.1 for the VisDA dataset.

C Additional Experiments

C.1 Additional Results on Common Corruption Datasets

In addition to the bar plot in Figure 3 from the main paper, we summarize the classification errors on CIFAR10-C with different severity levels of corruptions in Tables C.1-C.3. Across all three levels, our proposed TTT++ outperforms other strong baselines [8, 36, 37] by a clear margin. Specifically, our method leads to $\sim$23% lower classification errors on average than prior state-of-the-art methods.

C.2 Additional Results with Different Random Seeds

We follow the evaluation protocol of previous work [6, 8] and run all methods on the same pre-trained model with the same seed. As shown in Table C.4, the variance across different random seeds is minimal. We therefore report our main experimental results with only one random seed.

C.3 Additional Qualitative Results

In addition to Figure 3 from the main paper, we visualize the learned representation of test images on three other types of corruption in Figures C.1. These qualitative results confirm that while TTT-C itself leads to semantically more separated feature clusters, it cannot resolve the distributional shifts in the feature space. In comparison, the full version of our proposed TTT++ is able to improve both the feature alignment and the discriminative power of the test-time representations simultaneously.

Table C.1: Classification error (%) on CIFAR10-C, level-5 corruptions.

	brit	contr	defoc	elast	fog	frost	gauss	glass	impul	jpeg	motn	pixel	shot	snow	zoom	Average
Test	7.01	13.27	11.84	23.38	29.41	28.24	48.73	50.78	57	19.46	23.38	47.88	44	21.93	10.84	29.14
BN [36]	8.22	8.27	9.66	19.54	19.95	19.5	17.11	25.95	27.7	13.67	13.72	11.50	16.17	15.88	7.93	15.65
TENT[8]	7.14	7.16	8.28	16.86	14.49	11.99	14.64	21.39	22.1	12.01	11.28	9.6	13.34	12.16	7.15	12.64
SHOT [37]	8.01	7.95	9.51	18.93	18.88	13.15	16.42	24.74	26.27	13.55	13.39	11.23	15.38	15.55	7.74	14.71
TFA	7.44	7.40	8.89	15.73	12.82	11.49	12.94	18.46	19.13	11.66	10.77	9.93	12.67	11.73	7.03	11.87
TTT-C	5.32	5.7	8.05	15.37	8.39	11.11	14.63	19.87	12.41	**9.54**	8.76	11.93	13.06	9.91	7.1	10.74
TTT++	**5.20**	**5.43**	**7.73**	**13.08**	**8.09**	**9.73**	**12.73**	**15.70**	**12.45**	10.39	**8.52**	**8.87**	**11.07**	**8.75**	**6.31**	**9.60**

Table C.2: Classification error (%) on CIFAR10-C, level-4 corruptions.

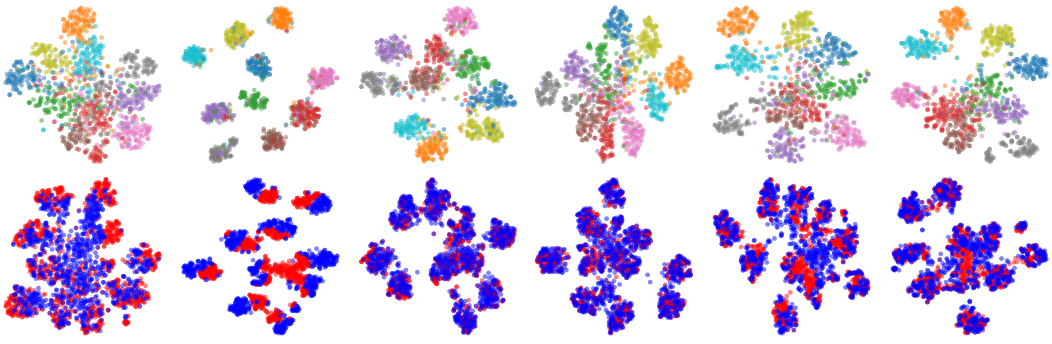
	brit	contr	defoc	elast	fog	frost	gauss	glass	impul	jpeg	motn	pixel	shot	snow	zoom	Average
Test	5.88	7.45	8.32	13.04	13.02	20.07	43.31	52.34	43.78	17.12	16.72	26.45	34.34	19.31	8.12	21.95
BN [36]	7.33	7.48	8.19	13.46	13.10	11.50	15.63	25.36	21.65	12.11	12.35	8.98	12.91	16.70	7.05	12.92
TENT [8]	6.71	6.62	7.08	11.73	9.13	10.66	13.61	20.39	17.12	10.77	10.02	8.56	11.04	13.41	6.59	10.90
SHOT [37]	6.71	6.90	7.66	12.31	11.22	10.77	14.30	22.49	18.68	11.33	11.13	8.51	11.58	15.05	6.68	11.69
TFA	6.55	6.51	7.38	11.76	9.96	10.03	12.65	18.46	15.39	10.45	10.36	8.36	10.69	12.79	6.47	10.52
TTT-C	4.85	5.02	6.14	10.17	6.00	8.47	12.84	19.90	11.48	10.58	8.17	7.43	10.24	10.44	6.15	9.19
TTT++	**4.34**	**4.81**	**5.68**	**9.52**	**5.91**	**7.74**	**12.08**	**15.92**	**9.47**	**9.34**	**7.71**	**6.93**	**9.26**	**9.08**	**5.80**	**8.24**

Table C.3: Classification error (%) on CIFAR10-C, level-3 corruptions.

	brit	contr	defoc	elast	fog	frost	gauss	glass	impul	jpeg	motn	pixel	shot	snow	zoom	Average
Test	5.64	6.47	5.73	7.69	8.98	18.54	36.96	35.53	26.86	15.54	16.68	13.10	28.00	16.89	7.54	16.68
BN [36]	6.95	6.96	7.03	9.27	10.19	11.21	13.53	16.53	15.84	10.91	12.20	8.42	12.12	14.90	7.26	10.89
TENT [8]	6.51	6.44	6.36	8.63	7.90	9.87	11.88	14.26	12.99	10.38	10.58	7.24	9.97	11.87	6.67	9.44
SHOT [37]	6.58	6.66	6.80	8.67	9.12	10.46	12.14	15.17	14.06	10.40	10.93	7.74	10.72	12.78	6.59	9.92
TFA	6.32	6.46	6.63	8.61	8.78	10.17	11.10	13.23	11.54	9.99	10.20	7.49	10.21	12.03	6.70	9.30
TTT-C	4.51	4.81	4.77	6.79	5.34	8.99	11.38	12.93	8.63	9.86	8.09	6.49	9.49	8.70	5.95	7.78
TTT++	**4.26**	**4.50**	**4.68**	**6.47**	**5.18**	**7.84**	**9.92**	**10.99**	**8.06**	**8.51**	**7.66**	**5.97**	**8.43**	**7.78**	**5.46**	**7.05**

Table C.4: Classification error (%) of TTT+ with different random seeds on CIFAR10-C, level-5 corruptions.

Seed	brit	contr	defoc	elast	fog	frost	gauss	glass	impul	jpeg	motn	pixel	shot	snow	zoom	Average
0	5.20	5.43	7.73	13.08	8.09	9.73	12.73	15.70	12.45	10.39	8.52	8.87	11.07	8.75	6.31	9.60
1	5.09	5.37	7.47	12.62	7.95	9.44	12.63	16.19	12.25	10.40	8.59	8.51	11.22	8.71	6.12	9.50
2	5.25	5.50	7.69	13.04	8.17	9.46	13.05	16.21	11.95	10.49	8.57	8.48	11.14	8.76	6.34	9.61
Std	0.08	0.07	0.14	0.25	0.11	0.16	0.22	0.29	0.25	0.06	0.04	0.22	0.08	0.03	0.12	0.06



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

Figure C.1: T-SNE visualization of the representation for the CIFAR10 images with the level-5 elastic transform corruption. Top row: per-class feature distribution. Bottom row: marginal feature distribution on the original test images (red) and corrupted test images (blue).