ReC-TTT: Contrastive Feature Reconstruction for Test-Time Training

Marco Colussi[1*] Sergio Mascetti[1] Jose Dolz[2] Christian Desrosiers[2]

[1]Università degli studi di Milano
{marco.colussi, sergio.mascetti}@unimi.it

[2]ÉTS Montréal
{jose.dolz, christian.desrosiers}@etsmtl.ca

Abstract

The remarkable progress in deep learning (DL) showcases outstanding results in various computer vision tasks. However, adaptation to real-time variations in data distributions remains an important challenge. Test-Time Training (TTT) was proposed as an effective solution to this issue, which increases the generalization ability of trained models by adding an auxiliary task at train time and then using its loss at test time to adapt the model. Inspired by the recent achievements of contrastive representation learning in unsupervised tasks, we propose ReC-TTT, a test-time training technique that can adapt a DL model to new unseen domains by generating discriminative views of the input data. ReC-TTT uses cross-reconstruction as an auxiliary task between a frozen encoder and two trainable encoders, taking advantage of a single shared decoder. This enables, at test time, to adapt the encoders to extract features that will be correctly reconstructed by the decoder that, in this phase, is frozen on the source domain. Experimental results show that ReC-TTT achieves better results than other state-of-the-art techniques in most domain shift classification challenges. The code is available at: https://github.com/warpcut/ReC-TTT

1. Introduction

The ability of deep learning (DL) models to generalize to new data is one of the main challenges that research in the field is facing. In the standard scenario, models are trained to learn patterns and relations on a source dataset, and performances are evaluated on a set of images not seen during training but extracted from the same distribution. Despite the impressive performance achieved by advanced models on various datasets, maintaining the assumption of domain invariance between source and target data proves to be impractical in many real-world scenarios. As a result, the limited robustness of DL models to distribution shifts remains a key obstacle to their use [16, 26].

As a solution to this challenge, two broad research branches have emerged: domain generalization (DG) and unsupervised domain adaptation (UDA). DG approaches aim at training more robust models with a native ability to generalize on various domains. The main limitation of these techniques is that they rely on the availability at train time of large amounts of data from different sources, which is often impractical [1, 31]. Moreover, these techniques may also underperform in domains very different from those considered during training.

On the other hand, UDA tries to achieve higher generalizability without anticipating potential distribution shifts but instead by adapting the model accordingly, either with test-time adaptation (TTA) [14, 27] or test-time training (TTT) [9, 25]. TTA methods perform adaptation without accessing the initial training step on the source data nor the labels of test samples. Our work focuses on the second scenario, TTT, which relaxes the very strict TTA setting and allows the design of the source training in a way that facilitates adaptation. During training, TTT approaches follow a multi-task learning strategy where the model learns an auxiliary self-supervised, or unsupervised task, in addition to the main learning objective, sharing encoder features across tasks. Then, at test time, the model uses solely the auxiliary task to update the shared encoder with a loss that was already optimized for the source domain.

Our work leverages the concept of contrastive learning [3] for improving test-time training. The idea of this powerful technique is to learn, from unlabeled data, general-purpose features that are similar in related samples and different in unrelated ones. Previous work has shown the usefulness of contrastive learning in a variety of unsupervised and semi-supervised image tasks [3, 10, 22]. Among others, *ReContrast* [8], which inspired our approach, adopts feature reconstruction contrastive learning in *unsupervised anomaly detection*, demonstrating a good transfer ability to various image domains compared to other unsupervised techniques. However, this recent approach has not been investigated for adapting models at test time.

This paper presents *ReC-TTT* (Contrastive Feature Reconstruction for Test-Time Training), a test-time train-

*Corresponding author.

ing approach designed for image classification that leverages techniques from the field of contrastive representation learning in a novel way. The core idea of *ReC-TTT* is to use a pre-trained frozen encoder to generate a discriminative feature representation of the input image. This representation is then used as a positive pair in the learning of the auxiliary task. In particular, during the training phase, two encoders are trained in a supervised manner to classify the images and, at the same time, a decoder is trained to minimize the differences between the features extracted from the trainable encoders and the ones reconstructed from the frozen pre-trained encoder. The intuition is that during test-time training, the now frozen decoder works as a guide to extract more meaningful information by the trainable encoders.

Our main contributions can be summarized as follows.

- We propose to use contrastive feature reconstruction as self-supervised task in TTT, which has been overlooked in the literature.
- We enhance vanilla contrastive feature reconstruction with an ensemble learning strategy where two classifiers are trained with different image augmentations to yield consistent predictions.
- Comprehensive experiments on various datasets with different types of distribution shifts and relevant ablation studies demonstrate the superiority of our approach compared to recent TTA and TTT methods, yielding state-of-the-art performance.

In the next section, we provide an overview of existing TTA and TTT methods, emphasizing the novelty of our method presented in Section 3. Section 4 describes the experimental setting, reports the performances, and the ablation studies. Finally, our conclusions and limitations are discussed in Section 5.

2. Related work

Test-Time Adaptation (TTA). In visual recognition tasks, distribution shifts between training and test data can greatly degrade performance [24]. To overcome this issue, recent approaches proposed to dynamically adapt the models at test time to the new data. Unlike domain generalization [1, 30], where the source model is robustly trained but fixed at test time, TTA allows updating the model for the target domain. TTA techniques do not have access to the source data or training (*i.e.*, only the trained model is provided), and the adaptation happens only at test time. A variety of TTA methods have been proposed in recent years. Among others, in PTBN [17], the adaptation is carried out by updating the BatchNorm layer statistics using the test batch. Instead, TENT [28] tries to minimize the entropy of the predictions for the test set. Finally, TIPI [18] proposes to identify transformations that can approximate the domain shift, and trains the model to be invariant to such transformations.

Test-Time Training (TTT). In contrast to TTA, TTT techniques have access to the source data during initial training (but not at test time), and a secondary self-supervised task is trained jointly with the main learning objective. This learning paradigm was first introduced in TTT [25], where the auxiliary task consists in recovering a random rotation of multiples of $90°$. At test time, the adaptation is performed by updating only the parameters related to the secondary task. TTT-MAE [7] uses transformers as backbone for the supervised training, with a masked-autoencoders architecture trained as a self-supervised reconstruction task; at test time, the network is trained only to reconstruct the masked images, adapting the shared feature extractor. TTTFlow [20] adopts normalizing flows on top of a pre-trained network to map the features into a simple multivariate Gaussian distribution. At test time, the log-likelihood of this distribution is employed to adapt the model. ClusT3 [9] proposed an unsupervised clustering task that maximizes the Mutual Information between the features and the clustering assignment that should remain constant across different domains.

Contrastive learning as auxiliary task. Contrastive learning as self-supervised task is gaining remarkable attention in domain adaptation research, thanks to its ability to learn robust representations. Among the various approaches based on this technique, AdaContrast [2] takes advantage of both momentum contrastive learning and weak-strong consistency regularization for pseudo-label supervision. In DaC [29], the test set is divided into source-like and domain-specific, applying two different strategies to the sub-sets using adaptive contrastive learning. TTT^{++} [15] adopts a contrastive approach on top of a TTT framework, using two augmented versions of the same image as positive pairs and, as negative pairs, augmented versions of other images. This technique also leverages a batch-queue decoupling to regularize the adaptation with smaller batch sizes. More recently, NC-TTT [19] introduces a contrastive approach based on the synthetic generation of noisy feature maps.

The proposed *ReC-TTT* method leverages contrastive feature reconstruction, whose ability to identify relevant domain-specific features was recently shown in anomaly detection [8]. We extend this recent work by introducing the first TTT approach based on this technique. In addition to this new application setting, we enhance the vanilla contrastive feature reconstruction model of [8] in several important ways. First, we introduce a classification task to the trainable encoder, learned jointly with the feature re-

construction. Second, instead of having a single trainable encoder as in *ReContrast* [8], our model includes a second one with its own classifier, processing an augmented version of the image. Finally, we introduce a regularization loss between the two classifiers, which promotes consistency across features extracted from the original and augmented data for an improved model generalization.

Our novel architecture provides significant benefits compared to existing approaches for test-time training. While current TTT approaches based on contrastive learning rely solely on transformation invariance, our method also exploits the more general principle of reconstruction. However, unlike approaches such as TTT-MAE [7] that measure reconstruction error on the output image, our method operates on the features of different network layers. This enables it to capture a broader spectrum of domain shifts (from pixel-level noise to image-level distortion). Last, our method improves upon these approaches via an ensemble learning strategy that combines the predictions of two independent classifiers.

3. Methodology

Our *ReC-TTT* method addresses the problem of domain shift between a given training set, representing the source domain $\mathcal{S} = (X_S, Y_S)$, and a test set from a target domain $\mathcal{T} = (X_T, Y_T)$, where X_S, X_T are spaces containing images and Y_S, Y_T the spaces of corresponding labels. In this setting, we suppose that both domains have the same labels, i.e., $Y_S = Y_T$, but that images have a different conditional distribution, i.e., $p_S(x|y) \neq p_T(x|y)$ where $x \in X$ and $y \in Y$.

Figure 1 shows the overall framework of our method. The architecture employed in *ReC-TTT* consists of two trainable encoders $f_{\theta 1}$ and $f_{\theta 2}$, a pre-trained frozen encoder f_{θ_F}, and a decoder g_θ that takes in input the concatenated features extracted from the three encoders. As other TTT approaches, *ReC-TTT* requires two steps. In the first step, our method has access to the source domain and the model is trained to learn a function mapping $X_S \rightarrow Y_S$ using a classification loss ($\mathcal{L}_{CE}$) and an auxiliary loss ($\mathcal{L}_{aux}$). The second step occurs at test time, where our method has only access to the unlabeled target set. In our case, $f_{\theta 1}$ and $f_{\theta 2}$ are updated using only the auxiliary function to learn the new mapping $X_T \rightarrow Y_T$. This partial update enables the model to learn the association in $\mathcal{T}$ while maintaining the knowledge acquired during training on $\mathcal{S}$.

In the following sections, we detail the different components of our method.

3.1. Contrastive feature reconstruction

Contrastive learning extracts meaningful representations by maximizing the agreement between the features of different views of the input data during training. In our framework, illustrated in Figure 1, this is achieved using two separate encoders. The first one ($f_{\theta 1}$) is updated during training, and hence generates a domain-specific domain representation, while the other ($f_{\theta F}$) is instead frozen and thus generates a domain representation based on a pre-trained network.

The extracted features are then combined into a bottleneck that resembles the last ResNet layer, and subsequently fed into a shared decoder (g_θ) which has the opposite architecture of the encoders. For a fair comparison with previous TTA and TTT works [9, 15], our method uses a ResNet50 backbone for the convolutional feature extractors.

Learning objective. The network is trained using global cosine-similarity [8] between the features at different layers of the encoders and the features at the opposite level of the decoder. Specifically, the model is trained in a cross-reconstruction fashion where the decoder learns to reconstruct the features of the frozen encoder starting with the ones obtained by the trainable encoder, and vice versa, using the following loss:

$$\mathcal{L}_{aux} = \sum_{\ell=1}^{L} 1 - \frac{\langle sg(f_E^\ell), f_D^\ell \rangle}{sg(\|f_E^\ell\|_2)\, \|f_D^\ell\|_2} \tag{1}$$

where sg is the stop gradient operation [4] used to avoid propagating the gradient directly into the encoder, f_E^ℓ and f_D^ℓ represent the flattened features of the encoder and decoder respectively at the ℓ^{th} layer, and $\langle \cdot, \cdot \rangle$ is the dot product operation.

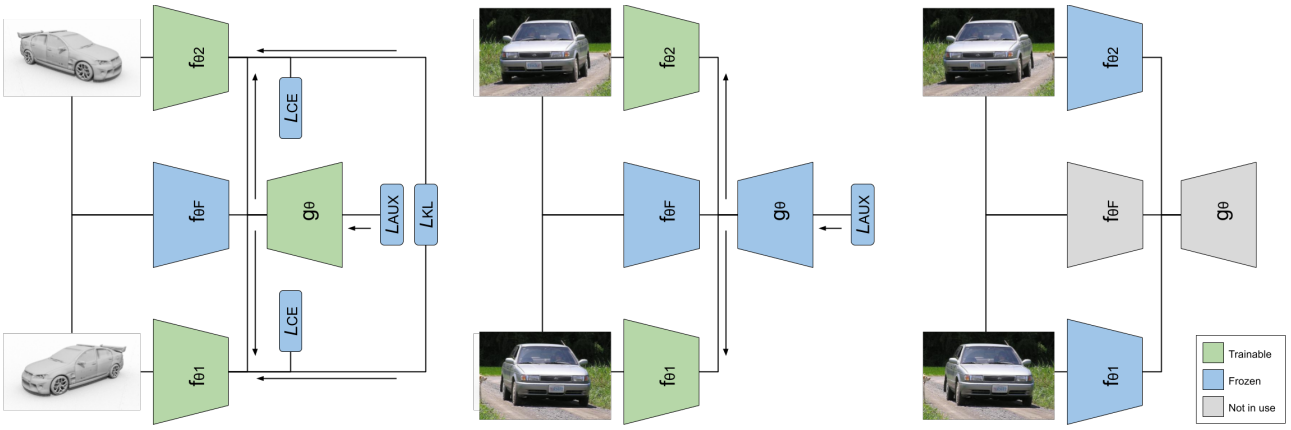
In the TTT framework, during training, the network is jointly trained on the two tasks: supervised classification and contrastive feature reconstruction combining the cross-entropy with the auxiliary loss described in Eq. (1), as follows:

$$\mathcal{L}_{train} = \mathcal{L}_{CE} + \mathcal{L}_{aux} \tag{2}$$

Experimental results (see ablation study reported in Section 2 of supplementary material) show that the contrastive reconstruction loss outperforms arguably simpler solutions such as *SimSiam* [4].

3.2. Encoder ensemble

As shown in Fig. 1a, our *ReC-TTT* method also leverages an ensemble learning strategy that integrates a secondary trainable encoder ($f_{\theta 2}$) and classification predictor. This encoder takes as input an augmented version of the original image to learn diversified representations of the data and add robustness to the contrastive learning process. The same image is fed to the frozen encoder ($f_{\theta F}$) to generate two contrastive pairs, such that the representations extracted by the decoder should be invariant to the augmentation applied. To avoid introducing information that could artificially facilitate the adaptation to specific domain shifts (as



(a) Train time (b) Test time (c) Inference time

Figure 1. **Overview of our *ReC-TTT* framework.** The directional flow of gradients is denoted by the symbol $\rightarrow$. $\mathcal{L}_{aux}$ is our cross reconstruction loss, which computes the global similarity between the features of the encoders and the features reconstructed by the decoder, $\mathcal{L}_{CE}$ is the cross-entropy between the predicted classes and the true labels, and $\mathcal{L}_{KL}$ is the Kullback–Leibler divergence between the two predicted distributions. The trainable components of our architecture are depicted in **green**, whereas the frozen components are represented in **blue**. (**a**) illustrates the training phase, where both the encoders and the decoder are trainable. At test-time training (**b**), the decoder is frozen. Finally, (**c**) shows the inference time when the entire network is frozen; modules represented in **gray** are not needed in this phase.

noise), we selected a weak, domain-agnostic augmentation: horizontal flip.

Learning objective. The model is trained with the loss of Eq. (2) applied to both encoders. Furthermore, we added a consistency loss between the two predictors, measuring their Kullback–Leibler (KL) divergence, to align the distributions predicted by the two encoders:

$$\mathcal{L}_{train} = \mathcal{L}_{CE} + \mathcal{L}_{aux} + \mathcal{L}_{KL} \tag{3}$$

Let P and Q be two discrete probability distributions over k classes. The KL divergence is computed as

$$D_{KL}(P \,\|\, Q) = \sum_{k} p_k \log\left(\frac{p_k}{q_k}\right). \tag{4}$$

Adaptation. Algorithm 1 describes how our method is used at test time for adapting the model to data from unseen domains. At this stage, we freeze the shared decoder and, for each test batch, reinitialize the weights of encoders. Afterward, since we have no access to labels for the supervised loss, the layers of the trainable encoders are updated using only Eq. (1) for a total of T iterations. For the final inference, the whole model is frozen, and we obtain the final classification by averaging the predictions of two independent encoders. As illustrated in Fig. 1c to reduce computational complexity during inference, the architecture can be optimized by removing the frozen encoder and decoder, which are no longer necessary for generating predictions.

Algorithm 1: Test-Time Training Algorithm

Data: Trained model parameters θ_0, test set X_T
Result: Predicted labels $\hat{Y}$

for *param* $\in \theta_g$ **do**
  param.trainable $\leftarrow$ False
end
for *batch* $\in X$ **do**
  $\theta \leftarrow \theta_0$; `// Initialize weights`
  for *iter* $t = 1..T$ **do**
    Get layers features of batch samples x using model with parameters θ;
    $\mathcal{L}_{aux} \leftarrow$ Auxiliary loss between encoders and decoder;
    $\nabla_\theta \mathcal{L} \leftarrow$ Compute gradient of $\mathcal{L}_{aux}$ with respect to θ_{t-1};
    $\theta_t \leftarrow \theta_{t-1} - \alpha \nabla_\theta \mathcal{L}$; `// Update model parameters`
  end
  $\hat{y} \leftarrow$ Make prediction using θ_T on examples x;
end
return $\hat{Y}$

4. Experiments

4.1. Experimental setup

Six publicly available datasets were selected for the evaluation. These datasets simulate various types of domain shift: image corruption, natural domain shift, and synthetic to real images.

CIFAR-10C, CIFAR-100C and TinyImageNet-C [12]. These three datasets are composed of 15 different types of corruptions, from various types of noise and blur to weather and digital corruptions. The images present five levels of severity for each perturbation and all the experiments were conducted using only the most severe category (level 5). The datasets consist of $10,000$ test images labeled into 10 classes for CIFAR-10C, 100 classes for CIFAR-100C, and 200 classes for TinyImageNet-C.

CIFAR-10.1 [23]. We also use the CIFAR-10.1 dataset to evaluate our model's ability to generalize to natural domain shift that takes place when images are re-collected after a certain time. The CIFAR-10.1 dataset is composed of $2,000$ images collected several years after the original CIFAR-10 dataset, with the same 10 classes.

VisDA [21]. The Visual Domain Adaptation (VisDA) dataset was designed to pose a new challenge in domain adaptation: from synthetic images to real-world images. This dataset is composed of $152,397$ train images consisting of 2D renderings, $55,388$ validation images extracted from the COCO dataset, and $72,372$ YouTube video frames that compose the test set. All images are labeled into 12 different classes. We evaluated the model's ability to generalize from the training set to the validation set ($train \rightarrow val$) and from the training set to the test set ($train \rightarrow test$).

Training protocol. Following previous work, our method employs Resnet50 as the backbone, using 32×32 images for the CIFAR datasets, 64×64 images for the TinyImageNet and 224×224 for the VisDA dataset. The backbone is pre-trained using ImageNet32 [5] for the first one and ImageNet [6] for the latter. Following existing literature, all CIFAR models were trained for 300 epochs with SGD as optimizer, a batch size of 128, and an initial learning rate of 0.1 with a multi-step scheduler, decreasing the learning rate by a factor of 0.1 every 25 epochs. In contrast, for VisDA, the model was trained for 100 epochs, a batch size of 64, and a learning rate of 0.001 without a scheduler.

Inference. At test time, the shared decoder is frozen, while the rest of the network is trained with SGD and a learning rate of 0.005 for CIFAR datasets and 0.0001 for VisDA, using only the auxiliary loss. Similarly to previous approaches, we reset the weights after each batch, hence enabling the consequential processing of batches with different domain shifts.

The experiments were run on a Ubuntu server with an `NVIDIA A100` GPU, 42Gb of RAM, and an `AMD EPYC` 8-core CPU. The code is implemented in python3 with `PyTorch` 1.12.0.

4.2. Empirical results

4.2.1 Comparison with state of the art

Our model was compared with seven recent approaches: ResNet50 [11] as the baseline, trained with the same strategy as our method, but only on the supervised task, PTBN [17], TENT [28], TTT++ [15][1], TIPI [18], ClusT3 [9] and NC-TTT [19]. For a fair comparison, all TTA methods were evaluated on the same pre-trained ResNet50, while TTT approaches were trained using the same ResNet50 base architecture and the same training strategy.

CIFAR-10 corruptions. Table 1 shows the comparison on the CIFAR-10C dataset of the different state-of-the-art methods, the baseline, and our approach. It is noticeable that *ReC-TTT* outperforms on average all previous methods, with a gain of 1.46% on TTT++ and 36.2% on the baseline. Also, our method is the only one able to outperform the baseline for the natural domain shift (CIFAR 10.1, see last line in Table 1). As discussed in previous papers [9], other techniques perform worse than the baseline possibly due to the small domain shift between CIFAR-10 and CIFAR-10.1. Instead, *ReC-TTT* can capture this small domain shift thanks to more robust training, thus achieving better performances, 5.5% higher than ClusT3 and around 3% better than NC-TTT. To be consistent with the other experiments, we report the performance with 20 adaptation iterations obtaining a gain of 0.27% of AUROC score, while without adaptation our model achieves an even better AUROC of 90.18 (see Section 4.2.5). A common limitation of TTT methods is that they are subject to high variability. To investigate this aspect, we repeated the experiments three times with different random seeds (Table 1 reports the average among three runs). The results show that the performance of TTT++ can vary by ±5.05, and those of ClusT3 by ±2.62, whereas *ReC-TTT* yields more consistent results with smaller variations (i.e., ±1.18).

CIFAR-100 and TinyImageNet-C corruptions. For the sake of the generability, the hyper-parameters used in the training of *ReC-TTT* for CIFAR-100C and TinyImageNet-C are the same as those used for CIFAR-10C with the only difference that, for CIFAR-100C, the best results are obtained when all the layers are trainable. Figure 2 shows that *ReC-TTT* again outperforms the other techniques on both datasets, with a gain of 29.47% when compared to the baseline for CIFAR-100C and a gain of 14.46% for TinyImageNet-C. This demonstrates the robustness of our method to adaptation settings involving a large number of classes.

VisDA. When training on VisDA, *ReC-TTT* achieves the best performance when all the layers of the encoders are

[1]Due to reproducibility issue, TTT++ results were extracted from the latest reported results [9].

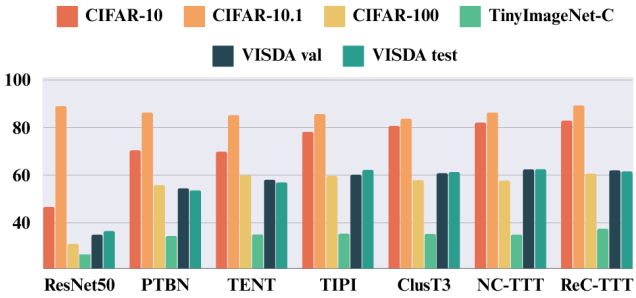


Figure 2. Quantitative results, compared to the state-of-the-art, on the CIFAR TinyImageNet-C and VISDA datasets (%). A detailed report for CIFAR-100, TinyImageNet and VISDA is provided in supplementary material.

trainable, and with 20 iterations of adaptation. Figure 2 also reports the results for $train \rightarrow val$ and $train \rightarrow test$ for VISDA. *ReC-TTT* performs better than all other approaches in $train \rightarrow val$ except NC-TTT while TIPI and NC-TTT show the best results for $train \rightarrow test$. It is worth mentioning that $train \rightarrow val$ and $train \rightarrow test$ model the same synthetic-to-real domain shift, but using two sources of real images with different characteristics. For instance, the ratio of images for each class varies greatly, and images from the test set are obtained from video frame crops and may thus be blurry, etc.

4.2.2 Visualization of the adaptation

To better understand the effect of adaptation, we consider Figure 3 showing the t-SNE plots of the test features before adaptation and after different numbers of iterations for two corruptions types: Brightness and Contrast. In the top row (Brightness), we can observe that *ReC-TTT* obtains a good separation of the features for the different classes also without iterations (AUROC of 92.31). Successive iterations further separate the clusters but have a marginal impact on performance (AUROC of 94.03 at iteration 20). The results are different in the bottom row (Contrast). In this case, without adaptation, most features overlap, without a clear separation, and, indeed, *ReC-TTT* reaches an AUROC of 48.14. With successive iterations, the cluster separation improves (AUROC of 89.56 at iteration 20) thus demonstrating the effectiveness of our adaptation technique on the extracted features. On the other hand, for a limited number of samples that are wrongly classified before the adaptation, the distance of these samples to the true class increases with the number of iterations.

4.2.3 Robustenss to smaller batch sizes

As demonstrated in [18], most domain adaptation approaches suffer from the need for large batch sizes to achieve competitive results. Most methods are usually evaluated with batches that have a size of 128 or more. This is a limitation in the application in which it is not possible to collect large batches before computing the inference. For this reason, we compared the performances using different batch sizes (8, 32, 64, 128) for the CIFAR-10C dataset. Table 2 reports the results of this study showing that most of the SOTA approaches lose up to 7% of AUROC when the number of samples is lower than the number of available classes (a performance degradation is also reported for TIPI in [18]). In contrast, the performance loss of *ReC-TTT* is less than 2% even with the smallest batch size.

4.2.4 Which layers to adapt?

Previous studies suggest that the selection of layers that are updated by the auxiliary task at test time can affect performance [9, 13, 15]. Table 3 reports the results of the adaptation of different layers on CIFAR-10C, having the best performance when updating only the first three ResNet blocks. While the difference in performance between three or four trainable layers is negligible (+0.17%), adapting the first two layers yields a reduction in performance of 2.3%, and using the first layer only results in a drop of performance of 18.13%. Differently, for CIFAR-100C and VISDA, we obtained the best results by updating all four layers of the trainable encoders. This can be explained by the greater challenge posed by these datasets, i.e., the larger number of classes for CIFAR-100C and the harder synth-to-real domain shift for VISDA, requiring adaptation of features in deeper layers.

4.2.5 Number of adaptation iterations

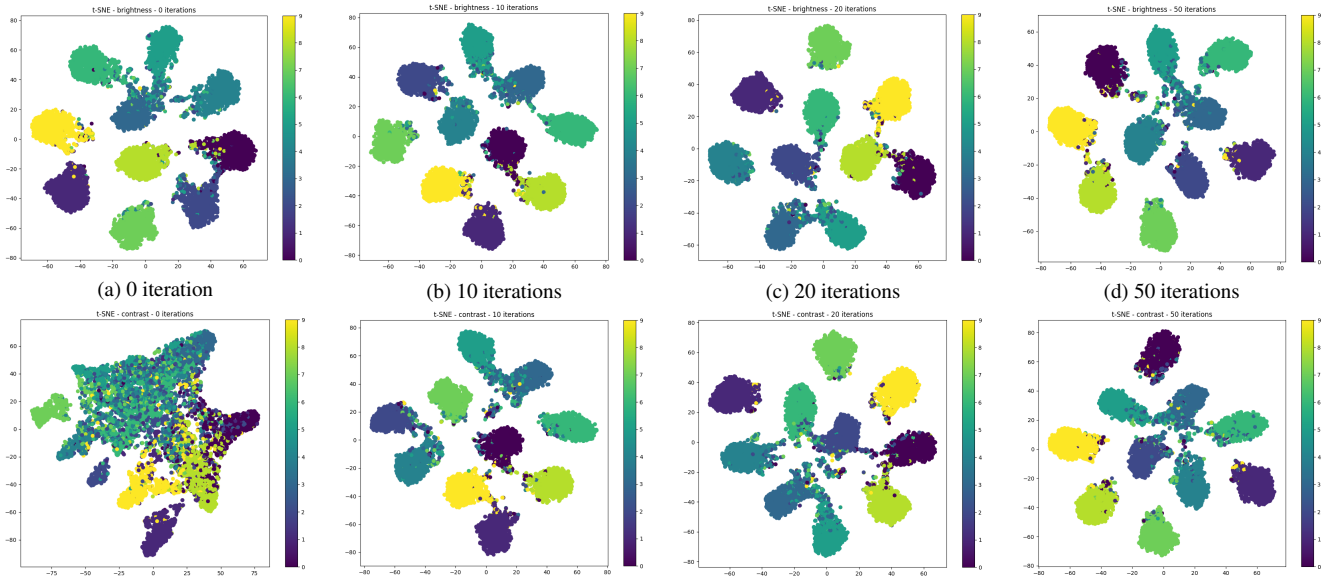
Another important aspect of test-time adaptation is the number of iterations needed at test time to obtain the best results. In line with previous studies [9, 20] Figure 4 shows, for the corruption types of CIFAR-10C, that the best results are obtained after 20 iterations. Successive iterations do not yield better results, on average. The same image for CIFAR-100 corruptions can be found in supplementary material. The same finding emerges from the other datasets with the only exception of CIFAR-10.1 where, as per previous experiments [9, 19, 20], adaptation tends to degrade the performances (90.18% of AUROC without adaptation, 89.27% of AUROC after 20 iterations).

4.2.6 Impact of removing the second trainable encoder

We evaluated the effectiveness of *ReC-TTT*'s ensemble learning strategy, which employs two trainable encoders, by comparing it with the base architecture with a single encoder. Table 4 shows the performance for three different

Corruption Type	**ResNet50**	**PTBN**	**TENT** (ICLR20)	**TTT++** (NeurIPS21)	**TIPI** (CVPR23)	**ClusT3** (ICCV23)	**NC-TTT** (CVPR24)	***ReC-TTT***
Gaussian Noise	23.65	57.49	57.67	75.87 ±5.05	71.90	**75.81** ±2.62	75.24 ±0.12	71.97 ±1.18
Shot Noise	27.68	61.07	60.82	77.18 ±1.36	78.24	77.32 ±2.14	**77.84** ±0.15	75.44 ±1.02
Impulse Noise	32.00	54.92	54.95	**70.47** ±2.18	59.64	67.97 ±2.78	68.77 ±0.15	69.28 ±0.27
Defocus Blur	38.73	82.23	81.39	86.02 ±1.35	84.67	88.10 ±0.20	88.22 ±0.04	**89.56 ±0.18**
Glass Blur	36.49	53.91	53.45	69.98 ±1.62	67.62	60.47 ±1.72	**70.19** ±0.18	69.38 ±0.73
Motion Blur	49.85	78.38	78.13	85.93 ±0.24	82.39	84.99 ±0.49	86.82 ±0.10	**88.94 ±0.03**
Zoom Blur	44.58	80.87	80.56	88.88 ±0.95	85.01	86.76 ±0.29	88.36 ±0.10	**89.65 ±0.27**
Snow	65.39	72.06	71.46	82.24 ±1.69	80.68	81.46 ±0.39	84.42 ±0.07	**86.75 ±0.44**
Frost	48.55	68.68	68.81	82.74 ±1.63	82.12	80.73 ±1.25	84.80 ±0.06	**86.83 ±0.59**
Fog	58.81	76.32	75.94	84.16 ±0.28	76.05	82.52 ±0.25	86.81 ±0.12	**88.87 ±0.33**
Brightness	84.72	85.38	84.87	89.97 ±1.20	88.96	91.52 ±0.24	92.52 ±0.04	**94.03 ±0.24**
Contrast	25.38	81.27	80.65	86.60 ±1.39	76.49	82.59 ±0.92	87.84 ±0.11	**89.56 ±0.48**
Elastic Transform	60.90	67.76	67.21	78.46 ±1.83	77.25	80.04 ±0.35	80.23 ±0.06	**81.66 ±0.32**
Pixelate	39.25	69.59	69.22	82.53 ±2.01	82.67	81.69 ±0.58	81.93 ±0.22	**82.13 ±0.34**
JPEG Compression	64.96	66.50	66.17	81.76 ±1.58	79.39	**81.58** ±1.18	78.49 ±0.09	79.69 ±0.12
Average	46.73	70.43	69.93	81.46	78.21	80.67	82.17	**82.92**
CIFAR 10.1	89.00	86.40	85.30	88.03	85.70	83.77	86.40	**89.27**

Table 1. Performance comparison with state-of-the-art on CIFAR-10C and CIFAR10.1 (%).



(a) 0 iteration (b) 10 iterations (c) 20 iterations (d) 50 iterations

(e) 0 iteration (f) 10 iterations (g) 20 iterations (h) 50 iterations

Figure 3. t-SNE plot of features after different adaptation iterations (0, 10, 20, 50) for the *Brightness* (top row) and *Contrast* (bottom row) corruptions of CIFAR-10C. The adaptation at test time helps separate the features of examples from the same class (represented by color).

corruptions and the average among all 15 corruptions available in CIFAR-10C. We observe that using two encoders performs better than having a single one in all cases. This demonstrates the effectiveness of our ensemble learning approach to stabilize training and provide a more robust prediction. To address the increased computational complexity introduced by our model, we investigated the impact of using only one encoder during inference when constrained by performance requirements. While this approach results in a slight reduction in performance, it still yields better re-

Batch size	PTBN	TENT	ClusT3	NC-TTT	*ReC-TTT*
8	61.97	62.11	73.73	79.75	**81.68**
32	68.46	68.46	80.14	81.80	**82.54**
64	69.45	69.54	80.38	82.01	**82.84**
128	70.43	70.09	80.67	82.43	**82.92**

Table 2. **Robustness to the batch size.** Qualitative performance of different approaches on CIFAR-10C (%) for different bacth sizes used during training.

Trainable layers	Impulse Noise	Brightness	Pixelate	Average
1 layer	16.12	93.26	34.83	64.79
2 layers	61.86	94.00	76.46	80.62
3 layers	69.25	94.06	82.03	**82.92**
4 layers	69.56	93.78	81.57	82.75

Table 3. **On the impact of the training different layers.** Performance comparison of training different layers of our approach on CIFAR-10C (%).

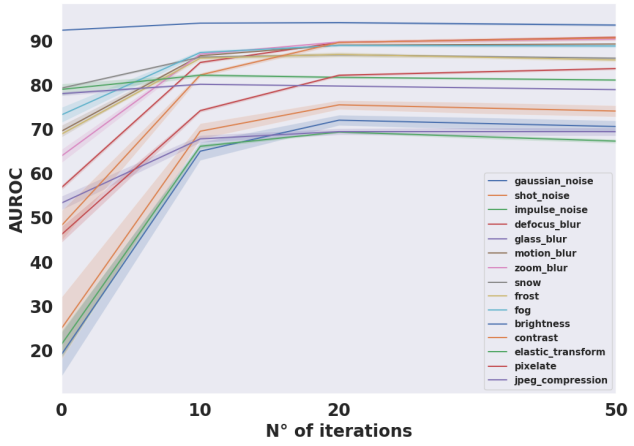


Figure 4. **How many iterations are needed for adaptation?** Performance (AUROC) obtained by our method with different number of adaptation iterations on CIFAR-10C. For most corruption types, our method provides a significant boost within few iterations and remains stable when the number of iterations is increased.

sults when compared to training the model without the ensemble architecture, remaining competitive with the other SOTA approaches.

5. Conclusion

Our work addressed the problem of domain shift between training and test data under the Test-Time Training framework. We presented *ReC-TTT*, a novel TTT approach based on contrastive feature reconstruction that can efficiently and effectively adapt the model to new unseen domains at test-time. Through a series of extensive experiments, we demonstrated that our model outperforms other state-of-the-art approaches across diverse datasets subject to different distribution shifts. An important limitation of previous approaches is their need for large batches of test samples to correctly adapt the model. Our results show that *ReC-TTT* is more robust to this factor, even when the number of classes is greater than the number of available samples. Furthermore, we highlight the robustness of our method against training variability, typically observed in current TTT approaches. Another key advantage of our approach is that it requires tuning few hyper-parameters at test-time, specifically, the layers to adapt, the learning rate, and the number of adaptation iterations. These results underscore the potential of our method in enhancing the applicability of TTT in various domains under challenging conditions.

	Impulse Noise	Brightness	Pixelate	Average
One encoder	65.19	93.17	80.36	81.07
Two encoders	69.28	94.03	82.13	**82.82**
One encoder (Inference)	67.54	93.31	80.89	81.59

Table 4. **Using one *vs.* two encoders.** Qualitative results on different configurations of our approach, on CIFAR-10C (%).

5.1. Limitations and future work

While the results presented in this work show how to tackle the domain-shift problem with test-time training effectively, we have to consider certain limitations: as in all TTT approaches, one key limitation is the need for the source data to train the model. Moreover, *ReC-TTT* was evaluated using only ResNet50 as backbone architecture, however employing different feature extractors might potentially yield a better performance. The architecture introduces multiple encoders that might affect the computational and memory requirements during training and adaptation, however, during inference, these additional components are not required, mitigating the resource demands. Last, although we validated our method on three classic benchmarks, further evaluation should be performed on different datasets representing different domain shifts.

Acknowledgments:. This project was partially supported by TEMPO – Tight control of treatment efficacy with tElemedicine for an improved Management of Patients with hemOphilia project, funded by the Italian Ministry of University and Research, Progetti di Ricerca di Rilevante Interesse Nazionale (PRIN) Bando 2022 - grant [2022PKTW2B] and MUSA (Multilayered Urban Sustainability Action) project funded by the NextGeneration EU program. We also thank *Compute Canada*.

References

[1] Junbum Cha, Sanghyuk Chun, Kyungjae Lee, Han-Cheol Cho, Seunghyun Park, Yunsung Lee, and Sungrae Park. Swad: Domain generalization by seeking flat minima. *Advances in Neural Information Processing Systems*, 34:22405–22418, 2021. 1, 2

[2] Dian Chen, Dequan Wang, Trevor Darrell, and Sayna Ebrahimi. Contrastive test-time adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 295–305, 2022. 2

[3] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020. 1

[4] 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. 3

[5] Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. A downsampled variant of imagenet as an alternative to the cifar datasets. *arXiv preprint arXiv:1707.08819*, 2017. 5

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 5

[7] Yossi Gandelsman, Yu Sun, Xinlei Chen, and Alexei Efros. Test-time training with masked autoencoders. *Advances in Neural Information Processing Systems*, 35:29374–29385, 2022. 2, 3

[8] Jia Guo, Lize Jia, Weihang Zhang, Huiqi Li, et al. Recontrast: Domain-specific anomaly detection via contrastive reconstruction. *Advances in Neural Information Processing Systems*, 36, 2024. 1, 2, 3

[9] Gustavo A Vargas Hakim, David Osowiechi, Mehrdad Noori, Milad Cheraghalikhani, Ali Bahri, Ismail Ben Ayed, and Christian Desrosiers. Clust3: Information invariant test-time training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6136–6145, 2023. 1, 2, 3, 5, 6

[10] 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. 1

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 5

[12] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019. 5

[13] Yoonho Lee, Annie S Chen, Fahim Tajwar, Ananya Kumar, Huaxiu Yao, Percy Liang, and Chelsea Finn. Surgical fine-tuning improves adaptation to distribution shifts. *arXiv preprint arXiv:2210.11466*, 2022. 6

[14] 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, 2020. 1

[15] Yuejiang Liu, Parth Kothari, Bastien Van Delft, Baptiste Bellot-Gurlet, Taylor Mordan, and Alexandre Alahi. Ttt++: When does self-supervised test-time training fail or thrive? *Advances in Neural Information Processing Systems*, 34:21808–21820, 2021. 2, 3, 5, 6

[16] John P Miller, Rohan Taori, Aditi Raghunathan, Shiori Sagawa, Pang Wei Koh, Vaishaal Shankar, Percy Liang, Yair Carmon, and Ludwig Schmidt. Accuracy on the line: on the strong correlation between out-of-distribution and in-distribution generalization. In *International Conference on Machine Learning*, pages 7721–7735. PMLR, 2021. 1

[17] Zachary Nado, Shreyas Padhy, D Sculley, Alexander D'Amour, Balaji Lakshminarayanan, and Jasper Snoek. Evaluating prediction-time batch normalization for robustness under covariate shift. *arXiv preprint arXiv:2006.10963*, 2020. 2, 5

[18] A Tuan Nguyen, Thanh Nguyen-Tang, Ser-Nam Lim, and Philip HS Torr. Tipi: Test time adaptation with transformation invariance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24162–24171, 2023. 2, 5, 6

[19] David Osowiechi, Gustavo A Vargas Hakim, Mehrdad Noori, Milad Cheraghalikhani, Ali Bahri, Moslem Yazdanpanah, Ismail Ben Ayed, and Christian Desrosiers. Nc-ttt: A noise constrastive approach for test-time training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6078–6086, 2024. 2, 5, 6

[20] David Osowiechi, Gustavo A Vargas Hakim, Mehrdad Noori, Milad Cheraghalikhani, Ismail Ben Ayed, and Christian Desrosiers. Tttflow: Unsupervised test-time training with normalizing flow. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2126–2134, 2023. 2, 6

[21] Xingchao Peng, Ben Usman, Neela Kaushik, Dequan Wang, Judy Hoffman, and Kate Saenko. Visda: A synthetic-to-real benchmark for visual domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 2021–2026, 2018. 5

[22] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 1

[23] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *International conference on machine learning*, pages 5389–5400. PMLR, 2019. 5

[24] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In *Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part IV 11*, pages 213–226. Springer, 2010. 2

[25] 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 *International conference on machine learning*, pages 9229–9248. PMLR, 2020. 1, 2

[26] Antonio Torralba and Alexei A Efros. Unbiased look at dataset bias. In *CVPR 2011*, pages 1521–1528. IEEE, 2011. 1

[27] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5018–5027, 2017. 1

[28] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. *arXiv preprint arXiv:2006.10726*, 2020. 2, 5

[29] Ziyi Zhang, Weikai Chen, Hui Cheng, Zhen Li, Siyuan Li, Liang Lin, and Guanbin Li. Divide and contrast: Source-free domain adaptation via adaptive contrastive learning. *Advances in Neural Information Processing Systems*, 35:5137–5149, 2022. 2

[30] Kaiyang Zhou, Ziwei Liu, Yu Qiao, Tao Xiang, and Chen Change Loy. Domain generalization: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. 2

[31] Kaiyang Zhou, Yongxin Yang, Yu Qiao, and Tao Xiang. Domain generalization with mixstyle. *arXiv preprint arXiv:2104.02008*, 2021. 1