# MT3: Meta Test-Time Training for Self-Supervised Test-Time Adaption

**Alexander Bartler** **Andre Bühler** **Felix Wiewel** **Mario Döbler** **Bin Yang**
Institute of Signal Processing and System Theory, University of Stuttgart, Germany

## Abstract

An unresolved problem in Deep Learning is the ability of neural networks to cope with domain shifts during test-time, imposed by commonly fixing network parameters after training. Our proposed method Meta Test-Time Training (MT3), however, breaks this paradigm and enables adaption at test-time. We combine meta-learning, self-supervision and test-time training to learn to adapt to unseen test distributions. By minimizing the self-supervised loss, we learn task-specific model parameters for different tasks. A meta-model is optimized such that its adaption to the different task-specific models leads to higher performance on those tasks. During test-time a single unlabeled image is sufficient to adapt the meta-model parameters. This is achieved by minimizing only the self-supervised loss component resulting in a better prediction for that image. Our approach significantly improves the state-of-the-art results on the CIFAR-10-Corrupted image classification benchmark.

## 1 INTRODUCTION

Deep neural networks have dramatically improved the results in a wide range of applications. However, after they are deployed the distribution of test data may be very different compared to the distribution of the training data. During testing, samples may be corrupted by, e.g., noise, different lighting conditions, or environmental changes such as snow or fog (see Figure 1). These corruptions and the resulting distribution shifts can cause a dramatic drop in performance (Azulay & Weiss, 2019; Hendrycks & Dietterich, 2019). Even truly unseen test images without a large distribution shift can harm the model performance (Recht, Roelofs, Schmidt, & Shankar, 2019). Adversarial perturbations are examples for an intentional distribution shift, that is not recognizable to humans, but also reduces model performance drastically.

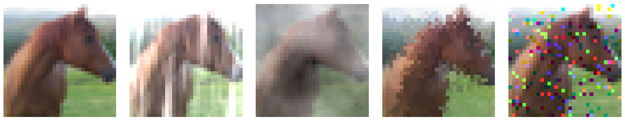

(a) orig (b) snow (c) fog (d) glass (e) impul

Figure 1: Example of a CIFAR-10 image with different corrupted versions of the most severe level taken from the CIFAR-10-Corrupted dataset.

To address changes in the test distribution and the resulting performance drop, recent work has mainly focused on the robustness to adversarial examples (Carlini & Wagner, 2017; Chen, Liu, et al., 2020; Dong et al., 2020; Jeddi, Shafiee, Karg, Scharfenberger, & Wong, 2020; Szegedy et al., 2013) or the generalization to out-of-distribution samples (Albuquerque, Naik, Li, Keskar, & Socher, 2020; Hendrycks et al., 2020; Krueger et al., 2020). Both areas aim to train a model in order to be robust against various types of unknown corruptions, distribution shifts, or domain shifts during testing. Another concept assumes that during training multiple unlabeled samples of the target domain are available and therefore unsupervised domain adaptation (UDA) can be performed (Tan et al., 2018; M. Wang & Deng, 2018; Wilson & Cook, 2020; Zhao et al., 2020). In the extreme case of one-shot UDA, only one unlabeled sample of the target domain is available during training (Luo, Liu, Guan, Yu, & Yang, 2020; Benaim & Wolf, 2018).

On the contrary, it is possible to account for distribution shifts only during test-time using a single test image under the assumption that the test image contains information about the distribution it originates from. Since the adaption to a single test sample is then

Proceedings of the 25th International Conference on Artificial Intelligence and Statistics (AISTATS) 2022, Valencia, Spain. PMLR: Volume 151. Copyright 2022 by the author(s).

performed by adapting the model at test-time, there is no need for any test data or information about the test distribution during the training stage. Additionally, in contrast to UDA, neither the original training data is needed for the adaption to a new test sample nor the test samples have to be drawn from the same distribution, since each test sample is processed individually. The assumption that each test sample can be corrupted differently could occur more likely in practice than having a persistent shift of the test distribution after deployment. For the concept of adaption during test-time, the model can be quickly adapted using only the sample itself, where in one-shot UDA the complete training dataset is additionally used to train a model on the target domain. If the target distribution is not stationary, the one-shot UDA has to be applied for each test sample individually, which would result in a tremendous testing complexity.

**Test-Time Training** The concept of adaption during test-time was first proposed by Sun et al. (2020) and is called Test-Time Training (TTT). In order to train a model which is able to adapt to unseen images, Sun et al. (2020) proposed to train the model using a supervised and a self-supervised loss jointly, denoted as joint training. During testing, only the self-supervised loss on one unlabeled test image is applied to adapt the model parameters which is hence called test-time training. After that, the refined model is used for inference. This test-time adaption is done for each test sample individually starting from the initially trained model.

The used architecture has two heads and a shared feature extractor. One head is used for the supervised downstream task, e.g., for classification the minimization of the categorical cross-entropy loss. The second head enables self-supervised learning. It solves a simple auxiliary task of rotation prediction, where four different rotation angles have to be predicted in a four-way classification problem (Gidaris, Singh, & Komodakis, 2018).

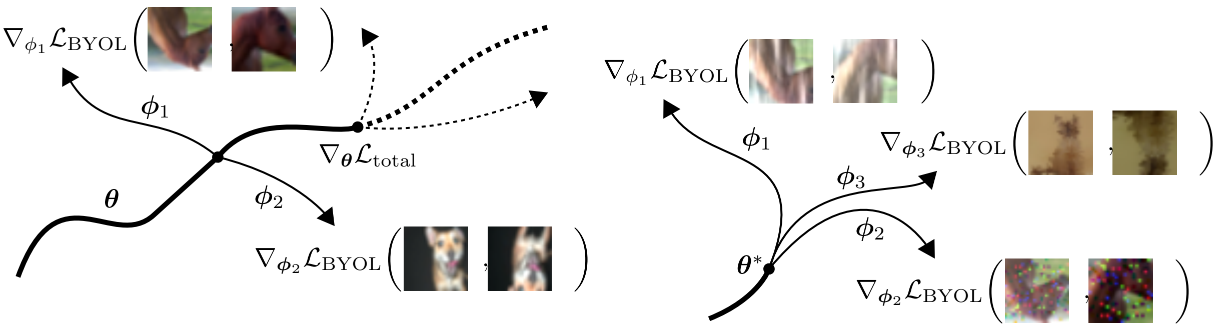
During testing, a batch of augmented views of a single image is used to minimize only the self-supervised loss subsequently to adapt the shared feature extractor while keeping the head for the supervised downstream task unchanged. The adapted model is used to make the prediction of the test sample. In addition, Sun et al. (2020) showed under strong assumptions that minimizing the self-supervised loss during testing implicitly minimizes the supervised loss.

While for the standard test procedure the adapted model is only used for a single image, the authors proposed an online setting where the model parameters are adapted sequentially during testing with a stream of test samples of a stationary or gradually changing test distribution. This online setting can be seen as online unsupervised domain adaption. Another recent approach for online test-time adaption is built upon entropy minimization (D. Wang, Shelhamer, Liu, Olshausen, & Darrell, 2020).

**Self-Supervised Learning** In the work of Sun et al. (2020) the rather simple auxiliary task of rotation prediction (Gidaris et al., 2018) is used for self-supervision. Recent state-of-the-art approaches for representation learning, on the other hand, rely on contrastive learning (Chen, Kornblith, Norouzi, & Hinton, 2020; Chen, Kornblith, Swersky, Norouzi, & Hinton, 2020; He, Fan, Wu, Xie, & Girshick, 2020; van den Oord, Li, & Vinyals, 2018). The key idea of contrastive learning is to jointly maximize the similarity of representations of augmented views of the same image while minimizing the similarity of representations of other samples, so called negatives. Another state-of-the-art technique for self-supervised representation learning is called *Bootstrap Your Own Latent* (BYOL) (Grill et al., 2020). Compared to contrastive losses, the main advantage of BYOL is that there is no need for negative samples. This makes BYOL suitable for test-time training since there is only a single image available during test-time.

BYOL consists of two neural networks, the online and target model. Both networks predict a representation of two different augmented views of the same image. The online network is optimized such that both the online and target predictions of the two augmented views are as similar as possible. This is realized by minimizing the mean squared euclidean distance of both $l_2$-normalized predictions. The parameters of the target network are updated simultaneously using an exponential moving average of the online network parameters.

**Meta-Learning** Another concept of adapting to unknown tasks or distributions is meta-learning, which is used in many state-of-the art results, e.g., for supervised few-shot learning (Antoniou, Edwards, & Storkey, 2018; Finn, Abbeel, & Levine, 2017; Hospedales, Antoniou, Micaelli, & Storkey, 2020; Li, Zhou, Chen, & Li, 2017; Nichol, Achiam, & Schulman, 2018) or unsupervised few-short learning (Hsu, Levine, & Finn, 2018; Khodadadeh, Bölöni, & Shah, 2018). Meta-learning has also shown its flexibility in the work of Metz, Maheswaranathan, Cheung, and Sohl-Dickstein (2018) where an unsupervised update rule is learned which can be used for pre-training a network in order to get powerful representations of unknown data distributions. In the work of Balaji, Sankaranarayanan, and Chellappa (2018), meta-learning is used to train models that generalize well to unknown


$\nabla_{\phi_1}\mathcal{L}_{\text{BYOL}}$
$\phi_1$
$\nabla_{\theta}\mathcal{L}_{\text{total}}$
$\theta$
$\phi_2$
$\nabla_{\phi_2}\mathcal{L}_{\text{BYOL}}$
$\nabla_{\phi_1}\mathcal{L}_{\text{BYOL}}$
$\phi_1$
$\nabla_{\phi_3}\mathcal{L}_{\text{BYOL}}$
$\phi_3$
$\phi_2$
$\theta^*$
$\nabla_{\phi_2}\mathcal{L}_{\text{BYOL}}$

(a) meta-training

(b) test-time adaption

Figure 2: MT3 training and test-time adaption: (a) In the outer loop the meta-parameters $\boldsymbol{\theta}$ are updated to minimize $\mathcal{L}_{\text{total}}$ which depends on multiple inner loop adaption steps. In the inner loop, different augmented views of the same image are used to adapt the meta-model to the task-specific model. (b) Test-time adaption to different corruptions starting from optimized meta-parameters $\boldsymbol{\theta}^*$.

domains. A widely used optimization based meta-learning algorithm is Model-Agnostic Meta-Learning (MAML) (Finn et al., 2017).

The main concept of MAML is to find the meta-model parameters $\boldsymbol{\theta}$ which can be adapted to a new task using a small number of samples and gradient steps. This means, it maximizes the sensitivity of the meta-model to changes in the task. In few-shot learning, tasks are defined as a set of new and unknown classes. During training, multiple tasks $\mathcal{T}_i$ are sampled from a distribution of tasks $P(\mathcal{T})$ and used to optimize the task-specific parameters $\boldsymbol{\phi}_i$ using a few gradient steps by minimizing a task-specific loss. This is often called the inner loop. The meta-parameters $\boldsymbol{\theta}$ are then optimized in the outer loop such that the adaption to $\boldsymbol{\phi}_i$ of each new task $\mathcal{T}_i$ maximizes the performance on that task. This results in an optimization over the gradient steps in inner loops, thus a second order optimization.

**Meta Test-Time Training (MT3)** In our work, we propose a novel combination of self-supervision and meta-learning to have the capability of adapting the model to unknown distributions during test-time. The combination of self-supervision and meta-learning has shown to be beneficial especially for few-short learning (Su, Maji, & Hariharan, 2020). In this work, a self-supervised and a supervised loss where jointly minimized by a meta-learner. In contrast to simply using joint training (Sun et al., 2020) or minimizing the sum of loss functions by meta-learning (Su et al., 2020), we propose to train the model such that it directly learns to adapt at test-time without supervision. We therefore train the meta-model, parameterized by $\boldsymbol{\theta}$, using a supervised and a slightly modified version of BYOL which are combined with MAML. During testing of a single sample, we start with the final meta-model parameters $\boldsymbol{\theta}^*$ and fine-tune them for each unlabeled test image to $\boldsymbol{\phi}^*$ using solely the self-supervised BYOL-like loss. The adapted model is in turn used for inference of that test image.

For training the meta-parameters in MT3, we define a batch of images as a task $\mathcal{T}_i$. The parameters $\boldsymbol{\theta}$ are transformed to $\boldsymbol{\phi}_i$ for each task $\mathcal{T}_i$ by minimizing the modified BYOL loss using two augmented versions of an unlabeled image. The meta-parameters are optimized such that the prediction of the updated model parameterized by $\boldsymbol{\phi}_i$ leads to a high performance for task $\mathcal{T}_i$. The optimization of $\boldsymbol{\theta}$ is performed over multiple tasks simultaneously as shown exemplarily in Figure 2 (a).

During testing, illustrated in Figure 2 (b), a batch of different augmented views of a single test sample defines a task for which we optimize the task-specific parameters with the BYOL-like loss in a self-supervised fashion using one or several gradient steps. This corresponds to the standard version of test-time training (Sun et al., 2020). The online setting of Sun et al. (2020) or D. Wang et al. (2020) is not considered further in our work. The optimized parameters $\boldsymbol{\phi}^*$ for a single sample are only used for the classification prediction of itself and are discarded afterwards. With this test-time adaption we aim for compensating the performance drop caused by unseen test distribution or distribution shifts.

Our contributions are as follows:

- We propose a novel combination of meta-learning and self-supervision which is able to adapt to unseen distribution shifts at test-time without supervision.
- We analyze MT3 and show that the combination

of meta-learning and BYOL achieves better performance than just joint training.

- Our method MT3 significantly outperforms the state-of-the-art in adapting to unseen test distribution shifts.

# 2 METHOD

**Tasks** In this work, the training dataset with $N$ input-output pairs is defined as $\mathcal{D}^{\text{train}} = \{\boldsymbol{x}^k, \boldsymbol{y}^k\}_{k=1}^N$ with inputs $\boldsymbol{x} \in \mathcal{X}$ and their corresponding class labels $\boldsymbol{y} \in \mathcal{Y}$. Each meta-training task $\mathcal{T}_i$ is associated to a batch of $K$ input-output pairs $\mathcal{D}_i^{\text{train}} = \{\boldsymbol{x}_i^k, \boldsymbol{y}_i^k\}_{k=1}^K$ uniformly sampled from $\mathcal{D}^{\text{train}}$. Each update of the meta-parameters is performed over a meta-batch which consists of $T$ tasks. In contrast to MAML, where the meta-objective is the adaption to new tasks, our meta-objective is to adapt the model to unknown data distribution shifts. Therefore, we do not sample the tasks $\mathcal{T}_i$ with respect to a different set of classes, but different distributions, which is further described in Section 2.2. During testing, a task $\mathcal{T}$ is defined as adapting the meta-model on $K_T$ augmented views of a single test sample $\boldsymbol{x}_{\text{test}}$.

**Architecture** Similar to previous work in representation learning (Chen, Kornblith, Norouzi, & Hinton, 2020; Chen, Kornblith, Swersky, et al., 2020; Grill et al., 2020), the overall architecture as shown in Figure 3 consists of a feature extractor $f$, a classification head $h$ for a supervised classification, a projector $p$ and a predictor $q$ for an auxiliary self-supervised task. The shared representation $\boldsymbol{g}$ will either be used for the classification prediction $\hat{\boldsymbol{y}}$ or to calculate the projection $\boldsymbol{z}$ and the prediction $\boldsymbol{r}$. As introduced by Chen, Kornblith, Norouzi, and Hinton (2020); Chen, Kornblith, Swersky, et al. (2020), the similarity in the self-supervised loss is calculated in the projection space $\boldsymbol{z}$, instead of the representation space $\boldsymbol{g}$. The meta-model is parameterized by the meta-parameters $\boldsymbol{\theta} = [\boldsymbol{\theta}_f, \boldsymbol{\theta}_h, \boldsymbol{\theta}_p, \boldsymbol{\theta}_q,]^T$. The task-specific parameters are denoted as $\boldsymbol{\phi} = [\boldsymbol{\phi}_f, \boldsymbol{\phi}_h, \boldsymbol{\phi}_p, \boldsymbol{\phi}_q,]^T$.

## 2.1 Meta Test-Time Training

Since the final model parameters $\boldsymbol{\theta}^*$ are adapted at test-time using a single unlabeled test sample $\boldsymbol{x}_{\text{test}}$, the sample-specific parameters $\boldsymbol{\phi}^*$ are then used for prediction. This test procedure breaks the classical learning paradigm where the model parameters are fixed at test-time.

In order to make the model adaptable at test-time, there are two crucial problems which need to be addressed. First, we need an unsupervised loss function which is used to adapt the model parameters at test-time, and second, the model has to be optimized during training such that adaptation during test-time results in a better classification performance.

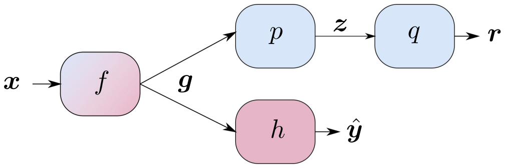


Figure 3: Used architecture in MT3 with shared feature extractor $f$, classification head $h$, and self-supervision head with projector $p$ and predictor $q$.

Following Sun et al. (2020), a self-supervised loss is minimized in order to update the model parameters at test-time. We make use of BYOL (Grill et al., 2020), since the rather simple self-supervised rotation loss used by Sun et al. (2020) can fail to provide enough information for adapting the model, specifically if the input sample is rotation invariant. Sun et al. (2020) further proposed to jointly minimize a supervised and a self-supervised loss during training. Although the authors have shown a correlation between both loss functions under strong assumptions, joint training may not lead to a quickly adaptable model for different self-supervised loss functions as will be shown in our experiments.

In contrast to this, we propose a novel training procedure to directly train a model such that it learns to adapt to unseen samples.

**Meta-Training** The goal of the meta-training phase is to find the meta-parameters $\boldsymbol{\theta}$ which are quickly adaptable to different unseen samples at test-time for achieving a more accurate classification under unknown distribution shifts.

During one meta-training outer loop step, the minimization of the self-supervised loss $\mathcal{L}_{\text{BYOL}}^{i,k}$ leads to the task-specific parameters $\boldsymbol{\phi}_i$ for each task $\mathcal{T}_i$ in the inner loop. The meta-parameters $\boldsymbol{\theta}$ are then optimized such that the optimization step to the task-specific parameters leads to high classification accuracy on these $T$ tasks $\mathcal{T}_i$.

For each task $\mathcal{T}_i$, two augmentations $\boldsymbol{a}_i^k, \tilde{\boldsymbol{a}}_i^k$ are generated from $\boldsymbol{x}_i^k$ using the sample augmentation $\mathcal{A}$ in order to calculate a variation of the BYOL loss as explained in detail in Section 2.2. To further enlarge the differences between the training tasks $\mathcal{T}_i$, a random batch augmentation $\mathcal{B}$ is applied to $\boldsymbol{x}_i, \boldsymbol{a}_i$ and $\tilde{\boldsymbol{a}}_i$. Note that the parameters of $\mathcal{B}$ are fixed for all $K$ images within one task and differ across tasks. Therefore, $\mathcal{B}$

artificially generates a distribution shift between tasks and and facilitates meta-learning.

To calculate our modified BYOL-like loss for each pair $\boldsymbol{a}_i^k$ and $\tilde{\boldsymbol{a}}_i^k$, the predictions $\boldsymbol{r}_{\boldsymbol{\phi}_i}^k$ and $\tilde{\boldsymbol{r}}_{\boldsymbol{\phi}_i}^k$ are calculated by the task-specific model parameterized by $\boldsymbol{\phi}_i$ and the projections $\boldsymbol{z}_{\boldsymbol{\theta}}^k$ and $\tilde{\boldsymbol{z}}_{\boldsymbol{\theta}}^k$ using the meta-model $\boldsymbol{\theta}$. This differs from the original idea of BYOL where the target model is parameterized by an exponential moving average (EMA) of the online model parameters. In our approach, the meta-model model can be regarded as a smooth version of our task-specific models and therefore a separate target model is obsolete. Our modified BYOL loss for optimizing the task-specific model is defined as

$$\mathcal{L}_{\text{BYOL}}^{i,k} = \bar{\mathcal{L}}(\boldsymbol{r}_{\boldsymbol{\phi}_i}^k, \tilde{\boldsymbol{z}}_{\boldsymbol{\theta}}^k) + \bar{\mathcal{L}}(\tilde{\boldsymbol{r}}_{\boldsymbol{\phi}_i}^k, \boldsymbol{z}_{\boldsymbol{\theta}}^k), \tag{1}$$

$$\text{with} \quad \bar{\mathcal{L}}(\boldsymbol{a}, \boldsymbol{b}) = 2 - 2 \cdot \frac{\boldsymbol{a}^T \boldsymbol{b}}{\|\boldsymbol{a}\|_2 \cdot \|\boldsymbol{b}\|_2}$$

denoting the squared $l_2$-norm of the difference between $l_2$-normalized versions of two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$. The first loss term at the right hand side of Eq. 1 measures the closeness of the prediction $\boldsymbol{r}_{\boldsymbol{\phi}_i}^k$ of the task-specific model to the projection $\tilde{\boldsymbol{z}}_{\boldsymbol{\theta}}^k$ of the meta-model. The second loss term symmetrizes the first one. Note that this loss is only differentiated with respect to the task-specific model parameters $\boldsymbol{\phi}_i$ excluding the classification head parameters $\boldsymbol{\phi}_h$. Hence, the $M$ update steps with the inner learning rate $\alpha$ are performed by

$$\boldsymbol{\phi}_i \leftarrow \boldsymbol{\phi}_i - \alpha \nabla_{\boldsymbol{\phi}_i} \frac{1}{K} \sum_k \mathcal{L}_{\text{BYOL}}^{i,k}, \tag{2}$$

where $\boldsymbol{\phi}_i$ is initialized with the meta-parameters $\boldsymbol{\theta}$.

Now, making use of all optimized task-specific parameters within a meta-batch, the classification predictions for each task $\hat{\boldsymbol{y}}_{\boldsymbol{\phi}_i}^k$ are calculated by the task-specific models parameterized by $\boldsymbol{\phi}_i$ and, in combination with $\boldsymbol{y}_i^k$, are used to optimize the meta-parameters by minimizing the cross-entropy loss $\mathcal{L}_{\text{CE}}(\hat{\boldsymbol{y}}_{\boldsymbol{\phi}_i}^k, \boldsymbol{y}_i^k)$. Additionally, the BYOL-like loss function weighted by $\gamma$ is minimized here since $\mathcal{L}_{\text{CE}}$ is not differentiable with respect to the parameter of the predictor $p$ and the projector $q$. The total loss function in the outer loop is defined as

$$\mathcal{L}_{\text{total}}^{i,k} = \mathcal{L}_{\text{CE}}^{i,k} + \gamma \cdot \mathcal{L}_{\text{BYOL}}^{i,k} \tag{3}$$

and is calculated using the task-specific parameters $\boldsymbol{\phi}_i$. The update of the meta-parameters $\boldsymbol{\theta}$ is done by

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \beta \nabla_{\boldsymbol{\theta}} \frac{1}{KT} \sum_i \sum_k \mathcal{L}_{\text{total}}^{i,k}, \tag{4}$$

where $\beta$ is the meta-learning rate. Note that the meta-gradient $\nabla_{\boldsymbol{\theta}}$ is a gradient over the optimization steps from $\boldsymbol{\theta}$ to every $\boldsymbol{\phi}_i$. The pseudo-code of the meta-training procedure is described in Algorithm 1.

**Algorithm 1** Meta-Training

```
Require: Training data $\mathcal{D}^{\text{train}}$, number of inner steps $M$, meta-batch size $T$, task batch size $K$, meta-learning rate $\beta$, inner learning rate $\alpha$, loss weight $\gamma$, sample augmentation $\mathcal{A}$, batch augmentation $\mathcal{B}$
while not converged do
    Sample $T$ tasks $\mathcal{T}_i$ each with batch size $K$
    for each $\mathcal{T}_i$ do
        Get augmentations $\boldsymbol{a}_i^k$ and $\tilde{\boldsymbol{a}}_i^k$ with $\mathcal{A}(\boldsymbol{x}_i^k)$
        Apply batch augmentation $\mathcal{B}$ to $\boldsymbol{x}_i^k, \boldsymbol{a}_i^k, \tilde{\boldsymbol{a}}_i^k$
        for $step = 1, 2, \ldots, M$ do
            Calculate $\boldsymbol{r}_{\boldsymbol{\phi}_i}^k, \tilde{\boldsymbol{r}}_{\boldsymbol{\phi}_i}^k, \boldsymbol{z}_{\boldsymbol{\theta}}^k, \tilde{\boldsymbol{z}}_{\boldsymbol{\theta}}^k$
            Optimize task-specific parameters:
            $\boldsymbol{\phi}_i \leftarrow \boldsymbol{\phi}_i - \alpha \nabla_{\boldsymbol{\phi}_i} \frac{1}{K} \sum_k \mathcal{L}_{\text{BYOL}}^{i,k}$    ▷ Eq. 1
        end for
    end for
    Update meta-parameters:
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \beta \nabla_{\boldsymbol{\theta}} \frac{1}{KT} \sum_i \sum_k \mathcal{L}_{\text{total}}^{i,k}$    ▷ Eq. 3
end while
Return: $\boldsymbol{\theta}^*$
```

**Algorithm 2** Test-Time Adaption

```
Require: Meta-model $\boldsymbol{\theta}^*$, test sample $\boldsymbol{x}_{\text{test}}$, number of steps $M$, test batch size $K_T$, learning rate $\alpha$, sample augmentation $\mathcal{A}$
for $step = 1, 2, \ldots, M$ do
    Initialize $\boldsymbol{\phi}$ with $\boldsymbol{\theta}^*$
    Copy $\boldsymbol{x}_{\text{test}}$ $K_T$ times
    Get $K_T$ pairs of augmentations $(\boldsymbol{a}^k, \tilde{\boldsymbol{a}}^k)$
    from $\boldsymbol{x}_{\text{test}}$ with $\mathcal{A}$
    Optimize task-specific model parameters:
    $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} - \alpha \nabla_{\boldsymbol{\phi}} \frac{1}{K_T} \sum_k \mathcal{L}_{\text{BYOL}}^{k}$    ▷ Eq. 1
end for
Get final classification prediction $\hat{\boldsymbol{y}}_{\text{test}}$
```

**Meta-Testing** At test-time, the optimized meta-model parameters $\boldsymbol{\theta}^*$ are adapted to a single test sample $\boldsymbol{x}_{\text{test}}$ using the self-supervised BYOL loss in Equation 1. Since only one sample is available during testing, an artificial batch is generated using $K_T$ different augmentation pairs of $\boldsymbol{x}_{\text{test}}$ by using the sample augmentation $\mathcal{A}$ to minimize the BYOL-like loss. Using the adapted model, the final classification prediction $\hat{\boldsymbol{y}}_{\text{test}}$ is performed. After a prediction, the adapted parameters $\boldsymbol{\phi}^*$ are discarded and we return back to the final meta-model parameters $\boldsymbol{\theta}^*$ for the next test sample. The pseudo-code for processing a single test sample is illustrated in Algorithm 2.

### 2.2 Implementation Details

**Architecture** We use a ResNet architecture (He, Zhang, Ren, & Sun, 2016) with 26 layers as our feature extractor $f$ with 32 initial filters for all of our experiments. Although the original implementation uses batch normalization (BN), we use group normalization (GN) (Wu & He, 2018) with 16 groups similar to Sun et al. (2020). The projector $p$ and predictor $q$ are each a two-layer MLP with 256 hidden neurons and output dimension of 128. The classifier $h$ shares the first hidden layer with the projector $p$ as proposed by Chen, Kornblith, Swersky, et al. (2020) followed by a 10-dimensional softmax activated output layer. We empirically found that using no GN in the projector and predictor improves performance.

**Augmentations** For the BYOL-like loss in Equation 1, the sample augmentation $\mathcal{A}$ generates two augmentations of one image. Similar to Chen, Kornblith, Norouzi, and Hinton (2020); Chen, Kornblith, Swersky, et al. (2020); Grill et al. (2020), we adjusted the random cropping for CIFAR-10 (uniform between 20 and 32 pixels) and resize back to the original image size of $32 \times 32$. In contrast to other approaches, we apply random vertical flipping with a probability of 50 %, since horizontal flipping is already used in the batch augmentation $\mathcal{B}$ and could be reversed if it is applied twice. Lastly, color jittering and color dropping are applied. We use the same types of color jittering as in (Grill et al., 2020) with the adapted strength of 0.2 compared to 1.0 for ImageNet (Chen, Kornblith, Swersky, et al., 2020). The color jittering is applied with a probability of 80 % and color dropping with a probability of 20 %.

Additionally, to simulate larger distribution shifts between tasks $\mathcal{T}_i$ during meta-training, batch augmentation $\mathcal{B}$ is applied to the complete batch $\{\boldsymbol{x}_i^k, \boldsymbol{a}_i^k, \tilde{\boldsymbol{a}}_i^k\}_{k=1}^K$. The parameters of $\mathcal{B}$ are randomly chosen for each task, but fixed for each image within the current task. Random horizontal flipping (50 % probability), Gaussian blurring (20 % probability) with a $3 \times 3$ filter with a standard deviation of 1.0, brightness adjustment (uniformly distributed delta between $-0.2$ and 0.2) and Gaussian noise with a uniformly distributed standard deviation between 0 and 0.02 are applied.

**Optimization** We use SGD for the meta-optimization with a fixed learning rate of $\beta = 0.01$ and a momentum of 0.9. The inner optimization is done using only one ($M = 1$) gradient step with a step size of $\alpha = 0.1$. During testing, we use the same fixed learning rate of $\alpha = 0.1$ and one gradient step since the same parameters are used during training. Weight decay is applied to the meta-model parameters with a strength of $1.5 \cdot 10^{-6}$. We set the weight of the BYOL loss to $\gamma = 0.1$. Gradient $l_2$-norm clipping with a clipping norm of 10 is applied to both the inner- and meta-gradient to stabilize the training (Finn et al., 2017). The meta-batch size is set to $T = 4$ and each task consists of $K = 8$ images. During test-time adaption, the batch size is set to $K_T = 32$. In all experiments the meta-model is trained for 200 epochs which takes approximately 48 hours on a single RTX 2080 Ti (11 GB). Note that the hyper-parameters are only chosen such that the training loss converges. No extensive hyper-parameter optimization was performed.

**Dataset** For training, the CIFAR-10 training dataset (Krizhevsky, Hinton, et al., 2009) is used. For evaluating the test-time training, we use the CIFAR-10-Corrupted dataset (Hendrycks & Dietterich, 2019). It consists of all 10,000 CIFAR-10 validation images with 15 different types of simulated corruptions for 5 different levels. All our results are reported for the most severe level 5. An example image with different corruptions is shown in Figure 1. The corruption types come from the four major categories noise, blur, weather, and digital. Exemplary subcategories are impulse noise, Gaussian blurring, frost, and JPEG compression.

# 3 EXPERIMENTS

In our experiments, we first analyze the training behavior of our method MT3[1] followed by a detailed analysis of the impact of meta-learning. For this, we compare to our own baseline and pure joint training (JT) without the meta-learning component. Finally, we compare our results with the state-of-the art method TTT (Sun et al., 2020). An overview of all methods is given in Table 1.

## 3.1 Ablation Studies

**Convergence of MT3** To show its stability and the ability of adaption, we evaluate the classification accuracy during training twice. First, we measure the classification accuracy of each task $\mathcal{T}_i$ with the meta-model parameters $\boldsymbol{\theta}$ before applying the self-supervised adaption. Second, we evaluate the model with the task-specific parameters $\boldsymbol{\phi}_i$ after the adaption in the inner loop. As shown in Figure 4, the training of MT3 leads to a stable convergence without large deviations. The small deviations, especially after adaption, highlight the reproducibility and stability of MT3. Furthermore, even at an early stage of training, MT3 learns to adapt such that the accuracy increases as shown by the large gap before and after the self-supervised adaption. This clearly shows that the learned meta-parameters $\boldsymbol{\theta}^*$ are

[1] A reference implementation is available on GitHub https://github.com/AlexanderBartler/MT3

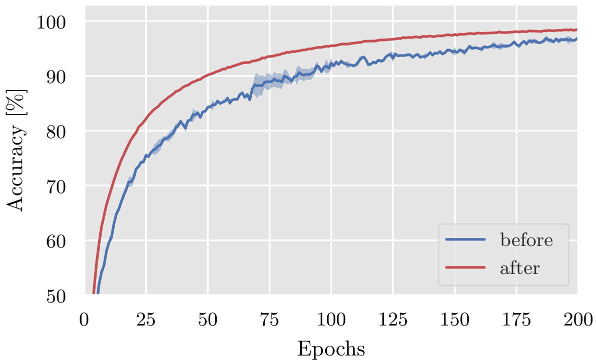


Figure 4: Training accuracy of tasks $\mathcal{T}_i$ before and after adaption to task-specific model parameters $\boldsymbol{\phi}_i$ over 3 runs.

| Method | self-superv. | $\mathcal{B}$ | training | test-time adaption |
|---|---|---|---|---|
| Baseline (Sun et al., 2020) | - | - | CE | - |
| JT (Sun et al., 2020) | rotation | - | joint | - |
| TTT (Sun et al., 2020) | rotation | - | joint | ✓ |
| Baseline (ours) | - | - | CE | - |
| JT (ours) | BYOL | ✓ | joint | - |
| TTT (ours) | BYOL | ✓ | joint | ✓ |
| MT (ours) | BYOL | ✓ | meta | - |
| MT3 (ours) | BYOL | ✓ | meta | ✓ |

Table 1: Component overview of all considered methods: Baseline, joint training (JT) and test-time training (TTT) with either cross-entropy (CE), rotation augmentation (Gidaris et al., 2018) or BYOL (Grill et al., 2020); usage of batch augmentation $\mathcal{B}$; and used training principles, joint training or meta training with or without test-time adaption.

able to be adapted with a single gradient step and image resulting in an improved classification accuracy.

**Baseline** In order to compare MT3 to classical supervised training, we choose the same architecture as described in Section 2.2 without the projector and predictor. This baseline model is simply trained by minimizing the cross-entropy loss. We use SGD with a fixed learning rate of 0.1 and a momentum of 0.9. The strength of weight decay is set to $5 \cdot 10^{-4}$. We train the baseline model for 200 epochs with a batch size of 128. We use the standard data augmentation protocol by padding with 4 pixels followed by random cropping to $32 \times 32$ pixels and random horizontal flipping (He et al., 2016; Lee, Xie, Gallagher, Zhang, & Tu, 2015). The hyper-parameters of the baseline training are optimized independently of other methods in order to have a fair comparison.

**Joint Training** To show the improvement caused by meta-learning, we compare MT3 to a second baseline, namely joint training (JT). We use exactly the same architecture as for MT3 and minimize the joint loss function similar to Equation 3 but without any inner step, i.e., without meta-learning. Additionally, we use the exponential moving average of the online model as the target model as originally proposed by Grill et al. (2020) with an update momentum of 0.996. The BYOL-like loss is weighted by $\gamma = 0.1$. For minimizing the joint loss function, we use SGD with a learning rate of 0.1 and a momentum of 0.9. The strength of weight decay is set to $1.5 \cdot 10^{-6}$. We train the model for 200 epochs with a batch size of 128. In order to have a fair comparison and to show the impact of meta-learning in MT3, we use the same data augmentation $\mathcal{A}$ for minimizing the BYOL-like loss. Furthermore, we use the same batch augmentation $\mathcal{B}$ to simulate distributions shifts here as well. The only major difference to MT3 is the use of joint training instead of meta-learning.

On the one hand, we use joint training to compare it to MT3 by fixing the learned model at test-time. On the other hand, similar to Sun et al. (2020), we adapt our jointly trained model at test-time using only the self-supervised loss (TTT). During test-time adaption, we use the same test-time parameters as for MT3 except the learning rate is set to 0.01 which is more comparable to the effective learning rate during joint training (due to $\gamma = 0.1$). The hyper-parameters of joint training are again optimized independently of other methods.

**Comparison to our baselines** We first compare our baseline and joint training without test-time adaption against our proposed method MT3. Additionally, we show the results of MT3 with fixed parameters at test-time (MT), thus without a gradient step at test-time. The results on the 15 corruption types of the CIFAR-10-Corrupted images are shown in Table 2 with their mean and standard deviation estimated over 3 runs. Furthermore, the average accuracy over all corruption types for each run is given by its mean and standard deviation. In case of TTT, the model parameters are adapted before the prediction of each single test image. The final accuracy is then calculated over the predictions of the 10,000 adapted models.

Our baseline model has on average the worst performance with an accuracy of 64.3 %. In comparison, our JT with stronger data augmentation and the utilization of the BYOL loss leads to a 9.7 % increase in accuracy achieving 74.0 %. Applying test-time training to our jointly trained model, the average accuracy drops down to 73.5 %, contrary to our expectations. Although joint training followed by test-time training is expected to help improving the result as shown in (Sun et al., 2020), we did not experience this in our case,

| | Baseline<br>(Sun et al., 2020) | JT<br>(Sun et al., 2020) | TTT<br>(Sun et al., 2020) | Baseline<br>(ours) | JT<br>(ours) | TTT<br>(ours) | MT<br>(ours) | MT3<br>(ours) |
|---|---|---|---|---|---|---|---|---|
| brit | 86.5 | 87.4 | **87.8** | 86.7 ± 0.44 | 86.5 ± 0.13 | 86.6 ± 0.26 | 84.3 ± 1.15 | 86.2 ± 0.47 |
| contr | 75.0 | 74.7 | 76.1 | 54.0 ± 6.42 | 75.4 ± 2.02 | 75.1 ± 2.38 | 69.3 ± 2.63 | **77.6 ± 1.21** |
| defoc | 76.3 | 75.8 | 78.2 | 68.1 ± 2.34 | 84.7 ± 0.11 | **84.7 ± 0.09** | 82.7 ± 1.33 | 84.4 ± 0.44 |
| elast | 72.6 | 76.0 | **77.4** | 74.3 ± 0.27 | 74.6 ± 0.80 | 74.4 ± 1.19 | 74.2 ± 1.08 | 76.3 ± 1.18 |
| fog | 71.9 | 72.5 | 74.9 | 70.7 ± 0.98 | 70.3 ± 0.86 | 70.4 ± 0.67 | 72.0 ± 1.03 | **75.9 ± 1.26** |
| frost | 65.6 | 67.5 | 70.0 | 65.2 ± 0.93 | 79.8 ± 0.62 | 79.5 ± 0.73 | 76.6 ± 1.16 | **81.2 ± 0.20** |
| gauss | 49.5 | 50.6 | 54.4 | 49.9 ± 3.17 | **71.7 ± 1.13** | 70.4 ± 1.08 | 63.6 ± 1.17 | 69.9 ± 0.34 |
| glass | 48.3 | 51.5 | 53.9 | 50.7 ± 2.96 | 62.8 ± 0.97 | 61.9 ± 1.10 | 62.8 ± 1.35 | **66.3 ± 1.24** |
| impul | 43.9 | 46.6 | 50.0 | 43.4 ± 4.31 | **59.3 ± 3.04** | 58.5 ± 3.17 | 50.3 ± 1.68 | 58.2 ± 1.25 |
| jpeg | 70.2 | 71.3 | 72.8 | 76.0 ± 0.86 | 78.6 ± 0.37 | **79.0 ± 0.44** | 75.2 ± 0.06 | 77.3 ± 0.26 |
| motn | 75.7 | 75.2 | 77.0 | 71.6 ± 0.46 | 70.7 ± 0.45 | 69.8 ± 0.46 | 72.6 ± 3.17 | **77.2 ± 2.37** |
| pixel | 44.2 | 48.4 | 52.8 | 60.1 ± 2.73 | 65.0 ± 0.32 | 62.1 ± 0.44 | 67.8 ± 5.13 | **72.4 ± 2.29** |
| shot | 52.8 | 54.7 | 58.2 | 52.3 ± 2.17 | **72.3 ± 1.36** | 71.0 ± 1.09 | 64.0 ± 1.24 | 70.5 ± 0.72 |
| snow | 74.4 | 75.0 | 76.1 | 74.5 ± 0.46 | 77.2 ± 0.58 | 77.2 ± 0.55 | 77.1 ± 0.51 | **79.8 ± 0.63** |
| zoom | 73.7 | 73.6 | 76.1 | 67.4 ± 1.70 | 81.6 ± 0.69 | **81.7 ± 0.66** | 78.7 ± 1.72 | 81.3 ± 0.58 |
| **avg.** | 65.4 | 66.7 | 69.0 | 64.3 ± 0.42 | 74.0 ± 0.77 | 73.5 ± 0.80 | 71.4 ± 0.42 | **75.6 ± 0.30** |

Table 2: Performance on the CIFAR-10-Corrupted dataset for MT3 compared to the results from (Sun et al., 2020), including their baseline, joint training with rotation loss (JT), and test-time adaption (TTT). Additionally, we report our results of MT3 without test-time adaption (MT), our baseline, joint training (JT), and test-time adaption (TTT) using BYOL. Mean and standard deviation are reported over 3 runs.

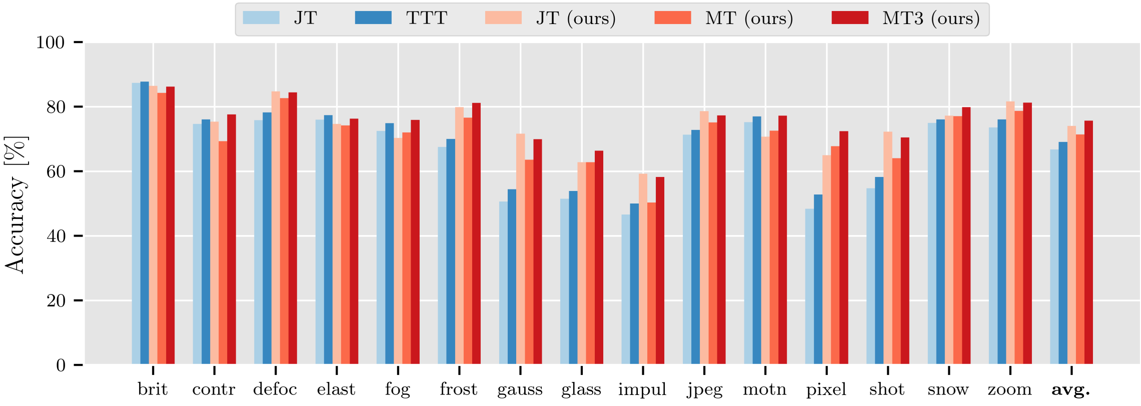


Figure 5: Performance on the CIFAR-10-Corrupted dataset for MT3 compared to MT3 without test-time adaption (MT), joint training with BYOL (JT (ours)), joint training with rotation loss (JT) (Sun et al., 2020), and test-time training (TTT) (Sun et al., 2020). Mean and standard deviation are reported over 3 runs.

where BYOL instead of a rotation loss is used. For some corruption types, e.g. jpeg compression (jpeg), a small improvement can be achieved with our TTT, but in 9 of 15 cases the test accuracy decreases with test-time training, e.g. for pixelate (pixel) by almost 3 %. In contrast, our method MT3 achieves a higher classification accuracy for all types of corruption after performing test-time adaption. MT3 raises the average accuracy of the meta-model from 71.4 % before to 75.6 % after adaption. Considering the average over all corruptions, MT3 has the lowest standard deviation, which highlights the stability and reproducibility of our method. Similar to JT, the results of the two corruption types Gaussian noise (gauss) and brightness (brit), which overlap with the applied batch augmentation $\mathcal{B}$, have improved compared to our baseline. The improvement on these datasets is mainly caused by the applied data augmentation and should therefore be handled carefully. Still, our method MT3 outperforms JT on average despite both methods using the same data augmentations.

In summary, the results suggest that our proposed method MT3 has learned during training to adapt at test-time, while joint training using BYOL combined with test-time adaption did not show that behavior. Furthermore, our analysis shows that the absolute improvement of MT3 is caused by meta-training and not only by using joint training with stronger data augmentation.

| | Baseline | JT | TTT | MT | MT3 |
|---|---|---|---|---|---|
| brit | **54.5 ± 0.79** | 53.2 ± 0.75 | 53.3 ± 0.79 | 51.7 ± 0.42 | 52.2 ± 0.44 |
| contr | 22.2 ± 1.60 | 29.0 ± 2.45 | 28.2 ± 2.49 | 28.7 ± 0.55 | **31.6 ± 1.53** |
| defoc | 37.9 ± 2.47 | 55.8 ± 0.32 | **55.9 ± 0.49** | 54.8 ± 0.62 | 55.0 ± 0.55 |
| elast | **45.0 ± 0.67** | 44.0 ± 0.90 | 44.0 ± 0.88 | 44.1 ± 0.61 | 44.2 ± 0.81 |
| fog | 30.8 ± 0.94 | 31.9 ± 1.31 | 32.0 ± 1.35 | 32.4 ± 0.27 | **33.3 ± 0.45** |
| frost | 31.3 ± 0.96 | 44.7 ± 0.78 | 44.5 ± 0.94 | 43.8 ± 0.90 | **45.5 ± 1.00** |
| gauss | 18.7 ± 2.86 | **32.8 ± 2.01** | 32.1 ± 1.94 | 30.7 ± 1.05 | 32.8 ± 0.84 |
| glass | 25.8 ± 1.21 | 30.5 ± 1.77 | 30.2 ± 1.95 | 31.7 ± 1.02 | **33.0 ± 0.93** |
| impul | 14.2 ± 2.33 | 17.1 ± 0.19 | 16.9 ± 0.26 | 17.3 ± 0.21 | **18.4 ± 0.09** |
| jpeg | **44.1 ± 1.51** | 43.1 ± 0.85 | 43.2 ± 0.83 | 42.4 ± 0.27 | 42.7 ± 0.46 |
| motn | 40.7 ± 2.08 | 44.8 ± 0.71 | 44.4 ± 0.74 | 44.4 ± 0.94 | **45.4 ± 0.81** |
| pixel | 28.1 ± 0.67 | 34.7 ± 0.59 | 33.2 ± 0.79 | 40.8 ± 1.86 | **41.2 ± 2.06** |
| shot | 20.0 ± 2.69 | 32.9 ± 1.52 | 32.2 ± 1.52 | 31.1 ± 1.28 | **33.1 ± 1.41** |
| snow | 38.4 ± 0.77 | 42.9 ± 0.85 | 42.7 ± 0.90 | 43.2 ± 0.62 | **43.7 ± 1.12** |
| zoom | 39.7 ± 2.10 | 53.8 ± 1.04 | **54.0 ± 1.04** | 52.2 ± 0.56 | 52.0 ± 0.62 |
| **avg.** | 32.8 ± 0.49 | 39.4 ± 0.42 | 39.1 ± 0.40 | 39.3 ± 0.37 | **40.3 ± 0.27** |

Table 3: Accuracy on the CIFAR-100-Corrupted dataset for MT3 compared to our baselines. Mean and standard deviation are reported over 3 runs.

### 3.2 Comparison with State-of-the Art

We compare our method to the state-of-the-art TTT (Sun et al., 2020) as shown in Table 2 and Figure 5. Besides our results, we discuss the baseline, joint training (JT) and joint training with test-time adaption (TTT) of Sun et al. (2020). The difference between all analyzed methods are shown in Table 1.

Our baseline as well as the baseline of Sun et al. (2020) have similar average performance over all corruption types. This highlights that both models have a comparable capacity or generalization capability and possible improvements are not caused by the model structure itself. In our work, joint training with the BYOL-like loss leads to a much higher average accuracy compared to the previous method where rotation classification as self-supervision was used. The large gap of 7.3 % might be caused by the stronger data augmentations or the use of BYOL in our method. Despite this, one important result is that simple joint training does not enable the ability to adapt at test-time in general. In the previous work of Sun et al. (2020), the adaption with only the self-supervised loss to a single test image using ten gradient steps leads on average to an improvement of 2.3 %. In comparison, test-time adaption for our jointly trained model using BYOL leads to an average degradation of 0.5 %. We also investigated the case of ten gradient steps at test-time, but found that on average the performance further degrades.

Our method MT3, on the other hand, shows the ability to adapt by a large improvement of 4.2 % before and after a single gradient step. Furthermore, the final average accuracy of 75.6 % over all corruption types is the best among all considered methods. For 7 out of 15 corruption types, MT3 has the highest accuracy compared to our baselines and previous work. This again highlights the ability of our method to adapt to unseen distribution shifts using a single gradient step during test-time.

### 3.3 CIFAR-100-Corrupted

To show the success and scalability of MT3, we evaluate our method on the more challenging CIFAR-100-Corrupted dataset (Krizhevsky et al., 2009; Hendrycks & Dietterich, 2019). Since Sun et al. (2020) did not evaluate this dataset and D. Wang et al. (2020) only for the online adaption, we only show the results compared to our baselines. We use the same hyperparameter as for CIFAR-10 except the test learning rate is lowered to $\alpha = 0.05$. As shown in Table 3, our method is also capable to learn to adapt on the more complex dataset CIFAR-100 with a similar behavior as for the CIFAR-10-Corrupted dataset.

## 4 CONCLUSION

We proposed a novel algorithm that allows to adapt to distribution shifts during test-time using a single sample. We show that our approach, based on meta-learning (MAML) and self-supervision (BYOL), effectively enables adaptability during test-time. In contrast to the previous work, where simply joint training was used, meta-learning has the explicit purpose to learn meta-parameters that can be rapidly adapted which we showed in our experiments. Our combination of meta-learning and self-supervision improves the average accuracy on the challenging CIFAR-10-Corrupted dataset by 6.6%, a 9.57% relative increase, compared to the state-of-the-art TTT.

## References

Albuquerque, I., Naik, N., Li, J., Keskar, N., & Socher, R. (2020). Improving out-of-distribution generalization via multi-task self-supervised pretraining. *arXiv preprint arXiv:2003.13525*.

Antoniou, A., Edwards, H., & Storkey, A. (2018). How to train your maml. *arXiv preprint arXiv:1810.09502*.

Azulay, A., & Weiss, Y. (2019). Why do deep convolutional networks generalize so poorly to small image transformations? *Journal of Machine Learning Research*, *20*(184), 1-25.

Balaji, Y., Sankaranarayanan, S., & Chellappa, R. (2018). Metareg: Towards domain generalization using meta-regularization. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 31). Curran Associates, Inc.

Benaim, S., & Wolf, L. (2018). One-shot unsupervised cross domain translation. In *Neurips.*

Carlini, N., & Wagner, D. (2017). Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th acm workshop on artificial intelligence and security* (pp. 3–14).

Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. (2020, 13–18 Jul). A simple framework for contrastive learning of visual representations. In H. D. III & A. Singh (Eds.), *Proceedings of the 37th international conference on machine learning* (Vol. 119, pp. 1597–1607). PMLR.

Chen, T., Kornblith, S., Swersky, K., Norouzi, M., & Hinton, G. E. (2020). Big self-supervised models are strong semi-supervised learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, & H. Lin (Eds.), *Advances in neural information processing systems* (Vol. 33, pp. 22243–22255). Curran Associates, Inc.

Chen, T., Liu, S., Chang, S., Cheng, Y., Amini, L., & Wang, Z. (2020, June). Adversarial robustness: From self-supervised pre-training to fine-tuning. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition (cvpr).*

Dong, Y., Fu, Q.-A., Yang, X., Pang, T., Su, H., Xiao, Z., & Zhu, J. (2020, June). Benchmarking adversarial robustness on image classification. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition (cvpr).*

Finn, C., Abbeel, P., & Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning* (pp. 1126–1135).

Gidaris, S., Singh, P., & Komodakis, N. (2018). Unsupervised representation learning by predicting image rotations. *CoRR*, *abs/1803.07728*.

Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P. H., Buchatskaya, E., ... others (2020). Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*.

He, K., Fan, H., Wu, Y., Xie, S., & Girshick, R. (2020). Momentum contrast for unsupervised visual representation learning. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 9729–9738).

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 770–778).

Hendrycks, D., Basart, S., Mu, N., Kadavath, S., Wang, F., Dorundo, E., ... others (2020). The many faces of robustness: A critical analysis of out-of-distribution generalization. *arXiv preprint arXiv:2006.16241*.

Hendrycks, D., & Dietterich, T. (2019). Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*.

Hospedales, T., Antoniou, A., Micaelli, P., & Storkey, A. (2020). Meta-learning in neural networks: A survey. *arXiv preprint arXiv:2004.05439*.

Hsu, K., Levine, S., & Finn, C. (2018). Unsupervised learning via meta-learning. *arXiv preprint arXiv:1810.02334*.

Jeddi, A., Shafiee, M. J., Karg, M., Scharfenberger, C., & Wong, A. (2020, June). Learn2perturb: An end-to-end feature perturbation learning to improve adversarial robustness. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition (cvpr).*

Khodadadeh, S., Bölöni, L., & Shah, M. (2018). Unsupervised meta-learning for few-shot image classification. *arXiv preprint arXiv:1811.11819*.

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.

Krueger, D., Caballero, E., Jacobsen, J.-H., Zhang, A., Binas, J., Zhang, D., ... Courville, A. (2020). Out-of-distribution generalization via risk extrapolation (rex). *arXiv preprint arXiv:2003.00688*.

Lee, C.-Y., Xie, S., Gallagher, P., Zhang, Z., & Tu, Z. (2015). Deeply-supervised nets. In *Artificial intelligence and statistics* (pp. 562–570).

Li, Z., Zhou, F., Chen, F., & Li, H. (2017). Meta-sgd: Learning to learn quickly for few-shot learning. *arXiv preprint arXiv:1707.09835*.

Luo, Y., Liu, P., Guan, T., Yu, J., & Yang, Y. (2020). Adversarial style mining for one-shot unsupervised domain adaptation. *arXiv preprint* *arXiv:2004.06042*.

Metz, L., Maheswaranathan, N., Cheung, B., & Sohl-Dickstein, J. (2018). Meta-learning update rules for unsupervised representation learning. *arXiv preprint arXiv:1804.00222*.

Nichol, A., Achiam, J., & Schulman, J. (2018). On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*.

Recht, B., Roelofs, R., Schmidt, L., & Shankar, V. (2019, 09–15 Jun). Do ImageNet classifiers generalize to ImageNet? In K. Chaudhuri & R. Salakhutdinov (Eds.), *Proceedings of the 36th international conference on machine learning* (Vol. 97, pp. 5389–5400). PMLR.

Su, J.-C., Maji, S., & Hariharan, B. (2020). When does self-supervision improve few-shot learning? In *European conference on computer vision* (pp. 645–666).

Sun, Y., Wang, X., Liu, Z., Miller, J., Efros, A., & Hardt, M. (2020). Test-time training with self-supervision for generalization under distribution shifts. In *International conference on machine learning* (pp. 9229–9248).

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2013). Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., & Liu, C. (2018). A survey on deep transfer learning. In *International conference on artificial neural networks* (pp. 270–279).

van den Oord, A., Li, Y., & Vinyals, O. (2018). Representation learning with contrastive predictive coding. *CoRR*, *abs/1807.03748*.

Wang, D., Shelhamer, E., Liu, S., Olshausen, B., & Darrell, T. (2020). Tent: Fully test-time adaptation by entropy minimization. *arXiv preprint arXiv:2006.10726*.

Wang, M., & Deng, W. (2018). Deep visual domain adaptation: A survey. *Neurocomputing*, *312*, 135–153.

Wilson, G., & Cook, D. J. (2020). A survey of unsupervised deep domain adaptation. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *11*(5), 1–46.

Wu, Y., & He, K. (2018). Group normalization. In *Proceedings of the european conference on computer vision (eccv)* (pp. 3–19).

Zhao, S., Yue, X., Zhang, S., Li, B., Zhao, H., Wu, B., ... others (2020). A review of single-source deep unsupervised visual domain adaptation. *IEEE Transactions on Neural Networks and Learning Systems*.