Test-Time Adaptation via Conjugate Pseudo-labels

Sachin Goyal[⋆1] **Mingjie Sun**[⋆1] **Aditi Raghunathan**[1] **Zico Kolter**[1,2]
[1]Carnegie Mellon University, [2]Bosch Center for AI
{sachingo, mingjies, raditi, zkolter}@cs.cmu.edu

Abstract

Test-time adaptation (TTA) refers to adapting neural networks to distribution shifts, with access to only the unlabeled test samples from the new domain at test-time. Prior TTA methods optimize over unsupervised objectives such as the entropy of model predictions in TENT [50], but it is unclear what exactly makes a good TTA loss. In this paper, we start by presenting a surprising phenomenon: if we attempt to *meta-learn* the "best" possible TTA loss over a wide class of functions, then we recover a function that is *remarkably* similar to (a temperature-scaled version of) the softmax-entropy employed by TENT. This only holds, however, if the classifier we are adapting is trained via cross-entropy loss; if the classifier is trained via squared loss, a different "best" TTA loss emerges. To explain this phenomenon, we analyze test-time adaptation through the lens of the training losses's *convex conjugate*. We show that under natural conditions, this (unsupervised) conjugate function can be viewed as a good local approximation to the original supervised loss and indeed, it recovers the "best" losses found by meta-learning. This leads to a generic recipe that can be used to find a good TTA loss for *any* given supervised training loss function of a general class. Empirically, our approach consistently dominates other TTA alternatives over a wide range of domain adaptation benchmarks. Our approach is particularly of interest when applied to classifiers trained with *novel* loss functions, e.g., the recently-proposed PolyLoss [25] function, where it differs substantially from (and outperforms) an entropy-based loss. Further, we show that our conjugate based approach can also be interpreted as a kind of self-training using a very specific soft label, which we refer to as the *conjugate pseudo-label*. Overall, our method provides a broad framework for better understanding and improving test-time adaptation. Code is available at `https://github.com/locuslab/tta_conjugate`.

1 Introduction

Modern deep networks perform exceeding well on new test inputs that are close to the training distribution. However, this performance dramatically decreases on test inputs drawn from a different distribution. While there is a large body of work on improving the robustness of models, most robust training methods are highly specialized to the setting they cater to. For e.g., they assume pre-specified perturbations, subpopulations, and spurious correlations, or access to unlabeled data from the target distribution, and most methods offer close to no improvement on general distribution shifts beyond what they were trained for [12, 21].

In practice, it is often cumbersome (or even impossible) to precisely characterize all possible distribution shifts a model could encounter and then train accordingly. Instead, a model already trained on some source data must be able to *adapt* at test-time to new inputs from a different domain. This setting of *test-time adaptation* (TTA) has gained interest in recent years [6, 47, 50, 54]. TTA is typically accomplished by updating the source model parameters via a few steps of optimization on an *unsupervised objective* involving the new test sample from the target distribution. The choice

⋆ Equal Contribution

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

of this unsupervised objective, which we call the TTA loss, dictates the success of the adaptation procedure. [47] uses a self-supervised objective on the test sample, [50] uses the entropy of model predictions, and several follow-ups have proposed variants or alternatives [40, 54]. However, it remains unclear as to how to choose or guide the selection of this TTA loss, and thus far the choice of these losses has remained largely heuristic in nature.

In this work, we begin by presenting a set of intriguing experiments where we attempt to *learn* the "best" TTA loss for a given source classifier and distribution shift. We parameterize the TTA loss by another neural network whose parameters are learnt via meta-learning [3, 9] where we differentiate through the adaptation process to find the TTA loss that achieves the best adaptation on distribution shifts. Surprisingly, we ultimately learn a TTA loss that looks *remarkably* similar to (a temperature-scaled version of) the softmax-entropy loss, which was already proposed by [50]. Why did we recover the commonly used softmax-entropy loss despite the fact that the procedure is capable of learning a very general class of losses and the meta-learning process could potentially *specialize* to both the source classifier and the distribution shift of interest? Furthermore, we find that this pattern only holds when the loss used to train the source classifier is cross-entropy loss; when a different loss such as squared loss is used instead, the meta-learning procedure recovers a TTA loss that itself looks more like a negative squared error, and is very different from the softmax-entropy loss (Section 3).

In order to explain this phenomenon, we propose to consider TTA through the lens of the *convex conjugate* function. Specifically, given a hypothesis function $h(x)$ and label y, several common losses (cross-entropy and the squared loss amongst them, but not limited to these) can be written in the form $\mathcal{L}(h(x), y) = f(h(x)) - y^T h(x)$ for some function f. In these cases, we show that "natural" TTA loss for such classifiers is precisely the (negation of) the convex conjugate evaluated at the gradient of h, $\mathcal{L}_{\text{TTA}}(x) = -f^*(\nabla f(h(x))$, where f^* is the convex conjugate of f. This framework not only recovers the results of our meta-learning experiments, but also justifies why some specific choices of TTA loss in the previous literature work well (e.g., this framework recovers TENT's choice of softmax-entropy for cross-entropy-trained classifier). Moreover, it also provides a broad framework for what the TTA loss should be when the source model is trained using various different loss functions (for example the recently-proposed PolyLoss [25, 29]) as is becoming increasingly common in machine learning. Further, we show that our proposed conjugate adaptation loss is in fact a kind of self-training with pseudo-labels [42], a classic approach in machine learning. Various formulations of the pseudo-label have been proposed in the literature, and our conjugate analysis provides a general recipe for the "correct" choice of soft pseudo-labels given by $\hat{y}(x) = \nabla f(h(x))$. We thus refer to these as *conjugate pseudo-labels* (Conjugate PL's), and believe our work provides a broad framework for understanding adaptation with unlabeled data in general.

Finally, we empirically verify the effectiveness of our proposed conjugate adaptation loss across several datasets and training losses, such as cross-entropy and squared loss, along with the recently-proposed PolyLoss [25] (which itself has shown higher standard test accuracy on a wide range of vision tasks). Over *all* models, datasets and training losses, we find our proposed conjugate pseudo-labeling consistently outperforms prior TTA losses and improves TTA performance over the current state of the art.

2 Background and preliminaries.

Test-time adaptation. We are interested in mapping an input $x \in \mathbb{R}^d$ to a label $y \in \mathcal{Y}$. We learn a model $h_\theta : \mathbb{R}^d \mapsto \mathbb{R}^{|\mathcal{Y}|}$ parameterized by θ that maps an input x to predictions $h_\theta(x)$. We assume access to a trained source model and adapt at test-time over the test input, before making the final prediction. This is the standard test-time adaptation (TTA) setting [47, 50]. During TTA, we update the model parameters on an unsupervised objective $\mathcal{L}(x, h_\theta)$. For example, in TENT [50], this loss is the entropy of the softmax-normalized predictions of the model. At each time step of adaptation, we observe a batch of test inputs and we take a gradient step towards optimizing the TTA loss on this test batch. As is standard, we measure the average online performance of models across all steps (number of test batch inputs seen) in the adaptation process.

Meta learning the loss function. In order to explore the existence of different TTA losses, we employ the meta-learning procedure where we attempt to *learn* the TTA loss. We use a similar procedure as prior work on meta-learning loss functions [3, 37] and parameterize the loss function via a neural network $m_\phi : \mathbb{R}^{|\mathcal{Y}|} \mapsto \mathbb{R}$ that takes in the model predictions/logits and outputs a loss value. We want to learn parameter ϕ such that when we update θ via the loss function m_ϕ, our final

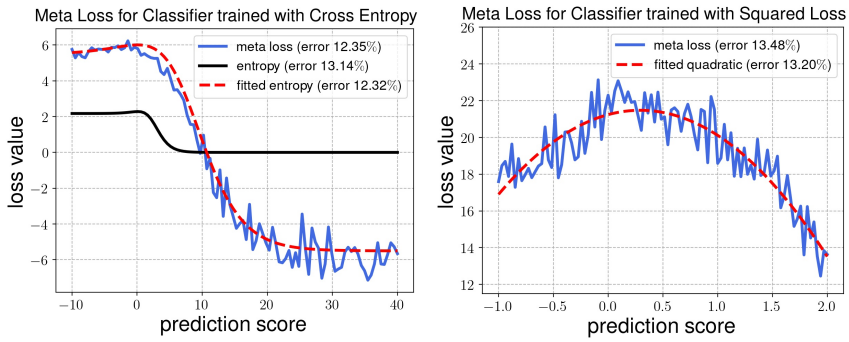
performance is optimal. In order to do so, let x be the unlabeled test samples to adapt to, and y be the corresponding labels. We update θ and ϕ alternatively as follows.

$$\theta^{t+1} \leftarrow \theta^t - \alpha \frac{\partial m_{\phi^t}(h_{\theta^t}(x))}{\partial \theta^t}, \quad \phi^{t+1} \leftarrow \phi^t - \beta \frac{\partial \mathcal{L}(h_{\theta^{t+1}}(x'), y')}{\partial \phi^t}, \tag{1}$$

where $\mathcal{L}$ is some supervised surrogate loss function such as cross-entropy. Please refer to Appendix A3 for further details regarding meta-learning setup. Note that the meta-learning process above assumes access to labels y of test inputs. In this paper, we do *not* propose meta-learning the TTA loss as an approach. Rather, we use meta-learning to explore what the "best" TTA losses look like. We discuss our findings from this exploration in the next section.

3 Test-time Adaptation via Meta-Learnt Losses

The objective used in TENT is the softmax-entropy of the model predictions which essentially makes the classifier more confident in its current predictions. The same can be achieved by various other loss formulations such as those mentioned in [40]. With so many possible choices for the loss function, what should we use for TTA? In this section, we attempt to answer this empirically and present some intriguing observations.



(a) (b)

Figure 1: Visualization of meta loss (blue) by varying one input prediction score. (a) For cross-entropy loss trained model, the learnt meta loss can be approximated with a scaled softmax-entropy function (dashed red). (b) When the source model is trained with a squared loss for classification, the learnt meta loss (blue) can be fitted closely with a quadratic function (dashed red), shown in Figure 1b. The range (max/min) of the prediction score (logit) in x-axis is chosen to cover the empirical range of the predicted logits.

Experiment 1. We learn the TTA loss parameterized by a neural network via meta-learning as described in Section 2. Our source classifier is a ResNet-26 trained on CIFAR-10 and we adapt to distribution shifts in CIFAR-10-C. We use the 4 labeled validation noises in CIFAR-10-C to learn the meta-loss network parameters and we denote the resulting learnt loss function by meta-TTA loss. We then adapt the source classifier to the test set of 15 corruptions by optimizing the meta-TTA loss.

Observations. First, we find that TTA using meta-TTA loss performs better than TENT (12.35% vs 13.14%), suggesting that there are better TTA losses than previous losses based on softmax-entropy.

However, on examining this meta-TTA loss, we find a surprising observation. Figure 1a (blue curve) visualizes the learnt meta-loss over model predictions as we vary a single class prediction with the rest fixed. Qualitatively, the learnt meta-loss looks very similar to softmax-entropy in one dimension. In fact, we can fit it closely with a scaled softmax-entropy function (dashed red curve): $\alpha \cdot \mathcal{H}(\text{softmax}(h_\theta(x)/T))$, where α is a magnitude parameter and T is a temperature scaler. We want to test if the meta-loss is basically learning the softmax-entropy function. Hence, we perform test-time adaptation with the fitted softmax-entropy function instead (dashed red curve) and achieve an error of 12.32%, essentially recovering the performance of meta-TTA.

Despite the ability to represent many different loss functions and potentially specialize to the CIFAR-10-C setting, the meta-loss procedure gave back the standard entropy objective. *Do we always recover a loss that looks like softmax-entropy?*

Experiment 2. In an attempt to isolate when we get back the entropy objective, we vary several things. We tried different architectures for the source classifier, different losses $\mathcal{L}$ during the meta-learning process (1) and different training losses for the source classifier.

Results. We observed that we consistently recovered the temperature scaled softmax-entropy function in all cases *except* when we varied the training loss for the source classifier (Appendix A.10). On using the squared loss function [18], a strikingly different meta-TTA loss emerges. Figure 1b (blue curve) shows the learnt meta-loss (13.48% error) for this network. Here again, the meta-TTA loss outperforms entropy (14.57%) but it is not simply due to a scaling factor. The loss now looks like the negative squared error (red curve). Like previously, we tried fitting a quadratic loss directly to the meta loss in Figure 1b, and this time we even slightly outperformed the meta-TTA loss.

To summarize, we used a meta-learning procedure to search for the "best" TTA loss, where the loss itself was parameterized by a neural network that could potentially represent arbitrarily complex loss functions. However, we ended up with loss functions displaying remarkable structure: across different architectures and different variants of meta-learning, for a classifier trained with cross-entropy, the meta-TTA loss was temperature scaled softmax-entropy and for a classifier trained with squared loss, the meta-TTA loss was a negative squared loss. This is interesting from both a practical and conceptual standpoint where the "best" TTA loss depends on the loss used to train the source classifier in a clean fashion. We attempt to understand and explain this phenomenon in the next section.

4 Conjugate Pseudo Labels

Results in the previous section raise an obvious question: *why* does softmax-entropy as used in TENT seem to be the "best" possible test time adaptation loss for classifiers trained via cross-entropy (at least, best in the sense that meta-learning consistently recovers something which essentially mimics softmax-entropy, even though meta-loss is parameterized by a neural network and hence could learn much more complex functions specific to the model and the particular shift)? And why, alternatively, does a quadratic TTA loss seem to perform best when the classifier is trained via squared loss?

In this section, we offer an explanation of this phenomenon via the construct of the convex conjugate function [1]. As we will see, our method recovers softmax-entropy *and* quadratic loss as the "natural" objectives for classifiers trained via cross-entropy and squared loss respectively. Furthermore, for classifiers trained via *other* loss functions, as is becoming increasingly common in deep learning, our approach naturally suggests corresponding test-time adaptation losses, which we show in the next section to comparatively outperform alternatives. Thus, we argue that our framework overall provides a compelling recipe for specifying the "correct" method for TTA for a large class of possible losses.

4.1 Losses and the convex conjugate

We begin by formally considering loss functions between a hypothesis output $h_\theta(x)$ (e.g., the logit outputs of a classifier, or the direct prediction of a regressor) and target y that take the following form

$$\mathcal{L}(h_\theta(x), y) = f(h_\theta(x)) - y^T h_\theta(x) \tag{2}$$

for some function f; when there is no risk of confusion, we will use h in place of $h_\theta(x)$ for simplicity of notation. While not every loss can be expressed in such a form, this captures a wide variety of common losses (possibly scaled by a constant value). For example, cross-entropy loss corresponds to the choice $f(h) = \log \sum_i \exp(h_i)$ and where y denotes a one-hot encoding of the class label; similarly, squared loss corresponds to the choice $f(h) = \frac{1}{2}\|h\|_2^2$.

When training an over-parameterized classifier, we can roughly view the training process as (approximately) attaining the *minimum* over hypotheses h for each training example

$$\min_\theta \frac{1}{t} \sum_{i=1}^{t} \mathcal{L}(h_\theta(x_i), y_i) \approx \frac{1}{t} \sum_{i=1}^{t} \min_h \mathcal{L}(h, y_i) \tag{3}$$

where t is the number of training samples. However, in the case of losses in the form (2), the minimization over h in this form represents a very specific and well-known optimization problem: it is known as the *convex conjugate* [1] of the function f

$$\min_h \mathcal{L}(h, y) = \min_h \{f(h) - y^T h\} = -f^\star(y) \tag{4}$$

where $f^\star$ denotes the convex conjugate of f. $f^\star$ is a convex function in y (and indeed, is convex regardless of whether or not f is convex). Furthermore, for the case that f is convex differentiable, the optimality condition of this minimization problem is given by $\nabla f(h^{\text{opt}}) = y$, so we also have that

$$f^\star(y) = f^\star(\nabla f(h^{\text{opt}})) \tag{5}$$

where h^{opt} refers to the optimal classifier (used interchangeably with $h_{\theta^{\text{opt}}}$). Putting this all together, we can state (admittedly, in a rather informal manner) that under the assumption that θ^{opt} is chosen so as to approximately minimize the empirical loss on the source data in the over-parameterized setting, we have that for t inputs

$$\frac{1}{t}\sum_{i=1}^{t} \mathcal{L}(h_{\theta^{\text{opt}}}(x_i), y_i) \approx \frac{1}{t}\sum_{i=1}^{t} -f^\star(\nabla f(h_{\theta^{\text{opt}}}(x_i))) \tag{6}$$

i.e., the empirical loss can be approximated by the (negative) conjugate applied to the gradient of the f, at least in a region close to the optimal θ^{opt} that minimizes the empirical loss. But the later expression has the notable benefit that it does not require any label y_i in order to compute the loss, and thus can be used as a basis for TTA on target domain of the hypothesis function $h_{\theta^{\text{opt}}}$.

Definition 1 (conjugate adaptation loss) *Consider a loss function that takes the form given in 2, used for training a hypothesis h_θ in the over-parameterized regime. We define the conjugate adaptation loss $\mathcal{L}^{conj}(h_\theta(x)) : \mathbb{R}^{|\mathcal{Y}|} \mapsto \mathbb{R}$ as follows.*

$$\mathcal{L}^{conj}(h_\theta(x)) = -f^\star(\nabla f(h_\theta(x))) = f(h_\theta(x)) - \nabla f(h_\theta(x))^\top h_\theta(x). \tag{7}$$

4.2 Recovery of existing test-time adaptation strategies

Cross-entropy The interesting aspect to this formalism is that when applied to classifiers trained with cross-entropy, it recovers exactly the TENT approach to TTA : minimizing the softmax-entropy of $h_\theta(x)$. And indeed, this loss was *also* recovered when using meta-learning to learn the "optimal" test-time adaptation loss. To see this, note that for cross-entropy, we have that $f(h) = \log \sum_i \exp(h_i)$, giving the optimality condition $y = \nabla f(h^{\text{opt}}) = \frac{\exp(h^{\text{opt}})}{\sum_i \exp(h_i^{\text{opt}})}$ and the conjugate function

$$f^\star(y) = \begin{cases} \sum_i y_i \log y_i & \text{if } \sum_i y_i = 1 \\ \infty & \text{otherwise} \end{cases}. \tag{8}$$

In other words,

$$\mathcal{L}^{\text{conj}}(h_\theta(x)) = -f^\star(\nabla f(h_\theta(x))) = -\sum_i \frac{\exp(h_i)}{\sum_j \exp(h_j)} \log \frac{\exp(h_i)}{\sum_j \exp(h_j)} \tag{9}$$

i.e. softmax-entropy of the model prediction, which is exactly the TTA loss that TENT uses.

Squared loss For the squared loss, we have that $f(h) = \frac{1}{2}\|h\|_2^2$, leading to the optimality condition $y = h$ and conjugate function $f^\star(y) = \frac{1}{2}\|y\|_2^2$. Hence, the adaptation loss in this case would be simply given by $\mathcal{L}^{\text{conj}}(h_\theta(x)) = -f^\star(\nabla f(h_\theta(x))) = -\frac{1}{2}\|h\|_2^2$ which is also what we observed in the meta-learning experiments discussed in Section 3.

4.3 Conjugate pseudo-labels

We now emphasize that by the nature of our approximations, there is an additional simple interpretation of the conjugate loss: it is also equal to the original loss (2) applied to the "psuedo-labels" $\tilde{y}_\theta^{\text{CPL}}(x) = \nabla f(h_\theta(x))$, where CPL refers to conjugate pseudo-labels, i.e.,

$$\mathcal{L}^{\text{conj}}(h_\theta(x)) = -f^\star(\nabla f(h_\theta(x))) = f(h_\theta(x)) - \nabla f(h_\theta(x))^T h_\theta(x) = \mathcal{L}(h_\theta(x), \nabla f(h_\theta(x))). \tag{10}$$

This property is known as the Fenchel-Young inequality, that is $f(x) + f^\star(u) \geq x^T u$ holding with equality when $u = \nabla f(x)$. In other words, our conjugate adaptation loss is precisely equivalent to self-training under the specific soft pseudo-labels given by $\tilde{y}^{\text{CPL}} = \nabla f(h_\theta(x))$. And indeed, for many cases, this may be a more convenient form to compute than explicitly computing the conjugate function at all. For this reason, we refer to our method as that of *conjugate pseudo-labels*.

In the case of cross-entropy loss, this approach then corresponds exactly to self-training using labels given by the softmax applied to the current hypothesis. We must emphasize, however, that while our conjugate formulation indeed has this "simple" form for the case of cross-entropy loss, the real advantage comes in that it provides the "correct" pseudo-label for use with *other* losses, which may result in pseudo-labels different from the "common" softmax operation.

Example: conjugate pseudo-labels for PolyLoss. PolyLoss [25] is a recently-proposed simple alternative to cross-entropy loss than has been shown to improve performance across a wide variety of compute tasks. This loss is given by the form

$$\mathcal{L}_{\text{poly}}(h_\theta(x), y) = \mathcal{L}_{\text{ce}}(h_\theta(x), y) + \epsilon \cdot y^T(1 - \text{softmax}(h_\theta(x))) \tag{11}$$

We note that this can be put exactly into our conjugate form (equation 2) by writing the loss in a slightly more involved fashion, which we refer to as the *expanded conjugate form*

$$\mathcal{L}_{\text{poly}}(h_\theta(x), y) = f(h_\theta(x)) - y^T g(h_\theta(x)). \tag{12}$$

where f is the log-sum-exp function as before, and $g(h) = h - \epsilon(1 - \text{softmax}(h))$. In order to formally put this into the form of the previous loss function (equation 2), we can simply define an alternative hypothesis as the function $h'_\theta(x) = g(h_\theta(x))$, and then define PolyLoss in the conjugate form as

$$\mathcal{L}_{\text{poly}}(h'_\theta(x), y) = f(g^{-1}(h'_\theta(x))) - y^T h'_\theta(x). \tag{13}$$

Typically, however, it is easier to simply operate on the expanded conjugate form, which yields the optimality condition for the pseudo-label $\nabla f(h^{\text{opt}}) = \mathsf{D}g(h^{\text{opt}})\tilde{y}^{\text{CPL}}_\theta(x)$, where D is the Jacobian operator. For the case of PolyLoss, this leads to the conjugate pseudo-label of the following form: $\tilde{y}^{\text{CPL}}_\theta(x) = (I + \epsilon \text{diag}(z) - \epsilon z z^T)^{-1} z, \quad z \equiv \text{softmax}(h_\theta(x))$.

Test-time adaptation. Finally, we note that the above discussion doesn't actually address any topics related to test-time adaptation to OOD data, but merely provides a generic characterization of a self-training procedure for generic loss functions of the form (2). However, the application to TTA on OOD data is fairly straightforward: as long as the learnt source parameters θ is a reasonable approximation to the true optimal θ^{opt} on the shifted domain, self-training with the conjugate pseudo-labels provides a reasonable proxy for fine-tuning the network on the true OOD loss. We emphasize that, common to most approaches for TTA , there are still some amount of design decisions that must be put in place; these are detailed in Section 5.1. In practice, we observe OOD generalization typically benefits (across all baselines) from an additional "temperature" scaling, i.e., applying the TTA loss to $h_\theta(x)/T$ for some fixed temperature T, although it requires a held-out validation dataset for tuning T. However, we should emphasize that truly unsupervised TTA would require making an informed guess for the value of these hyper-parameters. The full procedure for test time adaptation via conjugate pseudo-labels is shown in Algorithm 1.

Algorithm 1 Conjugate pseudo-labeling (Conjugate PL)

Input: Source classifier θ_0 trained using loss $\mathcal{L}(h_\theta(x), y) = f(h_\theta(x)) - h_\theta(x)^\top y$.
N batches of test data $D_{\text{test}} = [x_1, x_2, \ldots, x_N]$
Hyperparams: learning rate η and temperature T.

Let $\bar{h}_\theta(x) \stackrel{\text{def}}{=} h_\theta(x)/T$ be the temperature scaled predictor.

Let $\tilde{y}^{\text{CPL}}_\theta(x)$ denote the conjugate pseudo-label function $\tilde{y}^{\text{CPL}}_\theta(x) = \nabla(f(\bar{h}_\theta(x)))$.

for $n = 0, 1, \ldots N-1$ **do**

$\theta_{n+1} = \theta_n - \eta \nabla \mathcal{L}\Big(\bar{h}_\theta(x_n), \tilde{y}^{\text{CPL}}_\theta(x_n)\Big)$ [Self-training with conjugate pseudo-labels]

5 Experiments

In this section, we empirically evaluate the effectiveness and generality of the proposed conjugate pseudo-labeling procedure (Algorithm 1) for test-time adaptation on a variety of datasets.

5.1 Setup

Datasets. We evaluate on the three common corruption benchmarks: adapting a classifier trained on CIFAR-10 to CIFAR-10-C, CIFAR-100 to CIFAR-100-C and ImageNet to ImageNet-C [15]. Following the previous works [47, 50], we report the error averaged across corruptions at the highest severity for CIFAR-10/100-C and averaged across corruptions and severity level for ImageNet-C. We also evaluate on three domain adaptation datasets: adapting a classifier trained on SVHN to MNIST, an ImageNet classifier to ImageNet-R [16] and adapting from synthetic to real data in VISDA-C [38].

Models and Training losses. Following previous works on TTA[47, 50], we use ResNet-26 [14] as the source classifier architecture for CIFAR-10/100 experiments, ResNet-18 for SVHN to MNIST and a ResNet-50 for ImageNet and source synthetic data on VisDA-C. We consider source classifiers trained via the following loss functions: the de-facto cross-entropy, recently proposed polyloss [25] and squared loss [18].

Baselines. Our proposed conjugate pseudo-label is the classic approach of self-training with a specific form of pseudo-labels. In self-training, we replace the label y with a pseudo-label $\tilde{y}(x)$ and adapt by optimizing the loss function $\mathcal{L}(h_\theta(x), \tilde{y}(x))$. Note that we could either instantaneously update the pseudo-labels using the current classifier, or generate pseudo-labels once with just the source classifier. Instantaneous updates have been shown to work better for domain adaptation [7, 40], and we perform instantaneous updates for all methods. While we propose using $\tilde{y}^{\text{CPL}}(x) = \nabla f(h_\theta(x))$ (See Section 4.3), we compare to the standard pseudo-labels used in the literature:

- (i) the "hard" pseudo-label (hard PL) where $\tilde{y}(x) = \arg\max_i \big(h_\theta(x)\big)_i$ is the most likely class as predicted by h_θ. As is common in the self-training literature, we perform confidence thresholding.
- (ii) The "soft" pseudo-label (soft PL) where $\tilde{y}(x)$ is obtained by applying a softmax function to the model predictions $h_\theta(x)$.

We also compare with the following recently proposed test-time adaptation methods.

- Entropy Minimization (ENT) [50] minimizes the entropy of model predictions.
- Robust Pseudo-Label [40] where we minimize a robust classification loss, $\mathcal{L}_{\text{rpl}} = q^{-1}(1 - p(i|x)^q)$ where $i = \text{argmax}_j p(j|x)$ and $q \in [0, 1]$.
- MEMO [54] minimizes entropy of a model's outputs across different augmentations of a test input. We implement a *batch version*, where we see multiple test points at once, for fair comparisons.

TTA methodology. Following [50] and [40], we fine-tune by updating the learnable scale and shift parameters of the batch normalization layers across all adaptation losses. For each batch, batch normalization statistics is also updated, as suggested in [41]. We report performance at the end of one round of test-time adaptation over the entire test set.

We tune the learning rate (LR) and temperature (T) on the validation noises in the corruption benchmark by grid-search. LR is selected from $\{1e^{-1}, 1e^{-2}, \ldots 1e^{-4}\}$ and T from $\{1, 2 \ldots 5\}$. All the experiments have been performed on A6000 GPU's. On domain adaptation benchmarks, where there is no held-out target domain, we set T to be 1 and use the LR suggested by [6, 50]. We use the same hyperparameter tuning protocol across all methods. We single out temperature as a very important hyperparameter, as we discuss in the results below.

5.2 Results on classifiers trained with cross-entropy

We study the effectiveness of our proposed conjugate pseudo-labels when the source classifier is trained via cross-entropy loss. In this case, baselines Softmax PL and ENT are the same as Conjugate PL. Thus we omit them in our results. Table 1, reports the performance of various TTA methods.

When the source classifier is trained via cross-entropy, our conjugate pseudo-label algorithm exactly corresponds to entropy minimization with an additional temperature scaling. Entropy minimization as

Dataset	Temperature (T)	Hard PL	Robust PL	MEMO	Conjugate PL (**ENT**)
CIFAR-10-C	✗	13.95 (±0.06)	13.97 (±0.04)	**12.60** (±0.04)	13.07 (±0.05)
	✓	13.95 (±0.06)	12.85 (±0.04)	**12.51** (±0.01)	**12.51** (±0.03)
CIFAR-100-C	✗	45.22 (±0.4)	39.80 (±0.18)	**38.52** (±0.16)	41.15 (±0.25)
	✓	45.22 (±0.4)	36.37 (±0.10)	37.38 (±0.06)	**36.10** (±0.07)
ImageNet-C	✗	**45.43** (±0.05)	45.68 (±0.01)	48.91(±0.03)	45.82(±0.01)
	✓	45.43 (±0.05)	45.61 (±0.01)	48.91(±0.04)	**45.36**(±0.01)

Table 1: Mean errors when adapting to corruptions using a source classifier trained via cross-entropy loss. Here, conjugate pseudo-labeling becomes softmax-entropy minimization. With the right temperature scaling, softmax-entropy minimization matches or outperforms other approaches. Prior reported gains of other methods over softmax-entropy minimization disappear when we use temperature scaling. For additional context, the source classifier errors without adaptation are: CIFAR-10-C (29.54%), CIFAR-100-C (62.26%), ImageNet-C (61.89%)

proposed in prior work [50] does not tune the temperature parameter, and some newer objectives such as robust PL or MEMO outperform vanilla entropy minimization. For example, on CIFAR-100-C, vanilla ENT obtaines 41.15% average error, while robust PL improves this to 39.80% and MEMO to 38.52%. However, with the right temperature scaling, entropy minimization obtains 36.10% error which outperforms the newer objectives (with and without temperature scaling). A similar observation holds for CIFAR-10-C and ImageNet-C as well. Essentially, the gains over vanilla entropy minimization vanish when we do temperature scaling, and entropy minimization (i.e. conjugate pseudo-labeling corresponding to cross-entropy) turns out to be the best objective after all.

5.3 Results on classifiers trained with polyloss and squared loss

In the case of cross-entropy, conjugate pseudo-labeling reduces to the familiar notion of entropy minimization. We now explore the performance of our method on different loss functions where the conjugate pseudo-labels differ substantially from entropy minimization (section 4.3). Table 2 presents the results on the corruption benchmarks and Table 3 presents the results on the other domain adaptation datasets for source classifiers trained with PolyLoss.

Dataset	T	Hard PL	Robust PL	ENT	MEMO	Softmax PL	Conjugate PL (**Ours**)
CIFAR-10-C	✗	13.81(±0.12)	14.23(±0.02)	13.46(±0.06)	13.23(±0.07)	14.64(±0.11)	**13.02**(±0.09)
	✓	13.81(±0.12)	12.45(±0.05)	12.23(±0.06)	12.33(±0.04)	12.26(±0.04)	**12.08**(±0.05)
CIFAR-100-C	✗	40.47(±0.05)	42.86(±0.11)	40.12(±0.08)	39.90(±0.05)	41.00(±0.11)	**38.17**(±0.17)
	✓	40.47(±0.05)	39.80(±0.08)	38.23(±0.05)	39.23(±0.04)	37.04(±0.06)	**36.83**(±0.08)
ImageNet-C	✗	45.44(±0.21)	46.27(±0.03)	46.10(±0.03)	48.21(±0.05)	44.63(±0.03)	**44.01**(±0.01)
	✓	45.44(±0.21)	46.27(±0.03)	45.50(±0.02)	48.21(±0.04)	44.45(±0.03)	**44.01**(±0.01)

Table 2: Mean errors when adapting to corruptions using a source classifier trained via recently proposed Poly-1 Loss [25]. Conjugate pseudo-labeling consistently outperforms all previous approaches. For additional context, source classifier errors without adaptation : CIFAR-10-C (30.22%), CIFAR-100-C (63.91%) and ImageNet-C (62.18%).

First, we note that, across all datasets in Table 2 and Table 3, our conjugate PL approach outperforms all other TTA losses. With polyloss classifiers, entropy minimization is no longer the best method—on CIFAR-100-C, entropy minimization achieves 38.23% error while our conjugate PL achieves 36.83%. We see similar *consistent gains* on CIFAR-10-C, ImageNet-C, ImageNet-R and VisDA-C. On digit adaptation tasks from SVHN to MNIST/USPS/MNISTM, where there is a larger shift between source and target, the gains are especially pronounced. Figure 2 compares how the task loss (polyloss $\epsilon = 6$) on the test data decreases as we adapt the model through conjugate PL and other baselines. We use CIFAR-10-C as an example. Observe that our proposed conjugate PL indeed reduces the task loss the most among other baselines.

Dataset	Source Error	Hard PL	Robust PL	Entropy	Softmax PL	Conjugate PL **Ours**
SVHN → MNIST	28.33	20.21	19.73	14.28	16.54	**10.73**
SVHN → USPS	31.58	23.32	26.12	23.12	24.07	**21.62**
SVHN → MNISTM	61.69	50.73	51.35	49.33	50.47	**47.59**
ImageNet-R	64.19	58.52	59.46	58.25	56.62	**55.63**
VisDA-C	58.13	40.43	45.44	44.11	39.63	**38.42**

Table 3: Target error when adapting models trained via polyloss on source domains across different domain adaptation benchmarks. Conjugate pseudo-labeling offers consistent and substantial gains over previous approaches across three datasets.

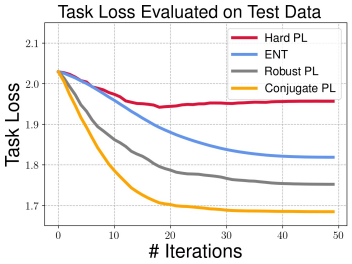


Figure 2: Task Loss (PolyLoss $\epsilon = 6$) evaluated on CIFAR-10-C test data during test-time adaptation.

Furthermore, on CIFAR-10-C and ImageNet-C, we find that adapting polyloss classifiers via conjugate PL improves the performance over all methods applied to cross-entropy trained source classifiers. For e.g., on ImageNet-C, the performance improves from 45.34% to 44.01%. However, this is only true when using the proposed conjugate PL. If we just did softmax-entropy minimization (even with temperature scaling), the final adapted performance of a polyloss classifier (45.5%) is in fact worse than that of a cross-entropy classifier (45.34%). Our results suggest that as we develop new training losses that improve the source classifiers, it is important to adapt via conjugate pseudo-labeling to reap the maximum gains.

Similarly, we experiment with the case when the source classifier is trained using squared loss on the CIFAR-10 and CIFAR-100 datasets, and observe consistent gains using the proposed conjugate pseudo-labels over the baselines. For example, on CIFAR-10-C, TTA using conjugate PL gives and error of 12.87%, outperforming baselines like ENT (13.24%) and Softmax PL (31.81%). Table 5 in Appendix A.7 shows the detailed results.

Comparing Table 1 and Table 2, we see that the relative ordering between the various baselines differs. This is further evidence that the adaptation loss has to depend on the training loss, and we believe our conjugate pseudo-label approach captures this appropriately by offering consistent gains across the various settings we experimented with.

6 Related Works

Test-time adaptation methods. In recent years, the setting of test-time adaptation has gained a lot of interest with a host of different approaches proposed in the literature. One family of TTA approaches update the source classifier by minimizing an unsupervised loss on the target distribution [4, 6, 20, 22, 35, 36, 40, 43, 44, 50, 51, 54]. TENT [50] proposes to minimize the entropy of model predictions at test time. Several follow ups like [6, 35, 40, 44, 54] propose alternative TTA objectives, e.g. robust pseudo-labelling [40], likelihood ratio loss [35], entropy of marginal probability averaged across augmentations [54] and self-supervised contrastive losses [6, 49]. However, most of these objectives are heuristically designed or chosen. In this paper, we provide a principled approach of designing unsupervised objectives for TTA .

Another family of approaches for test-time adaptation such as [2, 8, 13, 31, 34, 47] leverage an auxiliary self-supervised task (e.g. rotation prediction [47], masked autoencoders [10]) to update model parameters on each test sample. Crucially, these methods require modifying the source model training by augmenting the supervised training objective with an auxiliary self-supervised loss. Hence it cannot be applied to typical standard classifiers that are trained by minimizing a supervised loss on the source data.

Source-free domain adaptation. A very related setting to test-time adaptation is source-free domain adaptation, where a trained source classifier must be adapted to a target distribution of interest, although the entire target unlabeled data is available at once. SHOT [28] proposes to optimize the source hypothesis (i.e. feature extractor) with a combination of entropy minimization, diversity and self-training on pseudo-labels on the unlabeled target data. [53] promotes feature clustering on features from target distributions. [24, 26] use generative modeling to estimate the underlying source distributions for enforcing feature invariance. Such approaches typically require multiple epochs over the target data and cannot be easily adopted to work in an online fashion.

Unsupervised domain adaptation. The most canonical setting of domain adaptation involves access to labeled source data and unlabeled target data, all during training. The availability of source and target data during training lends itself to approaches that "align" the source and target representations in some way: [32, 33, 45, 48] match distribution statistics, [11] uses a discriminator, [46] uses self-supervised learning. However, such approaches require access to source data which might not always be feasible due to data privacy and efficiency issues.

Pseudo-labels and self-training. Self-training is a classic idea for leveraging unlabeled data, developed first for the semi-supervised setting. Self-training generates pseudo-labels on the unlabeled data, allowing us to use any "supervised" loss on this pseudo-labeled data. Self-training has shown promising results in various settings like semi-supervised learning [19] and improving adversarial robustness [5]. Self-training has also been gaining attention in the setting of unsupervised domain adaptation [28, 39], where pseudo-labels generated on the unlabeled data from target domain is used to supervise the adaptation process. [7, 23, 52] provide theoretical insights into how self-training with pseudo-labels can help under distribution shift. TENT [50] (i.e entropy minimization) can be viewed as a form of self-training with instantaneous softmax pseudo-labels. Our work provides a general framework for the choice of soft pseudo-labels based on the conjugate analysis of the source training objective. Some prior works like [7, 17, 27, 30, 55, 56] have documented the improvement in performance when using instantaneous pseudo-labels over pre-computed pseudo-labels, and thus lend further support to the benefits of our proposed conjugate pseudo-labeling approach. The experiment results presented in this work supporting conjugate pseudo-labels suggest that conjugate pseudo-labels is a promising direction of pseudo-labeling in a broader context.

7 Conclusion, Limitations and Future Directions

In this work, we proposed a general test-time adaptation loss, based on the convex conjugate formulation which in turn was motivated by the intriguing meta learning experiments. The fact that meta-learning recovers the proposed loss hints at some kind of optimality of the loss. In Section 4, we prove that for a broad set of loss functions, the proposed (unsupervised) conjugate loss is close to the oracle supervised loss. However, this still does not completely answer what the optimal test-time adaptation loss is and why.

The meta-learning framework in this work was constrained to learn functions over the logits of each individual input. It can be expanded to more involved setups, where we consider functions over the intermediate representations too and also consider learning functions over a batch of input while accounting for their interactions.

Beyond the choice of the adaptation loss itself, achieving good test-time adaptation generally involves several heuristics like updating only the batch norm parameters [50]. While our work was motivated by the loss function, via the meta-learning experiments, we discovered that temperature scaling is another important hyper-parameter that improves the performance of all previous baselines as well. At a high level, test-time adaptation has to be appropriately regularized to prevent the updates over batches from taking the model too far: updating only a few batch norm parameters is one way to do that, and perhaps temperature scaling provides a similar beneficial regularization effect by making the network predictions on unlabeled inputs less confident. Understanding the role of these heuristics more concretely is an interesting direction for future work. It also remains an open problem to understand under what sort of real-world distribution shifts would self-training based approaches would help.

Finally, it is also worth extending and applying the conjugate pseudo-labeling to other settings like semi-supervised learning.

8 Acknowledgments

We thank Shubhang Bhatnagar and Asher Trockman for helping with running the ImageNet experiments. We thank Zhili Feng for useful feedback. Sachin Goyal and Mingjie Sun were supported by funding from the Bosch Center for Artificial Intelligence. Aditi Raghunathan was supported by an Open Philanthropy AI Fellowship.

References

[1] https://en.wikipedia.org/wiki/Convex_conjugate.

[2] Pratyay Banerjee, Tejas Gokhale, and Chitta Baral. Self-supervised test-time learning for reading comprehension. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 2021.

[3] Sarah Bechtle, Artem Molchanov, Yevgen Chebotar, Edward Grefenstette, Ludovic Righetti, Gaurav Sukhatme, and Franziska Meier. Meta-learning via learned loss. *arXiv preprint arXiv:1906.05374*, 2019.

[4] Malik Boudiaf, Romain Mueller, Ismail Ben Ayed, and Luca Bertinetto. Parameter-free online test-time adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[5] Yair Carmon, Aditi Raghunathan, Ludwig Schmidt, John C Duchi, and Percy S Liang. Unlabeled data improves adversarial robustness. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/32e0bd1497aa43e02a42f47d9d6515ad-Paper.pdf.

[6] 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 (CVPR)*, 2022.

[7] Yining Chen, Colin Wei, Ananya Kumar, and Tengyu Ma. Self-training avoids using spurious features under domain shift. In *Advances in Neural Information Processing Systems*, 2020.

[8] Mohammad Zalbagi Darestani, Jiayu Liu, and Reinhard Heckel. Test-time training can close the natural distribution shift performance gap in deep learning based compressed sensing. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, 2022.

[9] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017.

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

[11] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario March, and Victor Lempitsky. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(59):1–35, 2016.

[12] Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization. In *International Conference on Learning Representations*, 2021.

[13] Nicklas Hansen, Rishabh Jangir, Yu Sun, Guillem Alenya, Pieter Abbeel, Alexei A. Efros, Lerrel Pinto, and Xiaolong Wang. Self-supervised policy adaptation during deployment. In *International Conference on Learning Representations*, 2021.

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

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

[16] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, Dawn Song, Jacob Steinhardt, and Justin Gilmer. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *In IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[17] Yosuke Higuchi, Niko Moritz, Jonathan Le Roux, and Takaaki Hori. Advancing momentum pseudo-labeling with conformer and initialization strategy. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2022.

[18] Like Hui and Mikhail Belkin. Evaluation of neural architectures trained with square loss vs cross-entropy in classification tasks. In *International Conference on Learning Representations*, 2021.

[19] Dong hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks.

[20] Yusuke Iwasawa and Yutaka Matsuo. Test-time classifier adjustment module for model-agnostic domain generalization. In *Advances in Neural Information Processing Systems*, 2021.

[21] Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, Tony Lee, Etienne David, Ian Stavness, Wei Guo, Berton A. Earnshaw, Imran S. Haque, Sara Beery, Jure Leskovec, Anshul Kundaje, Emma Pierson, Sergey Levine, Chelsea Finn, and Percy Liang. Wilds: A benchmark of in-the-wild distribution shifts. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 2021.

[22] Takeshi Kojima, Yutaka Matsuo, and Yusuke Iwasawa. Robustifying vision transformer without retraining from scratch by test-time class-conditional feature alignment. In *International Joint Conference on Artificial Intelligence*, 2022.

[23] Ananya Kumar, Tengyu Ma, and Percy Liang. Understanding self-training for gradual domain adaptation. In *Proceedings of the 37 th International Conference on Machine Learning (ICML)*, 2020.

[24] Vinod K Kurmi, Venkatesh K Subramanian, and Vinay P Namboodiri. Domain impression: A source data free domain adaptation method. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2021.

[25] Zhaoqi Leng, Mingxing Tan, Chenxi Liu, Ekin Dogus Cubuk, Jay Shi, Shuyang Cheng, and Dragomir Anguelov. Polyloss: A polynomial expansion perspective of classification loss functions. In *International Conference on Learning Representations*, 2022.

[26] Rui Li, Qianfen Jiao, Wenming Cao, Hau-San Wong, and Si Wu. Model adaptation: Unsupervised domain adaptation without source data. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[27] Xinzhe Li, Qianru Sun, Yaoyao Liu, Qin Zhou, Shibao Zheng, Tat-Seng Chua, and Bernt Schiele. Learning to self-train for semi-supervised few-shot classification. In *Advances in Neural Information Processing Systems*, 2019.

[28] Jian Liang, Dapeng Hu, and Jiashi Feng. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020.

[29] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In IEEE/CVF International Conference on Computer Vision (ICCV), 2017.

[30] Hong Liu, Jianmin Wang, and Mingsheng Long. Cycle self-training for domain adaptation. In *Advances in Neural Information Processing Systems*, 2021.

[31] 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? In *Advances in Neural Information Processing Systems*, 2021.

[32] Mingsheng Long, Jianmin Wang, Guiguang Ding, Jiaguang Sun, and Philip S. Yu. Transfer feature learning with joint distribution adaptation. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2013.

[33] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I. Jordan. Learning transferable features with deep adaptation networks. In *Proceedings of the 32nd International Conference on Machine Learning*, 2015.

[34] Xuan Luo, Jia-Bin Huang, Richard Szeliski, Kevin Matzen, and Johannes Kopf. Consistent video depth estimation. In *SIGGRAPH*, 2020.

[35] Chaithanya Kumar Mummadi, Robin Hutmacher, Kilian Rambach, Evgeny Levinkov, Thomas Brox, and Jan Hendrik Metzen. Test-Time Adaptation to Distribution Shift by Confidence Maximization and Input Transformation. *arXiv preprint arXiv: 2106.14999*, 2021.

[36] Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Yaofo Chen, Shijian Zheng, Peilin Zhao, and Mingkui Tan. Efficient test-time model adaptation without forgetting. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, 2022.

[37] Junhyuk Oh, Matteo Hessel, Wojciech M. Czarnecki, Zhongwen Xu, Hado P van Hasselt, Satinder Singh, and David Silver. Discovering reinforcement learning algorithms. In *Advances in Neural Information Processing Systems*, 2020.

[38] Xingchao Peng, Ben Usman, Neela Kaushik, Judy Hoffman, Dequan Wang, and Kate Saenko. Visda: The visual domain adaptation challenge, 2017.

[39] Viraj Prabhu, Shivam Khare, Deeksha Kartik, and Judy Hoffman. Sentry: Selective entropy optimization via committee consistency for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[40] Evgenia Rusak, Steffen Schneider, George Pachitariu, Luisa Eck, Peter Vincent Gehler, Oliver Bringmann, Wieland Brendel, and Matthias Bethge. If your data distribution shifts, use self-learning, 2022. URL `https://openreview.net/forum?id=1oEvY1a67c1`.

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

[42] H. Scudder. Probability of error of some adaptive pattern-recognition machines. *IEEE Transactions on Information Theory*, 1965.

[43] Manli Shu, Weili Nie, De-An Huang, Zhiding Yu, Tom Goldstein, Anima Anandkumar, and Chaowei Xiao. Test-time prompt tuning for zero-shot generalization in vision-language models. In *Advances in Neural Information Processing Systems*, 2022.

[44] Prabhu Teja Sivaprasad and François Fleuret. Test time adaptation through perturbation robustness. *arXiv preprint arXiv: 2110.10232*, 2021.

[45] Baochen Sun, Jiashi Feng, and Kate Saenko. Correlation alignment for unsupervised domain adaptation. *arXiv preprint arXiv: 1612.01939*, 2016.

[46] Yu Sun, Eric Tzeng, Trevor Darrell, and Alexei A. Efros. Unsupervised domain adaptation through self-supervision. *arXiv preprint arXiv:1909.11825*, 2019.

[47] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A. Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019.

[48] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014.

[49] Dequan Wang, Shaoteng Liu, Sayna Ebrahimi, Evan Shelhamer, and Trevor Darrell. On-target adaptation. *arXiv preprint arXiv: 2109.01087*, 2021.

[50] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations*, 2021.

[51] Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[52] Sang Michael Xie, Ananya Kumar, Robbie Jones, Fereshte Khani, Tengyu Ma, and Percy Liang. In-n-out: Pre-training and self-training using auxiliary information for out-of-distribution robustness. In *International Conference on Learning Representations*, 2021.

[53] Shiqi Yang, Yaxing Wang, Joost van de Weijer, Luis Herranz, and Shangling Jui. Generalized source-free domain adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[54] Marvin Zhang, Sergey Levine, and Chelsea Finn. Memo: Test time robustness via adaptation and augmentation. In *Advances in Neural Information Processing Systems*, 2022.

[55] Yang Zou, Zhiding Yu, B. V. K. Vijaya Kumar, and Jinsong Wang. Domain adaptation for semantic segmentation via class-balanced self-training. European Conference on Computer Vision, 2018.

[56] Yang Zou, Zhiding Yu, Xiaofeng Liu, B. V. K. Vijaya Kumar, and Jinsong Wang. Confidence regularized self-training. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] We add the limitations of our work in the end. See Section 7
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 7. We do not foresee any specific negative societal impact of our work, apart from general guidelines valid for deep learning in general.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We will be submitting the code along with the supplementary material. We will be adding a detailed readme for replicating the experiments.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] All the implementation details like the gridsearch and validation data have been mentioned in the paper, see section 5.1. Further, we will be adding exact hyper-parameters used for all the datasets in the appendix
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] All the results in the paper have the standard deviations reported, obtained by running the experiment 5 times with random seeds.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 5.1
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We will be adding the code to replicate experiments in this paper
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] We use all the publicly available datasets
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Appendix

A.1 Conjugate Derivations

Cross-Entropy Loss :

$$\begin{aligned}\mathcal{L}(h,y) &= -\sum_{i=1}^{c} y_i \log \frac{\exp(h_i)}{\sum_{j=1}^{c}\exp(h_j)} \\ &= -\sum_{i=1}^{c} y_i * h_i + \log \sum_{j=1}^{c}\exp(h_j) \\ &= f(h) - y^\top h,\end{aligned} \tag{14}$$

where $f(h)$ is $\log \sum_{j=1}^{c}\exp(h_j)$ and the constraint that $\sum_{i=1}^{c} y_i = 1$. Now, the conjugate $f^\star(y)$ is given by :

$$f^\star(y) = -\min_h\{f(h) - y^T h\} = -\min_h\{\log \sum_{j=1}^{c}\exp(h_j) - y^T h\} \tag{15}$$

with the constraint $\sum_{i=1}^{c} y_i = 1$. At the optimality,

$$y_i = (\nabla f(h))_i = \frac{\exp(h_i)}{\sum_j \exp(h_j)} \tag{16}$$

Then,

$$\begin{aligned}f^\star(y) &= -\log \sum_{j=1}^{c}\exp(h_j) + \sum_{i=1}^{c} h_i \frac{\exp(h_i)}{\sum_j \exp(h_j)} \\ &= \sum_i \frac{\exp(h_i)}{\sum_j \exp(h_j)} \log \frac{\exp(h_i)}{\sum_j \exp(h_j)},\end{aligned} \tag{17}$$

if the constraint $\sum_{i=1}^{c} y_i = 1$ is satisfied, otherwise $f^\star(y) = \infty$ by duality.

This in turn gives, the conjugate loss for cross-entropy (when the constraint is satisfied) :

$$\mathcal{L}^{\text{conj}}(h) = -f^\star(y) = -f^\star(\nabla f(h)) = -\sum_i \frac{\exp(h_i)}{\sum_j \exp(h_j)} \log \frac{\exp(h_i)}{\sum_j \exp(h_j)} \tag{18}$$

Squared Loss :

$$\begin{aligned}\mathcal{L}(h,y) &= \frac{1}{2}||h-y||_2^2 \\ &\approx \frac{1}{2}||h||_2^2 - y^\top h \ \text{ [ignoring the constant term]} \\ &= f(h) - y^\top h,\end{aligned} \tag{19}$$

Now, the conjugate $f^\star(y)$ is given by:

$$\begin{aligned}f^\star(y) &= -\min_h\{f(h) - y^T h\} = -\min_h\{\frac{1}{2}||h||_2^2 - y^T h\} \\ &= -\frac{1}{2}||h||_2^2\end{aligned} \tag{20}$$

A.2 Experiments on Binary Classification with Exponential Loss

Here we present the results on a binary classification task over a synthetic dataset of 100 dimensional gaussian clusters.

Dataset Creation For the binary classification task, we create a synthetic dataset similar to [23]. Specifically, let the data $\mathcal{X} \sim \mathcal{N}(\mu, \Sigma) \in \mathbb{R}^{100}$ and labels $\mathcal{Y} \in \{-1, +1\}$. We sample $\mu \sim \mathcal{N}(k, \mathcal{I}_{100})$. For Σ, similar to [23], we sample a diagonal matrix D, where each entry is sampled uniformly from a specified range, and a rotation matrix U from a HAAR distribution, giving $\Sigma = UDU^T$.
For the source data, we sample $\mu_s^{-1}, \mu_s^{+1}, \Sigma_s^{-1}, \Sigma_s^{+1}$ as specified above with $k = 0$. Now to create a distribution shifted data of various severity, we sample $\mu_t^{-1}, \mu_t^{+1}, \Sigma_t^{-1}, \Sigma_t^{+1}$ as specified above with $k = 1$, which are then used to sample the shifted data as follows :

$$\mu_\lambda^1 = \lambda\mu_t^1 + (1-\lambda)\mu_s^1$$
$$\mu_\lambda^{-1} = \lambda\mu_t^{-1} + (1-\lambda)\mu_s^{-1}$$
$$\Sigma_\lambda^1 = \lambda\Sigma_t^1 + (1-\lambda)\Sigma_s^1$$
$$\Sigma_\lambda^{-1} = \lambda\Sigma_t^{-1} + (1-\lambda)\Sigma_s^{-1}$$
$$\mathcal{X}_\lambda \sim \mathcal{N}(\mu_\lambda, \Sigma_\lambda)$$

In the following experiments, easy shift refers to $\lambda = 0.6$, moderate shift to $\lambda = 0.65$ and hard shift to $\lambda = 0.7$.

Exponential Loss for Binary Classification Let z be the classification score $h_\theta(x)$. For logistic training loss, conjugate adaptation loss would default to entropy with sigmoid probability. Thus, here we experiment with a different but also commonly used surrogate loss to 0/1 loss: exponential loss, which is defined as:

$$\mathcal{L}_{\text{exp}}(z, y) = \exp(-yz) \tag{21}$$

where $y \in \{-1, +1\}$. It can be rewritten in the expanded conjugate form of:

$$\mathcal{L}_{\text{exp}}(z, y) = \frac{1}{2} \cdot \left(e^z + e^{-z}\right) - \frac{1}{2} \cdot y \cdot \left(e^z - e^{-z}\right) \tag{22}$$

For exponential loss, the conjugate pseudo-label function and the conjugate pseudo-label loss are:

$$y_{\text{exp}}^{\text{CPL}}(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}, \quad \mathcal{L}_{\text{exp}}^{\text{CPL}}(z) = \frac{2}{e^z + e^{-z}} \tag{23}$$

The model is adapted on shifted gaussian clusters and we compare the conjugate loss with two baseline approaches: 1) Hard pseudo-labelling $\exp(-y_{\text{hard pl}} \cdot z)$; 2) Entropy applied to sigmoid probability $P(y = +1) = \sigma(z)$. The losses are compared on three degrees of shift (easy, moderate and hard), which is controlled by the drifted distance of Gaussian clusters. The results are shown in Figure 3, where we plot the accuracy curve with respect to adaptation iterations. With easy and moderate shift, conjugate loss (green) generalizes faster to shifted test data; with hard shift, only conjugate loss improves model accuracy on shifted test data while entropy (blue) deteriorates model performance.

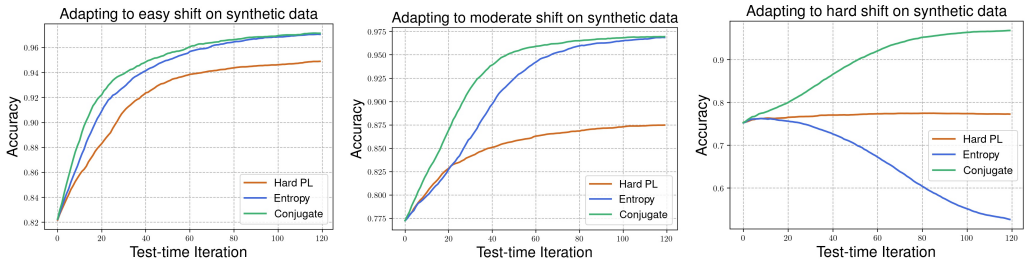


Figure 3: Test-time adaptation result on synthetic data with three shift levels ranging from easy, moderate and hard (detailed in section A.2). The source model is a linear classifier trained with exponential loss $\mathcal{L}_{\text{exp}} = e^{-yh_\theta(x)}$. Adaptation with the conjugate loss generalizes better compared to baseline losses.

A.3 Meta Learning Experiment Details

In section 3 we talked about learning the meta-loss function parameterized by a neural network $m_\phi : \mathbb{R}^{|\mathcal{Y}|} \mapsto \mathbb{R}$, that takes in the model predictions/logits and outputs a loss value. Here we discuss the architecture chosen and the implementation details. Further, in Appendix A.4 we empirically show that the learnt meta-loss is not affected by the choice of task loss / surrogate loss used in meta learning ($\mathcal{L}$ in Equation 1). Note that the task loss / surrogate loss function is used to update the meta-loss m_ϕ during meta-learning. The surrogate loss is calculated on updated source model's predictions on labeled samples from test domain. The surrogate loss tries to update the meta-loss in the outer loop such that when meta-loss is later used to update the source model in the inner loop, the source model generalizes better to the test domain.

Architecture and Implementation Details Figure 4 gives an overall schema for meta-learning the loss function and algorithm 2 gives the pseudo-code for meta-learning the loss function. Below we describe this in further detail. We use a transformer (denoted by $\mathcal{T}$) with a MLP (denoted by $\mathcal{P}$) over the output of transformer as the architecture for m_ϕ, i.e. $m_\phi(x) = \mathcal{P}(\mathcal{T}(x))$. Specifically, for a given source trained model h_θ and input $x \sim D_{\text{test}}$:

1. Let $h_\theta(x) \in \mathbb{R}^{|\mathcal{Y}|}$ be the model predictions/logits, where $|\mathcal{Y}|$ denotes the number of classes.
2. Let $h^j_\theta(x) \in \mathbb{R}, \forall j \in |\mathcal{Y}|$ be the prediction corresponding to class j.
3. The input to transformer is then given by $z \in \mathbb{R}^{|\mathcal{Y}| \times (1+e)}$, where $z^j \in \mathbb{R}^{1+e}, \forall j \in |\mathcal{Y}|$ is the concatenation of $h^j_\theta(x)$ and the learnable positional embedding $pe_j \in \mathbb{R}^e$.
4. The transformer output is given by $w = \mathcal{T}(z) \in \mathbb{R}^d$, where d denotes the feed-forward dimension of the transformer.
5. The transformer output w is finally passed through a MLP to get the meta-loss value $m_\phi(h_\theta(x)) = \mathcal{P}(w) \in \mathbb{R}$
6. The source model is updated by optimizing over the meta-loss.

$$\theta^{t+1} \leftarrow \theta^t - \alpha \frac{\partial m_{\phi^t}(h_{\theta^t}(x))}{\partial \theta^t} \tag{24}$$

7. The updated source model is then used to update the meta-loss by optimizing over some supervised loss function $\mathcal{L}_{\text{task}}$.

$$\phi^{t+1} \leftarrow \phi^t - \beta \frac{\partial \mathcal{L}_{\text{task}}(h_{\theta^{t+1}}(x'), y')}{\partial \phi^t}, \quad \text{where } (x', y') \sim D_{\text{test}} \tag{25}$$

Note that the last step assumes access to labels of test inputs. In this paper, we do not propose meta-learning the TTA loss as an approach. Rather, we use meta-learning to explore what the "best" TTA losses look like.

We select the trasformer input embedding dimension $(1 + e)$ from $\{16, 32, 64\}$ and transformer feed-forward dimension d from $\{32, 64, 128\}$. The number of transformer layers and the hidden layers in MLP are selected from $\{1, 2\}$. We use Adam optimizer with a learning rate of $1e^{-3}$ for learning the meta-loss (i.e. the transformer + MLP). We train the meta-loss for 100 epochs with a batch size of 200.

A.4 Effect of Task Loss in Meta Learning

In section 3, we show that the meta losses learned on different source classifiers differ substantially if the source classifiers are trained using different source loss functions. Here we further empirically verify that the learnt meta loss is not affected by the task loss used in meta learning ($\mathcal{L}$ in Equation 1). Thus the learnt meta loss is determined by the source model.

In Figure 5, we show the meta loss learnt on a ResNet-26 trained with Cross Entropy loss for two meta task losses: Cross Entropy Figure 5a and Squared Loss Figure 5b. We plot the meta loss as a function over one of its input prediction scores, while keeping other fixed. We can see that the task loss barely affects the learnt meta loss. Similar observations can be made for the classifier trained with squared loss Figure 6.

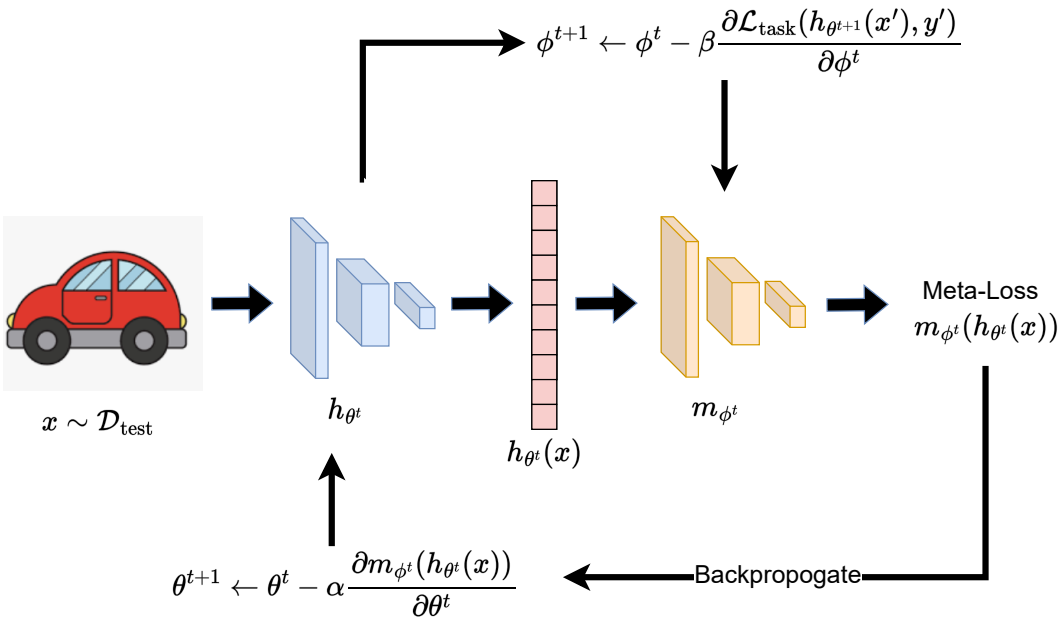


Figure 4: Meta-Loss learning procedure : The model predictions $h_{\theta^t}(x)$ are passed through the parameterized loss function m_{ϕ^t}, which outputs a loss value. We optimize ϕ such that when optimizing the source model over the loss $m_{\phi^t}(h_{\theta^t}(x))$, the updated θ^{t+1} has a better performance on the test domain. To do this, we take one gradient step over the meta-loss to get the update source model parameters θ^{t+1}, and then update ϕ by evaluating θ^{t+1} on the labeled validation data using some task loss $\mathcal{L}_{\text{task}}$.

Algorithm 2 Learning the Meta-Loss

Input: Source trained classifier h_{θ^0}. Randomly initialized meta-loss m_{ϕ^0}.
Task loss / Surrogate loss $\mathcal{L}_{\text{task}}$ like cross-entropy or squared loss for meta learning
N batches of test data $D_{\text{test}} = [(x_1, y_1), \ldots, (x_N, y_N)]$
Hyperparams: learning rates α and β.

for $epoch = 0, 1, 2, \ldots$ **do**
  for $n = 0, 1, \ldots N-1$ **do**
    $\theta^{t+1} \leftarrow \theta^t - \alpha \frac{\partial m_{\phi^t}(h_{\theta^t}(x_n))}{\partial \theta^t}$
    Sample $(x_r, y_r) \sim D_{\text{test}}$.
    $\phi^{t+1} \leftarrow \phi^t - \beta \frac{\partial \mathcal{L}_{\text{task}}(h_{\theta^{t+1}}(x_r), y_r)}{\partial \phi^t}$

A.5 Test-Time Adaptation Detail

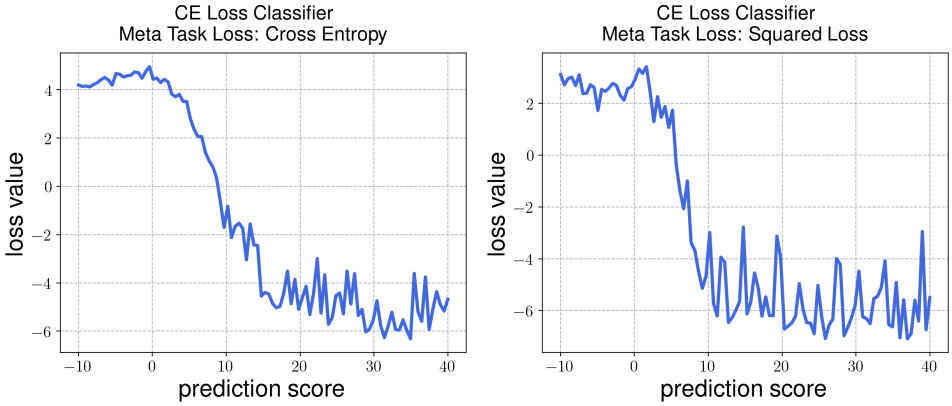
For completeness, we also give the test-time adaptation setup in Algorithm 3.

A.6 ImageNet results on each severity level

In continuation with results shown in Table 2 in Section 5.3, Table 4 shows the mean errors averaged across the 15 corruption types for each of the severity level on ImageNet-C, for a source classifier trained with PolyLoss ($\epsilon = 8$).

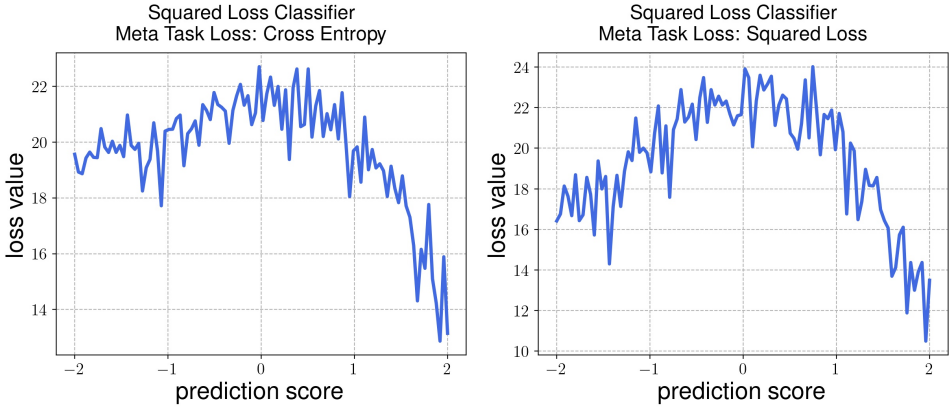
A.7 Square Loss Trained Source Classifier

In Section 5.3, we briefly discussed that similar to the other source training losses like cross-entropy and polyloss, our proposed conjugate loss outperforms the baselines when the source classifier is



(a) (b)

Figure 5: Visualizations of meta loss by varying one input dimension (prediction score). The source model is a ResNet-26 trained with Cross Entropy. Here we show meta loss trained by two different task losses: Cross Entropy Figure 5a and Squared Loss Figure 5b.



(a) (b)

Figure 6: Visualizations of meta loss by varying one input dimension (prediction score). The source model is a ResNet-26 trained with Squared Loss. Here we show meta loss trained by two different task losses: Cross Entropy Figure 6a and Squared Loss Figure 6b.

Algorithm 3 Test-Time Adaptation

Input: Source classifier θ_0 trained using loss $\mathcal{L}(h_\theta(x), y)$, An unsupervised loss function for test-time adaptation $\mathcal{L}_{\text{tta}}(x)$, N batches of test data $D_{\text{test}} = [x_1, \ldots, x_N]$
Hyperparams: learning rate η.

for $n = 0, 1, \ldots N-1$ **do**
$\theta_{n+1} = \theta_n - \eta \nabla \mathcal{L}_{\text{tta}}(x_n)$
$\hat{y}_n = h_{\theta_{n+1}}(x_n)$ [Predictions for the nth batch]

Corrution Severity	Temperature	Robust PL	Entropy	MEMO	Softmax PL	Conjugate
1	✗	34.27	33.17	34.39	32.49	**32.26**
	✓	34.27	32.84	34.39	32.70	**32.26**
2	✗	41.25	39.04	40.38	37.78	**37.40**
	✓	41.25	38.50	40.38	37.75	**37.40**
3	✗	47.37	44.04	45.67	42.30	**41.72**
	✓	47.37	43.33	45.67	42.14	**41.72**
4	✗	56.63	51.88	54.49	49.61	**48.84**
	✓	56.63	51.03	54.49	49.39	**48.84**
5	✗	67.11	62.53	66.13	60.94	**59.90**
	✓	67.11	61.80	66.13	60.30	**59.90**
Mean	✗	49.32	46.13	48.21	44.62	**44.02**
	✓	49.32	45.50	48.21	44.45	**44.02**

Table 4: Mean Errors across the 15 noises for various severity level on the ImageNet-C dataset, with source model trained using Poly-1 Loss. Note that Temperature scaling helped only in the case of Entropy and Softmax PL.

trained using a squared loss. Table 5 shows a detailed comparison with the baselines. We note that for the conjugate of squared loss, the temperature scaling can be wrapped into the learning rate as shown in Section 4.2. Further, on the CIFAR-10-C dataset we observe temperature scaling doesn't help any of the other baselines too, hence we do not include the temperature row in CIFAR-10-C.

Dataset	Temperature	Hard PL	Robust PL	ENT	MEMO	Softmax PL	Conjugate PL
CIFAR-10-C	✗	13.71 ($\pm$0.07)	13.06 ($\pm$0.05)	13.24 ($\pm$0.02)	13.22 ($\pm$0.04)	14.85 ($\pm$0.08)	**12.99** ($\pm$0.04)
CIFAR-100-C	✗	50.82 ($\pm$0.31)	44.53 ($\pm$0.13)	43.55 ($\pm$0.12)	51.35 ($\pm$0.04)	51.99 ($\pm$0.03)	**43.39** ($\pm$0.11)
	✓	50.82 ($\pm$0.31)	43.99 ($\pm$0.15)	**43.21** ($\pm$0.08)	51.35 ($\pm$0.04)	51.99 ($\pm$0.03)	43.39 ($\pm$0.11)

Table 5: Mean Errors on the common corruptions datasets for source classifier trained using squared loss. We note that temperature scaling didn't help on the CIFAR-10-C dataset. Source Classifier Errors without adaptation : CIFAR-10-C (28.34%), CIFAR-100-C (68.79%)

Dataset	Temperature (T)	Hard PL	Robust PL	MEMO	Conjugate PL (**ENT**)
CIFAR-10-C	✗	SGD,$1e^{-3}$, 1	SGD,$1e^{-3}$, 1	SGD,$1e^{-3}$, 1	SGD, $1e^{-3}$, 1
	✓	SGD,$1e^{-3}$, 1	SGD,$1e^{-2}$, 2	SGD,$5e^{-3}$, 3	Adam,$1e^{-3}$, 2
CIFAR-100-C	✗	SGD,$1e^{-2}$, 1	SGD,$1e^{-2}$, 1	SGD,$5e^{-3}$, 1	SGD, $1e^{-2}$, 1
	✓	SGD,$1e^{-2}$, 1	SGD,$1e^{-2}$, 2	SGD,$1e^{-2}$, 2	SGD,$1e^{-2}$, 2
ImageNet-C	✗	SGD,$1e^{-2}$, 1	SGD,$2.5e^{-3}$, 1	SGD,$1e^{-3}$, 1	SGD,$2.5e^{-3}$, 1
	✓	SGD,$1e^{-2}$, 1	SGD,$2.5e^{-3}$, 1.5	SGD,$1e^{-3}$, 1	SGD,$2.5e^{-3}$, 1.5

Table 6: Hyper-parameters (**Optimizer, Learning Rate, Temperature**) for the results in Table 1, where we showed the mean errors on the common corruptions dataset for a source classifier trained using cross-entropy loss.

A.8 Hyper-Parameters

We share the exact hyper-parameters found using gridsearch over the 4 validation noises for the common corruptions dataset.

Cross Entropy Classifier Experiments In Section 5.2, Table 1 shows the results when adapting a cross entropy trained classifier on various common corruptions dataset. Table 6 gives the optimizer, learning rate and optimal temperature for each of the baseline and our proposed conjugate loss.

PolyLoss Classifier Experiments In Section 5.3, Table 2 shows the results when adapting a polyloss trained classifier on various common corruptions dataset. Table 7 gives the optimizer, learning rate and optimal temperature for each of the baseline and our proposed conjugate loss.

Dataset	T	Hard PL	Robust PL	ENT	MEMO	Softmax PL	Conjugate PL (**Ours**)
CIFAR-10-C	✗	SGD,$1e^{-3}$, 1	SGD,$1e^{-3}$, 1	SGD,$1e^{-3}$, 1	SGD,$5e^{-3}$, 1	SGD, $1e^{-3}$, 1	SGD, $1e^{-3}$, 1
	✓	SGD,$1e^{-3}$, 1	SGD,$1e^{-2}$, 3	SGD,$1e^{-2}$, 3	SGD,$5e^{-3}$, 3	SGD, $1e^{-3}$, 2	SGD, $1e^{-3}$, 1.5
CIFAR-100-C	✗	SGD,$1e^{-2}$, 1	SGD,$1e^{-2}$, 1	SGD,$1e^{-2}$, 1	SGD,$1e^{-2}$, 1	SGD, $1e^{-2}$, 1	SGD, $1e^{-2}$, 1
	✓	SGD,$1e^{-2}$, 1	Adam,$1e^{-3}$, 3	SGD,$1e^{-2}$, 2	SGD,$1e^{-2}$, 2	SGD, $1e^{-2}$, 2.5	SGD, $1e^{-2}$, 1.5
ImageNet-C	✗	SGD,$1e^{-2}$, 1	SGD,$2.5e^{-3}$, 1	SGD,$2.5e^{-3}$, 1	SGD,$5e^{-3}$, 1	SGD, $2.5e^{-3}$, 1	SGD, $2.5e^{-3}$, 1
	✓	SGD,$1e^{-2}$, 1	SGD,$2.5e^{-3}$, 1	SGD,$2.5e^{-3}$, 1.5	SGD,$5e^{-3}$, 1	SGD, $2.5e^{-3}$, 2	SGD, $2.5e^{-3}$, 1

Table 7: Hyper-parameters (**Optimizer, Learning Rate, Temperature**) for the results in Table 2, where we showed the mean errors on the common corruptions dataset for a source classifier trained using poly-loss.

Squared Loss Classifier Experiments In Section 5.3, we briefly discussed the results when adapting a squared loss trained classifier on various common corruptions dataset. Table 8 gives the optimizer, learning rate and optimal temperature for each of the baseline and our proposed conjugate loss for the results in Table 5.

Digit Adaptation Datasets For the experiments on digits adaptation tasks, we do not have any validation set. Hence, we don't use temperature scaling here ($T = 1$) and fix the optimizer and LR as Adam and $1e^{-2}$ respectively for all the baselines.

A.9 Additional Experiments on Digit Adaptation Datasets

Similar to the setting of Table 1, we perform additional experiments on digit adaptation datasets when the source classifier is trained using the cross-entropy loss. Note that when the source classifier is trained using cross-entropy loss, the conjugate loss is equal to the softmax-entropy. In the absence of validation dataset in digit adaptation benchmarks, we used a fixed learning rate of 0.01 for all the baselines, optimizer as Adam and an informed temperature scaling guess of T=2.

Table 9 compares softmax-entropy minimization with various baselines. Here, again we observe that on SVHN $\rightarrow$ MNIST benchmark, without temperature scaling, MEMO (10.67% error) outperforms softmax-entropy (14.41% error). However, similar to the observations in Table 1, with temperature scaling, softmax-entropy minimization (9.26% error) is able to match the performance of MEMO (9.36% error). Further, on the SVHN $\rightarrow$ USPS benchmark, softmax-entropy (conjugate) and MEMO perform similar even without temperature scaling.

A.10 Additional Meta Learning the TTA Loss Experiments

In Section 3, we tried to learn a test-time adaptation (TTA) loss via meta-learning for adapting a CIFAR10 trained ResNet26 to distribution shifts on CIFAR10 corruptions. Figure 1 showed that the learnt meta-loss looks like a temperature scaled softmax-entropy.

In this section, we show the learnt meta loss across a range of settings as described below :

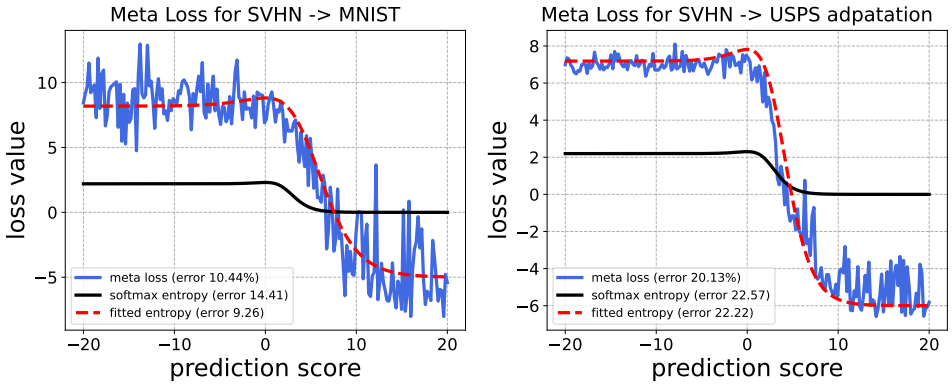
1. Digit Adaptation: Figure 7a and 7b show the learnt meta-loss when adapting a SVHN trained ResNet26 to MNIST dataset and USPS dataset respectively. We observe that the learnt meta-loss can be well approximated by a temperature scaled softmax-entropy.
2. Various Noise Types: In Figure 8, we show the learnt meta-loss when adapting a ResNet26 trained on CIFAR10 dataset using cross-entropy loss, to various noise types like speckle, gaussian, saturate and spatter. The severity level is kept fixed at the maximum i.e. 5.

Dataset	T	Hard PL	Robust PL	ENT	MEMO	Softmax PL	Conjugate PL (**Ours**)
CIFAR-10-C	✗	SGD,$1e^{-2}$, 1	SGD,$1e^{-2}$, 1	SGD,$1e^{-2}$, 1	SGD,$1e^{-2}$, 1	SGD,$1e^{-4}$, 1	SGD,$1e^{-2}$, 1
CIFAR-100-C	✗	Adam,$1e^{-3}$, 1	Adam,$1e^{-3}$, 1	Adam,$1e^{-3}$, 1	Adam,$1e^{-3}$, 1	Adam, $1e^{-4}$, 1	Adam, $1e^{-3}$, 1
	✓	Adam,$1e^{-3}$, 1	Adam,$1e^{-3}$, 0.5	Adam,$1e^{-3}$, 2	Adam,$1e^{-3}$, 2	Adam, $1e^{-4}$, 2.5	Adam, $1e^{-3}$, 1

Table 8: Hyper-parameters (**Optimizer, Learning Rate, Temperature**) for the results in Table 5, where we showed the mean errors on the common corruptions dataset for a source classifier trained using squared loss.

Dataset	Temperature (T)	Hard PL	Robust PL	MEMO	Conjugate PL (**ENT**)
SVHN → MNIST	✗	21.54	27.44	**10.67**	14.41
	✓	21.54	13.26	**9.36**	**9.26**
SVHN → USPS	✗	26.06	26.81	22.72	**22.57**
	✓	26.06	**22.32**	22.42	**22.27**

Table 9: Mean errors when adapting to digit adaptation benchmarks using a source classifier trained via cross-entropy loss. Here, conjugate pseudo-labeling becomes softmax-entropy minimization. Again we observe that with the right temperature scaling, softmax-entropy minimization matches other approaches. For additional context, the source classifier errors without adaptation are: SVHN → MNIST (34.17%), SVHN → USPS (31.84%).



(a) (b)

Figure 7: Visualizations of the learnt meta-loss by varying one input dimension (prediction score). The source model is a ResNet-26 trained with cross-entropy on the SVHN dataset. (a) The learnt meta-loss when adapting to the MNIST test dataset. (b) The learnt meta-loss when adapting to the USPS test dataset.

3. Various Severity Levels: In Figure 9, we vary the severity level of the noise, keeping the noise type fixed.

4. Dataset and Architecture: In Figure 10, we compare the learnt meta-loss when adapting to speckle noise, for different source classifier architectures (ResNet26 and ResNet50) and different source training dataset (CIFAR10 and CIFAR100). In all the cases, we again observe that the learnt meta-loss can be well approximated by a temperature scaled softmax-entropy.

5. Squared Loss : Finally, in Figure 11 we show the learnt meta-loss for classifiers trained with squared loss function instead of cross-entropy. We observe that in this case, the learnt meta loss mimics a quadratic function as expected from the conjugate formulation.

For each of the learnt meta losses, we also show the values (α, T, C) we use to fit the meta loss with softmax entropy function: $\alpha \cdot \mathcal{H}(\text{softmax}(x/T)) - C$. Note that although the learnt meta-loss can be approximated by the conjugate, the parameters α, T, C differ across the settings.

In the case of classifiers trained with squared loss, we fit the meta loss with a quadratic function $\sum_{i=1}^{K}(A \cdot x_i^2 + C)$, where K is the number of classes and x is the logit vector. Again, we also show the fitted parameter value A, C. The meta loss follows the trend of a quadratic function. The fitted quadratic function performs better or similar as the meta loss, while the parameters of the fitted quadratic function remain different across the meta learning setup (base classifier architectures and noise types).

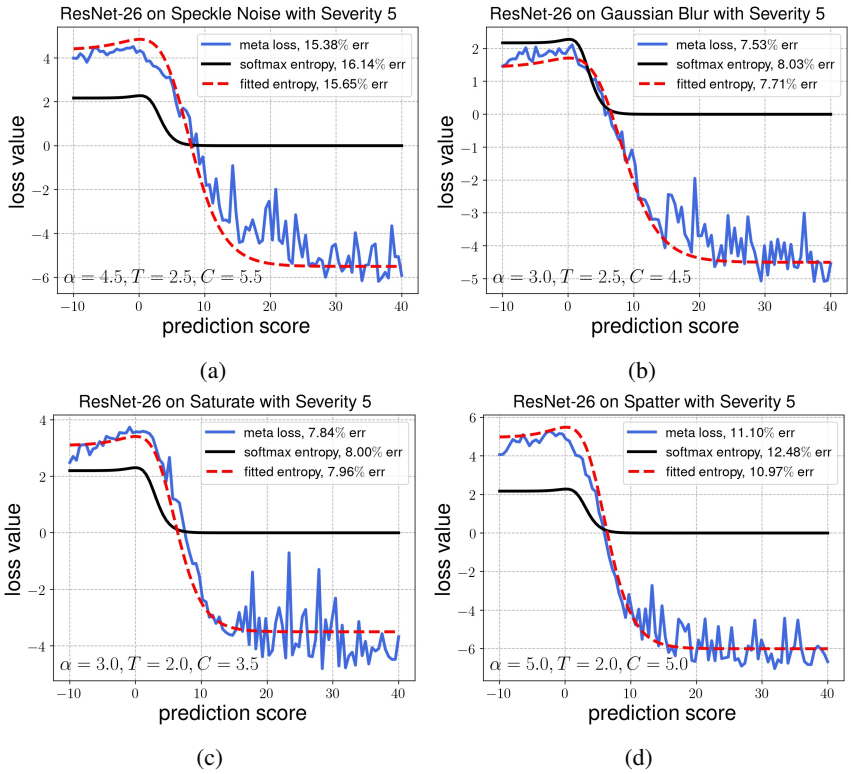


Figure 8: Visualization of meta loss (blue) learnt from various noise types in CIFAR-10-C validation set, where base classifiers are trained with cross-entropy loss. We show the error of meta loss, softmax entropy and fitted entropy for test-time adaptation on the corresponding noise types. We also show the parameters (α, T, C) in the fitted entropy.

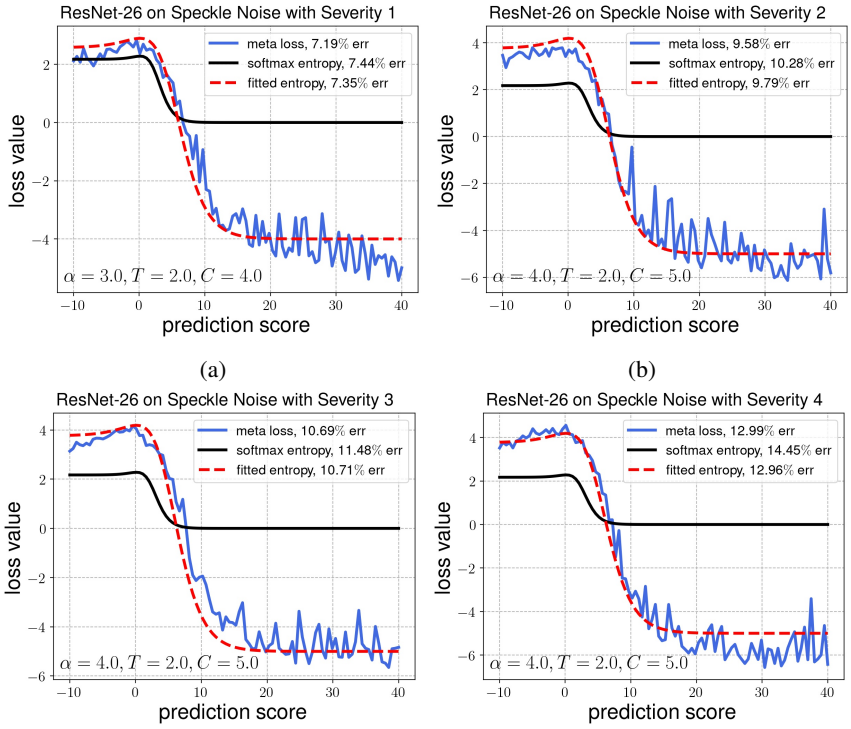


Figure 9: Visualization of meta loss (blue) learnt on speckle noise with different severity level for CIFAR-10-C, where base classifiers are trained with cross-entropy loss. We show the error of meta loss, softmax entropy and fitted entropy for test-time adaptation on the corresponding noise types. We also show the parameters (α, T, C) in the fitted entropy.

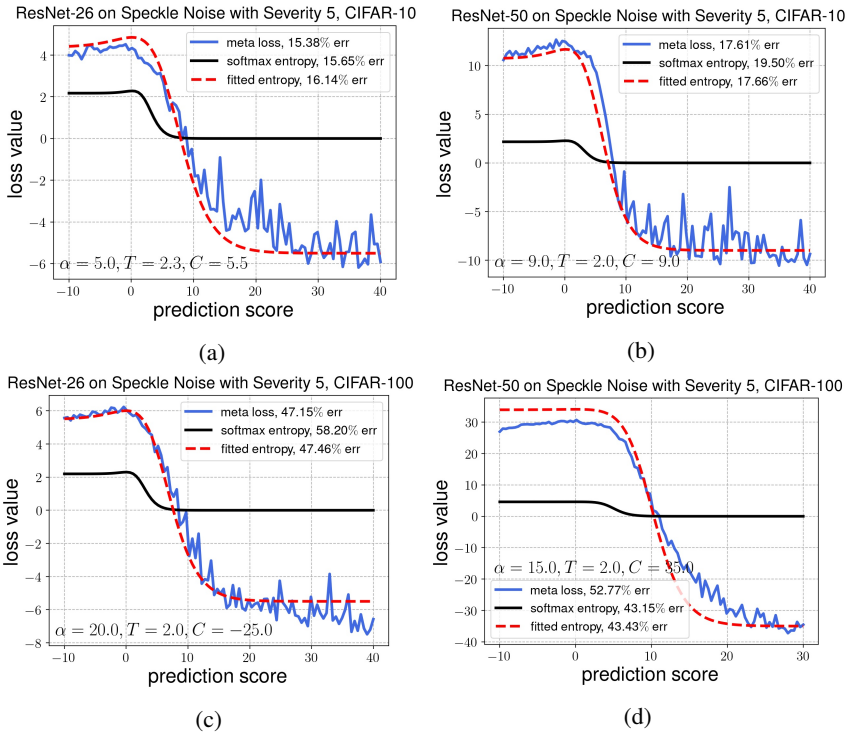


Figure 10: Visualization of meta loss (blue) learnt across datasets (CIFAR-10-C/CIFAR-100-C) and base classifier architectures (ResNet-26/ResNet-50), where base classifiers are trained with cross-entropy loss. We show the error of meta loss, softmax entropy and fitted entropy for test-time adaptation on the corresponding noise types. We also show the parameters (α, T, C) in the fitted entropy.

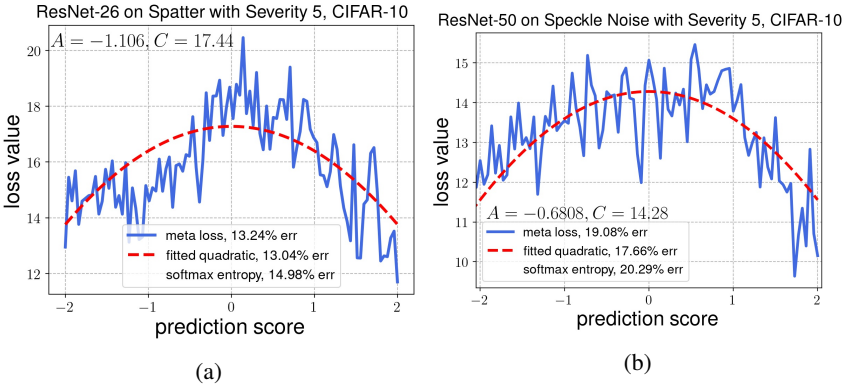


Figure 11: Visualization of meta loss (blue), where base classifier is trained with quadratic loss. We show the error of meta loss, softmax entropy and fitted quadratic function for test-time adaptation on the corresponding noise types. We also show the parameters (A, B, C) in the fitted quadratic function.