TIPI: Test Time Adaptation with Transformation Invariance

A. Tuan Nguyen[1], Thanh Nguyen-Tang[2],
Ser-Nam Lim[3*], Philip H.S. Torr[1*]
[1]University of Oxford, [2]Johns Hopkins University, [3]Meta AI
tuan@robots.ox.ac.uk, nguyent@cs.jhu.edu
sernamlim@meta.com, philip.torr@eng.ox.ac.uk

Abstract

When deploying a machine learning model to a new environment, we often encounter the distribution shift problem – meaning the target data distribution is different from the model's training distribution. In this paper, we assume that labels are not provided for this new domain, and that we do not store the source data (e.g., for privacy reasons). It has been shown that even small shifts in the data distribution can affect the model's performance severely. Test Time Adaptation offers a means to combat this problem, as it allows the model to adapt during test time to the new data distribution, using only unlabeled test data batches. To achieve this, the predominant approach is to optimize a surrogate loss on the test-time unlabeled target data. In particular, minimizing the prediction's entropy on target samples [34] has received much interest as it is task-agnostic and does not require altering the model's training phase (e.g., does not require adding a self-supervised task during training on the source domain). However, as the target data's batch size is often small in real-world scenarios (e.g., autonomous driving models process each few frames in real-time), we argue that this surrogate loss is not optimal since it often collapses with small batch sizes. To tackle this problem, in this paper, we propose to use an invariance regularizer as the surrogate loss during test-time adaptation, motivated by our theoretical results regarding the model's performance under input transformations. The resulting method (TIPI – Test tIme adaPtation with transformation Invariance) is validated with extensive experiments in various benchmarks (Cifar10-C, Cifar100-C, ImageNet-C, DIGITS, and VisDA17). Remarkably, TIPI is robust against small batch sizes (as small as 2 in our experiments), and consistently outperforms TENT [34] in all settings. Our code is released at https://github.com/atuannguyen/TIPI.

*The last two authors contributed equally

1. Introduction

Distribution shift is a common problem and is often faced in real-world applications. Specifically, despite taking various precautions while training a machine learning model to ensure a better generalization (e.g., collecting and training on multiple source domains [17, 27], finding flat minima [5], training with meta-learning objectives [16] etc.), the model often still struggles when the test data distribution shifts slightly. Note that it is also common not to have labels for the new shifted domain during test time. To tackle this problem, test time adaptation (also known as online domain adaptation) is a framework that allows the model to adapt to the target distribution using unlabeled test data batches. This is necessary since we typically do not have time to annotate the data during test time and can only make use of the unlabeled data. In this framework, the model needs to give predictions for the target data while simultaneously updating itself to improve its performance on that particular target distribution. We assume a situation in which the target data only arrive in small batches, which makes the adaptation task extremely challenging and renders traditional domain adaptation techniques such as representation alignment (via a distance metric) ineffective.

Within this test time adaptation framework, a common and effective approach is to optimize a surrogate objective function in lieu of the true loss function on the target data. The first group of surrogate objectives is the loss functions of self-supervised tasks. In particular, one would formulate a user-defined task (such as predicting the rotation angle of an image) and train it alongside the main task on the source domain; and keep training the self-supervised task on the test-time target data [19, 33]. However, these are not fully test-time adaptation methods, since they require altering the training procedure of the source domain. The second line of surrogate objectives is unsupervised loss functions. Among this group of unsupervised objectives, entropy minimization (TENT) [34] is the most successful method, and has been shown to be consistent across many benchmarks. Furthermore, different from the former group,

TENT is a fully test-time adaptation method. For these reasons, TENT has received much interest and a lot of follow-up papers/discussions.

However, as pointed out by its authors, TENT is not robust when using small batch sizes, as it often collapses to a trivial solution (i.e., it always predicts the same class for all input). This is detrimental to real-world applications since test data often arrive in small batches. For example, autonomous driving systems only process a few frames in real time – they typically do not accumulate the frames (let's say within one minute) to form a bigger batch. Note that TENT is previously evaluated mainly for large batch sizes (such as 64,128 or 200).

In this paper, we aim to tackle the aforementioned problem. Specifically, we aim to develop an unsupervised surrogate objective function for the test time adaptation problem such that it is task-agnostic, does not require altering the training procedure (e.g., does not require incorporating a self-supervised task into the training process), and is more resilient against small batch sizes.

We first provide theoretical results regarding a model's performance under input transformations. Specifically, we show that a model's loss on a data distribution is bounded by the KL distance on the predictive distribution of the data before and after the transformations (which we will use as a regularizer), and its loss on the transformed data distribution. Motivated by this result, we use small shifts in the input images that can simulate real source-target shifts, and enforce the network to be invariant under such data transformations. Our model outperforms TENT (and other relevant baselines) in all problem settings considered in the paper and is remarkably robust in the small-batch-size regime.

Our contributions in this paper are threefold:

- We provide theoretical results regarding a model's performance under transformations of the input data. Specifically, a model's performance on the target domain is bounded by an invariance term (maximum KL divergence of the predictive distributions before and after the transformations) and its loss on the transformed domain.

- We propose to find input transformations that can simulate the domain shifts, and enforce the network to be invariant under such transformations, using a regularizer based on our derived bound. The resulting method is **TIPI** (**T**est t**I**me ada**P**tation with transformation **I**nvariance).

- We perform extensive experiments on a wide range of datasets (Cifar10-C, Cifar100-C, ImageNet-C, DIGITS, and VisDA17) and settings (varying batch sizes) to validate our method. TIPI shows preferable performance compared to relevant baselines.

2. Related Works

Distribution Shift is an important and commonly faced problem when deploying machine learning models in real-world applications. There are two main approaches that try to alleviate this problem. The first setting is domain generalization [1, 17, 23, 26, 27], where one aims to collect multiple source domains and learn a generalizable model based on those domains. Another setting is (unsupervised) domain adaptation [2, 6, 18, 21, 28], where a model is trained on a labeled source dataset and we attempt to adapt that model to a target distribution with the help of unlabeled target data. This paper focuses on the latter setting, in particular, the test-time domain adaptation variation.

Test Time Domain Adaptation: In this setting, the unlabeled target data is not available during the training phase of the source model. Instead, the model only has access to the target data when deployed in the target environment, as the stream of test data batches. As a consequence, the unlabeled target data is not available in large quantities, but only in small batches. This makes the adaptation task extremely challenging. So far, there are two main lines of work dealing with this online adaptation problem. The first approach is based on the batch normalization layer in the network, with the idea of using the test-time batch statistics for these layers instead of the source statistics [24, 32]. This can partially help with the representation misalignment between the source and target distributions but is only applicable when the network has BatchNorm layers and the batch size is large. The other line of work is to use an unsupervised surrogate loss to "refine" the model on the target data. For example, TENT [34] minimizes the prediction entropy of the model on the target data. However, as pointed out by its authors, TENT needs a large batch size to avoid collapsing to a trivial prediction (only output one class), due to the characteristics of this surrogate objective. Meanwhile, TTT [33] and TTT+++ [19] train a self-supervised task alongside the main task on the source domain, and continue to train the self-supervised task during the online adaptation phase. Therefore, TTT and TTT++ are not fully off-the-shelf test-time adaptation methods, since they require modifications to the training phase of the source domain (to incorporate the self-supervised task). Recently, [8] generalizes TENT to other source loss functions (beyond the typical cross-entropy loss). There also have been several works [29] that study datapoints selection strategies for optimizing the unsupervised surrogate loss (e.g., TENT): only optimize the loss for reliable and informative datapoints. Note that this line of research is orthogonal/complementary to ours and we believe that our surrogate loss will also benefit from similar datapoints selection strategies. We leave exploring this direction to future work. Using a completely different approach, T3A [12] replaces the last linear layer (classification layer) with a nonparametric classifier and keeps updating the support set of each class with the new target

datapoints. T3A is robust against small batch sizes (since it is not optimization-based), but is not as effective as other optimization-based methods (e.g., TENT, TTT, and ours).

Connection between Adversarial Robustness and Robustness against Distribution shift: It has been observed that adversarially trained models generally perform better under certain types of domain shifts (for example, see the leaderboard of Robustbench [4] for Cifar10-C and Cifar100-C). Indeed, one might expect this to be the case if the space of adversarial attacks matches the space of distribution shifts. While it is not previously well understood, our paper sheds some light on this phenomenon. Note that different from previous works, our method enforces an unsupervised robustness loss during the online adaptation phase, not at the source training stage.

3. Approach

3.1. Problem setting

Assume that we have a source domain with the distribution $p_S(x, y)$, where $x \in \mathcal{X}$ is the input and $y \in \mathcal{Y}$ is the prediction target. Similar to prior works, we consider the classification problem, where $\mathcal{Y} = \{1, 2, \dots, C\}$ (however, the approach can be straightforwardly extended to regression problems). Suppose that using data from the source domain, we already successfully trained a model $f_\theta : \mathcal{X} \to \mathbb{R}^C$, where θ is the parameter of the neural network and $f_\theta(x)$ outputs the prediction probabilities for the C classes ($\|f_\theta(x)\|_1 = 1$ and $0 \leq f_\theta(x)_i \leq 1, \forall i \in \{1, 2, \dots, C\}$). This leads to a predictive distribution $\hat{p}_\theta(y|x)$, which is the categorical distribution where $\hat{p}_\theta(y = i|x) = f_\theta(x)_i$. Note that, for our method, we will not assume or require any modifications to the training process of the model f_θ on the source domain – a setting that is common and useful (e.g., a Resnet50 model downloaded directly from `torchvision`).

After that, this model is deployed to the target environment with data distribution $p_T(x, y)$. The model receives target data as a stream of unlabeled mini-batches (probably with small and varying sizes). The task is to adapt to this environment simultaneously and continually with only the unlabeled mini-batches.

The expected loss of this model on a data distribution $q(x, y)$ (for example, $q(x, y) = p_T(x, y)$) is:

$$\ell(\theta, q(x, y)) = \mathbb{E}_{q(x)} \left[l[q(y|x), \hat{p}_\theta(y|x)] \right], \qquad (1)$$

where l is a loss function (distance/divergence) between the true labeling distribution $q(y|x)$ and the predictive distribution $\hat{p}_\theta(y|x)$. This loss is not actually computed in practice for both the source and target data (we simply do not have labeled data for both of those), and it is used for theoretical discussion only. To make the theoretical discussion simpler, in this paper, we will consider the simple l_1 distance between the two distributions. However, note that our theoretical discussion also holds for other divergences such as the Jensen-Shannon divergence (or symmetric KL). Both of these distances satisfy the triangle inequality, which we use in our derivation.

3.2. Test Time Domain Adaptation with Transformation Invariance

Recall that the task is to gradually update the network parameter θ so that the model can adapt to the target domain. In this setting, we do not have access to the source data nor a large amount of target data (only small batches), so traditional domain adaptation approaches such as representation alignment are often infeasible.

Given access only to the batches of unlabeled target data, it is natural to try to come up with a surrogate objective to optimize in lieu of the true loss function (which is not available without the labels). In this sense, this has a close resemblance to the semi-supervised learning problem. In both settings, we aim to find a surrogate function to optimize for unlabeled data. In fact, both the entropy-minimization and pseudo-labeling techniques have been borrowed from the semi-supervised literature, with varying degrees of success. It should be noted that, in semi-supervised learning, the surrogate loss on the unlabeled data is minimized jointly with the supervised objective for labeled data, thus model collapsing is often not a problem. It is much more challenging, however, in this online adaptation setting, since the unsupervised surrogate loss is optimized alone, which can lead to the catastrophic collapse of the network parameter (such that it can not perform prediction anymore, even for the source domain). For example, as pointed out by its authors, TENT [34] tends to collapse to a trivial model if the batch size is small, which is often the case in practice.

In this paper, we examine the invariance regularization technique in the setting of test time domain adaptation. Note that a similar regularizer ([3, 22, 31]) has been applied to semi-supervised learning. However, to the best of our knowledge, we are the first to investigate this type of regularization in the online domain adaptation problem. We aim to investigate both **theoretically** and **empirically** how this technique can help with adapting to the new target domain. We also find that this surrogate loss is much more robust (against small batch sizes) compared to TENT.

To summarize, we propose to enforce the network to be invariant under a predefined space of transformations on the data on the target domain. The idea is that this space of transformations contains the actual domain shifts, or at least could simulate them. Specifically, we minimize θ with the following objective:

$$\min_\theta \mathbb{E}_{p_T(x)} \left[\max_{x' \in A(x)} \mathrm{KL} \left[\hat{p}_\theta(y|x'), \hat{p}_{\overline{\theta}}(y|x) \right] \right], \qquad (2)$$

where $\overline{\theta}$ is an identical copy of θ, and $A(x) = \{t(x)\}_{t \in S_t}$

Algorithm 1 Adapt and predict. θ is the network parameter, $\{x_i\}_{i=1}^n$ is the batch data, α is the learning rate for online adaptation

Input θ, $\{x_1, x_2, ..., x_n\}$
Output θ_{new} (newly adapted), $\{\hat{y}_1, \hat{y}_2, ..., \hat{y}_n\}$

function ADAPT-PREDICT(θ, $\{x_1, x_2, ..., x_n\}$)
  $p_i \leftarrow f_\theta(x_i)$`.detach()` ▷ stop gradient
  $x'_i \leftarrow \arg\max_{x' \in A(x_i)} \text{KL}[p_\theta(y|x'), p_i]$
  $l(\theta) = \frac{1}{n}\sum_{i=1}^n \text{KL}[p_\theta(y|x'_i), p_i]$
  $\theta_{\text{new}} = \theta - \alpha \nabla_\theta l(\theta)$
  $\hat{y}_i = \arg\max f_{\theta_{\text{new}}}(x_i)$
end function

(S_t is the predefined set of transformations).

We present the motivations and derivations of this objective function in the next subsection. The adapting and predicting procedure is depicted in Algorithm 1.

3.2.1 Invariance Regularization Technique

In this subsection, we introduce the invariance regularization technique for general transformations and provide theoretical results regarding the online domain adaptation problem, which will later motivate our method.

Let's consider a set of transformations S_t, where each $t \in S_t$ is a transformation of the data, i.e., $t : \mathcal{X} \to \mathcal{X}$. Ideally, S_t should contain the transformations responsible for the domain shift. This set can contain transformations such as data augmentations, GAN-like cross-domain transformations, random noise perturbation, etc. For now, we shall not care much about the type of transformations in S_t. The only important detail is that these transformations should preserve the class information (this is formalized in Assumption 1). Let $A(x) = \{t(x)\}_{t \in S_t}$ and let $p_t(t)$ be a distribution over S_t that assigns probability mass (or density) to each transformation $t \in S_t$. Each distribution $p_t(t)$ defines a conditional distribution $T(x^t|x)$ (with the support set $A(x)$), where $x^t = t(x), t \sim p_t(t)$. This in turn defines a joint distribution over x, x^t and y as:

$$p_T(x, x^t, y) = p_T(x, y)T(x^t|x). \tag{3}$$

It is also worth noting that we have a transformed data distribution $p_T(x^t, y)$ of x^t and y (just marginalizing out x from the above).

Assumption 1.

$$I_T(x, y) = I_T(x^t, y), \tag{4}$$

where I_T is the mutual information, calculated on the target domain (please refer to the Supplementary Material for their exact formula).

Intuitively, this assumption means that the transformed image x^t contains the same amount of information about the label y as the original image x does. This assumption is equivalent to $\mathbb{E}_{p_T(x,x^t)}\left[\text{KL}[p_T(y|x), p_T(y|x^t)]\right] = 0$ (a derivation can be found in the supplementary material), meaning that $p_T(y|x) = p_T(y|x^t)$ almost surely.

The transformation invariance regularizer on the target domain can be defined simply as:

$$h(\theta, x) = \max_{x' \in A(x)} D\left[\hat{p}_{\overline{\theta}}(y|x), \hat{p}_\theta(y|x')\right], \tag{5}$$

where $D[\cdot,\cdot]$ is a divergence/distance of choice and $\overline{\theta}$ is an identical copy of θ (so that the gradient is not back-propagated through it). We will see later (through an analysis of the gradient) that this is extremely important. Intuitively, it also helps to alleviate the network collapse problem, since, without this gradient blocking, the network would easily collapse to a constant output to minimize the regularizer term.

Intuitively, we can view this objective as using $\hat{p}_{\overline{\theta}}(y|x)$ as the target to optimize $\hat{p}_\theta(y|t(x))$ for all transformation t. This makes sense because, as discussed earlier, t does not change the semantic meaning of x significantly, so the predictive distribution of x and $t(x)$ should be similar.

Based on this regularizer, we can bound the loss on the target domain as follows:

Proposition 1. *Under Assumption 1, if D in Eq. 5 is either the forward or reverse KL divergence, we have:*

$$\ell(\theta, p_T(x, y)) \leq \mathbb{E}_{p_T(x)}\left[\sqrt{2h(\theta, x)}\right] + \ell(\theta, p_T(x^t, y)).$$

Proof. Provided in the Supplementary Material. □

This means that the loss on the target distribution is bounded by the invariance regularizer and the loss on the transformed distribution $p_T(x^t, y)$.

Now, assume for a moment (even if it is not realistic) that the set S_t contains all the transformations that can successfully transform the target data distribution $p_T(x, y)$ back to the source distribution $p_S(x, y)$ (i.e., there exists a distribution $p_t(t)$ over S_t such that the resulting transformed distribution $p_T(x^t, y)$ is the same with $p_S(x, y)$). Then, we have that the loss on the target domain is bounded by the loss on the source domain and our invariance regularizer. Assume also that while adapting to the new target domain, our model does not suffer from catastrophic forgetting of the source domain (which is often the case if the network does not collapse since we are adapting to the same task in a different environment). Then we have that the loss on the source domain would be small, and the invariance regularizer would also be small (since we explicitly minimize it). As a consequence, the loss on the target domain would be small and we can guarantee a good adaptation performance.

However, knowing the exact transformations from the target domain back to the source domains is often infeasible (only possible if we have some prior domain knowledge). This motivates us to find other transformations that

are task-agnostic and can simulate the distribution shift between the target and source domains.

3.2.2 Practical choice of the transformation set

We want to choose task-agnostic transformations such that they do not alter the semantic meaning of the image significantly, while they can simulate the distribution shift. The first idea that comes to mind is to use common image augmentations (e.g., random crop, random vertical flip, etc.) However, our early experiments suggest that this method is not consistent across corruption types/domain shifts. This makes sense since image augmentations are often heuristic, and they may be able to simulate certain types of distribution shifts, but not others. Therefore, we opt to use l_∞-norm based perturbations. Empirically, this leads to better performance and also is consistent across domains/tasks.

To summarize, we use the l_∞-norm based perturbations as the transformations of images. Empirically, we also found out that using the reverse KL leads to better performance (although almost all distances/divergences have positive results). We provide explanations/intuitions as to why we should prefer the reverse KL, and why adversarial examples can help with the distribution shift problem in the next subsection. Our final optimization problem is:

$$\min_\theta \mathbb{E}_{p_T(x)} \left[\max_{||x'-x||_\infty \leq \epsilon} \text{KL}\left[\hat{p}_\theta(y|x'), \hat{p}_{\overline{\theta}}(y|x)\right] \right]. \quad (6)$$

We name our method **TIPI** (**T**est t**I**me ada**P**tation with transformation **I**nvariance). To save computational cost, we use the single-step adversarial method FGSM [7] to find the adversarial example x'. Note that one can also use more advanced methods such as PGD [20] to find the adversarial perturbation to get better results (but with the trade-off of speed). We set ϵ to $2/255$ in all of our experiments.

3.2.3 Additional Discussions

Why preferring reverse KL over forward KL? Natural choices for D are the KL-divergences (forward or reverse KL), since they can be computed straightforwardly for the predictive distributions, and also are commonly used in practice for similar problems (e.g., VAT [22] uses the forward KL). Our experiments empirically suggest that using the reverse KL as the divergence D leads to better results compared to the forward KL. Here, we also give theoretical intuition as to why it is the case. First of all, let's consider the gradient of θ with respect to both the forward and reverse KL:
For the forward KL:

$$\nabla_\theta \text{KL}[\hat{p}_{\overline{\theta}}(y|x), \hat{p}_\theta(y|x')] = \sum_{y \in \mathcal{Y}} -\frac{\hat{p}_{\overline{\theta}}(y|x)}{\hat{p}_\theta(y|x')} \nabla_\theta \hat{p}_\theta(y|x'). \quad (7)$$

Meanwhile, for the reverse KL:

$$\nabla_\theta \text{KL}[\hat{p}_\theta(y|x'), \hat{p}_{\overline{\theta}}(y|x)] = \sum_{y \in \mathcal{Y}} \left(-\log \frac{\hat{p}_{\overline{\theta}}(y|x)}{\hat{p}_\theta(y|x')} + 1 \right) \nabla_\theta \hat{p}_\theta(y|x'). \quad (8)$$

Compare Eq. 8 to Eq. 7, we can see that the coefficient of $\nabla_\theta \hat{p}_\theta(y|x')$ in the reverse KL gets squashed down by the $\log$ function, leading to much smaller variance. We conjecture that the small variance in gradient is the reason why reverse KL performs better than forward KL.

How can invariance enforcing with adversarial examples help with the distribution shift problem? In our previous subsection, we provide theoretical evidence that motivates us to find transformations that can simulate common distribution shifts in practice. We argue here that l_∞ adversarial perturbation is a good candidate for this. First of all, l_∞ adversarial perturbation is a task-agnostic transformation that does not require much tuning, and we do not need to design heuristic transformations per domain. Secondly, a lot of the common domain shifts in practice are image corruptions that add noises to the images (gaussian noise, gaussian blur, jpeg compression noise, etc.), which can be simulated very well by adversarial noises. Lastly, even for more global domain shifts as seen in the DIGITS and VisDA17 experiments (Section 4), we conjecture that training a network that is robust against local and small changes (adversarial noise) would also help to make it more robust against larger shifts. Our empirical experiments also agree with this, as TIPI performs consistently across multiple types of domain shifts.

Implementation Details: Following TENT [34] and other test-time surrogate optimization methods, we only optimize the affine parameters of the model (i.e., the weight and bias parameters of the norm layers: BatchNorm, LayerNorm, InstanceNorm, or GroupNorm). This has been shown to lead to a more stable optimization, and also reduces the computational cost, as opposed to optimizing all parameters of the network. However, note that optimizing over the entire representation network (similar to SHOT [18]) is also possible. Furthermore, in case the network uses BatchNorm layers, we replace each BatchNorm with two BatchNorm layers (following [35]), one for the clean images and one for the adversarial ones. This is because the batch statistics of the adversarial examples are much different from that of the clean images [35], thus mixing them might cause performance degradation. Note that these two BatchNorm layers only keep separate batch statistics (feature mean and variance), while sharing the affine parameters (weight and bias) – the parameters being optimized.

4. Experiments

We test the effectiveness of our method on a wide variety of tasks (robustness against image corruptions and domain adaptation). Especially, we performed extensive experiments on various batch sizes of the test-time data batches (as small as 2 and as large as 200). The highlight of our empirical findings is that: **TIPI outperforms other test-time adaptation baselines in all scenarios and problem settings.** We discuss these experiments in more detail in the following subsections. As a side note, we also found that using TIPI with TENT further improves the performance of both methods, and helps TENT avoid network collapse. Details of our experiments (such as learning rate) can be found in our publicly available source code.

4.1. Evaluation Protocol and Baselines:

Evaluation Protocol: We consider a setting where the model is trained normally on the source domain. When the models are available online (e.g., ImageNet-trained Resnet50), we download the weights and keep them as is. Otherwise, we train the models ourselves. Following TENT [34], we perform online adaptation, and report the average performance across all batches (while the model is being adapted). This mimics a real-world scenario.

Baselines: The main purpose of our experiments is to compare our invariance regularizer against predictive entropy as an unsupervised surrogate subjective. Therefore, **TENT** [34] is our main baseline in all of the experiments. We also consider recent and relevant fully test-time adaptation methods as our baselines, such as BatchNorm with test-time statistics (**BN**) [24, 32] and **T3A** [12]. We also report the performance of the source-only model (**source**–no adaptation). Note that we also have tested T3A with test-time batch norm statistics (**T3A+BN**), but this model's performance is almost identical to (or even a little worse than) that of **BN** for all batch sizes, so we do not include it here. For a fair comparison, we rerun all methods in our experiment (to eliminate artifacts such as source model training), using their official source code. However, our numbers for these methods are very close to the reported numbers.

TTT [33] and TTT++ [19] are not fully test-time adaptation methods since they require altering the training process of the source domain (to incorporate a self-supervised task). Thus, they are not optimal for off-the-shelf models that have not been trained with the self-supervised task (e.g., a ResNet50 trained on ImageNet downloaded directly from `torchvision`). For this reason, we do not include these two methods in the comparisons. Note that as reported in Table 2 of the TTT++ paper [19], the variant of their model which has not been trained with the self-supervised task (which is the same as our evaluation protocol and most of the literature) largely underperforms our method.

EATA [29] proposes a datapoints selection strategy for TENT, and achieves state-of-the-art performance on the test time adaptation task. As discussed before, we believe this line of research is complementary to ours. Indeed, we are able to incorporate **TIPI into EATA, thereby improving its robustness** (EATA only uses TENT and is not robust against small batchsizes). Due to the page limit, we defer this comparison to the Supplementary Material.

4.2. Robustness to Image Corruptions

4.2.1 Experimental Settings

Following the literature, we conduct experiments on widely-used image-corruption datasets, namely:

- **CIFAR10-C** and **CIFAR100-C** [10]: Common image corruptions are applied to the CIFAR10 and CIFAR100 datasets [14]. We report the results for the highest level of severity, averaging over 15 types of corruption.
- **ImageNet-C**: Similar with the above two datasets, but for ImageNet [30]. We report the performance of all methods averaged over 15 types of corruption and 5 levels of severity.

For the CIFAR10-C and CIFAR100-C datasets, we use a WideResnet28-10 [36] model, which is the base model in Robustbench [4] and also used in TENT's official GitHub repository[1]. Note that in the original paper, TENT conducts experiments with a Resnet26 model [9]; however, the model/code was not publicly released, and there have been numerous reproducibility issues as reported in their GitHub page[2]. We use the Adam optimizer [13] for these two datasets.

For the ImageNet-C dataset, we follow previous works and use a Resnet50 network. We use SGD to optimize our invariance regularizer.

4.2.2 Results

Table 1, 2 and 3 show that our method is the best performer among the baselines. In the CIFAR10-C and CIFAR100-C experiments, our method performs similarly with TENT for large batch sizes (200). However, as the batch size decreases, TENT collapses and TIPI outperforms it significantly. For the challenging ImageNet-C dataset, our method performs better than TENT even for large batch sizes. TENT still collapses with small batch sizes while TIPI is very robust to this parameter. T3A is not an optimization-based adaptation approach so its performance is robust against the batch size, but it largely underperforms TIPI in all settings. Even more concerning, T3A performs extremely purely on ImageNet-C (worse than the source

[1] https://github.com/DequanWang/tent
[2] https://github.com/DequanWang/tent/issues

Table 1. **CIFAR10-C**: Results are averaged over 15 types of corruption with the highest level of severity

Method	Accuracy						
Source	56.5						
	Test Time Adaptation Batch Size						
	2	5	10	20	50	200	Average
T3A	57.8	57.8	57.8	57.8	57.9	57.9	57.8
BN	61.8	70.8	75.1	77.5	78.9	79.6	74.0
TENT	17.4	60.7	74.0	78.6	81.0	81.4	65.5
TIPI (ours)	**78.6**	**79.8**	**80.3**	**81.0**	**81.2**	**81.5**	**80.4**

Table 2. **CIFAR100-C**: Results are averaged over 15 types of corruption with the highest level of severity

Method	Accuracy						
Source	37.5						
	Test Time Adaptation Batch Size						
	2	5	10	20	50	200	Average
T3A	39.0	39.0	39.0	39.0	39.0	39.0	39.0
BN	26.0	48.0	54.1	57.4	59.5	60.7	50.9
TENT	1.5	7.4	39.8	59.9	64.4	65.7	39.8
TIPI (ours)	**56.3**	**62.3**	**64.1**	**65.0**	**65.5**	**65.8**	**63.2**

Table 3. **ImageNet-C**: Results are averaged over 15 types of corruption and 5 levels of severity

Method	Accuracy						
Source	39.5						
	Test Time Adaptation Batch Size						
	2	4	8	16	32	64	Average
T3A	30.2	30.2	30.2	30.3	30.3	30.3	30.3
BN	13.9	33.4	42.8	46.9	48.9	49.9	47.2
TENT	0.8	13.1	44.3	53.7	56.8	57.3	45.2
TIPI (ours)	**45.2**	**53.3**	**56.7**	**57.9**	**58.3**	**58.4**	**55.0**

model) – we conjecture this is because the non-parametric classifier in T3A is not effective when there is a large number of classes (1000) and each class's support set has a few datapoints (50).

4.3. Domain Adaptation

4.3.1 Experimental Settings

In this problem, we consider two main adaptation tasks, with the following two datasets:

- **DIGITS**: the task is digit classification (10 classes), and we have to adapt a model from SVHN [25] to MNIST [15], MNIST-M [6], and USPS [11]. For the MNIST and MNIST-M datasets, we only use the test set (10000 images) for the online adaptation. For the USPS dataset, we use both the training and test set to have roughly the same number of images for adaptation (9298 in total).

- **VisDA17** (12-class classification)[3]: is a more challenging dataset, with the adaption task from synthetic to real images. The source dataset contains 152397 3D-rendered synthetic images, while the target dataset contains 55388 real images.

Following the experiments performed in TENT, we use a ResNet26 for DIGITS and a ResNet50 for the VisDA17 dataset. For the test-time optimization of our surrogate loss, we use Adam for DIGITS and SGD for VisDA17. Other details such as the learning rate can be found in the provided source code.

[3]Oxford solely obtained and processed the VisDA17 dataset

Table 4. **DIGITS**: Online Domain Adaptation from SVHN to MNIST, MNIST-M and USPS

Method	SVHN→MNIST		SVHN→MNIST-M		SVHN→USPS	
	B.Size 4	B.Size 128	B.Size 4	B.Size 128	B.Size 4	B.Size 128
Source	85.8		60.3		84.9	
T3A	**86.3**	86.3	60.7	60.7	84.9	84.9
BN	75.6	83.5	56.0	60.9	72.4	80.8
TENT	27.0	90.7	19.2	61.4	38.7	90.4
TIPI (ours)	**86.3**	**91.6**	**64.2**	**63.8**	**86.8**	**93.9**

Table 5. **VisDA17**: Online Domain Adaptation of the VisDA classification challenge

Method	Accuracy							
Source	46.4							
	Test Time Adaptation Batch Size							
	2	4	8	16	32	64	128	Average
T3A	48.8	48.8	48.9	48.9	48.9	48.9	48.9	48.9
BN	38.3	49.5	54.6	57.3	58.7	59.4	59.8	53.9
TENT	12.3	26.5	53.0	59.9	62.1	62.0	62.0	48.3
TENTx2	10.0	22.1	45.9	59.2	62.5	63.1	63.2	46.6
TIPI (ours)	**51.2**	**60.3**	**62.7**	**64.0**	**64.3**	**64.6**	**64.5**	**61.7**

4.3.2 Results

Table 4 and Table 5 show a similar trend as observed in the image corruption experiments. In particular, TIPI performs the best among all the baselines. Especially, our method does not collapse with small batch sizes as most other models (such as BN or TENT) do. It is clear that TIPI achieves preferable performance in all settings while being much more robust compared to TENT.

4.4. Runtime and memory consumption

We would also like to discuss the runtime and memory (VRAM) consumption of our method. As TIPI is also an optimization-based adaptation, it is natural to compare it against TENT. Comparing our implementation of TIPI with TENT's official implementation, we observe that TIPI and TENT require roughly the same amount of VRAM, while our method is about two times slower than TENT (because it needs to find the adversarial examples). We believe that this is worth the trade-off, since our method clearly outperforms TENT, and is much more robust against small batch sizes – which is often the case in practice. To be more specific, using an NVIDIA Quadro RTX 6000, TIPI can reach 160 FPS (including both adaptation and prediction) with the standard 224x224 images of ImageNet and VisDA17.

However, just to be completely fair, we also compare TIPI with a variant of TENT where we perform two adaptation steps for each batch (thus its runtime is roughly the same as TIPI's), namely TENTx2 (second-last row in Table 5). We perform this comparison on the challenging adaptation dataset VisDA17 (for smaller datasets such as CIFAR and DIGITS, runtime and memory consumption are almost never a problem). TENTx2 does perform better than TENT (in the large-batch-size regime), but is still largely inferior to TIPI. Additionally, the collapsing problem for small batch sizes persists in this variant.

5. Conclusion

In this paper, we propose to use transformation invariance as an unsupervised surrogate loss function for online domain adaptation. Similar to TENT, our method (TIPI) is task-agnostic and does not require altering the source training process of the model, thus can be applied easily to a wide variety of off-the-shelf models. Remarkably, compared to TENT, our method is much more robust against small batch sizes, which proves to be extremely useful in challenging real-world scenarios. A promising future research direction is to investigate a data point selection strategy for optimizing the TIPI objective, to further improve the performance of the method.

Acknowledgments Author A. Tuan Nguyen acknowledges Meta AI for funding his PhD study. Oxford lab headed by author Philip Torr is supported by the UKRI grant: Turing AI Fellowship EP/W002981/1. Meta AI author Ser-Nam Lim is neither supported by the UKRI grant nor has any relationship to the grant.

References

[1] Yogesh Balaji, Swami Sankaranarayanan, and Rama Chellappa. Metareg: Towards domain generalization using meta-regularization. *Advances in neural information processing systems*, 31, 2018. 2

[2] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79(1):151–175, 2010. 2

[3] David Berthelot, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Kihyuk Sohn, Han Zhang, and Colin Raffel. Remixmatch: Semi-supervised learning with distribution alignment and augmentation anchoring. *arXiv preprint arXiv:1911.09785*, 2019. 3

[4] Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Edoardo Debenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. Robustbench: a standardized adversarial robustness benchmark. *arXiv preprint arXiv:2010.09670*, 2020. 3, 6

[5] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*, 2020. 1

[6] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030, 2016. 2, 7

[7] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. 5

[8] Sachin Goyal, Mingjie Sun, Aditi Raghunathan, and Zico Kolter. Test-time adaptation via conjugate pseudo-labels. *arXiv preprint arXiv:2207.09640*, 2022. 2

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 6

[10] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019. 6

[11] J. J. Hull. A database for handwritten text recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):550–554, 1994. 7

[12] Yusuke Iwasawa and Yutaka Matsuo. Test-time classifier adjustment module for model-agnostic domain generalization. *Advances in Neural Information Processing Systems*, 34:2427–2440, 2021. 2, 6

[13] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[14] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 6

[15] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010. 7

[16] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy Hospedales. Learning to generalize: Meta-learning for domain generalization. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018. 1

[17] Ya Li, Xinmei Tian, Mingming Gong, Yajing Liu, Tongliang Liu, Kun Zhang, and Dacheng Tao. Deep domain generalization via conditional invariant adversarial networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 624–639, 2018. 1, 2

[18] Jian Liang, Dapeng Hu, and Jiashi Feng. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *International Conference on Machine Learning*, pages 6028–6039. PMLR, 2020. 2, 5

[19] Yuejiang Liu, Parth Kothari, Bastien van Delft, Baptiste Bellot-Gurlet, Taylor Mordan, and Alexandre Alahi. Ttt++: When does self-supervised test-time training fail or thrive? *Advances in Neural Information Processing Systems*, 34:21808–21820, 2021. 1, 2, 6

[20] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017. 5

[21] Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain adaptation: Learning bounds and algorithms. *arXiv preprint arXiv:0902.3430*, 2009. 2

[22] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1979–1993, 2018. 3, 5

[23] Krikamol Muandet, David Balduzzi, and Bernhard Schölkopf. Domain generalization via invariant feature representation. In *International Conference on Machine Learning*, pages 10–18. PMLR, 2013. 2

[24] Zachary Nado, Shreyas Padhy, D Sculley, Alexander D'Amour, Balaji Lakshminarayanan, and Jasper Snoek. Evaluating prediction-time batch normalization for robustness under covariate shift. *arXiv preprint arXiv:2006.10963*, 2020. 2, 6

[25] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011. 7

[26] A Tuan Nguyen, Ser-Nam Lim, and Philip HS Torr. FedSR: A simple and effective domain generalization method for federated learning. 2022. 2

[27] A Tuan Nguyen, Toan Tran, Yarin Gal, and Atilim Gunes Baydin. Domain invariant representation learning with domain density transformations. *Advances in Neural Information Processing Systems*, 34:5264–5275, 2021. 1, 2

[28] A Tuan Nguyen, Toan Tran, Yarin Gal, Philip HS Torr, and Atılım Güneş Baydin. Kl guided domain adaptation. *arXiv preprint arXiv:2106.07780*, 2021. 2

[29] Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Yaofo Chen, Shijian Zheng, Peilin Zhao, and Mingkui Tan. Efficient test-time model adaptation without forgetting. *arXiv preprint arXiv:2204.02610*, 2022. 2, 6, 11

[30] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015. 6

[31] Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. *Advances in neural information processing systems*, 29, 2016. 3

[32] Steffen Schneider, Evgenia Rusak, Luisa Eck, Oliver Bringmann, Wieland Brendel, and Matthias Bethge. Improving robustness against common corruptions by covariate shift adaptation. *Advances in Neural Information Processing Systems*, 33:11539–11551, 2020. 2, 6

[33] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *International conference on machine learning*, pages 9229–9248. PMLR, 2020. 1, 2, 6

[34] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. *arXiv preprint arXiv:2006.10726*, 2020. 1, 2, 3, 5, 6

[35] Cihang Xie, Mingxing Tan, Boqing Gong, Jiang Wang, Alan L Yuille, and Quoc V Le. Adversarial examples improve image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 819–828, 2020. 5

[36] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016. 6

A. Derivations and Proofs

A.1. Assumption 1

First of all, the formulas for the two mutual information terms are:

$$I_T(x,y) = \int_{x\in\mathcal{X}} \sum_{y\in\mathcal{Y}} p_T(x,y) \log \frac{p_T(x,y)}{p_T(x)p_T(y)},$$

$$I_T(x^t,y) = \int_{x^t\in\mathcal{X}} \sum_{y\in\mathcal{Y}} p_T(x^t,y) \log \frac{p_T(x^t,y)}{p_T(x^t)p_T(y)}.$$

Recall that Assumption 1 states:

$$\begin{aligned}
& I_T(x,y) = I_T(x^{tr},y) \\
\Leftrightarrow & \mathbb{E}_{p_T(x,y)}\left[\log \frac{p_T(x,y)}{p_T(x)p_T(y)}\right] \\
& = \mathbb{E}_{p_T(x^{tr},y)}\left[\log \frac{p_T(x^{tr},y)}{p_T(x^{tr})p_T(y)}\right] \\
\Leftrightarrow & \mathbb{E}_{p_T(x,y)}\left[\log \frac{p_T(y|x)}{p_T(y)}\right] \\
& = \mathbb{E}_{p_T(x^{tr},y)}\left[\log \frac{p_T(y|x^{tr})}{p_T(y)}\right] \\
& = \mathbb{E}_{p_T(x^{tr},y)}\left[\log \frac{p_T(x^{tr},y)}{p_T(x^{tr})p_T(y)}\right] \\
\Leftrightarrow & \mathbb{E}_{p_T(x,x^{tr},y)}\left[\log \frac{p_T(y|x)}{p_T(y)}\right] \\
& = \mathbb{E}_{p_T(x,x^{tr},y)}\left[\log \frac{p_T(y|x^{tr})}{p_T(y)}\right] \\
\Leftrightarrow & \mathbb{E}_{p_T(x,x^{tr},y)}\left[\log p_T(y|x) - \log p_T(y|x^{tr})\right] = 0 \\
\Leftrightarrow & \mathbb{E}_{p_T(x,x^{tr})}\left[\mathbb{E}_{p_T(y|x)}\left[\log p_T(y|x) - \log p_T(y|x^{tr})\right]\right] \\
& = 0 \\
\Leftrightarrow & \mathbb{E}_{p_T(x,x^{tr})}\left[\mathrm{KL}[p_T(y|x), p_T(y|x^{tr})]\right] = 0
\end{aligned}$$

A.2. Proposition 1

Note that our result also holds if l is other distance/divergence (such as the Jensen Shannon divergence), only with a modified coefficient of the regularizer term. However, in this paper, we consider l as the l_1 distance.

Proof. First of all, due to Pinsker's inequality, we have that:

$$\begin{aligned}
& l[p_T(y|x), p_T(y|x^{tr})] \leq \sqrt{2\mathrm{KL}[p_T(y|x), p_T(y|x^{tr})]} \\
\Rightarrow & \mathbb{E}_{p_T(x,x^{tr})}\left[l[p_T(y|x), p_T(y|x^{tr})]\right] \\
& \leq \mathbb{E}_{p_T(x,x^{tr})}\left[\sqrt{2\mathrm{KL}[p_T(y|x), p_T(y|x^{tr})]}\right] \\
& \leq \sqrt{\mathbb{E}_{p_T(x,x^{tr})}\left[2\mathrm{KL}[p_T(y|x), p_T(y|x^{tr})]\right]} \\
& = 0 \\
\Rightarrow & \mathbb{E}_{p_T(x,x^{tr})}\left[l[p_T(y|x), p_T(y|x^{tr})]\right] = 0
\end{aligned}$$

We have:

$$\begin{aligned}
& \ell(\theta, p_T(x,y)) - \ell(\theta, p_T(x^{tr},y)) \\
= & \mathbb{E}_{p_T(x)}\left[l[p_T(y|x), p_\theta(y|x)]\right] \\
& \quad - \mathbb{E}_{p_T(x^{tr})}\left[l[p_T(y|x^{tr}), p_\theta(y|x^{tr})]\right] \\
= & \mathbb{E}_{p_T(x,x^{tr})}\left[l[p_T(y|x), p_\theta(y|x)]\right] \\
& \quad - \mathbb{E}_{p_T(x,x^{tr}}\left[l[p_T(y|x^{tr}), p_\theta(y|x^{tr})]\right] \\
& \quad - \mathbb{E}_{p_T(x,x^{tr})}\left[l[p_T(y|x), p_T(y|x^{tr})]\right] \\
\leq & \mathbb{E}_{p_T(x,x^{tr})}\left[l[p_\theta(y|x^{tr}), p_\theta(y|x)]\right]
\end{aligned}$$

This is because l satisfies the triangle inequality, so $l[p_T(y|x), p_\theta(y|x)] - l[p_T(y|x^{tr}), p_\theta(y|x^{tr})] - l[p_T(y|x), p_T(y|x^{tr})] \leq l[p_\theta(y|x^{tr}), p_\theta(y|x)]$.

Using Pinsker's inequality again, we have that:

$$\begin{aligned}
& \mathbb{E}_{p_T(x,x^{tr})}\left[l[p_\theta(y|x^{tr}), p_\theta(y|x)]\right] \\
\leq & \mathbb{E}_{p_T(x,x^{tr})}\left[\sqrt{2\mathrm{KL}[p_\theta(y|x^{tr}), p_\theta(y|x)]}\right] \\
\leq & \mathbb{E}_{p_T(x)}\left[\sqrt{2 \max_{x^{tr}\sim T(x^{tr}|x)} \mathrm{KL}[p_\theta(y|x^{tr}), p_\theta(y|x)]}\right]
\end{aligned}$$

and

$$\begin{aligned}
& \mathbb{E}_{p_T(x,x^{tr})}\left[l[p_\theta(y|x^{tr}), p_\theta(y|x)]\right] \\
\leq & \mathbb{E}_{p_T(x,x^{tr})}\left[\sqrt{2\mathrm{KL}[p_\theta(y|x), p_\theta(y|x^{tr})]}\right] \\
\leq & \mathbb{E}_{p_T(x)}\left[\sqrt{2 \max_{x^{tr}\sim T(x^{tr}|x)} \mathrm{KL}[p_\theta(y|x), p_\theta(y|x^{tr})]}\right]
\end{aligned}$$

This concludes our proof. □

B. Additional Experiments

B.1. Incorporating TIPI into EATA [29]

EATA [29] proposes a datapoints selection strategy for TENT, and achieves state-of-the-art performance on the test time adaptation task. As discussed before, this line of research (datapoint selection for surrogate optimization) is complementary to ours. Indeed, in this subsection, we show that TIPI can be incorporated into EATA, thereby improving EATA's robustness (EATA only uses TENT and is not robust against small batchsizes). We refer the readers to [29] for a detailed discussion on their datapoint selection strategy.

We conduct the experiment with ETA, the variation without weight regularization (ETA outperforms EATA for out-of-distribution data). Specifically, we compare ETA to ETA+TIPI, a variant of ETA with the same datapoint selection strategy, but with the TIPI surrogate objective. Following [29], we conduct the experiment on ImageNet-C with the highest level of severity. The 15 types of corruption

Table 6. **ImageNet-C, Batchsize 64**: Results for 15 types of corruption with the highest level of severity

Method	Classification Error (lower is better)															
	Gauss.	Shot	Impul.	Defoc.	Glass	Motion	Zoom	Snow	Frost	Fog	Brit.	Contr.	Elastic	Pixel	JPEG	Average
Source	97.0	96.3	97.4	82.1	90.3	85.3	77.5	83.4	76.9	76	40.9	94.6	83.5	79.1	67.4	81.8
ETA	64.9	62.1	63.4	66.1	67.1	52.2	47.4	48.1	54.2	39.9	32.1	55.0	42.1	39.1	45.1	51.9
ETA+TIPI	**63.0**	**60.9**	**62.1**	**65.2**	**65.4**	**51.6**	**46.7**	**47.6**	**53.7**	**39.9**	**31.6**	**54.1**	**41.2**	**38.2**	**43.6**	**51.0**

Table 7. **ImageNet-C, Batchsize 2**: Results for 15 types of corruption with the highest level of severity

Method	Classification Error (lower is better)															
	Gauss.	Shot	Impul.	Defoc.	Glass	Motion	Zoom	Snow	Frost	Fog	Brit.	Contr.	Elastic	Pixel	JPEG	Average
Source	97.0	96.3	97.4	**82.1**	90.3	85.3	77.5	83.4	76.9	76	40.9	94.6	83.5	79.1	67.4	81.8
ETA	97.7	97.4	97.5	97.8	97.9	95.1	93.3	92.2	92.3	89.2	78.4	96.4	90.2	90.1	92.3	93.2
ETA+TIPI	**87.8**	**88.8**	**87.4**	88.3	**89.3**	**82.4**	**70.5**	**73.2**	**75.8**	**60.7**	**40.8**	**93.9**	**64.0**	**58.8**	**65.4**	**75.4**

are: gaussian noise, shot noise, impulse noise, defocus blur, glass blur, motion blur, zoom blur, snow, frost, fog, brightness, contrast, elastic transform, pixelate, and jpeg compression.

Table 6 and Table 7 show the results of this comparison for batch sizes 64 and 2. For a large batch size, ETA+TIPI further improves over ETA, pushing a new SOTA result. For a small batch size, incorporating the TIPI objective help to stabilize the optimization, thus avoiding network collapse.

Furthermore, note that we keep the datapoint selection strategy in EATA as is in this experiment. This strategy is developed specifically for the TENT objective and we did not make any modifications for TIPI. Yet, using it for TIPI already improves the model's performance. This shows that indeed the two lines of research are complementary. It also suggests that investigating a datapoint selection strategy tailored for TIPI is a promising direction, and could potentially improve TIPI's performance even further.