TENT: FULLY TEST-TIME ADAPTATION BY ENTROPY MINIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

A model must adapt itself to generalize to new and different data during testing. This is the setting of fully test-time adaptation given only unlabeled test data and the model parameters. We propose test-time entropy minimization (tent): we optimize for model confidence as measured by the entropy of its predictions. During testing, we adapt the model features by estimating normalization statistics and optimizing channel-wise affine transformations. Tent improves robustness to corruptions for image classification on ImageNet and CIFAR-10/100 and achieves state-of-the-art error on ImageNet-C for ResNet-50. Tent shows the feasibility of target-only domain adaptation for digit classification from SVHN to MNIST/MNIST-M/USPS and semantic segmentation from GTA to Cityscapes.

1 INTRODUCTION

Deep networks can achieve high accuracy on training and testing data from the same distribution, as evidenced by tremendous benchmark progress (Krizhevsky et al., 2012; Simonyan & Zisserman, 2015; He et al., 2016). However, generalization to new and different data is limited (Hendrycks & Dietterich, 2019; Recht et al., 2019; Geirhos et al., 2018). Accuracy suffers when the training (source) data differ from the testing (target) data, a condition known as *dataset shift* (Quionero-Candela et al., 2009). Models can be sensitive to shifts during testing that were not known during training, whether natural variations or corruptions, such as unexpected weather or sensor degradation. Nevertheless, it can be necessary to deploy a model on different data distributions, so adaptation is needed.

During testing, the model must adapt given only its parameters and the target data. This *fully test-time adaptation* setting cannot rely on source data or supervision. Neither is practical when the model first encounters new testing data, before it can be collected and annotated, as inference must go on. Real-world usage motivates fully test-time adaptation by data, computation, and task needs:

1. Availability. A model might be distributed without source data for bandwidth, privacy, or profit.
2. Efficiency. It might not be computationally practical to (re-)process source data during testing.
3. Accuracy. A model might be too inaccurate without adaptation to serve its purpose.

We take the entropy of model predictions during testing as our adaptation objective. We call this the test entropy and name our method *tent* after it. Entropy is related to error, as more confident predictions are all-in-all more correct (Figure 1). On corrupted data, entropy is an indicator of the severity of the corruption, with a strong rank correlation to the loss for image classification (Figure 2).

To minimize entropy, tent normalizes and transforms the model features on target data by estimating statistics and optimizing affine parameters. This choice of low-dimensional, channel-wise feature modulation is efficient to update during testing, even for online adaptation. Tent does not restrict or alter model training: it is independent of the source data given the model parameters. If the model can be run, it can be adapted. Most importantly, tent effectively reduces not just entropy but error.

Our results evaluate robustness to common image corruptions and accuracy under domain shift for digit recognition. For reference results with more data and optimization, we evaluate methods for robust training, domain adaptation, and self-supervised learning given the labeled training data. Tent achieves less error given only the test data. Our analysis supports entropy as an objective, backs the generality of tent across architectures, and varies the amount of parameters and data for adaptation.

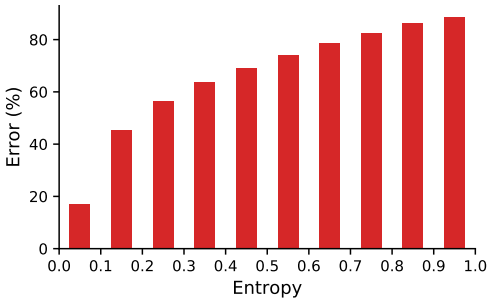


Figure 1: Predictions with lower entropy have lower error rates on corrupted CIFAR-100-C. Certainty can serve as supervision during testing.

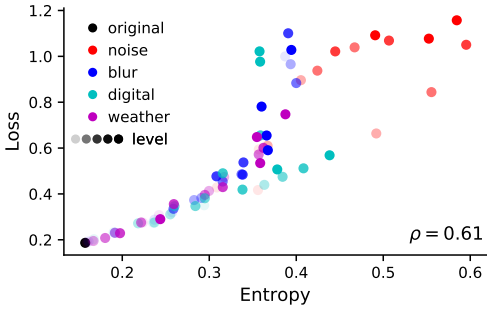


Figure 2: More corruption causes higher loss and entropy for the model on this data. Entropy can measure test shift without training data or labels.

Our contributions

- We highlight the setting of fully test-time adaptation with only target data and no source data. We suggest benchmarking adaptation with offline and online use of target data.
- We examine entropy as an adaptation objective, and propose tent: a test-time entropy minimization scheme to reduce generalization error by reducing the entropy of model predictions.
- For robustness to corruptions, tent reaches 44.0% error on ImageNet-C, better than the state-of-the-art for robust training (49.6%) and the strong baseline of test-time normalization (51.7%).
- For domain adaptation, tent is capable of target-only adaptation for digit classification and semantic segmentation, and even rivals methods that use source data and more optimization on SVHN→MNIST/USPS.

2 Setting: Fully Test-Time Adaptation

Adaptation addresses generalization from source to target. A model $f_\theta(x)$ with parameters θ trained on source data and labels x^s, y^s may not generalize when tested on shifted target data x^t. Table 1 summarizes adaptation settings, their required data, and types of losses. Our fully test-time adaptation setting uniquely requires only the model f_θ and unlabeled target data x^t.

Existing adaptation settings extend training given more data and supervision. Transfer learning by fine-tuning (Donahue et al., 2014; Yosinski et al., 2014) needs target labels to (re-)train with a supervised loss $L(x^t, y^t)$. Without target labels, our setting denies this supervised training. Domain adaptation (DA) (Quionero-Candela et al., 2009; Saenko et al., 2010; Ganin & Lempitsky, 2015; Tzeng et al., 2015) needs both the source and target data to train with a cross-domain loss $L(x^s, x^t)$. Test-time training (TTT) (Sun et al., 2019b) needs the source data to jointly train with a supervised loss and an unsupervised loss $L(x^s)$. Without source, our setting denies source supervision $L(x^s, y^s)$ for joint training across domains (DA) or losses (TTT). These settings have their purposes, but do not cover all practical cases when source, target, or supervision are not simultaneously available.

Unexpected target data during testing requires test-time adaptation. TTT and our setting adapt the model by optimizing an unsupervised loss during testing $L(x^t)$. During training, TTT jointly optimizes this same loss on source data $L(x^s)$ with a supervised loss $L(x^s, y^s)$, to ensure the parameters θ are shared across losses for compatibility with adaptation by $L(x^t)$. Fully test-time adaptation is independent of the training data and training loss given the parameters θ. By not changing training, our setting has the potential to require less data and computation for adaptation.

3 Method: Test Entropy Minimization via Feature Modulation

We adapt the model through test-time optimization to minimize the entropy of its predictions by modulating its features. We call this adaptation method *tent*, for test entropy. Tent requires a compatible model, an objective to minimize (Section 3.1), and parameters to optimize over (Section 3.2) to fully define the algorithm (Section Section 3.3). Figure 3 outlines our fully test-time adaptation method.

Table 1: Adaptation settings differ by their data and therefore losses during training and testing. Of the source ^{s} and target ^{t} data x and labels y, our fully test-time setting only needs the target data x^t.

setting	source data	target data	train loss	test loss
fine-tuning	-	x^t, y^t	$L(x^t, y^t)$	-
domain adaptation	x^s, y^s	x^t	$L(x^s, y^s) + L(x^s, x^t)$	-
test-time training	x^s, y^s	x^t	$L(x^s, y^s) + L(x^s)$	$L(x^t)$
fully test-time adaptation	-	x^t	-	$L(x^t)$

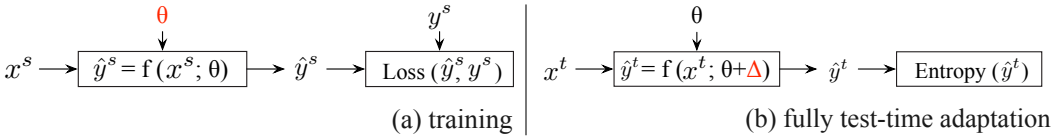


Figure 3: Method overview. Tent does not alter training (a), but minimizes entropy during testing (b) over a constrained modulation Δ, given the model parameters θ and unlabeled target data x^t.

Compatibility The model must be trained for the supervised task, probabilistic, and differentiable. No supervision is provided during testing, so the model must already be trained. Measuring the entropy of predictions requires a distribution over predictions, so the model must be probabilistic. Gradients are required for fast iterative optimization, so the model must be differentiable.

Choice of Model and Task We choose image classifiers as representative models for supervised learning with deep networks. These models are probabilistic by their prediction of softmax class distributions and end-to-end differentiable by their design.

3.1 Entropy Objective

Our test-time objective $L(x_t)$ is the entropy $H(\hat{y})$ of the model prediction during testing $\hat{y} = f_\theta(x_t)$. In particular, we measure the Shannon entropy (Shannon, 1948), $H(\hat{y}) = -\sum_c p(\hat{y}_c) \log p(\hat{y}_c)$ for the probability $\hat{y}_c$ of class c. As a measure of the task predictions, $H(\hat{y} = f_\theta(x))$ is a function of the model parameters, and therefore the supervised training. In this way entropy is task-general, as it is defined for any probabilistic task, but at the same time task-specific, as it varies with the task training.

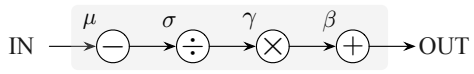
In contrast, proxy tasks for self-supervised learning are not directly related to the supervised task. Proxy tasks derive a self-supervised label y' from the input x_t without the task label y. For concreteness, examples of these proxies include rotation prediction (Gidaris et al., 2018), context prediction (Doersch et al., 2015), and cross-channel auto-encoding (Zhang et al., 2017). Too much progress on a proxy task could interfere with performance on the supervised task, and self-supervised adaptation methods have to limit or mix updates accordingly (Sun et al., 2019b;a). As such, care is needed to choose a proxy compatible with the domain and task, to design the architecture for the proxy model, and to balance optimization between the task and proxy objectives.

Our test entropy objective does not present these difficulties. Entropy is an unsupervised objective that depends only on $\hat{y}$. No further effort or choice is needed to adopt it. On the other hand, supervised learning and self-supervised learning take more effort in annotation or design choices.

3.2 Modulation Parameters

The model parameters θ are a natural choice for test-time optimization, and these are the choice of prior work for train-time entropy minimization in semi-supervised (Grandvalet & Bengio, 2005), few-shot (Dhillon et al., 2020), and domain adaptation (Carlucci et al., 2017) regimes. However, θ is the only representation of the training/source data in our setting, and altering θ could cause the model to drift from the training task. Furthermore, f can be nonlinear and θ can be high dimensional, making optimization too sensitive and inefficient for test-time usage.

For stability and efficiency, we instead only update feature modulations that are linear (scales and shifts), and low-dimensional (channel-wise). Figure 4 shows the two steps of our modulations: normalization by statistics and transformation by parameters. Normalization centers and standardizes



normalization $\mu \leftarrow \mathbb{E}[x_t], \sigma^2 \leftarrow \mathbb{E}[(\mu - x_t)^2]$

transformation $\gamma \leftarrow \gamma + \partial H / \partial \gamma, \beta \leftarrow \beta + \partial H / \partial \beta$

Figure 4: Tent modulates features during testing by estimating normalization statistics μ, σ and optimizing transformation parameters γ, β. Normalization and transformation apply channel-wise scales and shifts to the features. The statistics and parameters are updated on target data without use of source data. In practice, adapting γ, β is efficient because they make up <1% of model parameters.

the input x into $\bar{x} = (x - \mu)/\sigma$ by its mean μ and standard deviation σ. Transformation turns $\bar{x}$ into the output $x' = \gamma \bar{x} + \beta$ by affine parameters for scale γ and shift β. Note that the statistics μ, σ are estimated from the data while the parameters γ, β are optimized by the loss.

For implementation, we simply repurpose the normalization layers of the source model by updating their normalization statistics and affine parameters during testing for all layers and channels.

3.3 Algorithm

Initialization The optimizer collects the affine transformation parameters $\{\gamma_{l,k}, \beta_{l,k}\}$ for each normalization layer l and channel k in the source model. The remaining parameters $\theta \setminus \{\gamma_{l,k}, \beta_{l,k}\}$ are fixed. The normalization statistics $\{\mu_{l,k}, \sigma_{l,k}\}$ from the source data are discarded.

Iteration Each step updates the normalization statistics and transformation parameters on a batch of test data. The normalization statistics are updated by moving averages on the test data for each layer in turn, during the forward pass. The transformation parameters γ, β are updated by the gradient of the prediction entropy $\nabla H(\hat{y})$, during the backward pass. For online adaptation, forward is repeated once the model is updated, to improve inference for every test point. This only needs $2\times$ the inference time plus $1\times$ the gradient time per test point vs. the $1\times$ inference time of the unadapted model.

Termination For online adaptation, no termination is necessary, and iteration continues as long as there is test data. For offline adaptation, we update for a single epoch, as it is simple and efficient. Inference is then repeated for the whole test set. Of course, it is possible to extend adaptation by continuing to optimize for multiple epochs.

4 Experiments

We evaluate tent for corruption robustness on CIFAR-10/CIFAR-100 and ImageNet, and for domain adaptation on digit adaptation from SVHN to MNIST/MNIST-M/USPS. Our implementation is in PyTorch (Paszke et al., 2019) with the the `pycls` library (Radosavovic et al., 2019). Our (anonymized) code is included with this submission, and the code will be released for publication.

Datasets We run on image classification datasets for corruption and domain adaptation conditions. For large-scale experiments we choose ImageNet (Russakovsky et al., 2015), with 1,000 classes, a training set of 1.2 million, and a validation set of 50,000. For experiments at an accessible scale we choose CIFAR-10/CIFAR-100 (Krizhevsky, 2009), with 10/100 classes, a training set of 50,000, and a test set of 10,000. For domain adaptation we choose SVHN (Netzer et al., 2011) as source and MNIST (LeCun et al., 1998)/MNIST-M (Ganin & Lempitsky, 2015)/USPS (Hull, 1994) as targets, with ten classes for the digits 0–9. SVHN has color images of house numbers from street views with a training set of 73,257 and test set of 26,032. MNIST/MNIST-M/USPS have handwritten digits with a training sets of 60,000/60,000/7,291 and test sets of 10,000/10,000/2,007.

Models For corruption we use residual networks (He et al., 2016) with 26 layers (R-26) on CIFAR-10/100 and 50 layers (R-50) on ImageNet. For domain adaptation we use the R-26 architecture. For fair comparison, all methods in each experimental condition share the same architecture.

Our networks are equipped with batch normalization (Ioffe & Szegedy, 2015). For the source model without adaptation, the normalization statistics are estimated during training on the source data. For all test-time adaptation methods, we estimate these statistics during testing on the target data, as also done in concurrent work on adaptation by normalization (Schneider et al., 2020; Nado et al., 2020).

Table 2: Corruption benchmark on CIFAR-10-C and CIFAR-100-C for the highest severity. Tent has least error, with less optimization than domain adaptation (RG, UDA-SS) and test-time training (TTT), and improves on test-time norm (BN).

Method	Source	Target	Error (%)	
			C10-C	C100-C
Source	train		40.8	67.2
RG	train	train	18.3	38.9
UDA-SS	train	train	16.7	47.0
TTT	train	test	17.5	45.0
BN		test	17.3	42.6
PL		test	15.7	41.2
Tent (ours)		test	**14.3**	**37.3**

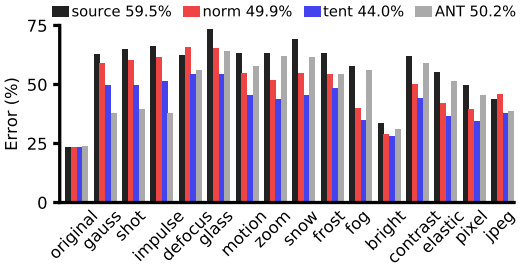


Figure 5: Corruption benchmark on ImageNet-C: error for each type averaged over severity levels. Tent improves on the prior state-of-the-art, adversarial noise training (Rusak et al., 2020), by fully test-time adaptation *without altering training*.

Optimization We optimize the modulation parameters γ, β following the training hyperparameters for the source model with few changes. On ImageNet we optimize by SGD with momentum, while on other datasets we optimize by Adam (Kingma & Ba, 2015). We lower the batch size (BS) to reduce memory usage, then lower the learning rate (LR) by the same factor to compensate (Goyal et al., 2017). On ImageNet, we set BS = 64 and LR = 0.00025, and on other datasets we set BS = 128 and LR = 0.001.We shuffle the test data to avoid ordering effects, and control for shuffling by sharing the order across methods.

Baselines We compare to domain adaptation, self-supervision, normalization, and pseudo-labeling:

- source applies the trained classifier to the test data without adaptation,
- adversarial domain adaptation (RG) reverses the gradients of a domain classifier on source and target to optimize for a domain-invariant representation (Ganin & Lempitsky, 2015),
- self-supervised domain adaptation (UDA-SS) jointly trains self-supervised rotation and position tasks on source and target to optimize for a shared representation (Sun et al., 2019a),
- test-time training (TTT) jointly trains for supervised and self-supervised tasks on source, then keeps training the self-supervised task on target during testing (Sun et al., 2019b),
- test-time normalization (BN) updates batch normalization statistics (Ioffe & Szegedy, 2015) on the target data during testing (Schneider et al., 2020; Nado et al., 2020),
- pseudo-labeling (PL) tunes a confidence threshold, assigns predictions over the threshold as labels, and then optimizes the model to these pseudo-labels before testing (Lee, 2013).

Only test-time normalization (BN), pseudo-labeling (PL), and tent (ours) are fully test-time adaptation methods. See Section 2 for an explanation and contrast with domain adaptation and test-time training.

4.1 Robustness to Corruptions

To benchmark robustness to corruption, we make use of common image corruptions (See appendix Section A for examples). The CIFAR-10/100 and ImageNet datasets are turned into the CIFAR-10/100-C and ImageNet-C corruption benchmarks by duplicating their test/validation sets and applying 15 types of corruptions at five severity levels (Hendrycks & Dietterich, 2019).

Tent improves error more with less data and computation. Table 2 reports errors averaged over corruption types at the severest level of corruption. On CIFAR-10/100-C we compare all methods, including those that require joint training across domains or losses, given the convenient size of this dataset. Adaptation is offline for fair comparison across this mixture of methods. Tent improves on the fully test-time adaptation baselines (BN, PL) but also the domain adaptation (RG, UDA-SS) and test-time training (TTT) methods that need the source data and several epochs of optimization.

Tent consistently improves error across corruption types. Figure 5 plots the errors for each corruption type averaged over levels on ImageNet-C. We compare the most efficient methods—source, normalization, and tent—given the large-scale size of the source data (>1 million images) that other methods rely on and the 75 target combinations of corruption types and levels. Tent and BN adapt online to rival the efficiency of inference without adaptation. Note that many conditions have

Table 3: Digit domain adaptation from SVHN to MNIST/MNIST-M/USPS. Target-only adaptation is not only feasible, but more efficient. Tent always improves on normalization (BN), and in 2/3 cases achieves less error than domain adaptation (RG, UDA-SS) without joint training on source & target.

Method	**Source**	**Target**	**Epochs**	**Error (%)**		
				MNIST	MNIST-M	USPS
Source	train		0	18.2	39.7	19.3
RG	train	train	10	15.0	33.4	18.9
UDA-SS	train	train	10	11.1	**22.2**	18.4
BN		test	1	15.7	39.7	18.0
Tent (ours)		test	1	10.0	37.0	16.3
Tent (ours)		test	10	**8.2**	36.8	**14.4**

over 50% error even after adaptation by normalization, but tent nevertheless further reduces the error. Tent reaches the least error for all corruptions types without harming the error on the original test set.

Tent reaches new state-of-the-art without extending training. The state-of-the-art methods for robustness extend training with adversarial noise (ANT) (Rusak et al., 2020) for 50.2% error or mixtures of data augmentations (AugMix) (Hendrycks et al., 2020) for 51.7% error. Combined with stylization from external images (SIN) (Geirhos et al., 2019), ANT+SIN reaches 47.4%. Tent reaches a new state-of-the-art of 44.0% by online adaptation and 42.3% by offline adaptation. It improves on ANT for all types except noise, on which ANT is trained. This requires just one gradient per test point, without further optimization on the training set, much less external images.

Among fully test-time adaptation methods, tent reduces the error beyond test-time normalization for 18% relative improvement. In concurrent work, Schneider et al. (2020) report 49.3% error for test-time normalization, for which tent still gives 14% relative improvement.

4.2 Target-Only Domain Adaptation

We adopt the established setting of digit adaptation (Ganin & Lempitsky, 2015; Tzeng et al., 2015; 2017; Shu et al., 2018) with adaptation from SVHN to MNIST/MNIST-M/USPS. Recall that unsupervised domain adaptation makes simultaneous use the labeled source data and unlabeled target data, while our fully test-time adaptation setting denies use of source data.

Tent adapts to target without source. Table 3 reports the target errors for domain adaptation and fully test-time adaptation methods. Test-time normalization (BN) marginally improves, while adversarial domain adaptation (RG) and self-supervised domain adaptation (UDA-SS) improve more by joint training on source and target. Tent always has lower error than the source model and BN, and it achieves the lowest error in 2/3 cases, even in just one epoch and without use of source data.

While encouraging for fully test-time adaptation, unsupervised domain adaptation remains necessary for the highest accuracy and harder shifts. For SVHN-to-MNIST, DIRT-T (Shu et al., 2018) achieves a remarkable 0.6% error [1]. For MNIST-to-SVHN, a difficult shift with source-only error of 71.3%, DIRT-T reaches 45.5% and UDA-SS reaches 38.7%. Tent fails on this shift and increases error to 79.8%. Success in these regimes presently requires joint optimization over source and target.

Tent needs less computation, but still improves with more. Tent reduces computation by (1) only using target data and (2) only taking one gradient per point. RG & UDA-SS use the source data (SVHN train), which is $\sim 7\times$ the size of the target data (MNIST test), and optimize for 10 epochs. Tent adapts with $\sim 70\times$ less computation. When tent is adapted for more epochs, its error improves further while still only using one-seventh the amount of data.

Tent scales to semantic segmentation. To show scalability to large models and inputs, we evaluate semantic segmentation (pixel-wise classification) on a domain shift from a simulated source to a real target. The source is GTA (Richter et al., 2017), a video game in an urban environment, and the target is Cityscapes (Cordts et al., 2016), an urban autonomous driving dataset. The model is HRNet-W18, a fully convolutional network (Shelhamer et al., 2017) with high-resolution architecture (Wang et al.,

[1] We exclude DIRT-T from our experiments because of incomparable differences in architecture and model selection. DIRT-T tunes with labeled target data, but we do not. Please refer to Shu et al. (2018) for more detail.

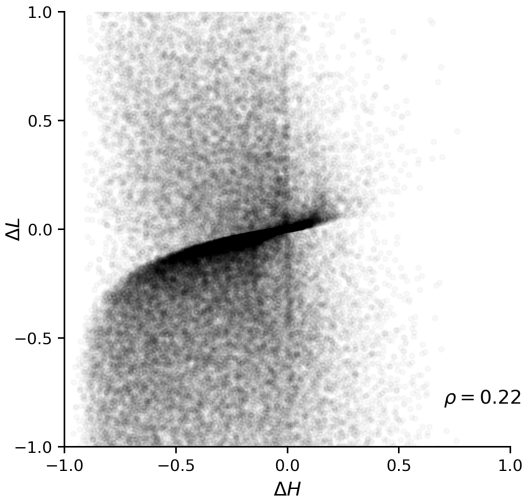


Figure 6: Tent reduces the entropy and loss. We plot changes in entropy ΔH and loss ΔL for all of CIFAR-100-C. Change in entropy rank-correlates with change in loss: note the dark diagonal and the rank correlation coefficient of 0.22.

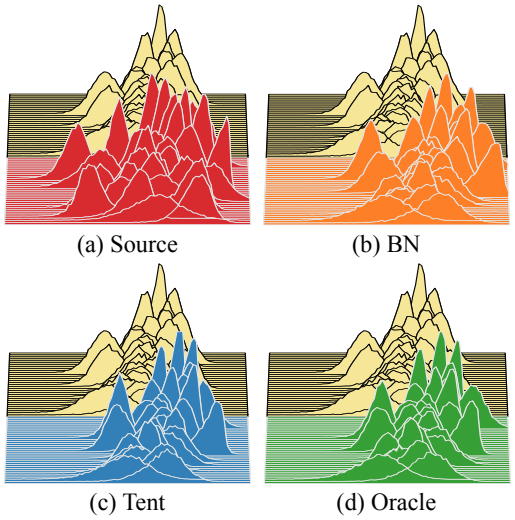


Figure 7: Adapted features on CIFAR-100-C with Gaussian noise (front) and reference features without corruption (back). Corruption shifts features from the reference, but BN lessens the difference. Tent increases the difference, shifting features closer to an oracle that optimizes on target labels.

2020). The target intersection-over-union scores (higher is better) are source 28.8%, BN 31.4%, and tent 35.8% with offline optimization by Adam. For *adaptation to a single image*, tent reaches 36.4% in 10 iterations with episodic optimization. See the appendix for a qualitative example (Section B).

4.3 Analysis

Tent reduces entropy and error. Figure 6 verifies tent does indeed reduce the entropy and the task loss (softmax cross-entropy). We plot changes in entropy and loss on CIFAR-100-C for all 75 corruption type/level combinations. Both axes are normalized by the maximum entropy of a prediction ($\log 100$) and clipped to ± 1. Most points have lower entropy and error after adaptation.

Tent needs feature modulation. We ablate the normalization and transformation steps of feature modulation. Not updating normalization increases errors, and can fail to improve over BN and PL. Not updating transformation parameters reduces the method to test-time normalization. Updating only the last layer of the model can improve but then degrades with further optimization. Updating the full model parameters θ never improves over the unadapted source model.

Tent modulation generalizes across target data. As tent simultaneously adapts and tests on target, its adaptation could be limited to these specific data points. We decouple specific adaptation to each point and general adaptation to the domain by adapting tent to target train rather than target test. Test errors drop: CIFAR-100-C error goes from 37.3% to 34.2% and SVHN-to-MNIST error goes from 8.2% to 6.5%. (Train is larger than test, so more data is resulting in less error.) The adapted modulation is not point-specific, but domain-general, as adapting on train helps predicting on test.

Tent modulation differs from normalization. Modulation normalizes and transforms, but what is the combined effect on features? Figure 7 contrasts adapted features on corrupted data against reference features on uncorrupted data. We examine features from the source model, normalization, tent, and an oracle that optimizes on the target labels. Normalization adapts features to be more like the reference, but tent does not. Instead, tent adapts features to be more like the oracle. This suggests a different, task-specific effect of test entropy minimization. See the appendix for feature visualizations of more layers (Section C).

Tent adapts alternative architectures In principle, tent is architecture agnostic. To gauge its generality, we evaluate new architectures based on self-attention (SAN) (Zhao et al., 2020) and

equilibrium solving (MDEQ) (Bai et al., 2020) for corruption robustness on CIFAR-100-C. Table 4 reports the errors for the source model, BN adaptation, and tent adaptation. Tent reduces their error with the same settings as convolutional residual networks, in spite of their distinct architectures.

Table 4: Tent adapts new architectures on CIFAR-100-C without tuning. Results are error (%).

SAN (pair)			**SAN-10 (patch)**			**MDEQ (large)**		
Source	BN	Tent	Source	BN	Tent	Source	BN	Tent
55.3	39.7	**36.7**	48.0	31.8	**29.2**	53.3	44.9	**41.7**

5 Related Work

We relate tent to existing adaptation, entropy minimization, and feature modulation methods.

Train-Time Adaptation Domain adaptation methods train a joint model of the source and target by cross-domain losses $L(x^s, x^t)$. These losses optimize feature alignment (Gretton et al., 2009; Sun et al., 2017), adversarial invariance (Ganin & Lempitsky, 2015; Tzeng et al., 2017), or shared proxy tasks (Sun et al., 2019a). While they are effective in their setting, they do not apply when joint use of source and target is denied. Tent adapts entirely to target without joint modeling of source.

Recent "source-free" methods (Li et al., 2020; Kundu et al., 2020) also adapt without source data. Both rely on generative modeling and optimize multiple models with multiple losses. Kundu et al. (2020) also alter training. Tent does not need generative modeling, nor does it alter training, making it much more computationally efficient and capable of online adaptation. We encourage the reader to examine each of these works on the frontier of adaptation without source.

To the best of our knowledge, Chidlovskii et al. (2016) are the first to motivate adaptation without source data due to legal or commercial concerns. They adapt predictions, rather than models, by applying denoising auto-encoders to the output. As such their method and experimental scope differ significantly from our end-to-end and online adaptation method.

Test-Time Adaptation Tent adapts by test-time optimization and normalization.

Test-time training (TTT) (Sun et al., 2019b) also optimizes during testing, but differs in its loss and alters training. TTT's loss relies on a proxy task, such as recognizing rotations of an image, and therefore depends on the choice of proxy. (Indeed, its authors caution that the proxy must be "both well-defined and non-trivial in the new domain"). Our test entropy loss is measured without any proxy task. Furthermore, TTT must alter training to include its proxy loss. Tent does not alter training, and so it applies to models alone withour their source data.

Normalizing feature statistics is common for domain adaptation (Gretton et al., 2009; Sun et al., 2017). For batch normalization Li et al. (2017); Carlucci et al. (2017) separate the source and target statistics during training. In concurrent work, Schneider et al. (2020); Nado et al. (2020) estimate target statistics during testing and show this improves generalization. Tent builds on test-time normalization to further reduce generalization error.

Entropy Minimization Entropy minimization is a key regularizer for domain adaptation (Carlucci et al., 2017; Shu et al., 2018; Saito et al., 2019; Roy et al., 2019), semi-supervised learning (Grandvalet & Bengio, 2005; Lee, 2013; Berthelot et al., 2019), and few-shot learning (Dhillon et al., 2020). Regularizing entropy penalizes decision boundaries at high densities in the data distribution to thereby improve accuracy for distinct classes (Grandvalet & Bengio, 2005). These methods all regularize entropy during training in concert with other supervised and unsupervised losses on additional data. Tent is the first method to minimize entropy during testing, for adaptation to corruption and domain shift, without other losses or data. Entropic losses are common; our contribution is to exhibit the effectiveness of entropy *as the sole loss* for fully test-time adaptation.

Feature Modulation Modulation adjusts a model so that it varies with its input. Following signal processing usage, the modulation is usually simpler than the model, for example in its lower dimensionality. We optimize feature modulations instead of the full model for stable and efficient adaptation. We choose modulation by channel-wise affine transformation, for its effectiveness in tandem with normalization (Ioffe & Szegedy, 2015; Wu & He, 2018), and for its flexibility to condition the model

for different tasks (Perez et al., 2018). These normalization and conditioning methods optimize the modulation during training by a supervised loss, but keep it fixed during testing. We optimize the modulation during testing by an unsupervised loss, so that it can adapt to different targets.

6 Conclusion

Tent reduces generalization error for fully-test time adaptation by entropy minimization. This is remarkable in that its entropy objective, while unsupervised, is defined by the supervised model training. In effect, it seems that the model has learned enough to supervise itself on shifted data. While there are still gaps in accuracy on corruptions and different domains, and therefore more adaptation is needed, this is an encouraging step. Improvements from tent show that a model knows more than it can infer in one go: it generalizes more by adapting to feedback from its own predictions. Our fully test-time adaptation setting and experiments should encourage more exploration of what models may already know about the data distribution, and how this can be turned into further self-improvement.

References

Shaojie Bai, Vladlen Koltun, and J Zico Kolter. Multiscale deep equilibrium models. *arXiv preprint arXiv:2006.08656*, 2020.

David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin A Raffel. Mixmatch: A holistic approach to semi-supervised learning. In *NeurIPS*, 2019.

Fabio Maria Carlucci, Lorenzo Porzi, Barbara Caputo, Elisa Ricci, and Samuel Rota Bulo. Autodial: Automatic domain alignment layers. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 5077–5085. IEEE, 2017.

Boris Chidlovskii, Stephane Clinchant, and Gabriela Csurka. Domain adaptation in the absence of source domain data. In *SIGKDD*, pp. 451–460, 2016.

Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016.

Guneet Singh Dhillon, Pratik Chaudhari, Avinash Ravichandran, and Stefano Soatto. A baseline for few-shot image classification. In *ICLR*, 2020.

Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *ICCV*, 2015.

J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *ICML*, 2014.

Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *ICML*, 2015.

Robert Geirhos, Carlos RM Temme, Jonas Rauber, Heiko H Schütt, Matthias Bethge, and Felix A Wichmann. Generalisation in humans and deep neural networks. In *NeurIPS*, 2018.

Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A. Wichmann, and Wieland Brendel. Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In *International Conference on Learning Representations*, 2019.

Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *ICLR*, 2018.

Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. In *NeurIPS*, 2005.

A. Gretton, AJ. Smola, J. Huang, M. Schmittfull, KM. Borgwardt, and B. Schölkopf. Covariate shift and local learning by distribution matching. In *Dataset Shift in Machine Learning*, pp. 131–160. MIT Press, Cambridge, MA, USA, 2009.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, June 2016.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *ICLR*, 2019.

Dan Hendrycks, Norman Mu, Ekin D Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. Augmix: A simple data processing method to improve robustness and uncertainty. In *ICLR*, 2020.

Jonathan J. Hull. A database for handwritten text recognition research. *TPAMI*, 1994.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. *NeurIPS*, 25, 2012.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Jogendra Nath Kundu, Naveen Venkat, R Venkatesh Babu, et al. Universal source-free domain adaptation. In *CVPR*, pp. 4544–4553, 2020.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Dong-Hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *ICML Workshop on challenges in representation learning*, 2013.

Rui Li, Qianfen Jiao, Wenming Cao, Hau-San Wong, and Si Wu. Model adaptation: Unsupervised domain adaptation without source data. In *CVPR*, June 2020.

Yanghao Li, Naiyan Wang, Jianping Shi, Jiaying Liu, and Xiaodi Hou. Revisiting batch normalization for practical domain adaptation. In *ICLRW*, 2017.

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.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. *NeurIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *AAAI*, 2018.

Joaquin Quionero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D Lawrence. *Dataset shift in machine learning*. MIT Press, Cambridge, MA, USA, 2009.

Ilija Radosavovic, Justin Johnson, Saining Xie, Wan-Yen Lo, and Piotr Dollár. On network design spaces for visual recognition. In *ICCV*, 2019.

Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do ImageNet classifiers generalize to ImageNet? In *ICML*, 2019.

Stephan R Richter, Zeeshan Hayder, and Vladlen Koltun. Playing for benchmarks. In *ICCV*, 2017.

Subhankar Roy, Aliaksandr Siarohin, Enver Sangineto, Samuel Rota Bulo, Nicu Sebe, and Elisa Ricci. Unsupervised domain adaptation using feature-whitening and consensus loss. In *CVPR*, 2019.

Evgenia Rusak, Lukas Schott, Roland S Zimmermann, Julian Bitterwolf, Oliver Bringmann, Matthias Bethge, and Wieland Brendel. A simple way to make neural networks robust against diverse image corruptions. In *ECCV*, 2020.

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. *IJCV*, 2015.

Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In *European conference on computer vision*, pp. 213–226. Springer, 2010.

Kuniaki Saito, Donghyun Kim, Stan Sclaroff, Trevor Darrell, and Kate Saenko. Semi-supervised domain adaptation via minimax entropy. In *ICCV*, 2019.

Steffen Schneider, Evgenia Rusak, Luisa Eck, Oliver Bringmann, Wieland Brendel, and Matthias Bethge. Improving robustness against common corruptions by covariate shift adaptation. *arXiv preprint arXiv:2006.16971*, 2020.

C.E. Shannon. A mathematical theory of communication. *Bell system technical journal*, 27, 1948.

Evan Shelhamer, Jonathan Long, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *PAMI*, 2017.

Rui Shu, Hung H Bui, Hirokazu Narui, and Stefano Ermon. A dirt-t approach to unsupervised domain adaptation. In *ICLR*, 2018.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

Baochen Sun, Jiashi Feng, and Kate Saenko. Correlation alignment for unsupervised domain adaptation. In *Domain Adaptation in Computer Vision Applications*, pp. 153–171. Springer, 2017.

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

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A Efros, and Moritz Hardt. Test-time training for out-of-distribution generalization. *arXiv preprint arXiv:1909.13231*, 2019b.

Eric Tzeng, Judy Hoffman, Trevor Darrell, and Kate Saenko. Simultaneous deep transfer across domains and tasks. In *ICCV*, 2015.

Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *CVPR*, 2017.

Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, et al. Deep high-resolution representation learning for visual recognition. *PAMI*, 2020.

Yuxin Wu and Kaiming He. Group normalization. In *ECCV*, 2018.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *NeurIPS*, 2014.

Richard Zhang, Phillip Isola, and Alexei A Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *CVPR*, 2017.

Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun. Exploring self-attention for image recognition. In *CVPR*, 2020.

APPENDIX

We report supplementary results for the method as described in Section 3. All of these results are collected with the *same method as-is*, with differences only in (1) the input and (2) the pre-trained model to be adapted.

A ROBUSTNESS TO CORRUPTIONS

In Section 4.1 we evaluate methods on a common image corruptions benchmark. In Table 2 we report results on the most severe level of corruption, level 5. In this appendix, we include examples of these image corruptions, and report results across corruption types and levels for completeness.

Example Corruptions We summarize the image corruption types for the benchmark in Figure 8.

Varying Severity Error rates for all corruption types and all levels 1–5 are reported in Tables 5, 6, 7, 8, and 9. The model and optimization details follow Section 4.1, and only the level of input corruption differs.

Table 5: Corruption benchmark measuring percentage error on CIFAR-10, CIFAR-100, and ILSVRC at severity level 1 (least severe).

	gauss	shot	impulse	defocus	glass	motion	zoom	snow	frost	fog	bright	contrast	elastic	pixelate	jpeg
CIFAR-10															
source-only	21.1	13.3	14.6	4.4	42.9	9.0	11.6	8.3	8.7	4.4	4.2	4.7	8.6	5.9	12.3
TTT	9.8	8.1	8.6	7.1	24.8	9.8	16.1	10.1	8.3	7.0	6.7	7.1	10.0	8.0	11.1
batch norm	9.4	7.1	8.9	4.0	21.5	5.5	5.9	7.2	6.3	4.3	4.0	4.2	7.0	5.5	10.9
pseudo label	9.1	7.7	8.7	5.1	19.7	6.5	7.4	6.8	7.5	5.7	5.2	5.1	8.4	6.5	11.7
tent (ours)	8.7	7.0	8.7	4.6	17.0	6.1	6.3	7.0	7.0	4.7	4.9	4.8	7.5	6.2	10.2
CIFAR-100															
source-only	56.6	44.4	44.6	21.5	79.2	30.5	33.8	30.0	32.9	21.9	21.4	22.3	29.5	25.7	37.9
TTT	34.3	31.3	31.0	26.8	52.9	31.4	35.9	33.2	31.3	27.9	27.4	27.7	33.5	29.7	35.8
batch norm	32.6	28.8	30.8	21.0	45.5	24.4	23.5	26.6	26.3	21.2	21.2	21.1	26.2	23.9	35.7
pseudo label	32.6	30.0	31.1	22.4	43.3	25.9	24.3	27.3	27.2	23.6	22.3	22.6	27.2	26.0	34.4
tent (ours)	29.6	27.4	28.6	21.6	39.7	24.3	23.1	25.9	26.3	21.8	21.9	21.9	26.9	24.0	31.7
ILSVRC															
source-only	37.5	38.5	47.0	41.1	44.6	35.8	46.8	46.8	39.8	40.2	27.1	35.8	32.8	32.7	33.8
batch norm	35.1	35.9	40.8	41.3	38.5	32.5	39.2	38.4	35.6	31.2	25.4	29.1	29.7	28.9	32.4
tent (ours)	31.3	31.7	34.9	33.8	32.2	29.6	34.2	33.0	32.9	29.1	25.4	27.6	28.6	27.5	29.9

Table 6: Corruption benchmark measuring percentage error on CIFAR-10, CIFAR-100, and ILSVRC at severity level 2.

	gauss	shot	impulse	defocus	glass	motion	zoom	snow	frost	fog	bright	contrast	elastic	pixelate	jpeg
CIFAR-10															
source-only	40.6	22.7	27.0	5.7	41.4	15.5	14.7	18.7	13.9	5.5	4.4	7.8	8.9	10.2	18.3
TTT	12.2	9.9	11.0	6.8	25.6	11.1	16.0	13.0	10.4	7.6	7.1	7.5	10.7	8.3	13.4
batch norm	13.2	9.1	13.1	4.2	21.1	6.8	5.8	11.5	8.2	4.6	4.3	5.1	6.6	6.6	16.9
pseudo label	13.2	9.0	12.1	5.0	19.1	7.6	6.6	10.9	9.3	6.2	5.4	6.7	8.0	7.8	18.4
tent (ours)	11.2	9.2	11.5	4.9	16.0	7.2	6.2	9.3	8.5	5.2	5.1	5.3	7.3	6.7	14.1
CIFAR-100															
source-only	74.8	57.9	67.1	25.1	78.8	39.1	38.6	47.7	42.6	25.2	22.0	28.7	30.0	33.2	46.6
TTT	38.9	34.6	34.7	27.4	50.9	34.3	37.8	36.9	35.6	28.7	28.1	29.4	33.9	30.9	40.8
batch norm	40.8	34.0	38.6	21.3	46.5	26.9	24.1	34.6	30.6	22.5	21.5	22.4	25.3	25.4	43.6
pseudo label	38.4	32.3	36.8	22.8	44.2	27.3	25.2	33.1	31.4	23.6	23.0	23.4	27.2	26.6	40.1
tent (ours)	35.7	30.4	34.0	22.0	39.6	26.3	23.6	31.3	29.7	22.6	22.1	22.7	26.3	25.2	37.7
ILSVRC															
source-only	47.6	50.7	57.0	48.3	59.6	46.6	56.7	69.6	57.3	47.0	28.8	42.3	54.6	35.4	36.9
batch norm	44.1	46.3	50.6	51.6	51.4	41.0	46.3	53.0	49.5	34.3	26.4	32.1	45.7	30.4	36.9
tent (ours)	36.4	36.9	40.6	39.2	39.0	33.6	38.2	41.4	42.1	30.5	26.2	29.3	40.9	28.1	32.3

Table 7: Corruption benchmark measuring percentage error on CIFAR-10, CIFAR-100, and ILSVRC at severity level 3.

	gauss	shot	impulse	defocus	glass	motion	zoom	snow	frost	fog	bright	contrast	elastic	pixelate	jpeg
CIFAR-10															
source-only	56.9	42.2	36.6	10.6	38.9	24.1	20.5	14.5	23.0	7.2	5.0	11.3	13.1	14.7	20.5
TTT	14.9	12.1	13.7	7.0	24.5	13.7	16.3	13.9	12.1	9.3	7.6	7.5	10.9	8.8	13.9
batch norm	19.6	15.2	17.8	4.7	20.6	9.0	6.7	11.5	10.9	5.2	4.6	5.5	7.4	7.8	18.4
pseudo label	17.0	15.8	16.0	5.6	18.3	9.2	6.8	11.5	11.2	6.9	6.1	5.7	8.1	8.7	16.3
tent (ours)	15.6	12.6	14.7	5.6	15.6	8.9	6.7	11.3	10.6	5.7	4.9	5.9	7.8	7.7	15.9
CIFAR-100															
source-only	84.4	77.5	79.9	34.8	74.8	48.3	45.3	41.4	54.7	29.0	23.3	35.3	36.8	39.8	49.2
TTT	43.9	39.4	37.5	28.6	53.3	37.2	41.9	38.6	40.0	36.6	28.4	30.1	35.8	31.4	42.2
batch norm	48.5	44.4	44.1	22.2	46.6	30.8	25.7	34.3	35.2	24.2	22.2	23.7	26.9	27.2	46.8
pseudo label	46.2	41.6	41.4	23.1	43.3	30.1	26.1	34.0	36.2	25.2	23.5	24.1	28.6	27.8	43.3
tent (ours)	40.7	36.9	38.6	22.4	39.4	29.1	24.8	32.3	33.0	24.0	22.8	23.2	26.8	26.7	39.8
ILSVRC															
source-only	64.2	66.5	65.6	63.6	83.0	65.4	64.8	66.4	69.1	56.6	31.5	55.2	45.9	46.3	39.6
batch norm	58.3	59.2	58.9	68.7	71.3	54.8	52.1	52.9	59.3	38.9	28.0	38.7	35.0	37.0	40.8
tent (ours)	44.5	44.4	45.0	50.8	52.8	40.1	41.6	41.1	50.1	32.7	27.5	32.0	30.2	31.6	34.2

Table 8: Corruption benchmark measuring percentage error on CIFAR-10, CIFAR-100, and ILSVRC following Hendrycks & Dietterich (2019) at severity level 4.

	gauss	shot	impulse	defocus	glass	motion	zoom	snow	frost	fog	bright	contrast	elastic	pixelate	jpeg
CIFAR-10															
source-only	63.2	48.5	49.9	19.7	50.8	23.9	26.4	17.2	25.2	10.9	5.7	20.4	20.0	31.9	23.5
TTT	16.1	14.2	16.5	8.6	37.0	13.4	17.2	15.9	12.6	14.0	7.7	8.5	14.2	10.1	15.4
batch norm	23.1	17.7	25.8	6.4	31.1	8.7	7.6	12.8	11.4	6.2	5.0	6.4	11.7	10.9	20.8
pseudo label	23.1	15.6	25.8	7.5	30.5	8.4	8.0	12.3	11.8	7.5	6.4	7.4	13.3	10.0	17.9
tent (ours)	18.2	14.0	20.7	6.8	24.6	8.7	7.2	12.0	10.7	6.7	5.6	6.0	11.9	9.0	17.1
CIFAR-100															
source-only	87.4	82.8	91.4	46.9	84.7	48.0	51.4	44.4	56.1	36.2	25.5	46.8	44.3	60.1	53.0
TTT	46.1	42.3	43.9	31.1	62.7	36.7	43.8	40.6	39.4	48.3	29.2	31.3	39.8	32.9	43.3
batch norm	51.6	47.3	54.4	25.3	57.5	30.0	28.0	36.6	35.7	28.1	23.4	25.3	34.4	31.7	50.3
pseudo label	48.0	43.8	50.0	26.0	56.3	30.3	29.3	37.9	35.2	28.5	24.7	26.1	35.1	30.9	47.0
tent (ours)	44.4	40.0	46.1	24.5	50.1	28.4	26.3	34.5	33.1	27.4	23.9	24.1	33.2	28.8	41.7
ILSVRC															
source-only	81.8	85.7	83.2	75.5	87.9	81.0	71.3	77.9	71.1	63.7	36.1	81.1	58.8	61.6	48.3
batch norm	73.5	76.9	75.0	79.5	78.4	69.2	58.2	63.5	60.8	42.3	30.9	61.5	41.3	47.9	52.9
tent (ours)	55.5	58.0	56.7	62.0	60.9	50.2	45.4	47.9	51.7	34.9	29.3	41.0	33.2	36.9	39.6

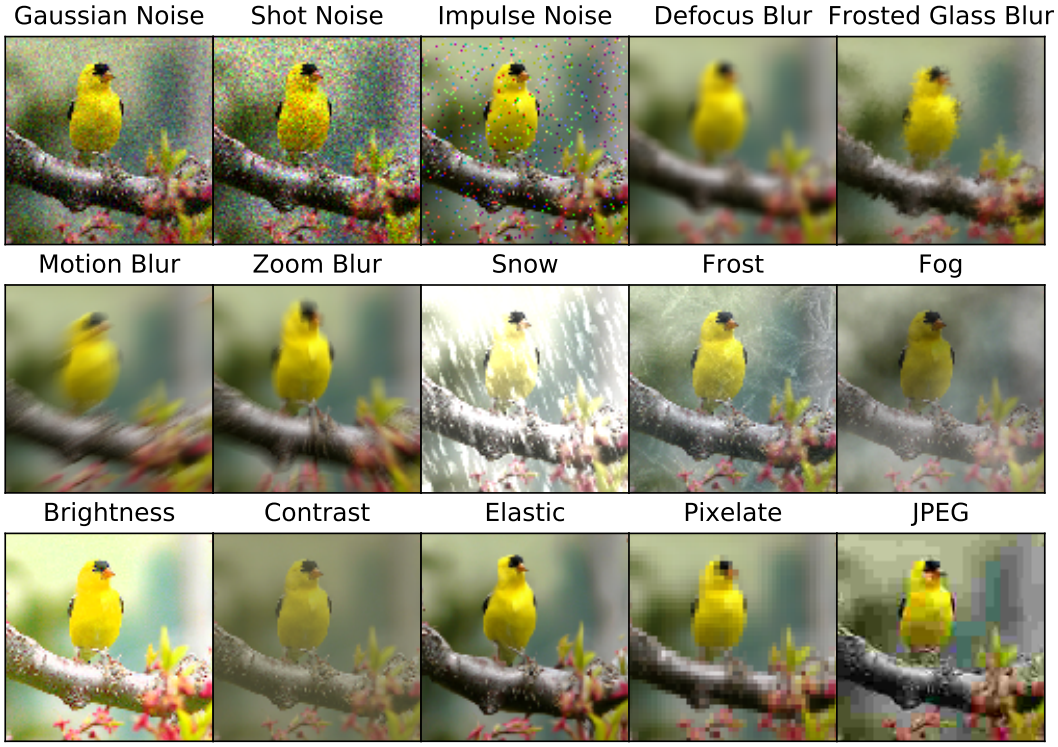


Figure 8: Examples of each corruption type in the image corruptions benchmark. (Figure reproduced from Hendrycks & Dietterich (2019)).

Table 9: Corruption benchmark measuring percentage error on CIFAR-10, CIFAR-100, and ILSVRC following Hendrycks & Dietterich (2019) at severity level 5 (most severe).

	gauss	shot	impulse	defocus	glass	motion	zoom	snow	frost	fog	bright	contrast	elastic	pixelate	jpeg
CIFAR-100															
source-only	89.9	88.1	95.1	65.6	81.4	55.6	59.0	53.0	65.9	59.1	31.9	77.8	51.4	76.5	57.6
RevGrad	49.9	47.6	57.1	**29.3**	51.8	**31.2**	**28.6**	37.0	37.7	34.3	**25.8**	27.1	40.8	34.8	50.0
UDA	54.3	51.9	**46.1**	35.8	59.1	44.5	40.4	45.0	42.2	41.3	31.1	37.4	46.3	36.7	47.6
TTT	47.7	46.0	50.3	34.4	63.3	41.7	47.4	42.6	44.9	61.9	31.2	36.4	43.8	36.5	47.0
batch norm	55.8	53.8	62.4	32.7	57.0	33.6	31.3	39.7	40.1	38.2	26.5	31.8	42.9	39.5	54.7
pseudo label	51.0	50.3	58.8	32.0	54.3	33.2	31.1	39.6	40.2	36.0	27.6	31.3	44.3	36.8	51.3
tent (online)	48.3	46.8	56.8	30.6	53.0	32.0	30.1	38.3	37.5	34.6	27.1	27.8	42.5	35.0	48.0
tent (offline)	**45.4**	**43.8**	53.9	29.6	**50.5**	**31.2**	**28.6**	**36.6**	**36.0**	**33.3**	26.2	**26.2**	**40.8**	**33.4**	**44.1**
ILSVRC															
source-only	94.2	92.9	93.9	84.2	91.7	87.8	77.2	84.5	77.9	79.6	42.5	95.4	84.0	71.7	61.3
batch norm	88.1	86.2	87.1	87.0	86.9	77.0	63.3	66.3	67.1	52.7	34.9	88.5	58.1	54.3	66.4
tent (online)	75.5	71.5	73.5	76.5	77.3	64.0	52.5	53.8	59.9	43.3	32.2	**84.3**	46.6	43.6	50.9
tent (offline)	**72.3**	**68.2**	**69.7**	**73.8**	**74.3**	**59.0**	**50.0**	**50.3**	**59.7**	**40.7**	**31.7**	88.0	**43.7**	**40.9**	**47.7**

B TARGET-ONLY DOMAIN ADAPTATION FOR SEMANTIC SEGMENTATION

We report a qualitative result on target-only domain adaptation for semantic segmentation (pixel-wise classification) with a simulation-to-real (sim-to-real) domain shift.

For this sim-to-real condition, the source data is simulated while the target data is real. Our source data is GTA Richter et al. (2017), a visually-sophisticated video game set in an urban environment, and our target data is Cityscapes Cordts et al. (2016), an urban autonomous driving dataset. The supervised model is HRnet-W18, a

fully convolutional network Shelhamer et al. (2017) in the high-resolution network family Wang et al. (2020). For this qualitative example, we run tent on a single image for multiple iterations, because an image is in effect a dataset of pixels. This demonstrates adaptation to a sole target instance, without any further access to the target distribution.

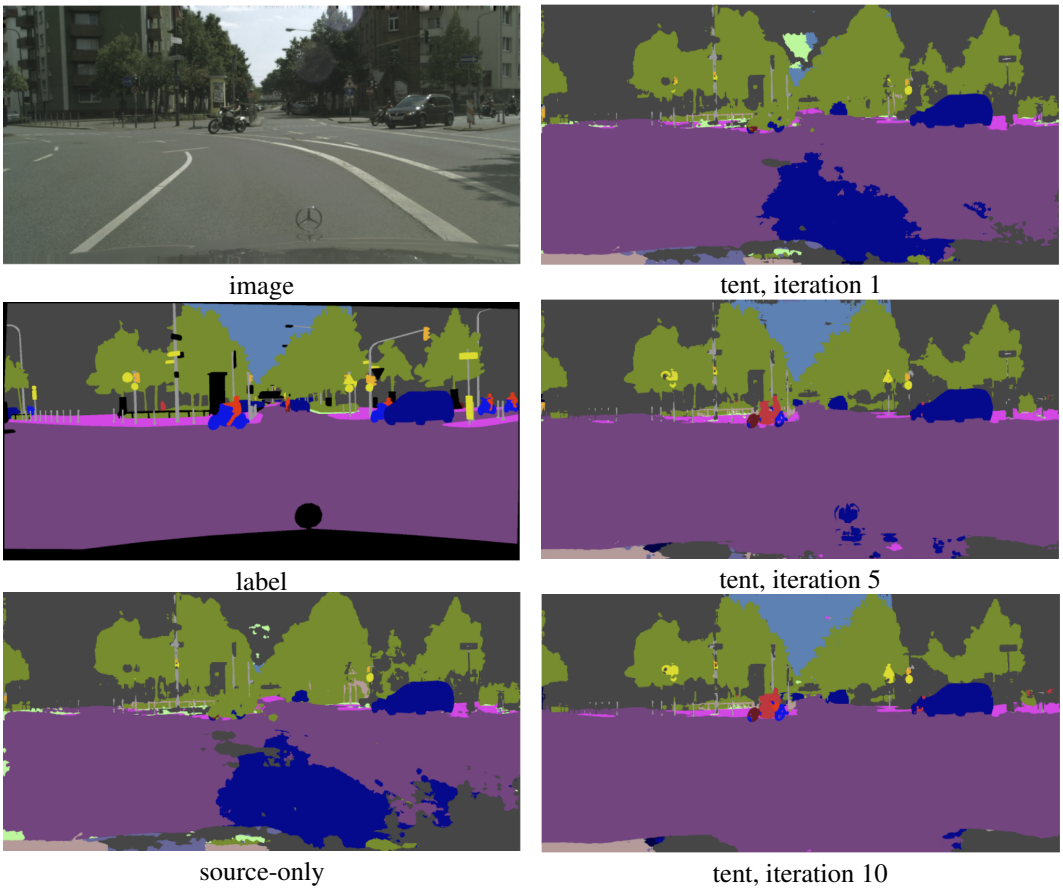


Figure 9: Adaptation for semantic segmentation with simulation-to-real shift from GTA Richter et al. (2017) to Cityscapes Cordts et al. (2016). Tent only uses the target data, and optimizes over a single image as a dataset of pixel-wise predictions. This episodic optimization in effect fits a custom model to each image of the target domain. In only 10 iterations our method suppresses noise (see the completion of the street segment, in purple) and recovers missing classes (see the motorcycle and rider, center).

C FEATURE DISTRIBUTIONS ACROSS LAYERS AND METHODS

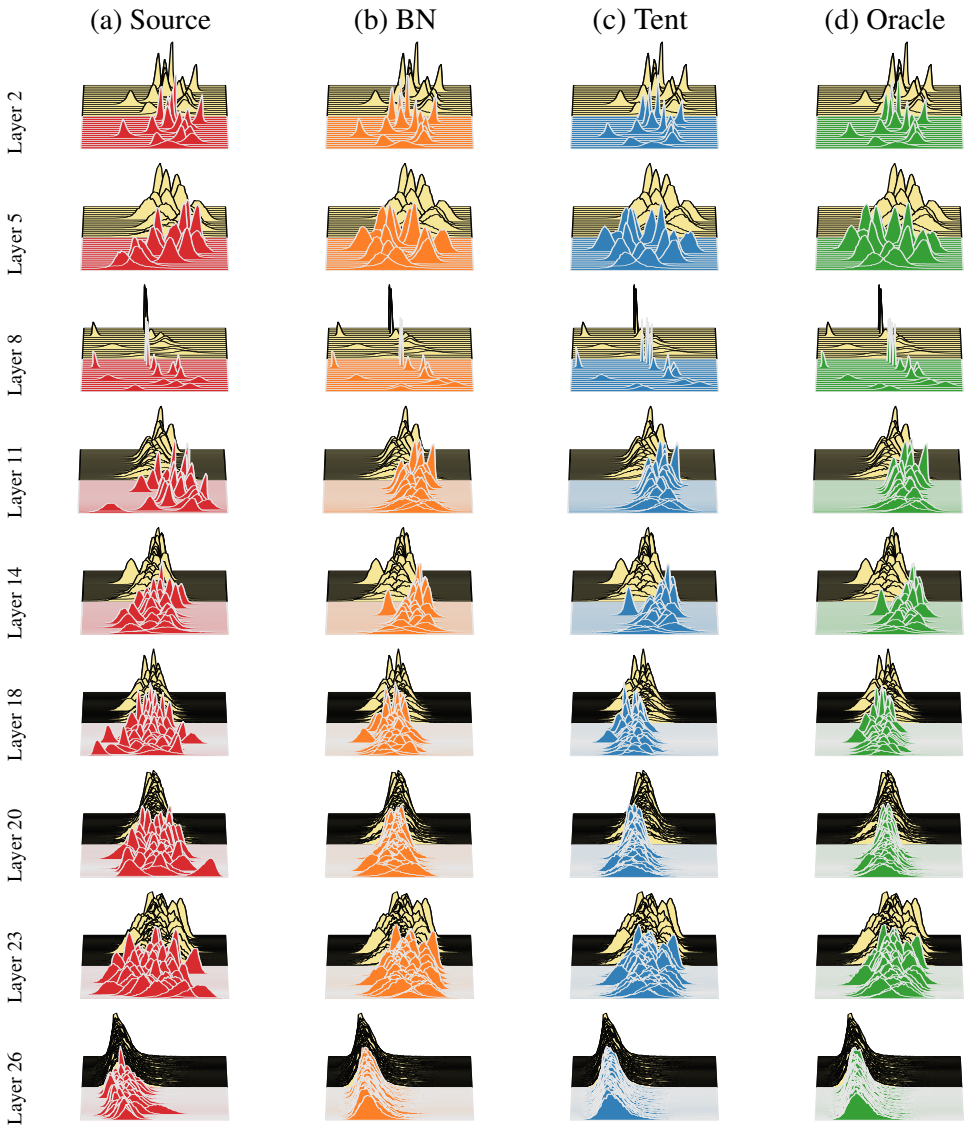


Figure 10: Adapted features on CIFAR-100-C with Gaussian noise (front) and reference features without corruption (back). Corruption shifts the source features from the reference. BN shifts the features back to be more like the reference. Tent shifts features to be less like the reference, and more like an oracle that optimizes on target labels.