Learning to Generalize in Dynamic Environments

Dequan Wang
Trevor Darrell, Ed.

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2022-229
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2022/EECS-2022-229.html

October 7, 2022

Learning to Generalize in Dynamic Environments

by

Dequan Wang

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Trevor Darrell, Chair
Associate Professor Joseph Gonzalez
Assistant Professor Angjoo Kanazawa
Research Scientist Evan Shelhamer

Fall 2022

Learning to Generalize in Dynamic Environments

by
Dequan Wang

Abstract

Learning to Generalize in Dynamic Environments

by

Dequan Wang

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Trevor Darrell, Chair

A model must be able to adapt itself to generalize to new environments. Deep networks achieved great success in the past decade, especially when training and testing data come from the same distribution. Unfortunately, the performance suffers when the training (source) differs from the testing (target) data, a condition known as domain shift. Models should update themselves to deal with these unexpected natural and adversarial perturbations, such as weather change, sensor degradation, adversarial attack, and so on. If we have some labeled target data, several transfer learning methods, such as fine-tuning and few-shot learning, could be utilized to optimize the model in a supervised way. However, the requirement for target labels is not practical for most real-world scenarios. Therefore, we instead focus on the unsupervised learning approach to generalize the model to the target domain.

In this dissertation, we study the setting of fully test-time adaptation, updating the model to the uncontrollable target data distribution, without access to target labels and source data. In other words, the model only has its parameters and unlabeled target data in this setting. The core idea is to leverage the test-time optimization objective, entropy minimization, as a feedback mechanism to the learnable model to close the loop during the test time. We optimize the model for confidence as measured by output entropy in either an online or offline manner. Such a simple yet effective method could reduce the generalization error for image classification on naturally corrupted and adversarial perturbed images. Also, the adaptive nature of the semantic segmentation model could be exploited to cope with the dynamic scale inference for scene understanding. With the help of contrastive learning and diffusion models, we could learn target domain features and generate source-style images to further boost the recognition performance in dynamic environments.

To my family.

Contents

List of Figures

List of Tables

Acknowledgments

My first and deepest thanks belong to my advisor, Trevor Darrell. He is the best mentor I could ever imagine. He is the most knowledgeable, open-minded, and supportive advisor, giving me so much support and encouragement. He is the best role model in my academic and personal life.

Many colleagues also have helped me along the way, playing a mentorship role. I am grateful to them for their support and guidance. In my first year, Judy Hoffman and Fisher Yu introduced me to domain adaptation and object recognition and taught me how to think and work like a computer scientist. Philipp Krähenbühl brought me into the research on autonomous driving, and then has been an indispensable supporter of my career. I thank Xinlei Chen for allowing me to work with him at Facebook AI Research. Evan Shelhamer is my long-time collaborator, friend, and mentor. I am grateful for his support and guidance along my journey to Ph.D. I also appreciate Joseph Gonzalez and Angjoo Kanazawa for serving on my committee. I thank them for their assistance, kindness, and support.

I have been fortunate to be part of the Berkeley Artificial Intelligence Research (BAIR) for six years. I am grateful to my colleagues at Berkeley, including Jon Long, Jeff Donahue, Pulkit Agrawal, Shubham Tulsiani, Saurabh Gupta, Hao Su, Kuan Fang, Boqing Gong, Ziwei Liu, Zhirong Wu, Dinesh Jayaraman, Chang Liu, Rowan McAllister, Shiry Ginosar, David Fouhey, Andrew Owens, Deepak Pathak, Anna Rohrbach, Marcus Rohrbach, Huijuan Xu, Qian Yu, Amir Zamir, Shanghang Zhang, Samaneh Azadi, Amir Bar, Andreea Bobu, Zhe Cao, Michael Chang, Jianbo Chen, Xi Chen, Xinyun Chen, Rodolfo Corona, Carlos Florensa, Daniel Fried, Hang Gao, Yang Gao, Xinyang Geng, Erin Grant, Devin Guillory, Ritwik Gupta, Zhi-Yang He, Lisa Anne Hendricks, Dan Hendrycks, Roi Herzig, Ronghang Hu, Zixi Hu, Sandy Huang, Phillip Isola, Biye Jiang, Chi Jin, Jinkyu Kim, Weicheng Kuo, Ke Li, Ruilong Li, Fangchen Liu, Zhuang Liu, Parsa Mahmoudieh, Wenlong Mou, Norman Mu, Medhini Narasimhan, Evonne Ng, Xinlei Pan, Seth Dong Huk Park, Taesung Park, Jason Peng, Suzie Petryk, Haozhi Qi, Ilija Radosavovic, Kate Rakelly, Colorado Reed, David Ren, Alexander Sax, Sheng Shen, Baifeng Shi, Richard Shin, Katie Stasaski, Yu Sun, Haoran Tang, Eric Tzeng, Guanhua Wang, Peter Wang, Ting-Chun Wang, Xiaolong Wang, Xin Wang, Olivia Watkins, Bichen Wu, Fangyu Wu, Yi Wu, Tete Xiao, Kelvin Xu, Huazhe Xu, Wilson Yan, Ge Yang, Hezheng Yin, Yang You, Yaodong Yu, Xiangyu Yue, Cecilia Zhang, Jiaheng Zhang, Marvin Zhang, Richard Zhang, Tianhao Zhang, Tianjun Zhang, Tinghui Zhou, Jun-Yan Zhu, Shizhan Zhu, and many others.

I thank the support of my co-authors for their help and collaboration: Zheng Zhang, Yu-Gang Jiang, Xiangyang Xue, Zhiqiang Shen, Jie Shao, Wei Zhang, Sheng Zeng, Xingchao Peng, Ben Usman, Neela Kaushik, Kate Saenko, Hou-Ning Hu, Qi-Zhi Cai, Ji Lin, Min Sun, Chiyu Jiang, Jingwei Huang, Philip Marcus, Matthias Nießner, Coline Devin, Bruno Olshausen, Shaoteng Liu, An Ju, Mong Ng, Kaahan Radia, Jianfei Chen, Ionel Gog, Qijing Huang, Zhen Dong, Yizhao Gao, Yaohui Cai, Tian Li, Kurt Keutzer, John Wawrzynek, Lianmin Zheng, Zhewei Yao, Ion Stoica, Michael Mahoney, Dian Chen, Sayna Ebrahimi, Jin Gao, Jialing Zhang, Xihui Liu, Xiaoxuan Liu, Yukuo Cen, Weize Chen, Xu Han, Zhiyuan Liu, Jie Tang, Alvin Cheung, among others.

I sincerely appreciate the support of my family. Thanks to my parents for their unconditional guidance, love, and support. This thesis is dedicated to them.

Chapter 1

Introduction

The world is full of unexpected changes, but our model is too finite to handle. It is worth noting that the modeling capacity is limited and pre-defined by the width and depth of the neural network, while the variations in the dynamic environments are not. Instead of deploying *one single model* for all variations, we pursue an adaptive, dynamic, learnable approach to create *a family of models* during the test time. Each member in such a model family is associated with the target input, generalizing from the same initial source model according to the test-time optimization objective. When the static model is independent of the given target data, our fully test-time adapted model learns to generalize to dynamic environments.

1.1 Summary of Contributions

This thesis presents a series of works around learning to generalize during test time in dynamic environments. We first introduce the setting of fully test-time adaptation. Then we propose an entropy minimization framework to update the model's parameters given unlabeled target data. When equipping a model with a test-time optimization objective, we observe a significant improvement in the robustness and generalization of image classification on both naturally corrupted and adversarially perturbed images. We also integrate the adaptivity into the dense prediction model to handle source-free domain adaptation and dynamic scale inference. All of the above works could be done in an online manner so that the latency of the overall system is not affected. If an offline optimization procedure is allowed, we could further boost the recognition performance by leveraging contrastive learning to obtain target domain representation. Finally, we identify the weak points of model adaptation—small batches, ordered data, and mixed domains—and introduce diffusion-based input adaptation, generating source domain style images to reduce multi-domain shifts simultaneously. The detailed description of the main contributions is summarized as follows:

In chapter 2, we introduce the first approach to fully test-time adaptation. The model has only the test data and its own parameters in this setting. We propose to adapt by test

entropy minimization: we optimize the model for confidence as measured by the entropy of its predictions. Our method, tent, estimates normalization statistics and optimizes channel-wise affine transformations to update online on each batch. Tent reduces generalization error for image classification on corrupted ImageNet and CIFAR-10/100 and reaches a new state-of-the-art error on ImageNet-C. Tent handles source-free domain adaptation on digit recognition from SVHN to MNIST/MNIST-M/USPS, semantic segmentation from GTA to Cityscapes, and VisDA-C benchmark. These results are achieved in one epoch of test-time optimization without altering training.

In chapter 3, we discuss the application of fully test-time adaptation as a novel dynamic defense approach. Adversarial attacks optimize against models to defeat defenses. Existing defenses are static and stay the same once trained, even while attacks change. We argue that models should fight back and optimize their defenses against attacks at test time. We propose dynamic defenses, to adapt the model and input during testing, by defensive entropy minimization (dent). Dent alters testing, but not training, for compatibility with existing models and train-time defenses. Dent improves the robustness of adversarially-trained defenses and nominally-trained models against white-box, black-box, and adaptive attacks on CIFAR-10/100 and ImageNet. In particular, dent boosts state-of-the-art defenses by 20+ points absolute against AutoAttack on CIFAR-10 at $\epsilon_\infty = 8/255$.

In chapter 4, we further investigate how to cope with dynamic scale variation in the context of semantic segmentation via test-time entropy minimization. Given the variety of the visual world, there is no one true scale for recognition: objects may appear at drastically different sizes across the visual field. Rather than enumerate variations across filter channels or pyramid levels, dynamic models locally predict the scale and adapt receptive fields accordingly. The degree of variation and diversity of inputs makes this a difficult task. Existing methods either learn a feedforward predictor, which is not totally immune to the scale variation it is meant to counter, or select scales by a fixed algorithm, which cannot learn from the given task and data. We extend dynamic scale inference from feedforward prediction to iterative optimization for further adaptivity. We propose a novel entropy minimization objective for inference, optimizing over task and structure parameters to tune the model to each input. Optimization during the test time improves semantic segmentation accuracy and generalizes better to extreme scale variations that cause feedforward dynamic inference to falter.

In chapter 5, we argue that source data should not be the only source of all model parameters. Domain adaptation seeks to mitigate the shift between training on the *source* data and testing on the *target* data. Most adaptation methods rely on the source data by joint optimization over source and target. Source-free methods replace the source data with source parameters by fine-tuning the model on target. Either way, the majority of the parameter updates for the model representation and the classifier are derived from the source and not the target. However, target accuracy is the goal, and so we argue for optimizing as much as possible on target. We show significant improvement by *on-target adaptation*, which learns the representation purely on target data, with only source predictions for supervision (without source data or parameter fine-tuning). In the long-tailed classification setting, we

demonstrate on-target class distribution learning, which learns the imbalance of classes on target data. On-target adaptation achieves state-of-the-art accuracy and computational efficiency on VisDA-C and ImageNet-Sketch.

In chapter 6, we revisit the idea of input adaptation, reducing the domain shift in pixel space. We leverage the latest image generation techniques, diffusion models, to generate a source-style image from the given target domain input. Test-time adaptation harnesses test inputs to improve the accuracy of a model trained on source data when tested on shifted target data. Existing methods update the source *model* by (re-)training on each target domain. While effective, re-training is sensitive to the amount and order of the data and the hyperparameters for optimization. We instead update the target *data* by projecting all test inputs toward the source domain with a generative diffusion model. Our diffusion-driven adaptation method, DDA, shares its models for classification and generation across all domains. Both models are trained on the source domain, then fixed during testing. We augment diffusion with image guidance and self-ensembling to automatically decide how much to adapt. Input adaptation by DDA is more robust than prior model adaptation approaches across a variety of corruptions, architectures, and data regimes on the ImageNet-C benchmark. With its input-wise updates, DDA succeeds where model adaptation degrades on too little data (small batches), dependent data (non-random order), or mixed data (multiple corruptions).

1.2 Summary of Publications

This dissertation covers the following publications:

- Material in chapter 2 is published as
  Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell, "Tent: Fully test-time adaptation by entropy minimization," in *ICLR*, 2021
- Material in chapter 3 is published as
  Dequan Wang, An Ju, Evan Shelhamer, David Wagner, and Trevor Darrell, "Fighting gradients with gradients: Dynamic defenses against adversarial attacks," *arXiv preprint arXiv:2105.08714*, 2021
- Material in chapter 4 is published as
  Dequan Wang, Evan Shelhamer, Bruno Olshausen, and Trevor Darrell, "Dynamic scale inference by entropy minimization," *arXiv preprint arXiv:1908.03182*, 2019
- Material in chapter 5 is published as
  Dequan Wang, Shaoteng Liu, Sayna Ebrahimi, Evan Shelhamer, and Trevor Darrell, "On-target adaptation," *arXiv preprint arXiv:2109.01087*, 2021
- Material in chapter 6 is published as
  Jin Gao, Jialing Zhang, Xihui Liu, Trevor Darrell, Evan Shelhamer, and Dequan Wang, "Back to the source: Diffusion-driven test-time adaptation," *arXiv preprint arXiv:2207.03442*, 2022

Chapter 2

Fully Test-Time Adaptation by Entropy Minimization

2.1 Introduction

Deep networks can achieve high accuracy on training and testing data from the same distribution, as evidenced by tremendous benchmark progress [6]–[8]. However, generalization to new and different data is limited [9]–[11]. Accuracy suffers when the training (source) data differ from the testing (target) data, a condition known as *dataset shift* [12]. 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.

To adapt during testing we minimize the entropy of model predictions. We call this objective the test entropy and name our method *tent* after it. We choose entropy for its connections to error and shift. Entropy is related to error, as more confident predictions are all-in-all more correct (Figure 2.1). Entropy is related to shifts due to corruption, as

more corruption results in more entropy, with a strong rank correlation to the loss for image classification as the level of corruption increases (Figure 2.2).

To minimize entropy, tent normalizes and transforms inference on target data by estimating statistics and optimizing affine parameters batch-by-batch. This choice of low-dimensional, channel-wise feature modulation is efficient to adapt during testing, even for online updates. 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 generalization to corruptions for image classification, to domain shift for digit recognition, and to simulation-to-real shift for semantic segmentation. For context with more data and optimization, we evaluate methods for robust training, domain adaptation, and self-supervised learning given the labeled source data. Tent can achieve less error given only the target data, and it improves on the state-of-the-art for the ImageNet-C benchmark. Analysis experiments support our entropy objective, check sensitivity to the amount of data and the choice of parameters for adaptation, and back the generality of tent across architectures.

Our contributions

- We highlight the setting of fully test-time adaptation with only target data and no source data. To emphasize practical adaptation during inference we benchmark with offline and online updates.
- 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 on test data.
- For robustness to corruptions, tent reaches 44.0% error on ImageNet-C, better than the state-of-the-art for robust training (50.2%) and the strong baseline of test-time normalization (49.9%).
- For domain adaptation, tent is capable of online and source-free adaptation for digit classification and semantic segmentation, and can even rival methods that use source data and more optimization.

2.2 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 2.1 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 for adaptation during inference.

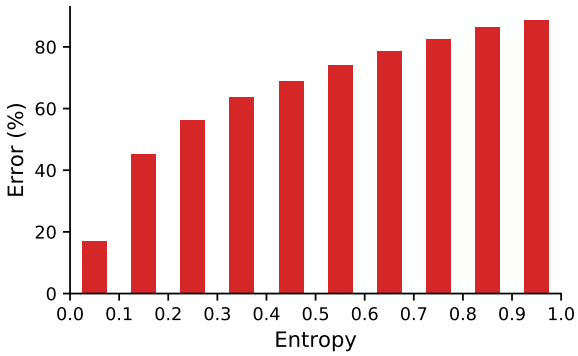


Figure 2.1: Predictions with lower entropy have lower error rates on corrupted CIFAR-100-C. Certainty can serve as supervision during testing.

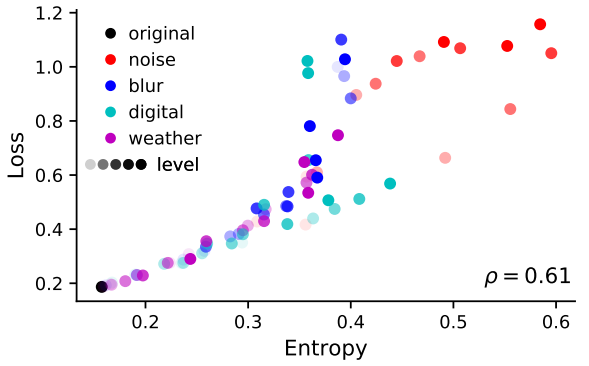


Figure 2.2: More corruption causes more loss and entropy on CIFAR-100-C. Entropy can estimate the degree of shift without training data or labels.

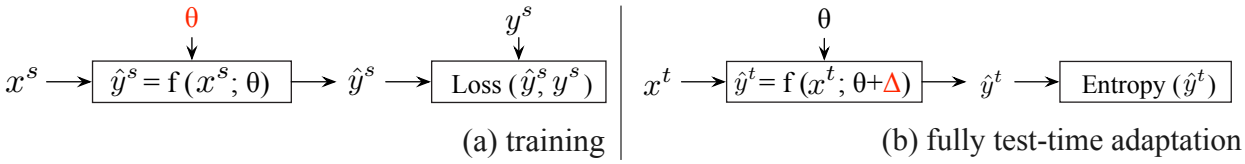


Figure 2.3: Method overview. Tent does not alter training (a), but minimizes the entropy of predictions during testing (b) over a constrained modulation Δ, given the parameters θ and target data x^t.

Existing adaptation settings extend training given more data and supervision. Transfer learning by fine-tuning [13], [14] 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) [12], [15]–[17] needs both the source and target data to train with a cross-domain loss $L(x^s, x^t)$. Test-time training (TTT) [18] adapts during testing but first alters training to jointly optimize its supervised loss $L(x^s, y^s)$ and self-supervised loss $L(x^s)$. Without source, our setting denies joint training across domains (DA) or losses (TTT). Existing 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.

Table 2.1: 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	-	$L(x^t)$

2.3 Method: Test Entropy Minimization via Feature Modulation

We optimize the model during testing to minimize the entropy of its predictions by modulating its features. We call our method *tent* for test entropy. Tent requires a compatible model, an objective to minimize, and parameters to optimize over to fully define the algorithm. Figure 2.3 outlines our method for fully test-time adaptation.

The model to be adapted 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. Typical deep networks for supervised learning satisfy these model requirements.

Entropy Objective

Our test-time objective $L(x_t)$ is to minimize the entropy $H(\hat{y})$ of model predictions $\hat{y} = f_\theta(x^t)$. In particular, we measure the Shannon entropy [19], $H(\hat{y}) = -\sum_c p(\hat{y}_c) \log p(\hat{y}_c)$ for the probability $\hat{y}_c$ of class c. Note that optimizing a single prediction has a trivial solution: assign all probability to the most probable class. We prevent this by jointly optimizing batched predictions over parameters that are shared across the batch.

Entropy is an unsupervised objective because it only depends on predictions and not annotations. However, as a measure of the predictions it is directly related to the supervised task and model.

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. Examples of these proxies include rotation prediction [20], context prediction [21],

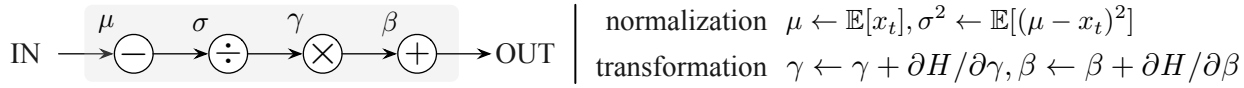


Figure 2.4: Tent modulates features during testing by estimating normalization statistics μ, σ and optimizing transformation parameters γ, β. Normalization and transformation apply channel-wise scales and shifts to the features. The statistics and parameters are updated on target data without use of source data. In practice, adapting γ, β is efficient because they make up $<1\%$ of model parameters.

and cross-channel auto-encoding [22]. 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 [18], [23]. 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 entropy objective does not need such efforts.

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 [24]–[26]. However, θ is the only representation of the training/source data in our setting, and altering θ could cause the model to diverge from its training. 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 2.4 shows the two steps of our modulations: normalization by statistics and transformation by parameters. Normalization centers and standardizes 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. We update their normalization statistics and affine parameters for all layers and channels during testing.

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 data. The normalization statistics are estimated 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. Note that the transformation update follows the prediction for the current batch, and so it only affects the next batch (unless forward is repeated). This needs just one gradient per point of additional computation, so we use this scheme by default for efficiency.

Termination For online adaptation, no termination is necessary, and iteration continues as long as there is test data. For offline adaptation, the model is first updated and then inference is repeated. Adaptation may of course continue by updating for multiple epochs.

2.4 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 [27] with pycls library [28].

Datasets We run on image classification datasets for corruption and domain adaptation conditions. For large-scale experiments we choose ImageNet [29], 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 [30], with 10/100 classes, a training set of 50,000, and a test set of 10,000. For domain adaptation we choose SVHN [31] as source and MNIST [32]/MNIST-M [16]/USPS [33] 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 [8] 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 [34]. 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 done in concurrent work on adaptation by normalization [35], [36].

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; on other datasets we optimize by Adam [37]. We lower the batch size (BS) to reduce memory usage for inference, then lower the learning rate (LR) by the same factor to

compensate [38]. On ImageNet, we set BS = 64 and LR = 0.00025, and on other datasets we set BS = 128 and LR = 0.001.We control for ordering by shuffling and 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 [16],
- self-supervised domain adaptation (UDA-SS) jointly trains self-supervised rotation and position tasks on source and target to optimize for a shared representation [23],
- 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 [18],
- test-time normalization (BN) updates batch normalization statistics [34] on the target data during testing [35], [36],
- 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 [39].

Only test-time normalization (BN), pseudo-labeling (PL), and tent (ours) are fully test-time adaptation methods. See Section 2.2 for an explanation and contrast with domain adaptation and test-time training.

Robustness to Corruptions

To benchmark robustness to corruption, we make use of common image corruptions . 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 [9].

Tent improves more with less data and computation. Table 2.2 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 sizes of these datasets. Adaptation is offline for fair comparison with offline baselines. 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 several epochs of optimization on source and target.

Table 2.2: 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**

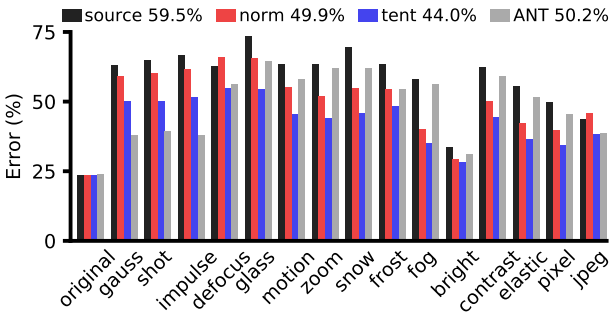


Figure 2.5: Corruption benchmark on ImageNet-C: error for each type averaged over severity levels. Tent improves on the prior state-of-the-art, adversarial noise training [40], by fully test-time adaptation *without altering training*.

Tent consistently improves across corruption types. Figure 2.5 plots the error for each corruption type averaged over corruption levels on ImageNet-C. We compare the most efficient methods—source, normalization, and tent—given the large scale of the source data (>1 million images) needed by other methods and the 75 target combinations of corruption types and levels. Tent and BN adapt online to rival the efficiency of inference without adaptation. Tent reaches the least error for most corruption types without increasing the error on the original data.

Tent reaches a new state-of-the-art without altering training. The state-of-the-art methods for robustness extend training with adversarial noise (ANT) [40] for 50.2% error or mixtures of data augmentations (AugMix) [41] for 51.7% error. Combined with stylization from external images (SIN) [42], 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 more optimization on the training set (ANT, AugMix) or use of external images (SIN). Among fully test-time adaptation methods, tent reduces the error beyond test-time normalization for 18% relative improvement. In concurrent work, [35] report 49.3% error for test-time normalization, for which tent still gives 14% relative improvement.

Source-Free Domain Adaptation

We benchmark digit adaptation [16], [17], [43], [44] for shifts from SVHN to MNIST/MNIST-M/USPS. Recall that unsupervised domain adaptation makes use the labeled source data

Table 2.3: Digit domain adaptation from SVHN to MNIST/MNIST-M/USPS. Source-free 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 (%)**		
			Source + Target	MNIST	MNIST-M	USPS
Source	train		-	18.2	39.7	19.3
RG	train	train	10 + 10	15.0	33.4	18.9
UDA-SS	train	train	10 + 10	11.1	**22.2**	18.4
BN		test	0 + 1	15.7	39.7	18.0
Tent (ours)		test	0 + 1	10.0	37.0	16.3
Tent (ours)		test	0 + 10	**8.2**	36.8	**14.4**

and unlabeled target data, while our fully test-time adaptation setting denies use of source data. Adaptation is offline for fair comparison with offline baselines.

Tent adapts to target without source. Table 2.3 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 [44] 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%. In this case success presently requires joint optimization over source and target.

Tent needs less computation, but still improves with more. Tent adapts efficiently on target data alone with just one gradient per point. RG & UDA-SS also 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$80$\times$ less computation. With more updates, tent reaches

[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 [44] for more detail.

8.2% error in 10 epochs and 6.5% in 100 epochs. With online updates, tent reaches 12.5% error in one epoch and 8.4% error in 10 epochs.

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 [45], a video game in an urban environment, and the target is Cityscapes [46], an urban autonomous driving dataset. The model is HRNet-W18, a fully convolutional network [47] with high-resolution architecture [48]. 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.

Tent scales to the VisDA-C challenge. To show adaptation on a more difficult benchmark, we evaluate on the VisDA-C challenge [49]. The task is object recognition for 12 classes where the source data is synthesized by rendering 3D models and the target data is collected from real scenes. The validation error for our source model (ResNet-50, pretrained on ImageNet) is 56.1%, while tent reaches 45.6%, and improves to 39.6% by updating all layers except for the final classifier as done by [50]. Although offline source-free adaptation by model adaptation [51] or SHOT [50] can reach lower error with more computation and tuning, tent can adapt online during testing.

Analysis

Tent reduces entropy and error. Figure 2.6 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 generalizes across target data. Adaptation could be limited to the points used for updates. We check that adaptation generalizes across points by adapting on target train and not 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; when subsampling to the

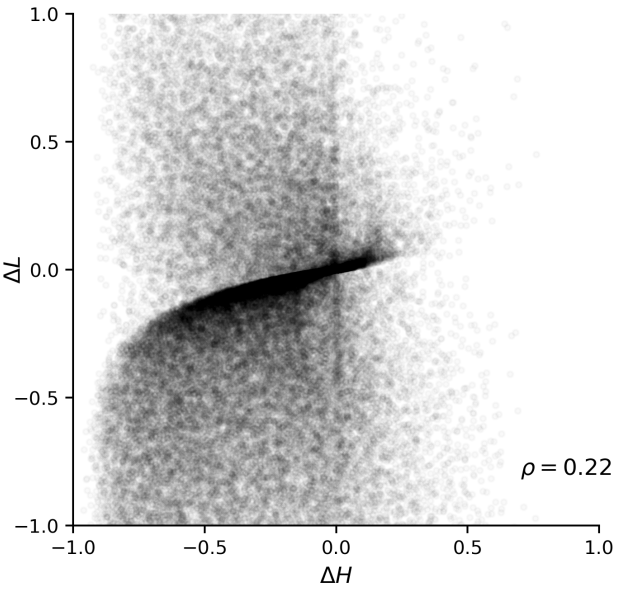


Figure 2.6: Tent reduces the entropy and loss. We plot changes in entropy ΔH and loss ΔL for all of CIFAR-100-C. Change in entropy rank-correlates with change in loss: note the dark diagonal and the rank correlation coefficient of 0.22.

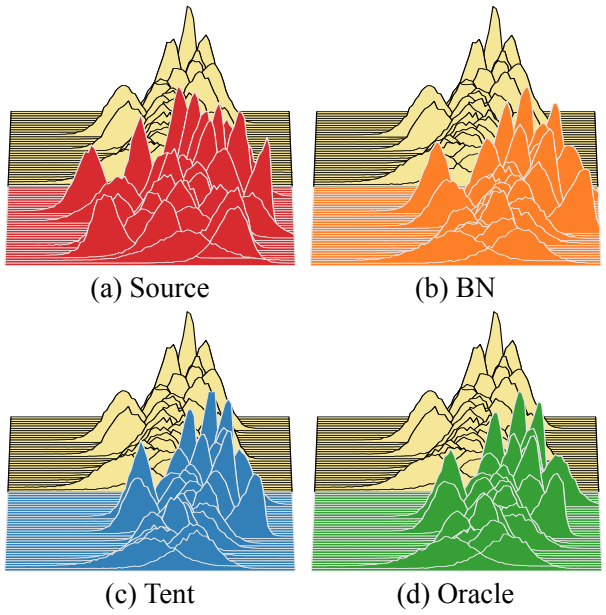


Figure 2.7: Adapted features on CIFAR-100-C with Gaussian noise (front) and reference features without corruption (back). Corruption shifts features away from the reference, but BN reduces the shifts. Tent instead shifts features more, and closer to an oracle that optimizes on target labels.

same size errors differ by <0.1%.) Therefore the adapted modulation is not point specific but general.

Tent modulation differs from normalization. Modulation normalizes and transforms features. We examine the combined effect. Figure 2.7 contrasts adapted features on corrupted data against reference features on uncorrupted data. We plot features from the source model, normalization, tent, and an oracle that optimizes on the target labels. Normalization makes features more like the reference, but tent does not. Instead, tent makes features more like the oracle. This suggests a different and task-specific effect.

Tent adapts alternative architectures. Tent is architecture agnostic in principle. To gauge its generality in practice, we evaluate new architectures based on self-attention (SAN) [52] and equilibrium solving (MDEQ) [53] for corruption robustness on CIFAR-100-C. Table 2.4 shows that tent reduces error with the same settings as convolutional residual networks.

Table 2.4: Tent adapts alternative architectures on CIFAR-100-C without tuning. Results are error (%).

SAN-10 (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**

2.5 Related Work

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

Train-Time Adaptation Domain adaptation jointly optimizes on source and target by cross-domain losses $L(x^s, x^t)$ to mitigate shift. These losses optimize feature alignment [54], [55], adversarial invariance [16], [43], or shared proxy tasks [23]. Transduction [56]–[58] jointly optimizes on train and test to better fit specific test instances. While effective in their settings, neither applies when joint use of source/train and target/test is denied. Tent adapts on target alone.

Recent "source-free" methods [50], [51], [59] also adapt without source data. [51], [59] rely on generative modeling and optimize multiple models with multiple losses. [50], [59] also alter training. Tent does not need generative modeling, nor does it alter training, and so it can deployed more generally to adapt online with much more computational efficiency. SHOT [50] adapts by information maximization (entropy minimization and diversity regularization), but differs in its other losses and its parameterization. These source-free methods optimize offline with multiple losses for multiple epochs, which requires more tuning and computation than tent, but may achieve more accuracy with more computation. Tent optimizes online with just one loss and an efficient parameterization of modulation to emphasize fully test-time adaptation during inference. We encourage examination of each of these works on the frontier of adaptation without source data.

[60] are the first to motivate adaptation without source data for legal, commercial, or technical concerns. They adapt predictions by applying denoising auto-encoders while we adapt models by entropy minimization. We share their motivations, but the methods and experiments differ.

Test-Time Adaptation Tent adapts by test-time optimization and normalization to update the model. Test-time adaptation of predictions, through which harder and uncertain cases are adjusted based on easier and certain cases [61], provides inspiration for certainty-based model adaptation schemes like our own.

Test-time training (TTT) [18] also optimizes during testing, but differs in its loss and must alter training. TTT relies on a proxy task, such as recognizing rotations of an image, and so its loss 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"). TTT alters training to optimize this proxy loss on source before adapting to target. Tent adapts without proxy tasks and without altering training.

Normalizing feature statistics is common for domain adaptation [54], [55]. For batch normalization [26], [62] separate source and target statistics during training. [35], [36] estimate target statistics during testing to improve generalization. Tent builds on test-time normalization to further reduce generalization error.

Entropy Minimization Entropy minimization is a key regularizer for domain adaptation [26], [44], [63], [64], semi-supervised learning [24], [39], [65], and few-shot learning [25]. Regularizing entropy penalizes decisions at high densities in the data distribution to improve accuracy for distinct classes [24]. These methods regularize entropy during training in concert with other supervised and unsupervised losses on additional data. Tent is the first to minimize entropy during testing, for adaptation to dataset shifts, without other losses or data. Entropic losses are common; our contribution is to exhibit entropy *as the sole loss* for fully test-time adaptation.

Feature Modulation Modulation makes a model vary with its input. We optimize modulations that are simpler than the full model for stable and efficient adaptation. We modulate channel-wise affine transformations, for their effectiveness in tandem with normalization [34], [66], and for their flexibility in conditioning for different tasks [67]. 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 target data.

2.6 Discussion

Tent reduces generalization error on shifted data by test-time entropy minimization. In minimizing entropy, the model adapts itself to feedback from its own predictions. This is truly self-supervised self-improvement. Self-supervision of this sort is totally defined by the supervised task, unlike proxy tasks designed to extract more supervision from the data, and yet it remarkably still reduces error. Nevertheless, errors due to corruption and other shifts remain, and therefore more adaptation is needed. Next steps should pursue test-time adaptation on more and harder types of shift, over more general parameters, and by more effective and efficient losses.

Shifts Tent reduces error for a variety of shifts including image corruptions, simple changes in appearance for digits, and simulation-to-real discrepancies. These shifts are popular as

standardized benchmarks, but other real-world shifts exist. For instance, the CIFAR 10.1 and ImageNetV2 test sets [10], [68], made by reproducing the dataset collection procedures, entail natural but unknown shifts. Although error is higher on both sets, indicating the presence of shift, tent does not improve generalization. Adversarial shifts [69] also threaten real-world usage, and attackers keep adapting to defenses. While adversarial training [70] makes a difference, test-time adaptation could help counter such test-time attacks.

Parameters Tent modulates the model by normalization and transformation, but much of the model stays fixed. Test-time adaptation could update more of the model, but the issue is to identify parameters that are both expressive and reliable, and this may interact with the choice of loss. TTT adapts multiple layers of features shared by supervised and self-supervised models and SHOT adapts all but the last layer(s) of the model. These choices depend on the model architecture, the loss, and tuning. For tent modulation is reliable, but the larger shift on VisDA is better addressed by the SHOT parameterization. Jointly adapting the input could be a more general alternative. If a model can adapt itself on target, then perhaps its input gradients might optimize spatial transformations or image translations to reduce shift without source data.

Losses Tent minimizes entropy. For more adaptation, is there an effective loss for general but episodic test-time optimization? Entropy is general across tasks but limited in scope. It needs batches for optimization, and cannot update episodically on one point at a time. TTT can do so, but only with the right proxy task. For less computation, is there an efficient loss for more local optimization? Tent and TTT both require full (re-)computation of the model for updates because they depend on its predictions. If the loss were instead defined on the representation, then updates would require less forward and backward computation. Returning to entropy specifically, this loss may interact with calibration [71], as better uncertainty estimation could drive better adaptation.

We hope that the fully test-time adaptation setting can promote new methods for equipping a model to adapt itself, just as tent yields a new model with every update.

Chapter 3

Fighting Gradients with Gradients: Dynamic Defenses against Adversarial Attacks

3.1 Introduction: Attack, Defend, and Then?

Deep networks are vulnerable to adversarial attacks: input perturbations that alter natural data to cause errors or exploit predictions [69]. As deep networks are deployed in real systems, these attacks are real threats [72], and so defenses are needed. The challenge is that every new defense is followed by a new attack, in a loop [73]. The strongest attacks, armed with gradient optimization, update to circumvent defenses that do not. Such iterative attacks form an even tighter loop to ensnare defenses. In a cat and mouse game, the mouse must keep moving to survive.

Current defenses, deterministic or stochastic, stand still: once trained, they are *static* and do not adapt during testing. Adversarial training [70], [74] learns from attacks during training, but cannot learn from test data. Stochastic defenses alter the network [75] or input [76], [77], but their randomness is independent of test data. Static defenses do not adapt, and so they may fail as attacks update.

Our *dynamic* defense fights adversarial updates with defensive updates by adapting during testing (Figure 3.1). In fact, our defense updates on every input, whether natural or adversarial. Our defense objective is entropy minimization, to maximize model confidence, so we call our method *dent* for defensive entropy. Our updates rely on gradients and batch statistics, inspired by test-time adaptation approaches [1], [18], [35], [50]. In pivoting from training to testing, dent is able to keep changing, so the attacker never hits the same defense twice. Dent has the last move advantage, as its update always follows each attack.

Dent connects adversarial defense and domain adaptation, which share an interest in the sensitivity of deep networks to input shifts. Just as models fail on adversarial attacks, they fail on natural shifts like corruptions. Adversarial data is a particularly hard shift, as

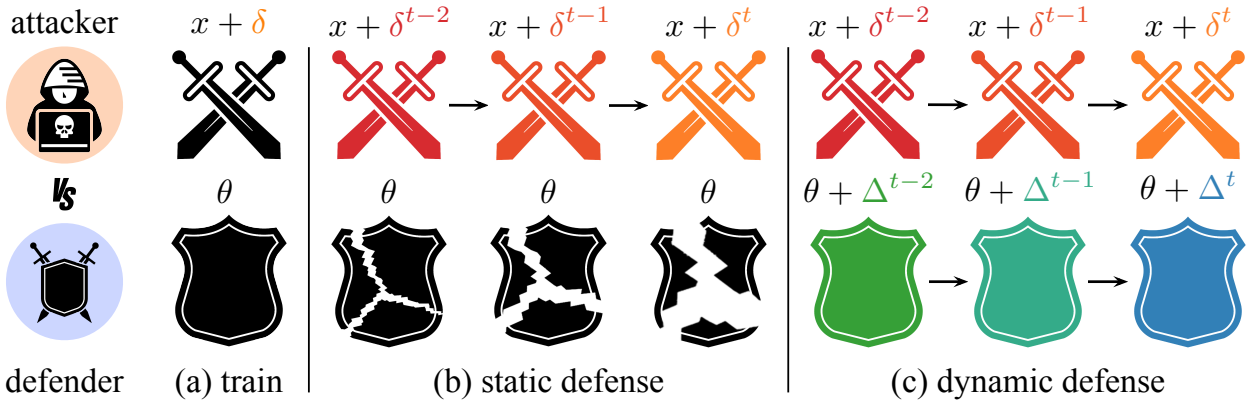


Figure 3.1: Attacks optimize the input $x + \delta$ against the model θ. Adversarial training optimizes θ for defense (a), but attacks update during testing while θ does not (b). Our *dynamic* defense improves robustness by adapting $\theta + \Delta$ during testing (c), so the attack cannot hit the same defense twice.

evidenced by the need for more parameters and optimization for adversarial training [70], and its negative side effect of reducing accuracy on natural data [78], [79]. Faced with these difficulties, we turn to adaptation, and change our focus to testing, rather than training more still.

Experiments evaluate dent against white-box attacks (APGD, FAB), black-box attack (Square), and adaptive attacks that are aware of its updates. Dent boosts state-of-the-art adversarial training defenses on CIFAR-10 by 20+ points against AutoAttack [80] at $\epsilon_\infty = 8/255$. Ablations inspect effects of iteration, parameterization, and batch size.

Our contributions

- We highlight an opportunity for dynamic defense: the last move advantage.
- We propose the first fully test-time dynamic defense: dent adapts both the model and input during testing without needing to alter training.
- Dent augments state-of-the-art adversarial training methods, improving robustness by 30% relative, and tops the AutoAttack leaderboard by 15+ points.
- We devise two adaptive attacks against dent: denying updates and mixing batches.

3.2 Related Work

Adversarial Defense For adaptive adversaries, which change in response to defenses, it is natural to consider dynamic defenses, which adapt in turn. [81] explain dynamic defenses

are promising in principle but caution they may not be effective in practice. Their analysis concerns randomized defenses, which do change, but their randomization does not adapt to the input. We argue for dynamic defenses that depend on the input to keep adapting along with the attacks. [82] supports dynamic defenses for similar reasons, but does not develop a specific defense. We demonstrate the first defense to optimize the model and input during testing for improved robustness.

Most defenses for deep learning focus on first-order adversaries [70], [74] that are equipped with gradient optimization but constrained by ℓ_p-norm bounds. Adversarial training and randomization are the most effective defenses against such attacks, but are nevertheless limited, as they are fixed during testing. Adversarial training [70], [74] trains on attacks, but a different or stronger adversary (by norm or bound) can overcome the trained defense [83], [84]. Randomizing the input [77], [85], [86] or network [75] requires the adversary to optimize in expectation [87], but can still fail with more iterations. Furthermore, these defenses gain adversarial robustness by sacrificing accuracy on natural data. Dent adapts during testing to defend against various attacks without more harm to natural accuracy.

Generative, self-supervised, and certified defenses try to align testing with training but are still static. Generative defenses optimize the input w.r.t. autoregressive [88], GAN [89], or energy [90] models, but the models do not adapt, and may be attacked by approximating their gradients [87]. Self-supervised defenses optimize the input w.r.t. auxiliary tasks [91], but again the models do not adapt. Certified defenses [77], [92] guarantee robustness within their training scope, but are limited to small perturbations by specific types of attacker during testing. Changing data distributions or adversaries requires re-training all of these defenses. Dent adapts during testing, without requiring (re-)training, and is the only method to update the model itself against attack.

Domain Adaptation Domain adaptation mitigates input shifts between the source (train) and target (test) to maintain model accuracy [12], [15]. Adversarial attack is such a shift, and adversarial error is related to natural generalization error [93], [94]. How then can adaptation inform dynamic defense? Train-time adaptation is static, like adversarial training, with the same issues of capacity, optimization, and re-computation when the data/adversary changes. We instead turn to test-time adaptation methods.

Test-time adaptation keeps updating the model as the data changes. Model parameters and statistics can be updated by self-supervision [18], normalization [35], and entropy minimization [1]. These methods improve robustness to natural corruptions [9], but their effect on adversarial perturbations is not known. We base our defense on entropy minimization as it enables optimization during testing without altering model architecture or training (as needed for self-supervision). For defense, we (1) extend the parameterization of adaptation with model and input transformations, (2) optimize for additional iterations, and (3) investigate usage on data that is adversarial, natural, or mixed. We are the first to report test-time model adaptation improves robustness to adversarial perturbations.

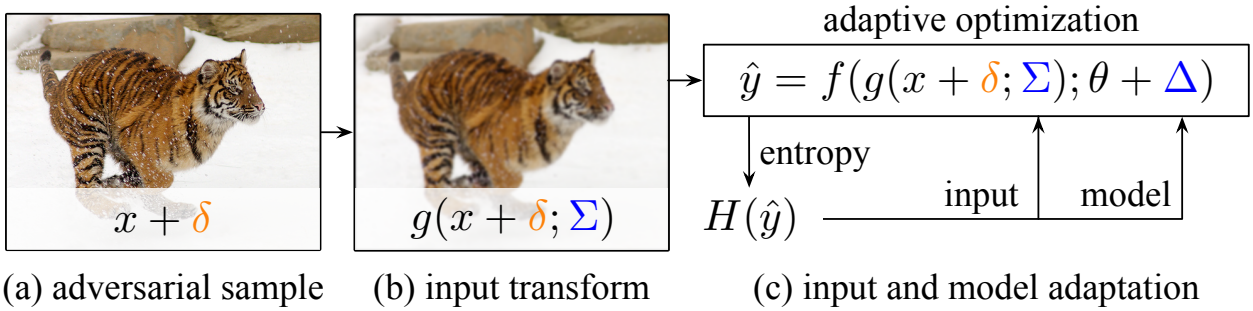


Figure 3.2: Dent adapts the model and input to minimize the entropy of the prediction $H(\hat{y})$. The model f is adapted by a constrained update Δ to the parameters θ. The input is adapted by smoothing g with parameters Σ. Dent updates batch-by-batch during testing.

Dynamic Inference A *dynamic* model conditionally changes inference for each input, while a *static* model unconditionally fixes inference for all inputs. There are various dynamic inference techniques, with equally varied goals, such as expressivity with more parameters or efficiency with less computation. All static models are alike; each dynamic model is dynamic in its own way.

Selection techniques learn to choose a subset of components [95], [96]. Halting techniques learn to continue or end computation [97], [98]. Mixing techniques learn to combine parameters [67], [99], [100]. Implicit techniques learn to iteratively update [101], [102]. While these methods learn to adapt during *training*, our method keeps adapting by directly optimizing during *testing*.

3.3 Method: Dynamic Defense by Test-Time Adaptation

Adversarial attacks optimize against defenses at test time, so defenses should fight back, and counter-optimize against attacks. Defensive entropy minimization (dent) does exactly this for dynamic defense by test-time adaptation.

In contrast to many existing defenses, dent alters testing, but not training. Dent only needs differentiable parameters for gradient optimization and probabilistic predictions for entropy measurement. As such, it applies to both adversarially-trained and nominally-trained models.

Preliminaries on Attacks and Defenses

Let $x \in \mathbb{R}^d$ and $y \in \{1, \ldots, C\}$ be an input sample and its corresponding ground truth. Given a model $f(\cdot; \theta) \colon \mathbb{R}^d \to \mathbb{R}^C$ parameterized by θ, the goal of the adversary is to craft

a perturbation $\delta \in \mathbb{R}^d$ such that the perturbed input $\tilde{x} = x + \delta$ causes a prediction error $f(x + \delta; \theta) \neq y$.

A targeted attack aims for a specific prediction of y', while an untargeted attack seeks any incorrect prediction. The perturbation δ is constrained by a choice of ℓ_p norm and threshold ϵ: $\{\delta \in \mathbb{R}^d \mid \|\delta\|_p < \epsilon\}$. We consider the two most popular norms for adversarial attacks: ℓ_∞ and ℓ_2.

Adversarial training is a standard defense, formulated by [70] as a saddle point problem,

$$\underset{\theta}{\operatorname{argmin}} \mathbb{E}_{(x,y)} \max_{\delta} L(f(x + \delta; \theta), y), \tag{3.1}$$

which the model minimizes and the adversary maximizes with respect to the loss $L(\hat{y}, y)$, such as cross-entropy for classification. The adversary iteratively optimizes δ by projected gradient descent (PGD), a standard algorithm for constrained optimization, for each step t via

$$\delta^t = \Pi_p(\delta^{t-1} + \alpha \cdot \operatorname{sign}(\nabla_{\delta^{t-1}} L(f(x + \delta^{t-1}; \theta), y))), \tag{3.2}$$

for projection Π_p onto the norm ball for $\ell_p < \epsilon$, step size hyperparameter α, and random initialization δ^0. The model optimizes θ against δ to minimize the loss of its predictions on perturbed inputs. This is accomplished by augmenting the training set with adversarial inputs from PGD attack.

Adversarial training is state-of-the-art, but static. Dynamic defenses offers to augment its robustness.

Defensive Entropy Minimization

Defensive entropy minimization (dent) counters attack updates with defense updates. While adversaries optimize to cross decision boundaries, entropy minimization optimizes to distance predictions from decision boundaries, interfering with attacks. As the adversary optimizes its perturbation δ, dent optimizes its adaptation Δ, Σ. Figure 3.2 shows dent's model (Δ) and input (Σ) updates.

Dent is dynamic because both Δ, Σ depend on the testing data, whether natural x or adversarial $x + \delta$. On the contrary, static defenses depend only on training data through the model parameters θ. Figure 3.3 contrasts static and dynamic defenses across the steps of attack optimization.

Entropy Objective Test-time optimization requires an unsupervised objective. Following tent [1], we adopt entropy minimization as our adaptation objective. Specifically, our defense objective is to minimize the Shannon entropy [19] $H(\hat{y})$ of the model prediction during testing $\hat{y} = f(x; \theta)$ for the probability $\hat{y}_c$ of class c:

$$H(\hat{y}) = -\sum_{c \in 1,\ldots,C} p(\hat{y}_c) \log p(\hat{y}_c) \tag{3.3}$$

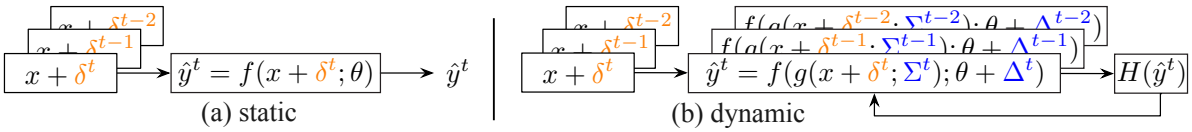


Figure 3.3: The adversary optimizes its attacks $\delta^{1 \cdots t}$ against the model f. Static defenses (left) do not adapt, and are vulnerable to persistent, iterative attacks. Our dynamic defenses (right) do adapt, and update their parameters Δ, Σ each time the adversary updates its attack δ.

Adaptation Parameters Dent adapts the model by Δ and input by Σ (Figure 3.2). For the model, dent adapts affine scale γ and shift β parameters by gradient updates and adapts mean μ and variance σ^2 statistics by estimation. These are a small portion of the full model parameters θ, in only the batch normalization layers [34]. However, they are effective for conditioning a model on changes in the task [67] or data [1], [35]. For the input, dent updates Gaussian smoothing g by gradient updates of the parameter Σ, while adjusting the filter size for efficiency [103]. This controls the degree of smoothing dynamically, unlike defense by static smoothing [77].

In standard models the scale γ and shift β parameters are shared across inputs, and so adaptation updates batch-wise. For further adaptation, dent can update sample-wise, with different affine parameters for each input. In this way it adapts more than prior test-time adaptation methods with batch-wise parameters [1], [35].

Our model and input parameters are differentiable, so end-to-end optimization coordinates them against attacks as layered defenses. This coordination is inspired by CyCADA [104], for domain adaptation, but dent differs in its purpose and its unified loss. CyCADA also optimizes input and model transformations but does so in parallel with separate losses. Our defensive optimization is joint and shares the same loss.

Update Algorithm In summary, when the adversary attacks with perturbation δ^t, our dynamic defense reacts with Σ^t, Δ^t. The parameters of the model f and smoothing g are updated by $\text{argmin}_{\Sigma,\Delta} H(f(g(x+\delta; \Sigma); \theta+\Delta))$ through test-time optimization. At each step, dent estimates the normalization statistics μ, σ and then updates the parameters γ, β, Σ by the gradient of entropy minimization. Figure 3.3 contrasts static defenses and dynamic defenses that update like dent.

Dent adapts on batches rather than samples. Batch-wise adaptation stabilize optimization for entropy minimization. The defense parameters reset between batches.

Discussion The purpose of a dynamic defense is to move when the adversary moves. When the adversary submits an attack $x+\delta^t$, the defense counters with Δ^t. In this way, the defense has the last move, and therefore an advantage.

Table 3.1: Dent boosts the robustness of adversarial training on CIFAR-10 against AutoAttack. Adversarial training is static, but dent is dynamic, and adapts during testing. Dent adapts batch-wise, while dent+ adapts sample-wise, surpassing the state-of-the-art for static defense at *robustbench.github.io*.

Accuracy(%)	Natural	Adversarial		
		static	dent	dent+
$\epsilon_\infty = 8/255$				
[105]	89.6	59.5	74.7	**82.3**
[106]	84.4	54.4	61.2	**75.2**
[107]	83.3	43.2	52.3	**71.8**
[108]	88.0	41.4	47.6	**64.4**
$\epsilon_2 = 0.5$				
[106]	89.5	73.4	77.8	**85.7**
[109]	88.7	67.7	69.7	**81.3**
[110]	89.1	66.4	73.4	**85.3**
[108]	88.0	66.1	70.3	**82.8**

Our dynamic defense changes the model, and therefore its gradients, but differs from gradient obfuscation [87]. Our defense does not rely on (1) shattered gradients, as the update does not cause non-differentiability or numerical instability; (2) stochastic gradients, as the update is deterministic given the input, model, and prior updates; nor (3) exploding/vanishing gradients, as the update improves robustness with even a single step (although more steps are empirically better). Dent forces the attack to rely on a *stale* gradient, as δ^t follows Δ^{t-1}, while the model adapts by Δ^t.

3.4 Experiments

We evaluate dent against white-box, black-box, and adaptive attacks with a variety of static defenses and datasets. For attacks, we choose the AutoAttack [80] benchmark, which includes four attack types spanning white-box/gradient and black-box/query attacks. For static defenses, we choose strong and recent adversarial training methods, and we also experiment with nominally trained models. For datasets, we evaluate on CIFAR-10/CIFAR-100 [30], as they are popular datasets for adversarial robustness, and ImageNet [29], as it is a large-scale dataset. We ablate the choice of model/input adaptation, parameterization, and the number of updates.

Setup

Metrics We score natural accuracy on the regular test data x and adversarial accuracy on the perturbed test data $x + \delta$. Each is measured as percentage accuracy (higher is better). We report the worst-case adversarial accuracy across attacks.

Test-time Optimization We optimize batch-wise Δ (dent) and sample-wise Δ (dent+). Dent updates by Adam [37] with learning rate 0.001. Dent+ updates by AdaMod [111] with learning rate 0.006. Σ updates use learning rate 0.25. All updates use batch size 128 and no weight decay. Dent+ regularizes updates by information maximization [50], [112]. We tuned update hyperparameters against PGD attacks.

Architecture For comparison with existing defenses, we keep the architecture and training the same, and simply load the public reference models provided by RobustBench [113]. For analysis and ablation experiments, we define a residual net with 26 layers and a width multiplier of 4 (ResNet-26-4) [8], [114], following prior work on adaptation [1], [18].

Attack Types & Threat Model

We evaluate standard white-box and black-box attacks with adversarially-trained models and nominally-trained models, as well as dent-specific adaptive attacks. We primarily evaluate against AutoAttack's ensemble of:

1. APGD-CE [70], [80], an untargeted white-box attack by cross-entropy,
2. APGD-DLR [80], a targeted white-box attack with a shift and scale invariant loss,
3. FAB [115], a targeted white-box attack for minimum-norm perturbation,
4. Square Attack [116], an untargeted black-box attack with square-shaped updates.

These attacks are cumulative, so a defense is only successful if it holds against each type. Following convention, we evaluate ℓ_∞ attacks with $\epsilon_\infty = 8/255$ and ℓ_2 attacks with $\epsilon_2 = 0.5$. This is the standard evaluation adopted by the popular RobustBench benchmark [113].

We devise and experiment with two adaptive attacks against dent and its dynamic updates. The first interferes with adaptation by denying updates: it optimizes offline against θ without Δ, Σ updates. The second interferes with adaptation by mixing data: it combines adversarial data and natural data in the same batch. Both are specific to dent to complement our general evaluation by AutoAttack.

These attacks fall under the usual white-box threat model. The adversary has full access to the classifier, including its architecture and parameters, and the defense, such as dent's adaptation parameters and statistics. With this access the adversary chooses an attack for each input, but it cannot choose the inputs (the test set is fixed).

Table 3.2: AutoAttack includes four attack types, and dent improves robustness to each on CIFAR-10 against ℓ_∞ attacks. We evaluate without dent (-) and with dent (+).

Accuracy(%)	APGD-CE		APGD-DLR		FAB		Square	
	-	+	-	+	-	+	-	+
[107]	45.9	57.6	43.2	52.3	43.2	52.3	43.2	52.3
[108]	50.1	60.2	41.6	48.0	41.5	47.7	41.4	47.6

We include one additional requirement: dent assumes access to test *batches* rather than individual test *samples*. While independent, sample-wise defense is ideal for simplicity and latency, batch processing is not impractical. For example, cloud deployments of deep learning batch inputs for throughput efficiency, and large-scale systems handle many inputs per unit time [117].

Dynamic Defense of Adversarial Training

We extend static adversarial training defenses with dynamic updates by dent. Compared to nominal training, adversarial training achieves higher adversarial accuracy but lower natural accuracy. The purpose of dent is to improve adversarial accuracy without further harming natural accuracy.

Dent improves state-of-the-art defenses. Table 3.1 shows state-of-the-art adversarial training defenses [105], [107]–[110] with and without dynamic defense by dent. Note that dent does not specialize to the choice of norm or bound, unlike adversarial training, but instead adapts to each attack during testing. In each case, dent significantly improves adversarial accuracy and maintains natural accuracy.

Dent updates batch-wise for 30 steps. Dent+ is more robust in fewer steps by sample-wise adaptation. With sample-wise (γ, β) parameters, dent+ needs only six steps to reach an adversarial accuracy within 90% of the natural accuracy. These experiments only include model adaptation of Δ, without input adaptation of Σ, as we found it unnecessary when combined with adversarial training.

Dent helps across attack types. Table 3.2 evaluates dent against each attack in the AutoAttack ensemble. Dent improves robustness to each attack type. We report the worst case across these types in the remainder of our experiments.

Table 3.3: Ablation of model adaptation (Δ), input adaptation (Σ), and steps on the accuracy of a nominally-trained model with dent.

Δ	Σ	Step	Time	Natural	Adversarial $\epsilon_\infty = \frac{1.5}{255}$	Adversarial $\epsilon_2 = 0.2$
×	None	0	1.0×	95.6	8.8	9.2
√	None	1	3.6×	95.6	15.0	13.5
×	Stat.	0	1.0×	86.2	25.8	23.6
√	Stat.	1	3.6×	86.3	27.5	24.4
√	Stat.	10	25.9×	86.3	37.6	30.9
√	Dyna.	10	26.1×	92.5	45.4	36.5

Dent helps across datasets and architectures. We experiment on ImageNet to check scalability. We evaluate the defense of [107], one of few defenses that scales to this dataset, against strong ℓ_∞-PGD attacks with 30 iterations, step size of 0.1, and five random starts. Dent improves the adversarial accuracy by 14 points against PGD at $\epsilon_\infty = 4/255$ and natural accuracy by 23 points.

Dynamic Defense of Nominal Training

Dent improves the adversarial accuracy of off-the-shelf, nominally-trained models. As dent does not assume adversarial training, it can apply to various models at test time.

For nominal training, we exactly follow the CIFAR reference training in pycls [28], [118] with ResNet-26-4/ResNet-32-10 architectures. Briefly, we train by stochastic gradient descent (SGD) for 200 epochs with batch size 128, learning rate 0.1 and decay 0.0005, momentum 0.9, and a half-period cosine schedule.

We evaluate against ℓ_∞ and ℓ_2 AutoAttack attacks on CIFAR-10. As the nominally-trained models have no static defense, we constrain the adversaries to smaller ϵ perturbations.

Dent defends nominally-trained models. Table 3.3 inspects how each part of dent affects adversarial accuracy and natural accuracy. When applying dent to nominally-trained models, model adaptation through Δ is further helped by input adaptation through Σ. In just a single step, the Δ update improves adversarial accuracy without affecting natural accuracy. from 8.8% to 15.0% against ℓ_∞ attacks with just a single step. With 10 steps, and Σ adaptation, dent improves the model's adversarial accuracy to 45.4% against ℓ_∞ attacks and 36.5% against ℓ_2 attacks. In total, dent boosts ℓ_∞ and ℓ_2 adversarial accuracy by almost

Table 3.4: Adaptive attack by denying updates. We transfer attacks from static models to dent and then evaluate nominal and adversarial training [70] against ℓ_∞ and ℓ_2 AutoAttack. Attacks break the static models (static-static), but fail to transfer to our dynamic defense (static-dent).

	Nominal		Adversarial	
	ϵ_∞=$\frac{1.5}{255}$	ϵ_2=0.2	ϵ_∞=$\frac{8}{255}$	ϵ_2=0.5
Static-Static	11.6	11.0	42.0	44.1
Static-Dent	82.5	81.6	50.0	50.2

Table 3.5: Adaptive attack by mixing adversarial and natural data. We report the adversarial accuracy on mixed batches, from low to high amounts of adversarial data. Dent improves on adversarial training (43.8%) across mixing proportions within 10 steps.

μ, σ	Step	1	10%	25%	50%	75%	90%
×	1	-	43.4	43.2	44.0	44.2	43.8
×	10	62.4	51.2	49.6	48.7	48.7	47.6
√	1	-	41.7	41.4	43.2	44.1	44.7
√	10	54.9	47.6	47.7	49.7	50.6	50.9

40 and 30 points while only sacrificing 3 points of natural accuracy. Dent delivers this boost at test-time, without re-training.

Input adaptation helps preserve natural accuracy. Gaussian smoothing significantly improves adversarial accuracy. This agrees with prior work on denoising by optimization [76] or randomized smoothing [77]. Tuned as a fixed hyperparameter, smoothing helps adversarial accuracy but hurts natural accuracy. Optimized end-to-end, our dynamic smoothing reduces the natural accuracy gap. On natural data, the learned Σ for the blur decreases to approximate the identity transformation.

Adaptive Attacks on Dent Updates

We adaptively attack dent through its use of adaptaton by (1) denying updates and (2) mixing batches. To deny updates, we attack the static model offline by optimizing against θ without Δ, Σ updates, then submit this attack to dent. This attempts to shortcircuit adaptation by disrupting the first update with a sufficiently strong perturbation. To mix batches, we mix adversarial and natural data in the same batch. This attempts to prevent adaptation by aligning batch statistics with natural data.

Denying Updates The aim of this attack is to defeat adaptation on the first move, before dent can update to counter it. We optimize against the static model alone to prevent defensive optimization until adversarial optimization is complete. Under this attack, the input to dent is the final perturbation derived by adversarial attack against the static model.

Table 3.6: Dynamic defenses can trade computation and adaptation. More steps are more robust on CIFAR-10 with ℓ_∞ AutoAttack. Dent+ reaches higher adversarial accuracy in fewer steps.

	STEPS			
DENT	0	20	30	40
[105]	59.5	68.3	74.7	76.1
[107]	43.2	48.2	52.3	55.1
[108]	41.4	45.4	47.6	48.7
DENT+	0	1	3	6
[108]	41.4	46.5	57.7	64.4

We examine whether these offline perturbations can disrupt adaptation. Table 3.4 shows that dent can still defend against this attack. This suggests that updating, and having the last move, remains an advantage for our dynamic defense.

Mixing Batches Dent adapts batch-wise, with the underlying assumption that one shared transformation can defend the whole batch. We challenge this assumption by evaluating mixed batches of adversarial and natural data. In Table 3.5, we vary the ratio of adversarial and natural data in each batch and measure accuracy on the adversarial portion.

At the extreme, we consider an adaptive attack with only one adversarial input per batch. Specifically, we batch one adversarial input with 15 natural inputs randomly chosen from the test set. This adaptive attack aims to reduce adaptation by the dynamic defense, as natural inputs do not need adaptation.

Dent is generally robust to batch mixing, and improves over adversarial training in 10 steps or less.

Ablations & Analysis

More updates deliver more defense. The number of steps can balance defense and computation. Table 3.6 shows that more steps offer stronger defense for both dent and dent+. However, more steps do nevertheless require more computation: ten-step optimization takes 25.9× more operations than the static model (Table 3.3). As a plus, dent+ is not only more robust, but also more efficient in needing fewer steps. Note that the computational difference between dent and dent+ is negligible, as the adaptation parameters are such a small fraction of the model.

Table 3.7: Ablation of model adaptation with and without normalization statistics (μ, σ) and affine parameters (γ, β) updates.

Accuracy(%)		Nominal		Adversarial	
μ, σ	γ, β	$\epsilon_\infty = \frac{1.5}{255}$	ϵ_2=0.2	$\epsilon_\infty = \frac{8}{255}$	ϵ_2=0.5
×	×	8.8	9.2	43.8	47.3
√	×	11.7	11.2	41.8	44.1
×	√	16.8	16.2	49.9	57.3
√	√	21.2	15.2	50.4	53.0

Table 3.8: Sensitivity analysis of batch size and adversarial accuracy with dent. With static batch statistics (×), small batch sizes are better. With dynamic batch statistics (√), small batch sizes are worse.

μ, σ	Type	1	2	4	8	16	32	64
×	Nat.	85.9	86.0	85.9	85.9	86.1	86.1	86.2
×	Adv.	70.4	69.5	67.8	65.3	61.9	58.6	55.1
√	Nat.	11.1	68.1	76.3	80.9	83.4	84.9	85.8
√	Adv	5.8	35.9	48.3	53.0	55.3	54.4	52.9

Model adaptation updates depend on the attack type. Dent adapts by adjusting normalization statistics and affine transformation parameters. Dent can fix or update the normalization statistics (μ, σ) by using static training statistics (×) or dynamic testing statistics (√); Dent can fix or update the affine parameters (γ, β) by not taking gradients (×) or applying gradient updates (√). Table 3.7 compares each combination: affine updates always help, but both updates together hurt ℓ_2 robustness.

Batch size We analyze dent's sensitivity to batch size and focus on small batch sizes. Some real-world tasks, such as autonomous driving, naturally provide a small batch of inputs (from consecutive video frames or various cameras, for example), and so we confirm that dent can maintain robustness on such small batches. Table 3.8 varies batch sizes to check dent's natural and adversarial accuracy.

3.5 Discussion

In advocating for dynamic defenses, we hope that test-time updates can help level the field for attacks and defenses. Our proposed defensive entropy method takes a first step by countering adversarial optimization with defensive optimization over the model and input. While more test-time computation is needed for the back-and-forth iteration of attacks and defenses, the cost of defense scales with the cost of attack, and some use cases may prefer slow and strong to fast and wrong.

Limitations Dent depends on batches to adapt, especially for fully test-time defense without adversarial training. It also relies on a particular choice of model and input parameters. A different objective could possibly lessen its dependence on batch size and reliance on constrained updates. More generally, dynamic defenses may present difficulties for certification

or deployment, as they could drift. Along with how to update, improved defenses could investigate when to reset, or how to batch inputs for joint optimization.

Domain Adaptation for Adversarial Defense Inquiry into adversarial defense and domain adaptation examines two sides of the same coin. Both trade in the currencies of accuracy and generalization but are not in close contact. We expect further exchange between these subjects to pay dividends in new kinds of dynamic inference for defense and adaptation alike. In particular, while dent is inspired by test-time adaptation, defenses could also be informed by open set/compound adaptation [119] to perhaps cope with multiple adversaries [84].

Benchmarking Standardized benchmarking, by AutoAttack and RobustBench for example, drives progress by competition and empirical corroboration. Dent brings adversarial accuracy on their benchmark within 90% of natural accuracy for three of the most accurate methods tested [105], [108], [120]. This is encouraging, but more research is needed to fully characterize dynamic defenses like dent. However, RobustBench is designed for static defenses, and disqualifes dent by its rule against test-time optimization. Continued progress could depend on a new benchmark to standardize rules for how attacks and defenses alike may adapt.

By fighting gradients with gradients, dent shows the potential for dynamic defenses to update and counter adversarial attacks. The next steps—by attacks and defenses—will tell.

Chapter 4

Dynamic Scale Inference by Entropy Minimization

4.1 Introduction

The world is infinite in its variations, but our models are finite. While inputs differ in many dimensions and degrees, a deep network is only so deep and wide. To nevertheless cope with variation, there are two main strategies: static enumeration and dynamic adaptation. Static enumeration defines a set of variations, processes them all, and combines the results. For example, pyramids enumerate scales [121], [122] and group-structured filters enumerate orientations [123]. Dynamic adaptation selects a single variation, conditioned on the input, and transforms processing accordingly. For example, scale-space search [124], [125] selects a scale transformation from input statistics and end-to-end dynamic networks select geometric transformations [126], [127], parameter transformations [128], and feature transformations [67] directly from the input. Enumeration and adaptation both help, but are limited by computation and supervision, because the sets enumerated and ranges selected are bounded by model size and training data.

Deep networks for vision exploit enumeration and adaptation, but generalization is still limited. Networks are enumerative, by convolving with a set of filters to cover different variations then summing across them to pool the variants [6], [32], [129]. For scale variation, image pyramids [121] and feature pyramids [47], [130] enumerate scales, process each, and combine the outputs. However, static models have only so many filters and scales, and may lack the capacity or supervision for the full data distribution. Dynamic models instead adapt to each input [131]. The landmark scale invariant feature transform [125] extracts a representation adapted to scales and orientations predicted from input statistics. Dynamic networks, including spatial transformers [126] and deformable convolution [127], make these predictions and transformations end-to-end. Predictive dynamic inference is however insufficient: the predictor may be imperfect in its architecture or parameters, or may not generalize to data it was not designed or optimized for. Bottom-up prediction, with only one step of

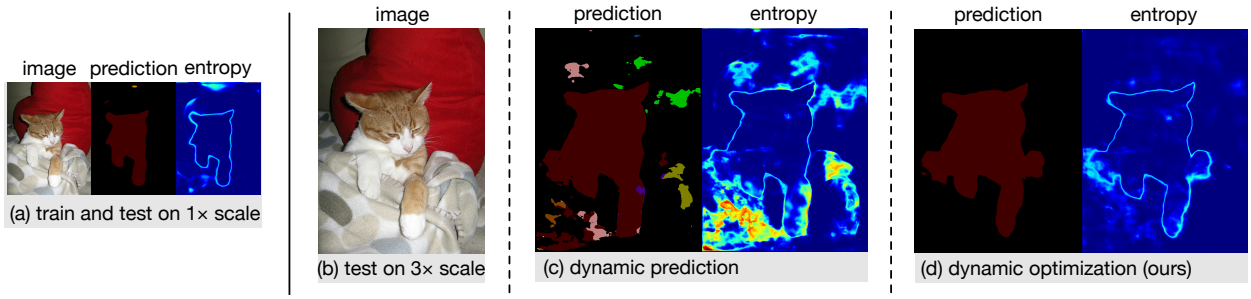


Figure 4.1: Generalization across scale shifts between training and testing conditions is difficult. Accuracy is high and prediction entropy is low for training and testing at the same scale (left). Accuracy drops and entropy rises when tested at 3x the training scale, even when the network is equipped with dynamic receptive fields to adapt to scale variation (middle). Previous approaches are limited to one-step, feedforward scale inference, and are unable to handle a 3x shift. In contrast our iterative gradient optimization approach is able to adapt further (right), and achieve higher accuracy by minimizing entropy with respect to task and scale parameters.

adaptation, can struggle to counter variations in scale and other factors that are too large or unfamiliar.

To further address the kinds and degrees of variations, including extreme out-of-distribution shifts, we devise a complementary third strategy: unsupervised optimization during inference. We define an unsupervised objective and a constrained set of variables for effective gradient optimization. Our novel inference objective minimizes the entropy of the model output to optimize for confidence. The variables optimized over are task parameters for pixel-wise classification and structure parameters for receptive field adaptation, which are updated together to compensate for scale shifts. This optimization functions as top-down feedback to iteratively adjust feedforward inference. In effect, we update the trained model parameters to tune a custom model for each test input.

Optimization during inference extends dynamic adaptation past the present limits of supervision and computation. Unsupervised optimization boosts generalization beyond training by top-down tuning during testing. Iterative updates decouple the amount of computation, and thus degree of adaptation, from the network architecture. Our main result is to demonstrate that adaptation by entropy optimization improves accuracy and generalization beyond adaptation by prediction (see Figure 4.1), which we show for semantic segmentation by inference time optimization of a dynamic Gaussian receptive field model [103] on the PASCAL VOC [132] dataset.

4.2 Method: Iterative Dynamic Inference by Optimization

Our approach extends dynamic scale inference from one-step prediction to multi-step iteration through optimization. For optimization during inference, we require an objective to optimize and variables to optimize over. Lacking task or scale supervision during inference, the objective must be unsupervised. For variables, there are many choices among parameters and features. Our main contribution is an unsupervised approach for adapting task and structure parameters via gradient optimization to minimize prediction entropy.

Note that our *inference* optimization is distinct from the *training* optimization. We do not alter training in any way: the task loss, optimizer, and model are entirely unchanged. In the following, optimization refers to our inference optimization scheme, and not the usual training optimization.

To optimize inference, a base dynamic inference method is needed. For scale, we choose local receptive field adaptation [103], [127], [133], because scale varies locally even within a single image. In particular, we adopt dynamic Gaussian receptive fields [103] that combine Gaussian scale-space structure with standard "free-form" filters for parameter-efficient spatial adaptation. These methods rely on feedforward regression to infer receptive fields that we further optimize.

Figure 4.2 illustrates the approach. Optimization is initialized by feedforward dynamic inference of Gaussian receptive fields [103]. At each following step, the model prediction and its entropy are computed, and the objective is taken as the sum of pixel-wise entropies. Model parameters are iteratively updated by the gradient of the objective, resulting in updated predictions and entropy. Optimization of the parameters for the Gaussian receptive fields is instrumental for adapting to scale.

Objective: Entropy Minimization

Unsupervised inference objectives can be bottom-up, based on the input, or top-down, based on the output. To augment already bottom-up prediction, we choose the top-down objective of entropy minimization. In essence, the objective is to reduce model uncertainty.

More precisely, for the pixel-wise output $\hat{Y} \in [0, 1]^{C \times H \times W}$ for C classes and an image of height H and width W, we measure uncertainty by the Shannon entropy [19]:

$$\mathbf{H}_{i,j}(\hat{Y}) = -\sum_{c} \mathbf{P}(y_{i,j} = c) \log \mathbf{P}(y_{i,j} = c) \tag{4.1}$$

for each pixel at index i, j to yield pixel-wise entropy of the same spatial dimensions as the output.

Entropy is theoretically motivated and empirically supported. By inspection, we see that networks tend to be confident on in-distribution data from the training regime. (Studying the probabilistic calibration of networks [71] confirms this.) In our case, this holds for testing

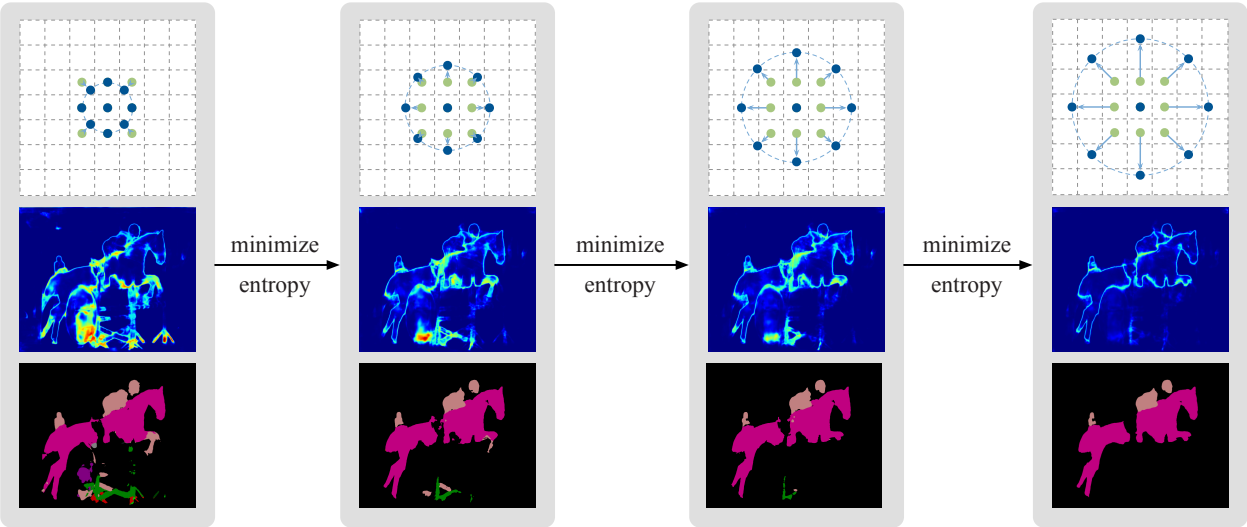


Figure 4.2: Overview. Dynamic receptive field scale (top) is optimized according to the output (bottom) at test time. We optimize receptive field scales and filter parameters to minimize the output entropy (middle). Optimizing during inference makes iterative updates shown from left to right: receptive field scale adapts, entropy is reduced, and accuracy is improved. This gives a modest refinement for training and testing at the same scale, and generalization improves for testing at different scales.

scales similar to the training scales, with high entropy on segment contours. On out-of-distribution data, such as scale shifts, the output entropy is higher and less structured. For qualitative examples, see Figures 4.1 and 4.2.

This objective is severe, in that its optimum demands perfect certainty (that is, zero entropy). As a more stable alternative, we consider adaptively thresholding the objective by the average entropy across output pixels. We calculate the mean entropy at each iteration, and only take the gradient of pixels with above-average entropy. This mildly improves accuracy.

Our final objective is then:

$$L(\hat{Y}) = \sum_{i,j \in \mathbf{S}} \mathbf{H}_{i,j}(\hat{Y}) \quad \text{for} \quad S = \{i, j : \mathbf{H}_{i,j} > \mathbf{H}_{\mu}\} \tag{4.2}$$

where $\mathbf{S}$ is the set of pixels with entropy above the average $\mathbf{H}_{\mu}$. At each step, we re-calculate the average entropy and re-select the set of violating pixels. In this way, optimization is focused on updating predictions where the model is the most uncertain.

Variables: Task and Structure Parameters

We need to pick the variables to optimize over so that there are enough degrees of freedom to adapt, but not so many that overfitting sets in. Furthermore, computation time and memory

demand a minimal set of variables for efficiency. Choosing parameters in the deepest layers of the network satisfy these needs: capacity is constrained by keeping most of the model fixed, and computation is reduced by only updating a few layers. The alternative of choosing all the parameters, and optimizing end-to-end during inference, is ineffective and inefficient: inference is slower and less accurate than feedforward prediction.

We select the task parameters θ_{score} of the output classification filter, for mapping from features to classes, and the structure parameters θ_{scale} of the scale regression filter, for mapping from features to receptive field scales. Optimizing over these parameters indirectly optimizes over the local predictions for classification scores $\hat{Y}$ and scales $\hat{\Sigma}$.

Why indirectly optimize the outputs and scales via these parameters, instead of direct optimization? First, dimensionality is reduced for regularization and efficiency: the parameters are shared across the local predictions for the input image and have fixed dimension. Additionally, this preserves dependence on the data: optimizing directly over the classification predictions admits degenerate solutions that are independent of the input.

Algorithm: Initialization, Iteration, and Termination

Initialization The unaltered forward pass of the base network gives scores $\hat{Y}^{(0)}$ and scales $\hat{\Sigma}^{(0)}$.

Iteration For each step t, the loss is the sum of thresholded entropies of the pixel-wise predictions $\hat{Y}^{(t)}$. The gradient of the loss is taken for the parameters $\theta^{(t)}_{\text{score}}$ and $\theta^{(t)}_{\text{scale}}$. The optimizer then updates both to yield $\theta^{(t+1)}_{\text{score}}$ and $\theta^{(t+1)}_{\text{scale}}$. Given the new parameters, a partial forward pass re-infers the local scales and predictions for $\hat{Y}^{(t+1)}$ and $\hat{\Sigma}^{(t+1)}$. This efficient computation is a small fraction of the initialization forward pass.

Termination The number of iterations is set and fixed to control the amount of inference computation. We do so for simplicity, but note that in principle convergence rules such as relative tolerance could be used with the loss, output, or parameter changes each iteration for further adaptivity. Figure 4.3 shows the progress of our inference optimization across iterations.

4.3 Experiments

We experiment with extending from predictive to iterative dynamic inference for semantic segmentation, because this task has a high degree of appearance and scale variation. In particular, we show results for iterative optimization of classifier and scale parameters in a dynamic Gaussian receptive field model [103] on the PASCAL VOC [132] dataset. By adapting both task and structure parameters, our approach improves accuracy on in-distribution inputs and generalizes better on out-of-distribution scale shifts. We ablate which variables to optimize and for how many steps, and analyze our choices by oracle and adversary results.

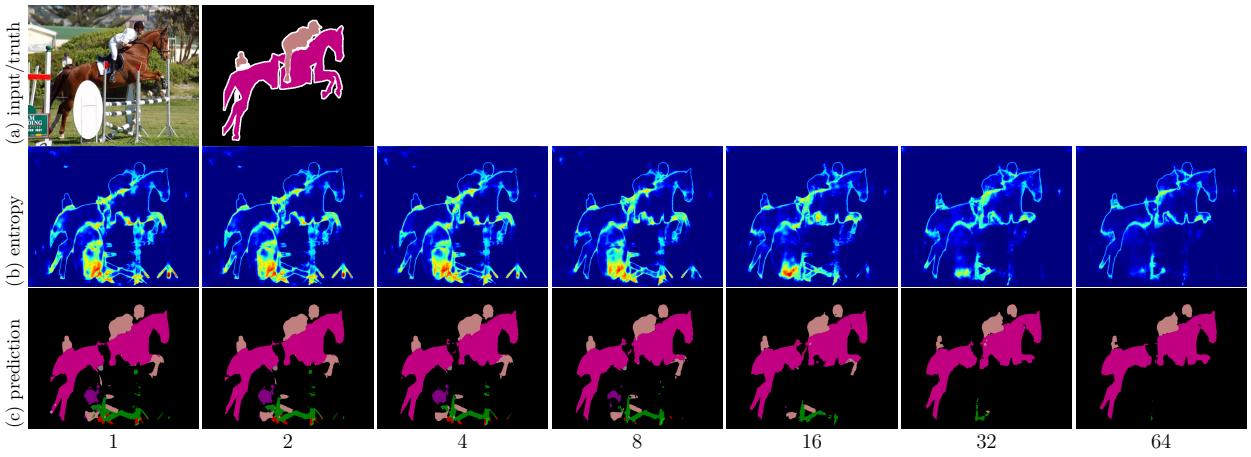


Figure 4.3: Iterative dynamic inference by our entropy minimization. We optimize output entropy with respect to task and scale parameters. (a) Input and ground truth. (b) Output entropy. (c) Output prediction. Our optimization reduces entropy and improves prediction accuracy.

These experiments establish the efficacy of entropy minimization during inference for scale adaptation, while oracle results show opportunity for further progress.

Data and Metric PASCAL VOC [132] is a well-established semantic segmentation benchmark with 20 semantic classes and a background class. The original dataset only has 1,464, 1,449 and 1,456 images with segmentation annotations for training, validation, and testing, respectively. As is standard practice, we include the additional 9,118 images and annotations from [134], giving 10,582 training samples in total. We measure accuracy by the usual metric of mean intersection-over-union (IoU). We report our results on the validation set.

Architecture We choose deep layer aggregation (DLA) [135] as a representative state-of-the-art architecture from the family of fully convolutional networks [47]. DLA utilizes the built-in feature pyramid inside the network via iterative and hierarchical aggregation. Our implementation is based on PyTorch [136].

Training We train our model on the original scale of the dataset. We optimize via stochastic gradient descent (SGD) with batch size 64, initial learning rate 0.01, momentum 0.9, and weight decay 0.0001 for 500 epochs. We use the "poly" learning rate schedule [137] with power 0.9. For the model with no data augmentation ("w/o aug"), the input images are padded to 512×512 . As for the "w/ aug" model, data augmentation includes (1) cropping to 512×512, (2) scaling in $[0.5, 2]$, (3) rotation in $[-10°, 10°]$, (4) color distortion [138], and (5) horizontal flipping.

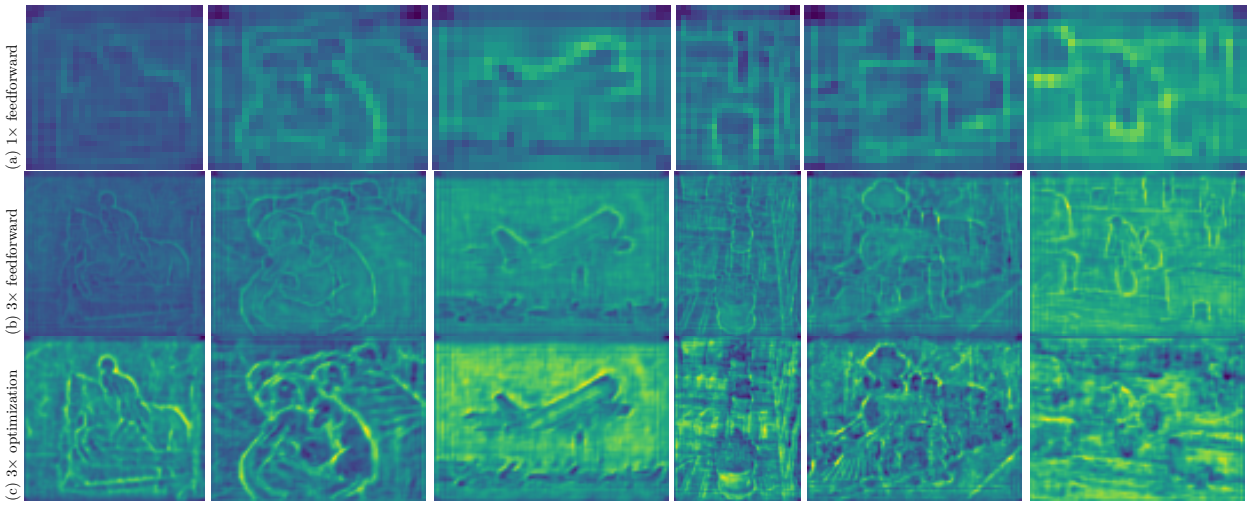


Figure 4.4: Visualization of dynamic receptive field sizes across scale shift. Darker indicates smaller, and brighter indicates larger. (a) is the feedforward inference at 1× scale while (b) and (c) are the feedforward prediction baseline and our iterative optimization at 3× scale. Observe that (a) and (b) are visually similar, in spite of the 3× scale shift, showing that the predictor has failed to adapt. Optimization adapts further by updating the output and scale parameters, and the dynamic receptive fields are accordingly larger. This is shown by how (c) is consistently brighter than (b).

Testing We test our model on different scales of the dataset in the $[1.5, 4.0]$ range. We optimize the model parameters for adaptation via Adam [37], batching all image pixels together, and setting the learning rate to 0.001. The model is optimized episodically to each input, and the parameters are reset between inputs. No data augmentation is used during inference to isolate the role of dynamic inference by the model.

Results

We compare the semantic segmentation accuracy of our optimization with a prediction baseline and optimization by oracle and adversary. The baseline is a one-step dynamic model using feedforward scale regression to adapt receptive fields following [103]. We train on a narrow range of scales and test on a broader range of scales to measure refinement, the improvement for the training scales, and generalization, the improvement for the new scales. This baseline is the initialization for our iterative optimization approach: the output and scale predictions for the initial iteration are inferred by the one-step model. For analysis results, the oracle and adversary optimize during inference to respectively minimize/maximize the cross-entropy loss of the output and the truth.

As reported in Table 4.1, our method consistently improves on the baseline by $\sim$2 points

Table 4.1: Comparison of our method with the feedforward scale regression baseline and the oracle. Results are scored by intersection-over-union (higher is better). "w/o aug" excludes data augmentation, where "w/ aug" includes scaling, rotation, and other augmentation. Even though data augmentation reduces the effect of scale variation, our method further improves accuracy for all scales.

		1.5	2.0	2.5	3.0	3.5	4
w/o aug	scale regression	68.2	59.3	50.2	41.8	34.0	27.5
	entropy optimization (ours)	**69.0**	**60.1**	**51.9**	**43.5**	**35.8**	**29.2**
	oracle	72.0	64.4	55.8	47.5	39.2	32.1
w/ aug	scale regression	74.2	70.8	65.8	59.8	53.5	46.8
	entropy optimization (ours)	**74.6**	**71.7**	**67.7**	**61.8**	**56.0**	**49.0**
	oracle	78.0	75.7	72.3	67.8	62.4	55.6

Table 4.2: Ablation of the number of iterations for optimization. Entropy minimization saturates after 32 steps, while oracle optimization continues to improve.

		1.5	2.0	2.5	3.0	3.5	4
step 0	scale regression	68.2	59.3	50.2	41.8	34.0	27.5
step 32	entropy optimization (ours)	**69.0**	60.1	51.9	**43.5**	**35.8**	**29.2**
	oracle	72.0	64.4	55.8	47.5	39.2	32.1
step 128	entropy optimization (ours)	69.0	**60.3**	**52.1**	**43.5**	35.2	28.5
	oracle	73.3	68.6	61.8	54.0	45.7	38.5

for all scales, which indicates that our unsupervised optimization for iterative inference helps the model generalize better across scales. When the scale shift is larger, there is likewise a larger gap.

To evaluate the effect of data augmentation, we experiment with ("w/ aug") and without ("w/o aug"). Data augmentation significantly improves generalization across scales. Note that our optimization during inference still improves the model with data augmentation by the same amount.

Ablations

We ablate the choice of parameters to optimize and the number of updates to make.

We optimize during inference to adapt the task parameters (score) of the classifier and structure parameters (scale) of the scale regressor. The task parameters map between the visual features and the classification outputs. Updates to the task parameters are the most

Table 4.3: Analysis of entropy minimization (compared to oracle and adversary optimization) and ablation of the choice of parameters for optimization (score, scale, or both). The oracle/adversary optimizations minimize/maximize the cross-entropy of the output and truth to establish accuracy bounds. The adversary results show that our method helps in spite of the risk of harm. The oracle results show there are still better scales to be reached by further progress on dynamic inference.

	test on 1×			test on 3×		
	score	scale	both	score	scale	both
scale regression	69.8	69.8	69.8	59.8	59.8	59.8
entropy optimization (ours)	70.2	**70.7**	**70.6**	61.1	61.8	**62.3**
oracle	73.7	75.6	77.7	63.9	67.8	71.3
adversary	67.4	55.9	52.4	57.4	47.4	44.4

direct way to alter the pixelwise output distributions. Updates to the structure parameters address scale differences by adjusting receptive fields past the limits of the feedforward scale regressor. From the experiments in Table 4.3, both are helpful for refining accuracy and reducing the generalization gap between different scales. Optimizing end-to-end, over all parameters, fails to achieve better than baseline results.

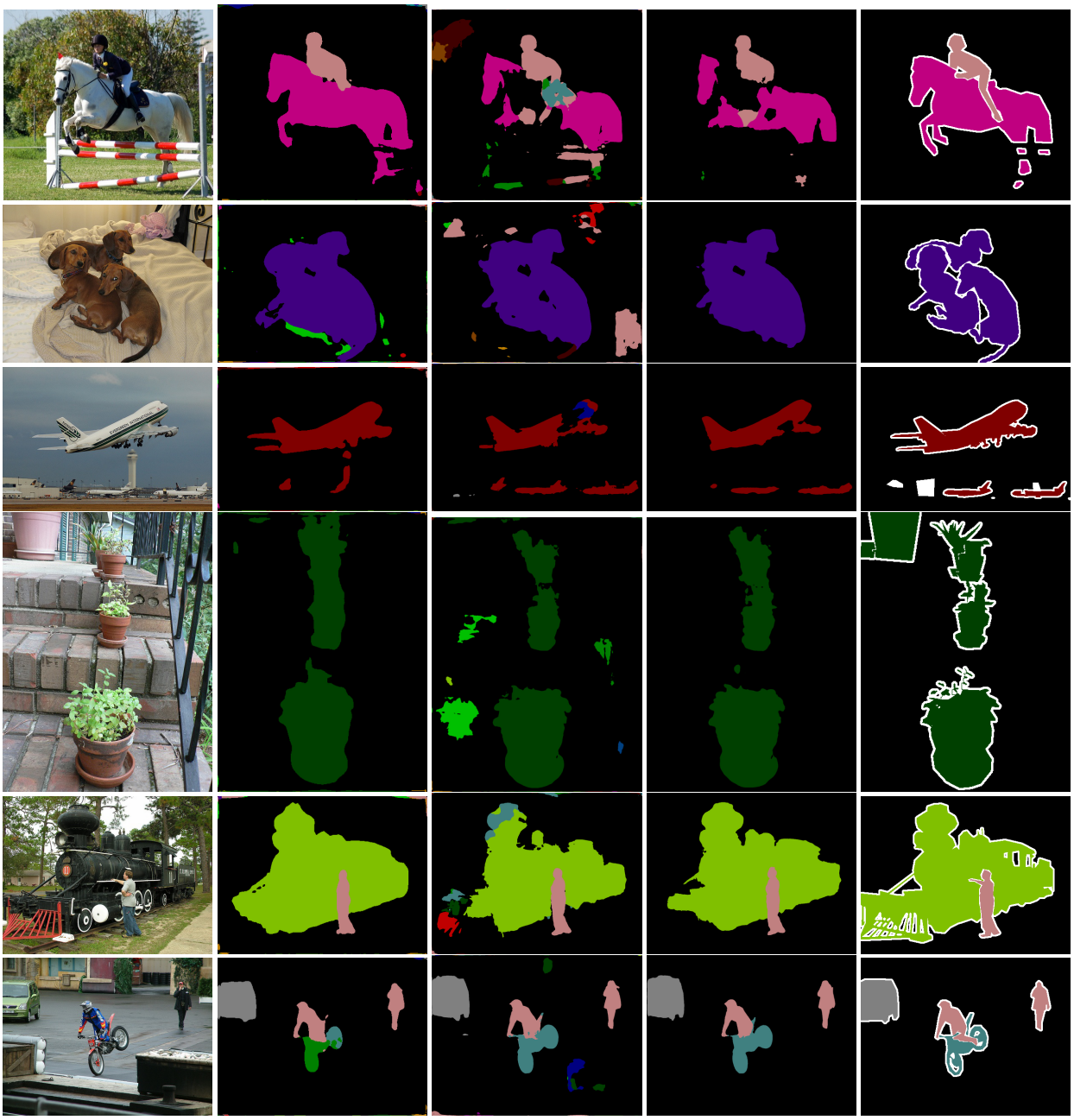
Iterative optimization gives a simple control over the amount of computation: the number of updates. This is a trade-off, because enough updates are needed for adaptation, but too many requires excessive computation. Table 4.2 shows that 32 steps are enough for improvement without too much computation. Therefore, we set the number of steps as 32 for all experiments in this chapter. For our network, one step of inference optimization takes $\sim\frac{1}{10}$ the time of a full forward pass.

Analysis

We analyze our approach from an adversarial perspective by maximizing the entropy instead of minimizing. To measure the importance of a parameter, we consider how much accuracy degrades when adversarially optimizing it. The more performance degrades, the more it matters. Table 4.3 shows that adversarial optimization of the structure parameters for scale degrades accuracy significantly, indicating the importance of dynamic scale inference. Jointly optimizing over the task parameters for classification further degrades accuracy.

While better compensating for scale shift is our main goal, our method also refines inference on in-distribution data. The results in Table 4.3 for 1× training and testing show improvement of ∼1 point.

We include qualitative segmentation results in Figure 4.5 and corresponding scale inferences in Figure 4.4.



(a) image (b) 1× feedforward (c) 3× feedforward (d) 3× opt. (ours) (e) truth

Figure 4.5: Qualitative results from the PASCAL VOC validation set [132]. Our model is trained on 1× scale and tested on 3× scale. (a) and (e) are the input image and ground truth. (b) indicates the reference in-distribution prediction on 1× scale. (c) is the out-of-distribution prediction for the feedforward dynamic baseline. (d) is the out-of-distribution prediction for our iterative optimization method. Our method corrects noisy, over-segmented fragments and false negatives in true segments.

4.4 Related Work

Dynamic Inference Dynamic inference adapts the model to each input [131]. Many approaches, designed [124], [125] and learned [67], [103], [126]–[128], rely on bottom-up prediction from the input. Our method extends bottom-up prediction with top-down optimization to iteratively update the model from the output. Recurrent approaches to iterative inference [139], [140] require changing the architecture and training more parameters. Our optimization updates parameters without architectural alteration.

Entropy Objective We minimize entropy during testing, not training, in effect tuning a different model to each input. The entropy objectives in existing work are optimized during training, especially for regularization. Entropy is maximized/minimized for domain adaptation [17], [63], [141], [142] and semi-supervised learning [24], [143]. In reinforcement learning, maximum entropy regularization improves policy optimization [144], [145]. We optimize entropy locally for each input during testing, while existing use cases optimize globally for a dataset during training.

Optimization for Inference We optimize an unsupervised objective on output statistics to update model parameters for each test input. Energy minimization models [146] and probabilistic graphical models [147], [148] learn model parameters during training then optimize over outputs during inference. The parameters of deep energy models [149], [150] and graphical models are fixed during testing, while our model is further optimized on the test distribution. Alternative schemes for learning during testing, like transduction and meta-learning, differ in their requirements. Transductive learning [57], [151] optimizes jointly over the training and testing sets, which can be impractical at deep learning scale. We optimize over each test input independently, hence scalably, without sustained need for the (potentially massive) training set. Meta-learning by gradients [152] updates model parameters during inference, but requires supervision during testing and more costly optimization during meta-training.

4.5 Conclusion

Dynamic inference by optimization iteratively adapts the model to each input. Our results show that optimization to minimize entropy with respect to score and scale parameters extends adaptivity for semantic segmentation beyond feedforward dynamic inference. Generalization improves when the training and testing scales differ substantially, and modest refinement is achieved even when the training and testing scales are the same. While we focus on entropy minimization and scale inference, more optimization for dynamic inference schemes are potentially possible through the choice of objective and variables.

Chapter 5

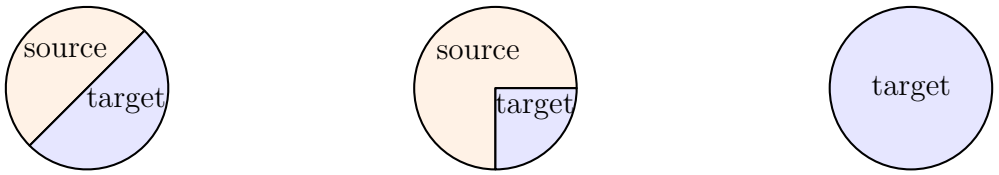
Target Data Is All You Need: On-Target Adaptation

5.1 Introduction

Deep networks achieve tremendous success on various visual tasks at the expense of massive data collection and annotation efforts. Even more data is needed when training (source) and testing (target) data differ, as the model must be adapted on the new data to maintain accuracy. To reduce the annotation effort on new data, unsupervised domain adaptation (UDA) approaches transfer knowledge from labeled source data to unlabeled target data. Standard UDA requires simultaneous optimization on the source and target data to do so. However, this requirement may not be entirely practical, in that *shifted* or *future* target data may not be available during training. Furthermore, (re-)processing source data during testing may be limited by computation, bandwidth, and privacy. Most importantly, it is the target data that ultimately matters for testing. In this chapter, we therefore turn our attention from source to target, and how to learn more from it.

Recent work adapts to the target data without the source data or even adapts during testing. However, these "source-free" and "test-time" approaches still rely heavily on the source parameters for fine-tuning. Source-free adaptation initializes from source parameters then optimizes on target data without the joint use of source data [50], [51], [59]. Test-time adaptation partially updates source parameters on the target data while testing [1], [18], [35]. Such approaches reduce reliance on the source data, and can even improve accuracy, but have they made full use of the target data? Many of the model parameters are fixed [1], [35], [50] or regularized toward the source parameters [51], [59]. We investigate whether more can be learned from target, and more accuracy gained, by not transferring the source parameters.

We propose on-target adaptation to unshackle the target representation from the source representation. To do so, we (1) factorize the representation from the classifier and (2) separate the source parameters from the source predictions. By factorizing the represen-



(a) UDA (b) Source-Free (c) On-Target (ours)

Figure 5.1: Domain adaptation adjusts a model trained on source data for testing on target data. We contrast methods by their updates on source and target. Unsupervised domain adaptation (UDA) jointly learns 50/50 on source/target. Source-free adaptation transfers source parameters, then selectively learns on target. Our *on-target* approach learns 100% of the testing model parameters on target by neither sharing nor transferring source parameters, but instead distilling source predictions.

tation from the classifier, we can train the representation entirely on the target data by self-supervision. Given this on-target representation, we can then supervise a new classifier from source predictions by distillation [153], without transferring the source parameters. Not transferring parameters frees our target model from the constraints of the source architecture, so that we can experiment with distinct target architectures. In this way, we can even change the model size to optimize a target-specific model that is more accurate and more efficient. In contrast to prior work on adaptation, this uniquely allows for learning 100% of the target model parameters on target data, as illustrated by Figure 5.1.

To realize our proposed factorization and separation, we employ contrastive learning, source-free adaptation, and teacher-student distillation. We initialize the target representation by self-supervision with contrastive learning. We turn the source model into a teacher model by source-free adaptation, and then generate pseudo-labels to supervise distillation. We lastly train the student model on the teacher supervision, starting from the target representation and new classifier parameters, and repeat this teacher-student cycle by resetting the student classifier parameters between epochs. Contrastive learning has recently enabled self-supervised representations to compete with or even surpass supervised representations [154]–[159]. We show it provides a sufficient target representation.

Our experiments show on-target adaptation achieves state-of-the-art accuracy and computational efficiency on common domain adaptation benchmarks. For model accuracy, our method brings ∼3% absolute improvement compared to state-of-the-art unsupervised and source-free domain adaptation methods on VisDA-C [49] and ImageNet Sketch [160] while reducing 50%+ of parameters. For computation, our method reduces FLOPs by 50+% and memory by 75+% for each forward pass of the target model. In the long-tailed classification setting, on-target class distribution learning equals the state-of-the-art learnable weight scaling [161] without needing source data. Ablation experiments support the generality of

on-target representation learning across architectures, contrastive learning methods, losses, and amount of optimization.

Our contribution is to investigate whether the source data should be the primary source of target model parameters, and to propose an alternative: on-target adaptation. Our insight is that the source representation can be fully decoupled from source supervision. Domain adaptation normally emphasizes the representation of source data, by either jointly optimizing on source data or transferring source parameters. On-target adaptation emphasizes the representation of target data instead, by distilling source predictions into a self-supervised target representation. We are the first to show this is feasible, as a new kind of source-free adaptation. Furthermore we show it improves accuracy and reduces computation on standard benchmarks like VisDA-C.

5.2 Related Work

Adaptation On-target adaptation is unique in its decoupling of the target representation from the source representation. Prior adaptation approaches transfer the source representation to the target, either by joint optimization or by initialization. To transfer the source model to a visually different target domain, unsupervised domain adaptation (UDA) learns a joint representation for both domains for visual recognition tasks, such as image classification [162], object detection [163], semantic segmentation [164]. Some of the most representative unsupervised domain adaptation ideas are 1) maximum mean discrepancy [165], [166]; 2) moment/correlation matching [167], [168]; 3) domain confusion [16], [43]; 4) GAN-based alignment [104], [169]. All these UDA methods need simultaneous access to both source and target data. In practice, it might be impossible to meet this requirement due to limited bandwidth, computational power, or privacy concerns. Therefore, test-time training [18], source-free adaptation [50], and fully test-time adaptation [1] settings focus on adapting a source model by fine-tuning on the target data without source data. Exciting concurrent work even adapts without the source model by only using source predictions [170]–[172]. These "black-box" adaptation methods exclusively optimize teacher and student predictions, with distillation losses and output regularizers. While we likewise apply teacher-student learning, our work is complementary in using contrastive learning as a loss on the input for the student representation.

Semi-supervised learning Many UDA methods follow the practice of semi-supervised learning, especially pseudo labeling [39] which is to utilize the model prediction to generate supervision for the unlabeled images. The typical setup of unsupervised domain adaptation methods is to jointly optimize with ground truths on the source and pseudo labels on the target [166], [173]–[175]. When source data annotations are not available, DeepCluster [176] and SHOT [50] further leverage weighted k-means clustering to reduce the side effects on noisy pseudo labels. Similarly, our method does not require access to labeled source data, while only relying on the target images with generated pseudo labels. In addition, our

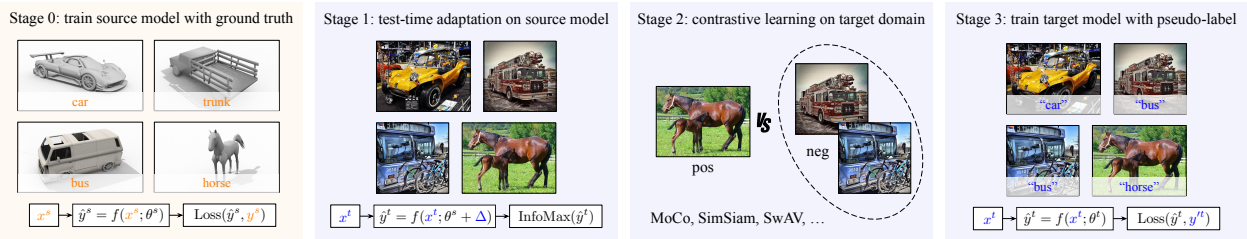


Figure 5.2: On-target adaptation proceeds in four stages. Source data is colored in orange while target data is colored in blue. Stage 0 is the only stage to use source data. Stage 3 is the stage that connects source and target: the target representation from stage 2 is fine-tuned as a student model on the predictions of the teacher model from stage 1. Note that no parameters are shared or transferred from source to target, so the target parameters are fully learned on target data.

method heavily benefits from the contrastive learned target domain representation, which is treated as initialization to overcome the misleading of noisy pseudo labels.

Long-tailed recognition Long-tailed recognition tackles imbalanced class distributions in real-world data. Existing work divides into three groups: 1) re-balancing the data distribution [177]–[180]; 2) designing class-balanced losses [181]–[189]; 3) transfer learning across classes [190], [191]. All of these methods address imbalance by altering training, so that the model may learn more balanced features, and a classifier that covers common (head) and rare (tail) classes. We instead adapt the classifier for long-tailed recognition during testing.

5.3 Method: On-Target Adaptation

The goal of the proposed on-target adaptation is to tackle domain shift during the test-time with only a source model, without the access of annotation and source data. Specifically, the supervised model with source parameter $f(\cdot;\theta^s)$ trained on source images x^s and labels y^s needs to generalize on unlabeled target data x^t when an unneglectable domain shift happened. Our on-target adaptation (Figure 5.2) is proposed to obtain target model parameter θ^t purely during test-time.

Stage 0 (source): train model with labeled source data We train a deep ConvNet and learn source parameter θ^s by minimizing vanilla cross-entropy loss $\mathcal{L}(\hat{y}^s, y^s)$ on labeled source data (x^s, y^s). Specifically, $\mathcal{L}(\hat{y}^s, y^s) = -\Sigma_c p(y^s_c) \log(p(\hat{y}^s_c))$ for the predicted probability $\hat{y}^s_c$ of class c, where target probability y^s_{gt} is 1 for the ground truth class gt and 0 for the rest.

Stage 1 (teacher): adapt source model without source data We update the source parameter θ^s during testing to minimize information maximization (InfoMax) loss [112]. Specifically, InfoMax loss augment entropy loss $\mathcal{L}_{ent} = -\Sigma_c p(\hat{y}_c^t) \log(p(\hat{y}_c^t)$ with diversity objective $\mathcal{L}_{div} = D_{KL}(\hat{y}^t \mid\mid \frac{1}{C}\mathbf{1}_C) - \log(C)$. where D_{KL} indicates the KullbackLeibler divergence, $\mathbf{1}_C$ is an all-one vector with C dimensions. Here $\frac{1}{C}\mathbf{1}_C$ indicates the target label vector with evenly distributed $\frac{1}{C}$ probabilities, where $\mathcal{L}_{div}$ is propose to enforce the global diversity over classes.

As for the parameters to optimize over, we follow the motivation of decoupling the representation and classifier. When the classifier is frozen, the goal of optimization is to mitigate domain shift by deriving proper target features from the source model. In particular, we keep the classifier the same on both source and target domain, and obtain Δ by the gradient of the test-time objective (InfoMax), to update the representation part of model parameter θ^s.

Stage 2 (student): initialize target model with contrastive learning Instead of fine-tuning from source model, we choose to initialize the target feature purely from target data. Benefiting from the recent advances in contrastive learning methods, we train an unsupervised model with purely unlabeled target images. Specifically, we initialized target representation via improved momentum contrast learning (MoCo v2) [156], [157]. It is worth noting that our method does not require a specific contrastive learning method. In other words, the default MoCo v2 could be easily replaced by a more recent self-supervised learning model, such as SwAV [155], SimSiam [154], Barlow Twins [159]. Such a modular design makes it easier to benefit from the latest advance in contrastive learning. We have performed an ablation study on the choice of contrastive learning method in Section 5.4.

Stage 3 (teacher-student): transfer knowledge from teacher to student We use the adapted source model $f(\cdot; \theta^s + \Delta)$ as the initial teacher model to generate pseudo labels y'^t on unannotated target images x^t. Then we fine-tune the student model $f(\cdot; \theta^t)$ initialized by contrastive learning on target data with cross-entropy loss $\mathcal{L}(\hat{y}^t, y'^t) = -\Sigma_c p(y_c'^t) \log(p(\hat{y}_c^t))$. Specifically, we use normal distribution with a mean of zero and standard deviation of 0.01 for the classification head, since contrastive learned model does not contain classifier. The teacher would be replaced with the latest student to gradually denoise pseudo labels for the subsequent phase. Meanwhile, the contrastive learned model would re-initialize the student feature to eliminate the accumulated errors from imperfect pseudo labels. In other words, the student model would start over one more time with only newer pseudo labels for the next transferring phase.

Figure 5.3 illustrates the procedure of transferring the knowledge from teacher to student. Specifically, the interaction between teacher and student models benefits from consistency regularization and pseudo-labeling, inspired by a recent semi-supervised learning approach called FixMatch [192]. During the transferring, we augment the target images with random cropping, random flipping, and AutoAugment with ImageNet policy, as "strong" augmen-

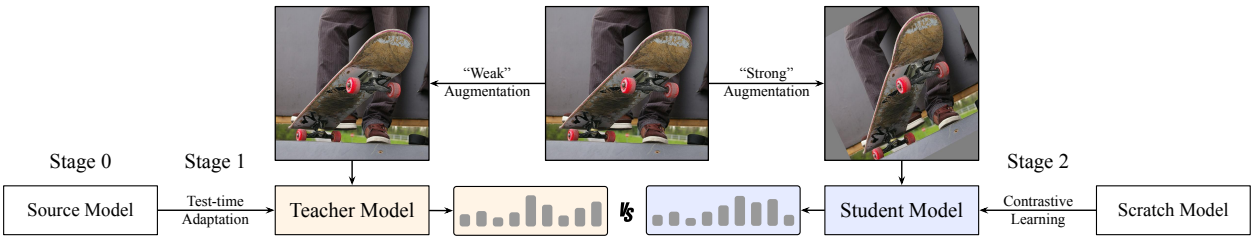


Figure 5.3: Teacher-student (stage 3) learning in our method. Transfer learning between the teacher (orange) and the student (blue), where pseudo labels are generated on the weakly-augmented images. The model is trained on the strongly-augmented target data to match the pseudo labels.

tation, while the "weak" augmentation is the combination of resizing and center cropping when generating pseudo labels. Relying on the assumption that the model should generate similar predictions on data-augmented versions of the same image [193]–[195], consistency regularization enforces the cross-entropy loss between student output on strongly-augmented images and teacher output on weakly-augmented images.

5.4 Experiments

Setup

Datasets We evaluate our method on both domain adaptation and long-tailed recognition benchmarks, including VisDA-C [49], Office Home [196], Sketch [160], ImageNet-LT [191], and iNaturalist-18 [197]. Figure 5.5 presents some of example images to illustrate domain shifts.

Metric We report top-1 accuracy (denoted as acc.) on the whole dataset for all datasets. On VisDA-C, we additionally report the percentage accuracy of each category and the corresponding average of all categorical accuracies (denoted as avg.), due to the imbalanced distributed label space. On Office Home, we calculate the average for all kinds of domain shifts as the summary number for each method. On long-tailed recognition benchmarks, we additionally report percentage accuracy on many-shot (more than 100 samples), medium-shot (20-100 samples), and few-shot (less than 20 samples) following the evaluation protocol from [161], [191].

Baselines We choose the most recent fully test-time adaptation method, TENT [1], and source-free adaptation framework, SHOT [50], as our "online" and "offline" adaptation baselines. TENT does not alter training, while SHOT minimally customizes the source architecture and training. Entropy minimization is the target optimization objective for both TENT

and SHOT. SHOT additionally regularizes optimization by the information maximization (InfoMax) loss [112], [198], [199] to augment entropy minimization on each sample with diversity maximization across samples. Following SHOT, we therefore augment TENT into TENT-IM by including this regularization. Alongside their role as baselines, these methods can serve as teacher models for our stage 1. We also compare with unsupervised domain adaptation (UDA) baselines, including DANN [16], DAN [165], ADR [200], CDAN+E [201], CDAN+BSP [202], CDAN+TN [203], SAFN [204], SWD [205], DSBN+MSTN [206], STAR [207]. It is worth noting that all these UDA methods are fine-tuning from the ImageNet pretrained ResNet-101 source model, with access to both source and target data. TENT-IM and SHOT likewise initialize their representation from the source model. In contrast, our method trains ResNet-18 from scratch, and entirely on the target data, by contrastive learning for the representation and teacher-student learning for the classification.

For long-tailed recognition, we choose learnable weight scaling (LWS) [161] as the train-time baseline. LWS first decouples the full model into representation and classification, and then only re-scales the classifier parameters with class-balanced sampling. Our on-target class distribution learning extends LWS to test-time, by re-scaling the parameters on unlabeled target data.

Architecture For comparability with state-of-the-art models, we choose 18/50/101-layer ResNet models [8] for both main results and ablation studies. When reproducing the prior works, we keep the architecture the same, for example, the weight-normalization [208] augmented ImageNet pretrained ResNet-101 for SHOT [50].

Implementation

Our implementation is in PyTorch [27] and depends on the VISSL [209], MMClassification [210], and Weights & Biases [211] libraries.

Stage 0 (source) We train residual networks [8] with various depths (including 18, 50, 101), and initializations (ImageNet pretraining or Kaiming init [212] when training from scratch). We optimize cross-entropy loss by SGD with an initial learning rate 0.1, momentum 0.9, weight decay 0.0001, batch size 256. We do not apply label smoothing [213] except for SHOT [50], as it specifically includes it. We adopt the standard data augmentation pipeline from ImageNet training, such as random cropping, random flipping, and color jitter. We choose ImageNet statistics as the default input mean and variance for all models.

Stage 1 (teacher) We experiment with three types of teacher models: 1) source-only, 2) TENT-IM [1], 3) SHOT [50]. To optimize this altered loss, we choose SGD with learning rate 0.0001, momentum 0.9, and weight decay 0.0001. In addition to batch normalization [34], we also update convolutional layers except the final classification layer. As for SHOT, We

Table 5.1: Each stage of our on-target adaptation improves target accuracy. Contrastive learning (stage 2), for fitting the representation on target data alone, helps whether or not the teacher is adapted (stage 1).

method	#1 test-time adaptation	#2 contrastive learning	#3 teacher student	VisDA-C train →val	VisDA-C train →test	Imagenet →Sketch	Office Home
source-only				21.8	23.9	27.6	51.7
test-time adaptation	✓			31.4	34.3	35.6	53.8
on-target adaptation			✓	28.3	31.8	27.9	54.3
without contrastive	✓		✓	43.3	46.0	40.5	56.8
on-target adaptation		✓	✓	29.1	33.9	✗	✗
	✓	✓	✓	49.9	51.2	✗	✗

execute the authors' open-sourced codebase with the same hyperparameters for various architectures (ResNet-50 and ResNet-101), initialization (from scratch and ImageNet pretrain), and domain shifts (train to val/test splits on VisDA-C).

Stage 2 (student) We experiment with two designs as students: 1) source-only, 2) contrastive learning. Specifically, we leverage some of off-the-shelf contrastive learning methods to initialize target-domain representation, such as MoCo v2 [157], SimSiam [154], SwAV [155], Barlow Twins [159]. Compared to their training recipes on ImageNet, we have more epochs on VisDA-C val/test with the same batch size, learning rate, data augmentation, and model architecture, to make the training procedure longer with the smaller amount of images.

Stage 3 (teacher-student) By default, the whole knowledge distillation consists of three phases, where each phase has 10 epochs to train the student model with the hard pseudo label. The student would be reset to the contrastive model to avoid error accumulation at the beginning of every phase. The teacher would be replaced with the latest student before starting the next phase, so that the quality of pseudo-labeling could be improved gradually. We utilize SGD with an initial learning rate 0.01, momentum 0.9, weight decay 0.0001, batch size 256, and cosine annealing scheduler [214].

On-target adaptation

Table 5.1 reports reports the change in target accuracy for each stage of on-target adaptation on VisDA-C. The source model is ResNet-50 trained from scratch. The target model is ResNet-18. Source-free adaptation is done by TENT-IM. In teacher-student learning, the student, with either the source (teacher) or our on-target (contrastive) representation, is

Require: model f_s # source model, for predictions
Require: model f_t # target model, for representation
Require: image transform t_w # weak augmentation
Require: image transform t_s # strong augmentation
  $f_0 \leftarrow f_s$ # source model for the first teacher
  for i $\leftarrow$ 1 to N **do** # for each epoch
    $f_i \leftarrow f_t$ # initialize by contrastive learning
    for b $\leftarrow$ 1 to B **do** # for each mini-batch
      $y' \leftarrow f_{i-1}(t_w(x))$ # teacher
      $\hat{y} \leftarrow f_i(t_s(x))$ # student
      $\ell \leftarrow \text{Loss}(\hat{y}, y')$ # hard-label cross-entropy
    end for
  end for
  return f_N # student model for testing

Figure 5.4: On-target pseudo-code.

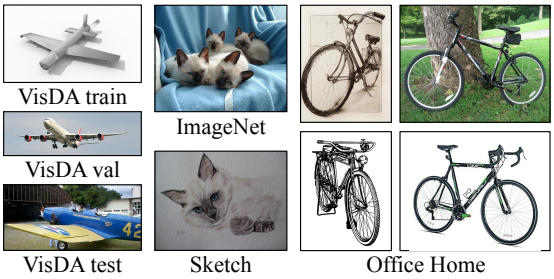


Figure 5.5: Example images from VisDA-C, ImageNet, ImageNet-Sketch, and Office-Home.

Table 5.2: Classification accuracy of on-target adaptation on VisDA-C (validation) across all categories and averaged over classes (avg.) and images (acc.). R18/50/101S denotes ResNet-18/50/101 randomly initialized from scratch and R18/50/101P denotes ResNet-18/50/101 pretrained on ImageNet.

method	network	plane	bcycl	bus	car	horse	knife	mcycl	person	plant	sktbrd	train	trunk	avg.	acc.
ADR [200]	R101P	94.2	48.5	84.0	72.9	90.1	74.9	92.6	72.5	80.8	61.8	82.2	28.8	73.6	73.8
CDAN+E [201]	R101P	85.2	66.9	83.0	50.8	84.2	74.9	88.1	74.5	83.4	76.0	81.9	38.0	73.9	71.0
CDAN+BSP [202]	R101P	92.4	61.0	81.0	57.5	89.0	80.6	90.1	77.0	84.2	77.9	82.1	38.4	75.9	73.4
SAFN [204]	R101P	93.6	61.3	84.1	70.6	94.1	79.0	91.8	79.6	89.9	55.6	89.0	24.4	76.1	75.6
SWD [205]	R101P	90.8	82.5	81.7	70.5	91.7	69.5	86.3	77.5	87.4	63.6	85.6	29.2	76.4	75.6
DSBN+MSTN [206]	R101P	94.7	86.7	76.0	72.0	95.2	75.1	87.9	81.3	91.1	68.9	88.3	45.5	80.2	79.2
STAR [207]	R101P	95.0	84.0	84.6	73.0	91.6	91.8	85.9	78.4	94.4	84.7	87.0	42.2	82.7	80.4
TENT-IM [1]	R50S	58.9	40.1	50.2	23.6	22.6	25.3	29.8	24.8	22.9	30.2	45.1	20.1	32.8	31.4
TENT-IM + Ours	R50S	90.1	66.0	75.2	41.3	29.2	11.2	57.0	60.8	40.1	51.1	73.6	23.8	51.6	50.9
TENT-IM + Ours	R18S	90.5	65.2	79.6	38.8	26.7	12.9	51.6	59.9	44.2	46.0	71.1	24.0	50.9	49.9
SHOT [50]	R101P	94.6	86.6	79.5	55.6	93.6	96.1	79.8	80.7	89.2	89.0	86.1	57.1	82.3	77.8
SHOT + Ours	R18S	96.0	89.5	84.3	67.2	95.9	94.2	91.0	81.5	93.8	89.9	89.1	58.2	85.9	82.8

trained on the predictions of the teacher. This is repeated for multiple epochs. The on-target representation (stage #2) improves accuracy with and without source-free adaptation of the teacher, as it only depends on the target data.

VisDA train $\rightarrow$ val Table 5.2 compares our method with state-of-the-art unsupervised (upper part) and test-time (lower part) domain adaptation approaches from VisDA-C train to val splits. The proposed on-target adaptation significantly improves the existing test-time adaptation methods. It is worth noting that all these existing methods need to keep the same architecture when joint-training or fine-tuning on the source model. On the contrary, our method could utilize a much more lightweight model, such as ResNet-18 as shown in this table. For example, our method brings 18+ points improvement compared to source-free

Table 5.3: Classification accuracy of our method supervised by three teachers: source-only, SHOT, and TENT-IM on VisDA-C (test). R50S/R101S denotes ResNet-50/101 randomly initialized from scratch and R50P/R101P denotes ResNet-50/101 pretrained on ImageNet.

	source-only		ours			TENT-IM		ours			SHOT		ours	
network	avg.	acc.	avg.	acc.	network	avg.	acc.	avg.	acc.	network	avg.	acc.	avg.	acc.
R50S	22.1	23.9	30.9	33.9	R50S	34.2	34.3	49.3	51.2	R101P	89.3	88.4	91.7	91.6
R50P	34.4	37.7	39.8	43.5	R50P	60.4	62.4	74.8	77.7	R50P	76.4	78.0	81.1	83.0

Table 5.4: Adaptation on ImageNet-Sketch.

method	network	accuracy
Anisotropic [215]	ResNet-50	24.5
Debiased [216]	ResNet-50	28.4
Crop [217]	ResNet-50	30.9
RVT [218]	DeiT-B	36.0
TENT-IM [1]	ResNet-50	35.6
TENT-IM + Ours	ResNet-18	37.5
TENT-IM + Ours	ResNet-50	40.5

Table 5.5: Adaptation on Office-Home.

method	network	accuracy
DANN [16]	ResNet-50	57.6
DAN [165]	ResNet-50	56.3
CDAN+E [201]	ResNet-50	65.8
CDAN+BSP [202]	ResNet-50	66.3
SAFN [204]	ResNet-50	67.3
CDAN+TN [203]	ResNet-50	67.6
TENT-IM [1]	ResNet-50	53.8
TENT-IM + Ours	ResNet-18	56.8

adapted teacher TENT-IM, while reducing over 50% parameters and runtime flops, and 75% memory consumption at each feed-forward of target model.

VisDA train → test Table 5.3 compares our method with state-of-the-art unsupervised (upper part) and test-time (lower part) domain adaptation approaches from VisDA-C train to test splits. Similar to Table 5.2, our method dramatically improves the performance of all kinds of teacher models.

In the following two paragraphs, we discuss the potential application of on-target adaptation *without contrastive learning*. When test-time data is not sufficient enough to finish the contrastive learning, we could skip contrastive learning on target data (stage 2) of the proposed method. In other words, we directly fine-tune the target model initialized by the source model. We believe that adaptation performance could be further improved once the contrastive learning could no longer be data-hungry or target domain data could be abundant.

ImageNet → Sketch Table 5.4 reports the empirical results on generalization regarding ImageNet/Sketch as source/target domain. For on-target adaptation, we try two student models: model as same as the teacher model (ResNet-50) and small supervised model pretrained on ImageNet (ResNet-18). Our method additionally brings ∼5/3% improvements

compared to the teacher model with the same/shallower student models, where the teacher model, TENT-IM, already outperforms the previous state-of-the-art by ∼5%.

Office Home Table 5.5 compares our method with state-of-the-art unsupervised (upper part) and test-time (lower part) domain adaptation approaches for the various domain shifts in Office Home. Our method advances the average accuracy over all domain shifts of teacher model (TENT-IM) by 3%.

On-target class distribution learning

In this section, we argue for calibrating classifier during the test-time, without any modification on training procedure, aiming at long-tailed recognition task. Here we treat the long-tailed data as the source domain while regarding the class-balanced data as the target domain. During training, instance-balanced sampling provides a generalizable representation to start with. Then the classifier is re-scaled during the test-time on the class-balanced data.

First, we train the source domain model with the instance-balanced sampling, which samples each sample with the same probability. In this way, the learned classifier has a higher prior probability on the head compared to the tail. Then we calibrate the parameters of the classifier while freezing the feature with test images and pseudo labels, following the practice of our on-target adaptation. It is worth mentioning that we do not utilize contrastive learning to re-initialize the representation on target data or tune the feature part in teacher-student (stage 3). The major reason for such a choice is to follow the practice of LWS [161], which points out that the domain shift only exists within class distribution so that only classifier needs to be calibrated.

The empirical results indicate that our method could automatically calibrate the categorical prior and adaptive fit the test data distribution without the access of training data. Table 5.6 demonstrate that our test-time on-target adaptation could achieve comparable performance compared to state-of-the-art training-time methods on ImageNet-LT and iNaturalist-18 datasets. We report results for two popular ConvNets, ResNet-50 and ResNet-101, as teacher models trained on long-tailed data. Our method achieves comparable overall performance with the train-time method (LWS) on both datasets. Compared to the vanilla ResNet-50 and ResNet-101, our fully test-time method significantly improves the overall performance by a large margin. Considering the accuracy of few-shot (less than 20 samples) categories, our method outperforms the train-time practice on three out of four cases, extending the usage scenarios of train-time long-tailed recognition methods.

Ablation Studies

Stage 0: network & initialization The upper part of Table 5.7 presents the numbers of SHOT/TENT-IM with ResNet in various depths (18, 50, 101) and initializations (from scratch, ImageNet pretrain). We observe that our method consistently improves the teacher

Table 5.6: Comparing our method performance with learnable weight scaling (LWS) on long-tailed benchmarks including iNaturalist18 and ImageNet-LT. Note that LWS adapts during training while our method can adapt during testing, which is more efficient.

	iNaturalist18				ImageNet-LT			
method	many	medium	few	acc.	many	medium	few	acc.
ResNet-50	72.2	63.0	57.2	61.7	64.0	33.8	5.8	41.6
+ LWS [161]	65.0	66.3	65.5	65.9	57.1	45.2	29.3	47.7
+ On-Target Class Distribution	64.2	66.3	65.9	65.9	55.7	46.0	28.6	47.4
ResNet-101	75.9	66.0	59.9	64.6	66.6	36.8	7.1	44.2
+ LWS [161]	69.6	69.1	67.9	68.7	60.1	47.6	31.2	50.2
+ On-Target Class Distribution	66.5	69.1	68.3	68.5	58.9	48.7	31.8	50.3

Table 5.7: Classification accuracy on VisDA-C (validation). "Imagenet pretrain" indicates whether we utilize ResNet pretrained on ImageNet at stage 0. "Source only" indicates whether we skip test-time adaptation (stage 1) and directly use the source model to generate pseudo labels at stage 3.

	imagenet	source	SHOT		ours			imagenet	source	TENT		ours	
network	pretrain	only	avg.	acc.	avg.	acc.	network	pretrain	only	avg.	acc.	avg.	acc.
ResNet-50	✗	✗	49.7	48.3	69.1	67.3	ResNet-50	✗	✗	32.8	31.4	50.9	49.9
ResNet-50	✓	✗	75.0	74.5	77.8	78.6	ResNet-50	✓	✗	60.9	60.2	75.1	73.8
ResNet-101	✓	✗	82.3	77.8	85.9	82.8	ResNet-18	✗	✗	34.0	33.1	51.4	51.4
ResNet-101	✓	✓	49.9	55.5	60.0	65.6	ResNet-50	✗	✓	17.8	21.8	22.3	29.1

models with various model architectures and initializations, which indicates the usability and versatility of the proposed framework. When adapting from synthetic to real-world domains, the ImageNet pretrained model should not be utilized to start with, due to the learned inductive bias of its parameters. Therefore we also experiment on training from scratch for teacher models. Larger capacity does not lead to better generalization when training from scratch. On the contrary, We observe that a deeper ImageNet pretrained ConvNet provides a stronger inductive bias from the beginning.

Stage 1: test-time adaptation The lower part of Table 5.7 presents the numbers of source-only models as teacher, without any source-free adaptation (TENT-IM/SHOT). The final accuracy after teacher-student suffers from the poorer quality of the initial pseudo label. ImageNet pretraining could alleviate such a phenomenon, but the numbers with test-time adapted teachers are still significantly better than the source-only ones. Comparing these empirical results with Table 5.2, TENT-IM boosts the initial/final accuracy by 15.0/28.6

Table 5.8: Classification accuracy on VisDA-C (validation). Left side: Ablation results on the student model with various initialization. Right side: Ablation results on the contrastive learning method using MoCo, SwAV, SimSiam, and Barlow Twins.

method	avg.	acc.	method	avg.	acc.
TENT	32.8	31.4	Ours (MoCo)	50.9	49.9
VisDA-C train	45.3	43.3	SwAV	50.5	49.4
ImageNet	47.3	46.2	SimSiam	48.7	47.6
ImageNet (MoCo)	46.6	45.5	Barlow Twins	46.3	44.8

points, while SHOT brings 32.4/25.9 points improvement. In a word, test-time adaptation should be leveraged in preparation for trustworthy pseudo labels.

Stage 2: on-target feature The left part of Table 5.8 presents the ablation study of student architecture and initialization. When the student model is not initialized on target data, such as source data (VisDA-C train) or the external large dataset (ImageNet), the overall accuracy drops 4+ points, which indicates the necessity of on-target feature learning. We also ablate the same contrastive learning algorithm (MoCo v2) on a different data source, such as VisDA-C val and ImageNet. The empirical results indicate that more external data does not bring any advantage, which echos our statement on the target-specific representation learning.

Stage 2: contrastive learning The right part of Table 5.8 presents the results with various contrastive learning frameworks, including SwAV, SimSiam, Barlow Twins. We train these contrastive learned models with the same number of epochs to have a fair comparison. Compared to the performance of the teacher model, all these learned features bring a noticeable improvement, achieving the comparable numbers with the reference performance of MoCo v2. We observe that our method is not sensitive to the choice of contrastive learning method. In this way, our on-target adaptation could be further improved by introducing a more advanced contrastive learning approach in the future.

Stage 3: more phases Table 5.9 presents the detailed numbers for each phase during teacher-student. We observe that the first phase already significantly outperforms the test-time adapted teacher model, which is also the state-of-the-art practice. The following several phases gradually improve the results, taking the last generation student as the next teacher. Considering the speed-accuracy trade-off, we choose to have three phases as our default setup, even though more phases could lead to a better result. For example, 9-phase ($3\times$) optimization brings up around 9 points improvement compared to our default 3-phase ($1\times$) one with TENT-IM as the initial teacher model.

Table 5.9: Classification accuracy of our method on VisDA-C (validation). "Soft" indicates that the hard-label cross-entropy loss is replaced with the soft-label KL divergence loss for each even-numbered phase. Note that our default number of phases (phase 3) is highlighted.

teacher	soft	phase 0	phase 1	phase 2	phase 3	phase 4	phase 5	phase 6	phase 7	phase 8	phase 9
TENT	✗	32.8	44.2	48.2	50.9	52.7	54.3	56.0	57.0	58.1	58.9
	✓	32.8	44.3	53.5	56.4	59.9	61.4	63.6	64.2	65.1	65.6
SHOT	✗	82.3	84.8	85.5	85.9	86.2	86.3	86.3	86.3	86.2	86.3
	✓	82.3	84.7	85.2	85.6	85.2	85.7	85.0	85.4	84.9	85.2

Stage 3: soft label Table 5.9 presents the ablation study on the design choice of loss function. Our default setup only utilizes hard labels with cross-entropy loss. Actually, our framework also benefits from the soft label with KullbackLeibler divergence loss, following the popular practice of knowledge distillation [153]. We observe that the mix of both hard and soft label bring up the best performance. we replace the cross-entropy loss (hard label) with KullbackLeibler divergence (soft label) for the *even* number of phases. We set the number of epochs as one for all these interpolated soft label phases for a more computational-friendly practice. The known drawback is that the soft label part typically needs specific tuning on learning rate, loss weight, temperature, and so on. Existing works [65], [192], [219] discuss sharpening (temperature) and thresholding (confidence threshold) to improve the performance of semi-supervised learning. Instead, we only ablate the default Kullback-Leibler divergence loss without bells and whistles like temperature and confidence threshold. Our default training objective chooses to be the most robust hard label with cross-entropy criterion for all the other experiments.

5.5 Discussion

Domain adaptation is itself adapted to many different needs: unsupervised domain adaptation jointly optimizes over labeled source and unlabeled target data, source-free adaptation adapts to target given source parameters instead of source data, and test-time adaptation even adapts while making predictions. Across each of these varieties, the source comes first. The target representation is either aligned to the source representation or it is initialized from it by transfer learning. On-target adaptation departs from this standard practice by transferring the source predictions without the source representation. This decoupling is unconventional, but useful, because it enables learning all of the target model parameters on the target data. Given enough target data, on-target adaptation improves accuracy by learning the model for target data on target data.

Chapter 6

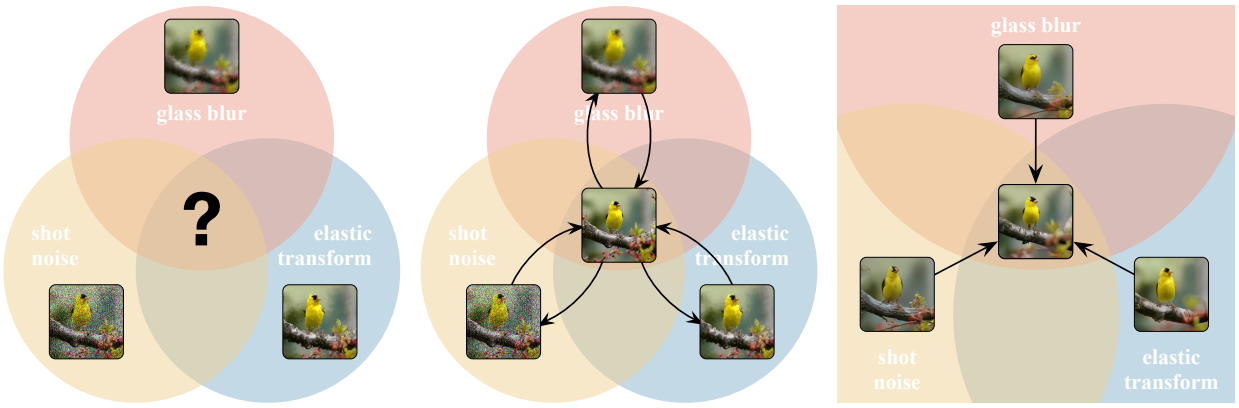
Back to the Source: Diffusion-Driven Test-Time Adaptation

6.1 Introduction

Deep networks achieve state-of-the-art performance for visual recognition [8], [220]–[222], but can still falter when there is a *shift* between the source data for training a recognition model and the target data for testing [12]. Shift can result from corruption [9], [223]; adversarial attack [74]; or natural shifts between simulation and reality, different locations and times, and other such differences [49], [224]. To cope with shift, adaptation and robustness techniques update inference to improve accuracy on target data. In this chapter, we examine two key axes of adaptation, when to adapt—during training or testing—and what to adapt—the model or the input, and propose a uniquely test-time input adaptation method driven by a generative diffusion model.

The dominant paradigm for adaptation is to update the model during training by joint optimization over source and target [15], [43], [104], [225], [226]. However train-time adaptation faces a fundamental issue: not knowing how the data may differ during testing. While train-time updates can cope with known target domains, what if new and different shifts should arise during deployment? In this case, test-time updates are needed to adapt the model (1) without the source data and (2) without halting inference. Source-free adaptation [18], [50], [51], [59], [227], [228] satisfies (1) by re-training the model on new targets without access to the source. Test-time adaptation [1], [18], [35], [229] satisfies (1) and (2) by iteratively updating the model while making predictions. Updating the model during testing makes inference more robust to shift, but with some additional disadvantages. Model updates could be too computationally intense to scale to many targets, which each need their own model, and the updates may be sensitive to different amounts or orders of data from the target(s), in which case it may fail to help or even harm robustness. In summary, existing methods concentrate on updating the source *model*.

We propose to update the target *data* during testing instead. To update the data, we



(a) Setting: Multi-Target Adaptation (b) Cycle-Consistent Paired Translation (c) DDA (ours): Many-to-One Diffusion

Figure 6.1: One diffusion model can adapt inputs from new and multiple targets during testing. Our adaptation method, DDA, projects inputs from all target domains to the source domain by a generative diffusion model. Having trained on the source data alone, our source diffusion model for generation and source classification model for recognition do not need any updating, and therefore scale to multiple target domains without potentially expensive and sensitive re-training optimization.

project test inputs back to the source domain by generative diffusion modeling. Our diffusion-driven adaptation method, DDA, learns a diffusion model of the source data during training, then projects inputs from all targets during testing. Figure 6.1 shows how just one generative model of source enables adaptation from multiple targets. DDA trains a diffusion model to replace the source data, for source-free adaptation, and adapts target inputs while making predictions, for test-time adaptation. Figure 6.2 illustrates how DDA adapts the input then applies the source classifier without model adaptation.

Our experiments measure robustness to input corruption to compare and contrast input updates and model updates. For input updates, we evaluate and ablate our proposed DDA method. For model updates, we evaluate entropy minimization methods (Tent [1] and MEMO [229]), which are the state-of-the-art for fully test-time online updates, and BUFR [230], which is the state-of-the-art for source-free offline updates. DDA achieves higher robustness than MEMO in all cases, and helps where Tent degrades due to limited, ordered, or mixed data. As a model-agnostic input adaptation method, DDA improves across standard (ResNet-50) and state-of-the-art convolutional (ConvNeXt [222]) and attentional (Swin Transformer [221]) architectures without re-tuning.

Our contributions:

- We propose the first diffusion modeling approach for test-time adaptation to corruption, DDA, and propose a novel self-ensembling scheme to select how much to adapt.

- We identify and empirically confirm weak points of test-time model updates—small batches, ordered data, and mixed domains—and highlight our test-time input updates to address these natural but currently challenging regimes.
- We experiment on the ImageNet-C benchmark to show that our DDA improves over test-time model adaptation across corruptions, architectures, and data regimes.

6.2 Related Work

We relate our uniquely generative test-time input adaptation method to existing methods for model and input adaptation. We also highlight diffusion modeling as the key approach for our generative modeling of the source data.

Model Adaptation Model adaptation aims to update the source model on target data to improve accuracy. While myriad varieties exist, we focus on source-free adaptation—not needing the source data while adapting—and on test-time adaptation—making predictions while adapting—because DDA is a source-free and test-time method. *Source-free* adaptation [18], [50], [51], [59] makes it possible to respect practical deployment constraints on computation, bandwidth, and privacy. Nevertheless, most methods involve a certain amount of complexity and computation by altering training [18], [50], [51], [59] and interrupting testing by re-training their model(s) offline on each target [50], [51], [59], [230]. DDA is source-free, as it replaces the source data with source diffusion modeling. However, it differs by adapting the data and not its diffusion model or recognition model. Furthermore, it does not alter training of the recognition model, as it can train the diffusion model in isolation. By keeping its models fixed, DDA handles multiple targets without halting testing for model re-training, as source-free model adaptation must. *Test-time adaptation* [1], [18], [35], [231] updates the model without holding up inference. (Fully test-time methods are more extreme still, and do so without any access to training or source data.) Such test-time model updates can be sensitive to their optimization hyperparameters along with the size, order, and diversity of the test data. On the contrary, DDA updates the data, which makes it independent across inputs, and thereby invariant to the batching, order, or mixture of the test data. DDA can even adapt to a single test input without augmentation, unlike test-time model adaptation.

Input Adaptation Input adaptation aims to translate data between source and target. DDA carries out test-time input adaptation from target to source by diffusion. Prior methods adapt during testing, but differ in their purpose and technique, or adapt during training, but cannot handle new target domains during testing. When testing, translation goes from target to source, allowing direct application of the source model without having to (re-)train it for target, as done by diffusion-driven defense [232] for robustness to attack. When training, translation goes from source to target—real or synthesized—to provide additional data or auxiliary losses, as done by style transfer [233]–[236], conditional image synthesis [104],

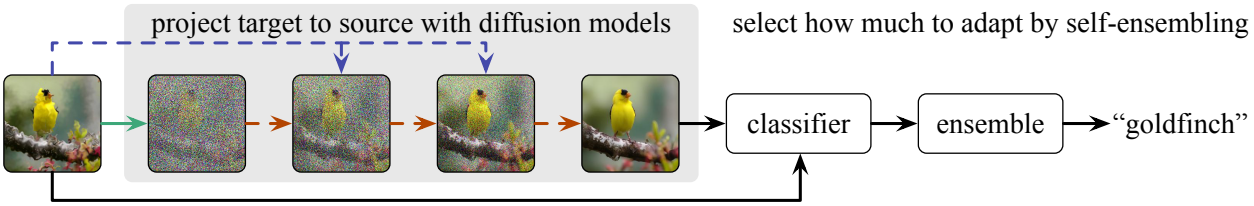


Figure 6.2: DDA projects target inputs back to the source domain. Adapting the input during testing enables direct use of the source classifier without model adaptation. The projection adds noise (forward diffusion, green arrow) then iteratively updates the input (reverse diffusion, red arrow) with conditioning on the original input (guidance, purple arrow). For reliability, we ensemble predictions with and without adaptation depending on their confidence.

[237]–[242], or adversarial generation [40] for robustness to shift. A key work in this line is CyCADA [104], which translates from source to target with generative modeling by CycleGAN [237]. While CyCADA is generative like DDA, CycleGAN requires paired source and target data for training, and so it cannot adapt to multiple and varied targets during testing. In contrast, DDA only requires source data during training, and can adapt to multiple target domains with a single model.

Diffusion Modeling Diffusion [243]–[248] is an emerging approach to generative modeling that operates on the input space by iteratively refining samples. In essence, diffusion models learn to “reverse” noise to generate an image by gradient updates w.r.t. the input. The type of noise matters, and standard diffusion models rely on Gaussian noise. In this chapter, we investigate how a strong diffusion model can project corrupted target data to the source data distribution, which involves corruptions that are highly non-Gaussian. We apply the denoising diffusion probabilistic model (DDPM) [249] in this new role of diffusion-driven adaptation. Guided diffusion models improve generation by optimization based on class labels [250], [251], text [252], [253] and images [254], but test-time adaptation denies the necessary data for their use. While diffusion has been applied to adversarial defense [232], we are the first to adopt it for test-time adaptation to corruptions. Diffusion is key to our source-free and test-time adaptation method, in replacing the source data with source diffusion, and in updating target data without needing to update the model across batches or target domains. We apply diffusion in this fashion to propose a source-free adaptation method that uniquely scales to multiple target domains without the addition or optimization of parameters.

6.3 Method: Test-Time Diffusion-Driven Adaptation

We propose diffusion-driven adaptation (DDA) for test-time input adaptation by generative modeling with diffusion. During training, we train a generative diffusion model on the source domain data, and train a discriminative classification model on the source domain data and annotations. During testing, given a test input from the target domain, the diffusion model projects it to the source domain, and then the classification model makes its prediction from the original and translated input. Figure 6.2 illustrates inference with DDA through projection, classification, and ensembling of the predictions over the target and source-projected inputs.

DDA does not need any target data during training, and in principle it can handle an arbitrary number of target domains during testing. Of course, its ability to effectively project a given input to the source domain may vary for each target. DDA can adapt a single test input at a time, making it an episodic method, which does not require batching or cumulative updates. In contrast, existing test-time model adaptation methods, such as Tent [1] and BUFR [230], degrade on on too little data (small batches), on dependent data (in non-random order), or on mixed data across different target domains (with multiple corruptions) See Sec. 6.4 for our comparison of input and model adaptation. By not relying on the batching or ordering of the data, our DDA approach can better address practical settings that require inference as the data arrives, such as perception for autonomous driving.

Background: Diffusion for Image Generation

Diffusion models are a class of latent variable generative models that have recently demonstrated state-of-the-art performance for image synthesis. Given a clean image sampled from the real image distribution $x_0 \sim q(x_0)$, the forward process of the diffusion model defines a fixed Markov chain, to gradually add Gaussian noise to the clean image x_0 over T time steps, producing a sequence of noised images $x_1, x_2, \cdots, x_T$. Mathematically, the forward process is defined as

$$q(x_{1:T}|x_0) := \prod_{t=1}^{T} q(x_t|x_{t-1}), \qquad q\left(x_t \mid x_{t-1}\right) := N\left(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t \mathbf{I}\right), \tag{6.1}$$

where the sequence, $\beta_1, ..., \beta_T$, is a fixed variance schedule to control the step sizes of the noise.

On the other hand, given the noise sampled from a Gaussian distribution $X_T \sim \mathcal{N}(0, \mathbf{I})$, the reverse process of the diffusion model iteratively removes the noise to generate a clean image in T time steps. The reverse process is formulated as a Markov chain with parameterized Gaussian transitions

$$p(x_{0:T}) := p(x_T) \prod_{t=1}^{T} p(x_{t-1}|x_t), \qquad p_\theta\left(x_{t-1} \mid x_t\right) := N\left(x_{t-1}; \mu_\theta\left(x_t, t\right), \sigma_t^2\left(x_t, t\right) \mathbf{I}\right). \tag{6.2}$$

In denoising diffusion probabilistic models (DDPM) [249], we set $\sigma_t\left(x_t, t\right) = \sigma_t \mathbf{I}$ to time-dependent constants. μ_θ is parameterized into a linear combination of x_t and $\epsilon_\theta(x_t, t)$, where $\epsilon_\theta(x_t, t)$ is a function that predicts the noise component of a noised sample x_t. The parameters of $\mu_\theta\left(x_t, t\right)$ are learned by optimizing the variational bound of negative log-likelihood $\mathbb{E}[-\log p_\theta(x_0)]$. With the above parameterization and following DDPM [249], the training objective $\mathcal{L}_{\text{simple}}$ simplifies to the mean-squared error loss between the actual noise $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ in x_t and the predicted noise:

$$\mathcal{L}_{\text{simple}} := ||\epsilon_\theta(x_t, t) - \epsilon||^2. \tag{6.3}$$

Since the training objective is derived from the variational bound on the negative log-likelihood $\mathbb{E}[-\log p_\theta(x_0)]$ of the data, the diffusion model learns a generative model of the source data distribution.

Diffusion-Driven Input Adaptation

Here we detail the key step of our diffusion-driven adaptation approach: the projection of target test inputs to the source data distribution by diffusion. Specifically, we project the test image to the source domain by running the forward process followed by the reverse process of the diffusion model. Note that our approach is orthogonal to the choice of diffusion model as long as the chosen model can be trained on the source data. Equipped with a diffusion model of the source data, our approach applies this single model to the projection of single/multiple/mixed target domain data to the source domain.

We describe the steps of our method and highlight each as it is illustrated in Fig. 6.2. Given an input image x_0 from the target domain and an unconditional diffusion model trained on the source domain, First the forward process (Eq. 6.1 of the diffusion model, the green arrow, perturbs the image with Gaussian noise. We denote the image sequence derived by N iterative forward steps as $x_0, x_1, \cdots, x_N$, where N is a hyper-parameter controlling the amount of noise added to the input image. We name N as "diffusion range" for simplicity. Then the reverse process (Eq. 6.2, the red dotted arrow, iteratively removes noise for N steps to generate the denoised image sequence $x^g_{N-1}, x^g_{N-2}, \cdots, x^g_0$. Since the diffusion model is trained on the source domain, the generated image x^g_0 should have higher likelihood under the source data distribution than the test image x_0, in so much as the diffusion model is fit to the domain.

While this projection can adapt the input, a trade-off arises when choosing the diffusion range N. Too little diffusion, when N is small, fails to project outside of the target domain back to the source. However, too much diffusion, when N is large, fails to project inside of the same class across domains. Our ideal goal is to adapt the input from the target domain to the source domain while preserving its discriminative content for the classification task. The issue is that domain and class information may be interdependent, which makes it difficult to identify the optimal trade-off between domain adaptation and class preservation.

Based on our observation that the class information can relate strongly to the structure (low-frequency signal) of an image, we regularize diffusion to better preserve this structure.

Algorithm 1 Diffusion-Driven Input Adaptation

Input: Reference image x_0
Output: Generated image x_0^g
M: refinement range for the iterative latent refinement
N: diffusion range
Sample $x_N \sim q(x_N \mid x_0)$ ▷ perturb input
$x_N^g \leftarrow x_N$
for $t \leftarrow N \dots 1$ **do**
    $\hat{x}_{t-1}^g \sim p_\theta\left(x_{t-1}^g \mid x_t^g\right)$ ▷ unconditional proposal
    if $t > M$ **then**
        $x_{t-1} \sim q(x_{t-1} \mid x_0)$ ▷ condition encoding
        $x_{t-1}^g \leftarrow \phi_D(x_{t-1}) + (\mathbf{I} - \phi_D)\left(\hat{x}_{t-1}^g\right)$ ▷ $\phi_D(\cdot)$: low-pass filter with scale factor D
    else
        $x_{t-1}^g \leftarrow \hat{x}_{t-1}^g$
    end if
end for
return x_0^g

Inspired by ILVR [254], we adopt an iterative latent refinement process, the purple dotted arrow, which conditions on the input image in the reverse process. This refinement enforces the structural, and therefore class, alignment between the generated image and the input image. In particular, we adopt a linear low-pass filtering operation implemented by $\phi_D(\cdot)$, a sequence of downsampling and upsampling operations by a scaling factor of D, The low-pass filtered image represents the structural information of an image. At each time step t in the reverse process, we force $\phi_D(x_t^g)$ to be identical to $\phi_D(x_t)$. Mathematically, we add a latent refinement operation after sampling $\hat{x}_{t-1}^g$ based on x_t^g,

$$\hat{x}_{t-1}^g \sim p_\theta\left(\hat{x}_{t-1}^g \mid x_t^g\right), \qquad x_{t-1}^g = \phi_D(x_{t-1}) + (\mathbf{I} - \phi_D)\left(\hat{x}_{t-1}^g\right). \tag{6.4}$$

The latent refinement is conducted when $t \geq M$, where M is a hyper-parameter named "refinement range".

In summary, we first perturb the input image from the target domain with noise by the forward process of the diffusion model, and then run the reverse process with iterative latent refinement to carry out input adaptation with minimal alteration of class-dependent information. In this way, we generate an image in the style of the source domain that preserves the class identity of the given target image. Algorithm 1 outlines our approach for projecting the target image to the source domain with diffusion.

Selecting How Much to Adapt by Self-Ensembling

Once we have adapted the target domain inputs to the source domain by diffusion, our source-trained classification model can make predictions on the adapted images. In most

cases, diffusion preserves enough discriminative information in the adapted inputs for correct classification. However, diffusion may occasionally generate imperfect or ambiguous images that are more like source data but result in misclassification. In such cases, the classification model may be more accurate on the original, unadapted input, even with its domain shift.

Motivated by these failure cases for diffusion, we propose a self-ensembling scheme to aggregate the predictions over the original and adapted inputs. Specifically, as we have both the original test image x_0 and generated image x_0^g from diffusion, we first apply the classification model to both inputs. The confidence of the C classification predictions on the original, unadapted and generated, adapted input are denoted as $p \in \mathbb{R}^C$ and $p^g \in \mathbb{R}^C$, respectively. The ensembled prediction fuses the predictions based on the average confidence, i.e., $\text{argmax}_c \frac{1}{2}(p_c + p_c^g)$, where $c \in \{1, \ldots, C\}$.

This self-ensembling scheme automatically selects how much to rely on the adapted and unadapted inputs. Selecting in this way increases the robustness of adaptation by rejecting unsuitable results of generative modeling.

6.4 Experiments

Setup

Dataset ImageNet-C [9] is a standard robustness benchmark for large-scale 1000-way image classification. It consists of synthetic but natural corruptions (e.g., natural noise and blur, digital artifact, and different weather conditions) applied to the ImageNet [29] validation set of 50,000 images. It includes 15 corruption types at 5 levels of severity. We measure robustness as the top-1 accuracy of predictions on the most severe corruptions (level 5) on ImageNet-C. We evaluate DDA with the same hyperparameters in all experiments, except as noted for ablation and analysis.

Adaptation Settings We consider adaptation with and without separation of the target domains/corruption types. The first *independent adaptation*: this is the standard setting for robustness experiments on ImageNet-C, where adaptation and evaluation are done independently for each corruption type. The second *joint adaptation*: this is a more natural and difficult setting, where adaptation and evaluation are done jointly over all corruptions by combining their data. The settings are equivalent for "episodic" methods that make independent predictions across test inputs. Note that the regular source-only model is episodic, as is our DDA method. However, many model adaptation methods are not—Tent [1] and BUFR [230] included—because model updates on one input alter predictions on other inputs. Experimenting with both settings allows for standardized comparison with existing work and exploration of adaptation without knowledge of the target domains.

Classification Models We experiment with multiple classification architectures to ensure general improvement. We select ResNet-50 [8] for a common architeture to standardize on,

Table 6.1: Input adaptation is more robust in the episodic setting of image-wise adaptation. Episodic inference is independent across inputs, which includes source-only prediction without adaptation and model updates by MEMO or input updates by DDA (ours). We evaluate standard accuracy on ImageNet and robustness to corruption on ImageNet-C with maximum severity (level 5). All results are top-1 accuracies (higher is better).

Architecture	Data/ Size	FLOPs/ Params	ImageNet Acc.	Source-Only	MEMO	DDA
				ImageNet-C Acc.		
RedNet-26	1K/224^2	1.7/9.2	76.0	15.0	20.6	**25.0**
ResNet-50	1K/224^2	4.1/25.6	76.6	18.7	24.7	**27.3**
Swin-T	1K/224^2	4.5/28.3	81.2	33.1	29.5	**37.0**
ConvNeXt-T	1K/224^2	4.5/28.6	81.7	39.3	37.8	**41.4**

RedNet-26 [255] for a compact architecture, plus Swin [221] and ConvNeXt [222] to evaluate the state-of-the-art in attentional and convolutional architectures. Experimenting with Swin and ConvNeXt sharpens our evaluation of adaptation as these architectures already improve robustness. Table 6.1 lists their FLOPS, number of parameters, and accuracy on the source data.

Benchmark Evaluation: Independent Adaptation

Input updates are more robust than model updates with episodic adaptation. We begin by evaluating source-only inference (without adaptation), model adaptation with MEMO, and input adaptation with our DDA. Each method is episodic, in making separate predictions for each input, for fair comparison. MEMO adapts to each input by augmentation and entropy minimization: it minimizes the entropy of the predictions w.r.t. the model parameters over different augmentations of the input. By relying on data augmentation, MEMO avoids trivial solutions to optimizing so many parameters on a single input. DDA circumvents this issue by not updating the model at all, and instead updating the input itself. Table 6.1 summarizes each source classifier and compares the robustness of each method. DDA achieves consistently higher robustness than MEMO. For the state-of-the-art architectures, the Swin-T transformer and the ConvNeXt-T convolutional network, DDA still delivers a ~ 2 point boost.

DDA consistently improves across corruption types and prevents catastrophic failure. Figure 6.3 analyzes the robustness of DDA across each corruption type individually.

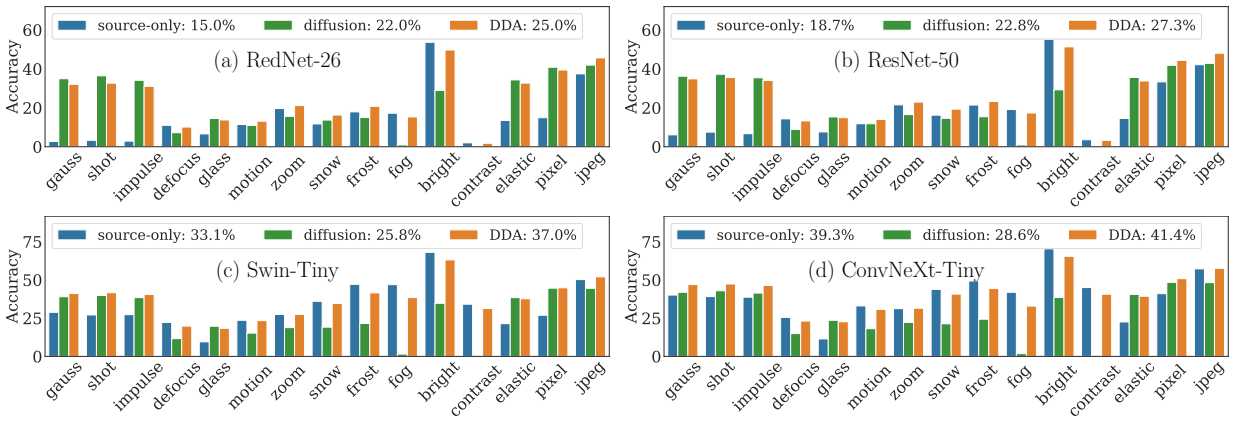


Figure 6.3: DDA reliably improves robustness across corruption types. We compare the source-only model, diffusion-only adaptation, and DDA with diffusion and self-ensembling. DDA is the best on average, and reliably improves over diffusion-only inference with few exceptions. Self-ensembling with DDA prevents catastrophic drops (on fog or contrast, for example).

We observe that diffusion without self-ensembling could consistently outperform source-only on most high-frequency corruptions. As for low-frequency corruptions, our proposed fusion by self-ensembling automatically selects how much to adapt. In this way, the failure cases of diffusion modeling do harm the final prediction, which avoids catastrophic drops on more global corruptions like fog and contrast.

DDA is not sensitive to small batches or ordered data. The amount and order of the data for each corruption type may vary in practical settings. For the amount, source-free methods use the entire test set at once, while test-time methods may choose different batch sizes. For the order of the target data, it is commonly shuffled (as done by Tent and other test-time methods). We evaluate at different batch sizes and with and without shuffling to understand the effect of these data regimes. Figure. 6.4 plots sensitivity these factors. DDA and MEMO are totally unaffected, as episodic methods, but Tent is extremely sensitive. Controlling the amount and order of data during deployment may not always be possible, but Tent requires it to ensure improvement (and not failure).

Challenge Exploration: Joint Adaptation

The joint adaptation setting, in which the data for all corruption types is combined, presents a new challenge. In this new setting, the amount, order, and mixture of the data can be varied to further complicated adaptation for methods that depend on batching or ordering of the domains. As episodic methods, which adapt to each input independently, MEMO and

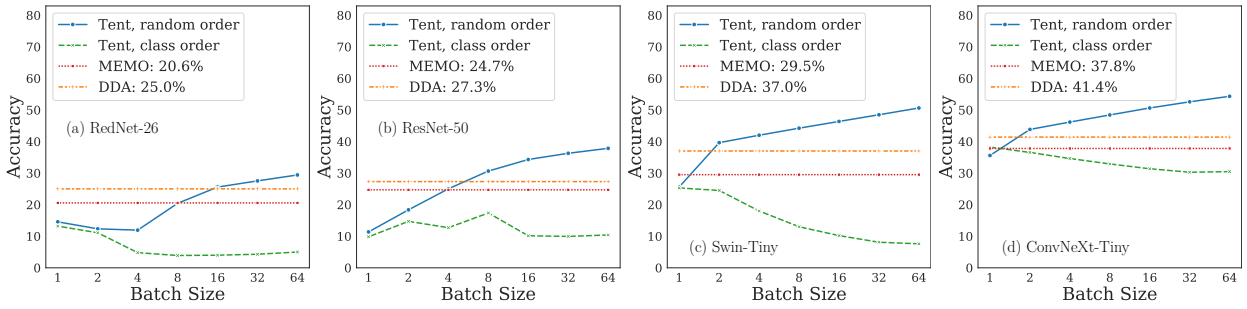


Figure 6.4: DDA is invariant to batch size and data order while Tent is extremely sensitive. To analyze sensivity to the amount and order of the data we measure the average robustness of independent adaptation across corruption types. DDA does not depend on these factors and consistently improves on MEMO. Tent fails on class-ordered data without shuffling and degrades at small batch sizes.

DDA can both address small batches, ordered data, and mixed domains. On the other hand, non-episodic methods, such as model adaptation with cumulative updates across inputs, have no such guarantee.

DDA is more robust than model adaptation in the joint setting. We further compare with MEMO and test-time batch normalization [35] (BN) in the joint setting. We evaluate with ResNet-50 because it is a standard architecture for these model adaptation methods. The accuracies are 27.3% for DDA, 24.7% for MEMO, and 10.3% for BN. Although BN is competitive in the independent setting, in the joint setting sharing the mean and variance across all corruptions types is insufficient for adaptation. BUFR [230] lacks ResNet-50 results, and we could not tune it to be better than source-only accuracy.

DDA assumes less and succeeds where Tent degrades. We compare to Tent [1], a representative fully test-time model adaptation method, which cumulatively updates during testing. Tent can help the most when its assumptions of large enough batches and randomly ordered data are met, but may otherwise harm robustness. In contrast, the accuracy of DDA is independent of batch size and data order, and helps robustness in each setting.

Ablation and Analysis of Diffusion-Driven Adaptation

We ablate the different diffusion steps that update the input. As described in Sec. 6.3, our diffusion-driven adaptation method is composed of a forward process, reverse process, and guidance (or refinement). We experiment with three settings as follows: (1) We first run the forward process (i.e., add Gaussian noise) on the input image and then run the reverse process of the diffusion model to denoise, without the iterative refinement module.

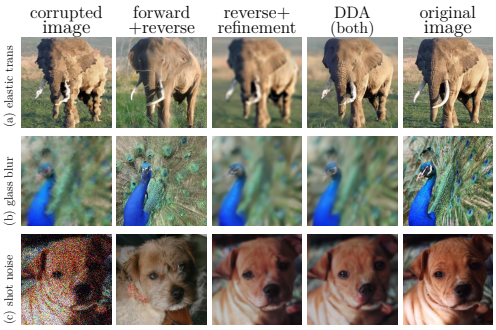
Table 6.2: DDA is reliably more robust when the target data is limited, ordered, or mixed. Deployment may supply target data in various ways. To explore these regimes, we vary batch size and whether or not the data is ordered by class or mixed across corruption types. We compare episodic adaptation by input updates with DDA (ours) and by model updates with MEMO against cumulative adaptation with Tent. DDA and MEMO are invariant to these differences in the data. However, Tent is highly sensitive to batch size and order, and fails in the more natural data regimes.

Method	Mixed Classes	Mixed Types	Batch Size	RedNet-26	ResNet-50	Swin-T	ConvNeXt-T
Source-Only				15.0	18.7	33.1	39.3
MEMO [229]	N/A		N/A	20.6	24.7	29.5	37.8
DDA (ours)				**25.0**	**27.3**	**37.0**	**41.4**
Tent [1]	✗	✗	1 / 64	0.8 / 0.2	0.1 / 0.4	2.8 / 2.3	10.5 / 9.6
	✗	✓	1 / 64	0.8 / 0.3	0.1 / 0.3	8.0 / 2.2	18.8 / 6.5
	✓	✗	1 / 64	0.8 / **7.7**	0.1 / **22.6**	3.0 / **41.0**	11.0 / **50.1**
	✓	✓	1 / 64	0.8 / 3.4	0.1 / 6.5	8.5 / 36.9	18.9 / 47.4

This setting is denoted as "*forward+reverse*". (2) We start from a random noise and run the reverse process of the diffusion model with the iterative refinement module as guidance, which we denote as "*reverse+refinement*". (3) Our DDA model with combines both, i.e., we run the forward process on the input image and then run the reverse process of the diffusion model with the iterative refinement module as guidance. Figure 6.5 shows performance of "forward-reverse", "reverse-refinement", and our *DDA* approach which includes the forward process, reverse process, and iterative refinement. The results demonstrates that each step contributes to the robustness of adaptation.

6.5 Discussion

DDA mitigates shift by test-time input adaptation with diffusion modeling. Our experiments on ImageNet-C confirm that diffusing target data back to the source domain improves robustness. In contrast to test-time model adaptation, which can struggle with scarce, ordered, and mixed data, our method is able to reliably boost accuracy in these regimes. In contrast to source-free model adaptation, which can require re-training to each target, we are able to scalably adapt from multiple targets by keeping our source models fixed. These practical differences derive from our conceptual shift from model adaptation to input adaptation and our adoption of diffusion modeling.



		RedNet-26	ResNet-50	Swin-T	ConvNeXt-T
(a) trans	forward+reverse	24.2	24.5	24.9	25.8
	reverse+refinement	29.6	30.5	35.0	37.0
	DDA (ours)	34.3	35.6	38.4	40.5
(b) blur	forward+reverse	13.8	13.9	14.4	15.0
	reverse+refinement	14.8	15.2	19.7	23.7
	DDA (ours)	14.5	15.3	19.7	23.6
(c) noise	forward+reverse	19.3	19.5	20.2	21.0
	reverse+refinement	30.5	32.0	36.0	39.1
	DDA (ours)	36.4	37.2	39.8	42.9

Figure 6.5: Ablation of diffusion updates justifies each step. We ablate the forward, reverse, and refinement updates of our DDA method. We omit self-ensembling from DDA to focus on these input updates. Forward adds noise, reverse denoises by diffusion, and refinement guides the reverse updates. DDA is best with all steps, but forward and reverse or reverse and refinement help on their own.

Having examined whether to adapt by input updates or model updates, we expect that reconciling the two will deliver more robust generalization than either alone.

Limitations The strengths and weaknesses of input adaptation complement those of model adaptation. Although our method can adapt to a single target input, it must adapt from scratch on each input, and so its computation cannot be amortized across deployment. In contrast, model adaptation by TTT [18] or Tent [1] can update on each batch while cumulatively adapting the model more and more. Although diffusion can project many targets to the source data, and does so without expensive model re-training, it can fail on certain shifts. If these shifts arise gradually, then model adaptation could gradually update too [256], but our fixed diffusion model cannot.

We rely on diffusion, and so we are bound to the quality of generation by diffusion. Diffusion does have its failure modes, even though our positive results demonstrate its present use and future potential. In particular, diffusion models may not only translate domain attributes but other image content, given their large model capacity. Our use of image guidance helps avoid this, but at the cost of restraining adaptation on certain corruptions. New diffusion architectures or new guidance techniques specific to adaption could correct these shortcomings.

Chapter 7

Conclusion

In this thesis, we have discussed how to generalize in dynamic environments via fully test-time adaptation. I firmly believe that we should focus more on building an adaptive, dynamic, generalizable agent, especially in the era of the foundation models and embodied AI. There are still quite a few open challenges and questions that need to be addressed. Below, I briefly summarize two promising future directions.

One of the most attractive directions is to leverage the most recent advance in foundation models [257]. The paradigm shift of foundation models comes from unsupervised learning on a large-scale multi-modality dataset and then adapting to various downstream tasks. The value of fully test-time adaptation is further highlighted in this scenario. Since re-training foundation models could be typically quite time-consuming, we cannot afford any modification to the training procedure for adaptation purposes. That is precisely the motivation behind *fully* where only test data is involved. Since foundation models could be seen as a reliable starting point to make a prediction, we could trust the confidence of foundation models and gradually refine the prediction via entropy minimization. With the help of fully test-time adaptation, I believe foundation models could be a solid basis for more downstream applications.

Another future direction is to extend the proposed test-time adaptation framework to other domains and modalities, such as language, touch, sound, and so on. Robot perception for embodied AI is naturally a multi-modality system, where each sensory modality could internally hold an inexplicit consistency among each other, previously explored in [258], [259]. Such a cross-task consistency should also play an important role, as a promising optimization objective, in the application of fully test-time adaptation.

We hope that the research on fully test-time adaptation in this thesis can identify a novel direction to enhance a model— adapting itself in a self-supervised manner and yielding a new model with every update. Such a dynamic model should be able to learn continuously to generalize to new environments throughout its lifetime.

Bibliography

[1] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell, "Tent: Fully test-time adaptation by entropy minimization," in *ICLR*, 2021.

[2] Dequan Wang, An Ju, Evan Shelhamer, David Wagner, and Trevor Darrell, "Fighting gradients with gradients: Dynamic defenses against adversarial attacks," *arXiv preprint arXiv:2105.08714*, 2021.

[3] Dequan Wang, Evan Shelhamer, Bruno Olshausen, and Trevor Darrell, "Dynamic scale inference by entropy minimization," *arXiv preprint arXiv:1908.03182*, 2019.

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

[5] Jin Gao, Jialing Zhang, Xihui Liu, Trevor Darrell, Evan Shelhamer, and Dequan Wang, "Back to the source: Diffusion-driven test-time adaptation," *arXiv preprint arXiv:2207.03442*, 2022.

[6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," in *NeurIPS*, 2012.

[7] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2015.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[9] Dan Hendrycks and Thomas Dietterich, "Benchmarking neural network robustness to common corruptions and perturbations," in *ICLR*, 2019.

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

[11] 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.

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

[13] 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.

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

[15] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell, "Adapting visual category models to new domains," in *ECCV*, 2010.

[16] Yaroslav Ganin and Victor Lempitsky, "Unsupervised domain adaptation by backpropagation," in *ICML*, 2015.

[17] Eric Tzeng, Judy Hoffman, Trevor Darrell, and Kate Saenko, "Simultaneous deep transfer across domains and tasks," in *ICCV*, 2015.

[18] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A Efros, and Moritz Hardt, "Test-time training for out-of-distribution generalization," in *ICML*, 2020.

[19] Claude Elwood Shannon, "A mathematical theory of communication," *Bell system technical journal*, vol. 27, 1948.

[20] Spyros Gidaris, Praveer Singh, and Nikos Komodakis, "Unsupervised representation learning by predicting image rotations," in *ICLR*, 2018.

[21] Carl Doersch, Abhinav Gupta, and Alexei A Efros, "Unsupervised visual representation learning by context prediction," in *ICCV*, 2015.

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

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

[24] Yves Grandvalet and Yoshua Bengio, "Semi-supervised learning by entropy minimization," in *NeurIPS*, 2005.

[25] Guneet Singh Dhillon, Pratik Chaudhari, Avinash Ravichandran, and Stefano Soatto, "A baseline for few-shot image classification," in *ICLR*, 2020.

[26] Fabio Maria Carlucci, Lorenzo Porzi, Barbara Caputo, Elisa Ricci, and Samuel Rota Bulo, "Autodial: Automatic domain alignment layers," in *ICCV*, 2017.

[27] 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.

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

[29] 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.

[30] Alex Krizhevsky, "Learning multiple layers of features from tiny images," University of Toronto, Tech. Rep., 2009.

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

[32] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[33] Jonathan J. Hull, "A database for handwritten text recognition research," *PAMI*, 1994.

[34] Sergey Ioffe and Christian Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *ICML*, 2015.

[35] 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.

[36] 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.

[37] Diederik Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.

[38] 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.

[39] 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.

[40] 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.

[41] 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.

[42] 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 *ICLR*, 2019.

[43] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell, "Adversarial discriminative domain adaptation," in *CVPR*, 2017.

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

[45] Stephan R Richter, Zeeshan Hayder, and Vladlen Koltun, "Playing for benchmarks," in *ICCV*, 2017.

[46] 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.

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

[48] 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.

[49] Xingchao Peng, Ben Usman, Neela Kaushik, Judy Hoffman, Dequan Wang, and Kate Saenko, "VisDA: The visual domain adaptation challenge," *arXiv preprint arXiv:1710.06924*, 2017.

[50] Jian Liang, Dapeng Hu, and Jiashi Feng, "Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation," in *ICML*, 2020.

[51] Rui Li, Qianfen Jiao, Wenming Cao, Hau-San Wong, and Si Wu, "Model adaptation: Unsupervised domain adaptation without source data," in *CVPR*, 2020.

[52] Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun, "Exploring self-attention for image recognition," in *CVPR*, 2020.

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

[54] 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*, Cambridge, MA, USA: MIT Press, 2009, pp. 131–160.

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

[56] A Gammerman, V Vovk, and V Vapnik, "Learning by transduction," in *UAI*, 1998.

[57] Thorsten Joachims, "Transductive inference for text classification using support vector machines," in *ICML*, 1999.

[58] Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf, "Learning with local and global consistency," in *NeurIPS*, 2004.

[59] Jogendra Nath Kundu, Naveen Venkat, R Venkatesh Babu, *et al.*, "Universal source-free domain adaptation," in *CVPR*, 2020.

[60] Boris Chidlovskii, Stephane Clinchant, and Gabriela Csurka, "Domain adaptation in the absence of source domain data," in *SIGKDD*, 2016.

[61] Vidit Jain and Erik Learned-Miller, "Online domain adaptation of a pre-trained cascade of classifiers," in *CVPR*, 2011.

[62] Yanghao Li, Naiyan Wang, Jianping Shi, Jiaying Liu, and Xiaodi Hou, "Revisiting batch normalization for practical domain adaptation," *arXiv preprint arXiv:1603.04779*, 2017.

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

[64] 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.

[65] 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.

[66] Yuxin Wu and Kaiming He, "Group normalization," in *ECCV*, 2018.

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

[68] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar, "Do cifar-10 classifiers generalize to cifar-10?" *arXiv preprint arXiv:1806.00451*, 2018.

[69] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus, "Intriguing properties of neural networks," in *ICLR*, 2014.

[70] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu, "Towards deep learning models resistant to adversarial attacks," in *ICLR*, 2018.

[71] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger, "On calibration of modern neural networks," in *ICML*, 2017.

[72] Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li, "Adversarial examples: Attacks and defenses for deep learning," *TNNLS*, 2019.

[73] Florian Tramer, Nicholas Carlini, Wieland Brendel, and Aleksander Madry, "On adaptive attacks to adversarial example defenses," in *NeurIPS*, 2020.

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

[75] Guneet S. Dhillon, Kamyar Azizzadenesheli, Jeremy D. Bernstein, Jean Kossaifi, Aran Khanna, Zachary C. Lipton, and Animashree Anandkumar, "Stochastic activation pruning for robust adversarial defense," in *ICLR*, 2018.

[76] Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens van der Maaten, "Countering adversarial images using input transformations," in *ICLR*, 2018.

[77] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter, "Certified adversarial robustness via randomized smoothing," in *ICML*, 2019.

[78] Dong Su, Huan Zhang, Hongge Chen, Jinfeng Yi, Pin-Yu Chen, and Yupeng Gao, "Is robustness the cost of accuracy?–a comprehensive study on the robustness of 18 deep image classification models," in *ECCV*, 2018.

[79] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael I Jordan, "Theoretically principled trade-off between robustness and accuracy," in *ICML*, 2019.

[80] Francesco Croce and Matthias Hein, "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks," in *ICML*, 2020.

[81] David Evans, Anh Nguyen-Tuong, and John Knight, "Effectiveness of moving target defenses," in *Moving target defense*, Springer, 2011, pp. 29–48.

[82] Ian Goodfellow, "A research agenda: Dynamic models to defend against correlated attacks," *arXiv preprint arXiv:1903.06293*, 2019.

[83] Yash Sharma and Pin-Yu Chen, "Attacking the madry defense model with L_1-based adversarial examples," *arXiv preprint arXiv:1710.10733*, 2017.

[84] Florian Tramer and Dan Boneh, "Adversarial training and robustness for multiple perturbations," in *NeurIPS*, 2019.

[85] Edward Raff, Jared Sylvester, Steven Forsyth, and Mark McLean, "Barrage of random transforms for adversarially robust defense," in *CVPR*, 2019.

[86] Tianyu Pang*, Kun Xu*, and Jun Zhu, "Mixup inference: Better exploiting mixup to defend adversarial attacks," in *ICLR*, 2020.

[87] Anish Athalye, Nicholas Carlini, and David Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," in *ICML*, 2018.

[88] Yang Song, Taesup Kim, Sebastian Nowozin, Stefano Ermon, and Nate Kushman, "Pixeldefend: Leveraging generative models to understand and defend against adversarial examples," in *ICLR*, 2018.

[89] Pouya Samangouei, Maya Kabkab, and Rama Chellappa, "Defense-GAN: Protecting classifiers against adversarial attacks using generative models," in *ICLR*, 2018.

[90] Mitch Hill, Jonathan Craig Mitchell, and Song-Chun Zhu, "Stochastic security: Adversarial defense using long-run dynamics of energy-based models," in *ICLR*, 2021.

[91] Changhao Shi, Chester Holtz, and Gal Mishne, "Online adversarial purification based on self-supervised learning," in *ICLR*, 2021.

[92] Huan Zhang, Hongge Chen, Chaowei Xiao, Sven Gowal, Robert Stanforth, Bo Li, Duane Boning, and Cho-Jui Hsieh, "Towards stable and efficient training of verifiably robust neural networks," in *ICLR*, 2020.

[93] David Stutz, Matthias Hein, and Bernt Schiele, "Disentangling adversarial robustness and generalization," in *CVPR*, 2019.

[94] Justin Gilmer, Nicolas Ford, Nicholas Carlini, and Ekin Cubuk, "Adversarial examples are a natural consequence of test error in noise," in *ICML*, 2019.

[95] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein, "Neural module networks," in *CVPR*, 2016.

[96] Andreas Veit and Serge Belongie, "Convolutional networks with adaptive inference graphs," in *ECCV*, 2018.

[97] Alex Graves, "Adaptive computation time for recurrent neural networks," *arXiv preprint arXiv:1603.08983*, 2016.

[98] Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E Gonzalez, "Skipnet: Learning dynamic routing in convolutional networks," in *ECCV*, 2018.

[99] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean, "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer," in *ICLR*, 2017.

[100] Brandon Yang, Gabriel Bender, Quoc V Le, and Jiquan Ngiam, "Condconv: Conditionally parameterized convolutions for efficient inference," in *NeurIPS*, 2019.

[101] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud, "Neural ordinary differential equations," in *NeurIPS*, 2018.

[102] Shaojie Bai, J Zico Kolter, and Vladlen Koltun, "Deep equilibrium models," in *NeurIPS*, 2020.

[103] Evan Shelhamer, Dequan Wang, and Trevor Darrell, "Blurring the line between structure and learning to optimize and adapt receptive fields," *arXiv preprint arXiv:1904.11487*, 2019.

[104] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell, "Cycada: Cycle-consistent adversarial domain adaptation," in *ICML*, 2018.

[105] Yair Carmon, Aditi Raghunathan, Ludwig Schmidt, John C Duchi, and Percy S Liang, "Unlabeled data improves adversarial robustness," in *NeurIPS*, 2019.

[106] Vikash Sehwag, Saeed Mahloujifar, Tinashe Handina, Sihui Dai, Chong Xiang, Mung Chiang, and Prateek Mittal, "Improving adversarial robustness using proxy distributions," *arXiv preprint arXiv:2104.09425*, 2021.

[107] Eric Wong, Leslie Rice, and J Zico Kolter, "Fast is better than free: Revisiting adversarial training," in *ICLR*, 2020.

[108] Gavin Weiguang Ding, Yash Sharma, Kry Yik Chau Lui, and Ruitong Huang, "Mma training: Direct input space margin maximization through adversarial training," in *ICLR*, 2020.

[109] Leslie Rice, Eric Wong, and Zico Kolter, "Overfitting in adversarially robust deep learning," in *ICML*, 2020.

[110] Jérôme Rony, Luiz G Hafemann, Luiz S Oliveira, Ismail Ben Ayed, Robert Sabourin, and Eric Granger, "Decoupling direction and norm for efficient gradient-based l2 adversarial attacks and defenses," in *CVPR*, 2019.

[111] Jianbang Ding, Xuancheng Ren, Ruixuan Luo, and Xu Sun, "An adaptive and momental bound method for stochastic learning," *arXiv preprint arXiv:1910.12249*, 2019.

[112] Ryan Gomes, Andreas Krause, and Pietro Perona, "Discriminative clustering by regularized information maximization," in *NeurIPS*, 2010.

[113] Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein, "Robustbench: A standardized adversarial robustness benchmark," *arXiv preprint arXiv:2010.09670*, 2020.

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

[115] Francesco Croce and Matthias Hein, "Minimally distorted adversarial examples with a fast adaptive boundary attack," in *ICML*, 2020.

[116] Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein, "Square attack: A query-efficient black-box adversarial attack via random search," in *ECCV*, 2020.

[117] Christopher Olston, Noah Fiedel, Kiril Gorovoy, Jeremiah Harmsen, Li Lao, Fangwei Li, Vinu Rajashekhar, Sukriti Ramesh, and Jordan Soyke, "Tensorflow-serving: Flexible, high-performance ml serving," in *NeurIPS Workshop*, 2017.

[118] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár, "Designing network design spaces," in *CVPR*, 2020.

[119] Ziwei Liu, Zhongqi Miao, Xingang Pan, Xiaohang Zhan, Dahua Lin, Stella X Yu, and Boqing Gong, "Open compound domain adaptation," in *CVPR*, 2020.

[120] Dongxian Wu, Shu-Tao Xia, and Yisen Wang, "Adversarial weight perturbation helps robust generalization," in *NeurIPS*, 2020.

[121] P. Burt and E. Adelson, "The laplacian pyramid as a compact image code," *Communications, IEEE Transactions on*, vol. 31, no. 4, pp. 532–540, 1983.

[122] Angjoo Kanazawa, Abhishek Sharma, and David Jacobs, "Locally scale-invariant convolutional neural networks," *arXiv preprint arXiv:1412.5104*, 2014.

[123] Taco S Cohen and Max Welling, "Steerable cnns," in *ICLR*, 2017.

[124] Tony Lindeberg, *Scale-space theory in computer vision*. Springer Science & Business Media, 1994, vol. 256.

[125] D.G. Lowe, "Distinctive image features from scale-invariant keypoints," *IJCV*, 2004.

[126] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu, "Spatial transformer networks," in *NeurIPS*, 2015.

[127] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei, "Deformable convolutional networks," in *ICCV*, 2017.

[128] Bert De Brabandere, Xu Jia, Tinne Tuytelaars, and Luc Van Gool, "Dynamic filter networks," in *NeurIPS*, 2016.

[129] MD Zeiler and R Fergus, "Visualizing and understanding convolutional networks," in *ECCV*, 2014.

[130] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie, "Feature pyramid networks for object detection," in *CVPR*, 2017.

[131] Bruno A Olshausen, Charles H Anderson, and David C Van Essen, "A neurobiological model of visual attention and invariant pattern recognition based on dynamic routing of information," *Journal of Neuroscience*, vol. 13, no. 11, pp. 4700–4719, 1993.

[132] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman, "The pascal visual object classes (voc) challenge," *IJCV*, 2010.

[133] Rui Zhang, Sheng Tang, Yongdong Zhang, Jintao Li, and Shuicheng Yan, "Scale-adaptive convolutions for scene parsing," in *ICCV*, 2017.

[134] Bharath Hariharan, Pablo Arbeláez, Lubomir Bourdev, Subhransu Maji, and Jitendra Malik, "Semantic contours from inverse detectors," in *ICCV*, 2011.

[135] Fisher Yu, Dequan Wang, Evan Shelhamer, and Trevor Darrell, "Deep layer aggregation," in *CVPR*, 2018.

[136] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer, "Automatic differentiation in pytorch," in *NeurIPS Workshop*, 2017.

[137] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *PAMI*, 2018.

[138] Andrew G Howard, "Some improvements on deep convolutional neural network based image classification," *arXiv preprint arXiv:1312.5402*, 2013.

[139] Pedro HO Pinheiro and Ronan Collobert, "Recurrent convolutional neural networks for scene labeling," in *ICML*, 2014.

[140] Joao Carreira, Pulkit Agrawal, Katerina Fragkiadaki, and Jitendra Malik, "Human pose estimation with iterative error feedback," in *CVPR*, 2016.

[141] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan, "Unsupervised domain adaptation with residual transfer networks," in *NeurIPS*, 2016.

[142] Tuan-Hung Vu, Himalaya Jain, Maxime Bucher, Mathieu Cord, and Patrick Pérez, "Advent: Adversarial entropy minimization for domain adaptation in semantic segmentation," *arXiv preprint arXiv:1811.12833*, 2018.

[143] Jost Tobias Springenberg, "Unsupervised and semi-supervised learning with categorical generative adversarial networks," in *ICLR*, 2016.

[144] Ronald J Williams and Jing Peng, "Function optimization using connectionist reinforcement learning algorithms," *Connection Science*, vol. 3, no. 3, pp. 241–268, 1991.

[145] Zafarali Ahmed, Nicolas Le Roux, Mohammad Norouzi, and Dale Schuurmans, "Understanding the impact of entropy on policy optimization," in *ICML*, 2019.

[146] Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang, "A tutorial on energy-based learning," *Predicting structured data*, 2006.

[147] Daphne Koller and Nir Friedman, *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[148] M.J. Wainwright and M.I. Jordan, "Graphical models, exponential families, and variational inference," *Foundations and Trendső in Machine Learning*, vol. 1, no. 1-2, pp. 1–305, 2008.

[149] David Belanger, Bishan Yang, and Andrew McCallum, "End-to-end learning for structured prediction energy networks," in *ICML*, 2017.

[150] Michael Gygli, Mohammad Norouzi, and Anelia Angelova, "Deep value networks learn to evaluate and iteratively refine structured outputs," in *ICML*, 2017.

[151] V. Vapnik, *Statistical Learning Theory*. Wiley, 1998.

[152] Chelsea Finn, Pieter Abbeel, and Sergey Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," *arXiv preprint arXiv:1703.03400*, 2017.

[153] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[154] Xinlei Chen and Kaiming He, "Exploring simple siamese representation learning," *arXiv preprint arXiv:2011.10566*, 2020.

[155] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin, "Unsupervised learning of visual features by contrasting cluster assignments," *arXiv preprint arXiv:2006.09882*, 2020.

[156] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick, "Momentum contrast for unsupervised visual representation learning," in *CVPR*, 2020.

[157] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He, "Improved baselines with momentum contrastive learning," *arXiv preprint arXiv:2003.04297*, 2020.

[158] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, *et al.*, "Bootstrap your own latent: A new approach to self-supervised learning," *arXiv preprint arXiv:2006.07733*, 2020.

[159] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny, "Barlow twins: Self-supervised learning via redundancy reduction," *arXiv preprint arXiv:2103.03230*, 2021.

[160] Haohan Wang, Songwei Ge, Zachary Lipton, and Eric P Xing, "Learning robust global representations by penalizing local predictive power," in *NeurIPS*, 2019.

[161] Bingyi Kang, Saining Xie, Marcus Rohrbach, Zhicheng Yan, Albert Gordo, Jiashi Feng, and Yannis Kalantidis, "Decoupling representation and classifier for long-tailed recognition," in *ICLR*, 2020.

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

[163] Yuhua Chen, Wen Li, Christos Sakaridis, Dengxin Dai, and Luc Van Gool, "Domain adaptive faster r-cnn for object detection in the wild," in *CVPR*, 2018.

[164] Judy Hoffman, Dequan Wang, Fisher Yu, and Trevor Darrell, "Fcns in the wild: Pixel-level adversarial and constraint-based adaptation," *arXiv preprint arXiv:1612.02649*, 2016.

[165] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan, "Learning transferable features with deep adaptation networks," in *ICML*, 2015.

[166] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan, "Deep transfer learning with joint adaptation networks," in *ICML*, 2017.

[167] Baochen Sun, Jiashi Feng, and Kate Saenko, "Return of frustratingly easy domain adaptation," in *AAAI*, 2016.

[168] Werner Zellinger, Thomas Grubinger, Edwin Lughofer, Thomas Natschläger, and Susanne Saminger-Platz, "Central moment discrepancy (cmd) for domain-invariant representation learning," *arXiv preprint arXiv:1702.08811*, 2017.

[169] Ming-Yu Liu, Thomas Breuel, and Jan Kautz, "Unsupervised image-to-image translation networks," *arXiv preprint arXiv:1703.00848*, 2017.

[170] Jian Liang, Dapeng Hu, Ran He, and Jiashi Feng, "Distill and fine-tune: Effective adaptation from a black-box source model," *arXiv preprint arXiv:2104.01539*, 2021.

[171] Haojian Zhang, Yabin Zhang, Kui Jia, and Lei Zhang, "Unsupervised domain adaptation of black-box source models," *arXiv preprint arXiv:2101.02839*, 2021.

[172] Kunhong Wu, Yucheng Shi, Yahong Han, Yunfeng Shao, Bingshuai Li, and Qi Tian, "Domain adaptation without model transferring," *arXiv preprint arXiv:2107.10174*, 2021.

[173] Weichen Zhang, Wanli Ouyang, Wen Li, and Dong Xu, "Collaborative and adversarial network for unsupervised domain adaptation," in *CVPR*, 2018.

[174] Jaehoon Choi, Minki Jeong, Taekyung Kim, and Changick Kim, "Pseudo-labeling curriculum for unsupervised domain adaptation," *arXiv preprint arXiv:1908.00262*, 2019.

[175] Yang Zou, Zhiding Yu, BVK Kumar, and Jinsong Wang, "Unsupervised domain adaptation for semantic segmentation via class-balanced self-training," in *ECCV*, 2018.

[176] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze, "Deep clustering for unsupervised learning of visual features," in *ECCV*, 2018.

[177] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *JAIR*, 2002.

[178] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao, "Borderline-smote: A new over-sampling method in imbalanced data sets learning," in *ICIC*, 2005.

[179] Li Shen, Zhouchen Lin, and Qingming Huang, "Relay backpropagation for effective learning of deep convolutional neural networks," in *ECCV*, 2016.

[180] Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens Van Der Maaten, "Exploring the limits of weakly supervised pretraining," in *ECCV*, 2018.

[181] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie, "Class-balanced loss based on effective number of samples," in *CVPR*, 2019.

[182] Salman H Khan, Munawar Hayat, Mohammed Bennamoun, Ferdous A Sohel, and Roberto Togneri, "Cost-sensitive learning of deep feature representations from imbalanced data," *TNNLS*, 2017.

[183] Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Arechiga, and Tengyu Ma, "Learning imbalanced datasets with label-distribution-aware margin loss," *arXiv preprint arXiv:1906.07413*, 2019.

[184] Salman Khan, Munawar Hayat, Syed Waqas Zamir, Jianbing Shen, and Ling Shao, "Striking the right balance with uncertainty," in *CVPR*, 2019.

[185] Chen Huang, Yining Li, Chen Change Loy, and Xiaoou Tang, "Deep imbalanced learning for face recognition and attribute prediction," *PAMI*, 2019.

[186] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár, "Focal loss for dense object detection," in *ICCV*, 2017.

[187] Jun Shu, Qi Xie, Lixuan Yi, Qian Zhao, Sanping Zhou, Zongben Xu, and Deyu Meng, "Meta-weight-net: Learning an explicit mapping for sample weighting," *arXiv preprint arXiv:1902.07379*, 2019.

[188] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun, "Learning to reweight examples for robust deep learning," in *ICML*, 2018.

[189] Munawar Hayat, Salman Khan, Waqas Zamir, Jianbing Shen, and Ling Shao, "Max-margin class imbalanced learning with gaussian affinity," *arXiv preprint arXiv:1901.07711*, 2019.

[190] Xi Yin, Xiang Yu, Kihyuk Sohn, Xiaoming Liu, and Manmohan Chandraker, "Feature transfer learning for face recognition with under-represented data," in *CVPR*, 2019.

[191] Ziwei Liu, Zhongqi Miao, Xiaohang Zhan, Jiayun Wang, Boqing Gong, and Stella X Yu, "Large-scale long-tailed recognition in an open world," in *CVPR*, 2019.

[192] Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel, "Fixmatch: Simplifying semi-supervised learning with consistency and confidence," *arXiv preprint arXiv:2001.07685*, 2020.

[193] Philip Bachman, Ouais Alsharif, and Doina Precup, "Learning with pseudo-ensembles," *arXiv preprint arXiv:1412.4864*, 2014.

[194] Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen, "Regularization with stochastic transformations and perturbations for deep semi-supervised learning," *arXiv preprint arXiv:1606.04586*, 2016.

[195] Samuli Laine and Timo Aila, "Temporal ensembling for semi-supervised learning," *arXiv preprint arXiv:1610.02242*, 2016.

[196] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan, "Deep hashing network for unsupervised domain adaptation," in *CVPR*, 2017.

[197] Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie, "The inaturalist species classification and detection dataset," in *CVPR*, 2018.

[198] Yuan Shi and Fei Sha, "Information-theoretical learning of discriminative clusters for unsupervised domain adaptation," in *ICML*, 2012.

[199] Weihua Hu, Takeru Miyato, Seiya Tokui, Eiichi Matsumoto, and Masashi Sugiyama, "Learning discrete representations via information maximizing self-augmented training," in *ICML*, 2017.

[200] Kuniaki Saito, Yoshitaka Ushiku, Tatsuya Harada, and Kate Saenko, "Adversarial dropout regularization," in *ICLR*, 2018.

[201] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan, "Conditional adversarial domain adaptation," in *NeurIPS*, 2018.

[202] Xinyang Chen, Sinan Wang, Mingsheng Long, and Jianmin Wang, "Transferability vs. discriminability: Batch spectral penalization for adversarial domain adaptation," in *ICML*, 2019.

[203] Ximei Wang, Ying Jin, Mingsheng Long, Jianmin Wang, and Michael I Jordan, "Transferable normalization: Towards improving transferability of deep neural networks," in *NeurIPS*, 2019.

[204] Ruijia Xu, Guanbin Li, Jihan Yang, and Liang Lin, "Larger norm more transferable: An adaptive feature norm approach for unsupervised domain adaptation," in *ICCV*, 2019.

[205] Chen-Yu Lee, Tanmay Batra, Mohammad Haris Baig, and Daniel Ulbricht, "Sliced wasserstein discrepancy for unsupervised domain adaptation," in *CVPR*, 2019.

[206] Woong-Gi Chang, Tackgeun You, Seonguk Seo, Suha Kwak, and Bohyung Han, "Domain-specific batch normalization for unsupervised domain adaptation," in *CVPR*, 2019.

[207] Zhihe Lu, Yongxin Yang, Xiatian Zhu, Cong Liu, Yi-Zhe Song, and Tao Xiang, "Stochastic classifiers for unsupervised domain adaptation," in *CVPR*, 2020.

[208] Tim Salimans and Diederik P Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," *arXiv preprint arXiv:1602.07868*, 2016.

[209] Priya Goyal, Quentin Duval, Jeremy Reizenstein, Matthew Leavitt, Min Xu, Benjamin Lefaudeux, Mannat Singh, Vinicius Reis, Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Ishan Misra, *Vissl*, `https://github.com/facebookresearch/vissl`, 2021.

[210] MMClassification Contributors, *Openmmlab's image classification toolbox and benchmark*, `https://github.com/open-mmlab/mmclassification`, 2020.

[211] Lukas Biewald, *Experiment tracking with weights and biases*, 2020. [Online]. Available: `https://www.wandb.com/`.

[212] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *ICCV*, 2015.

[213] Rafael Müller, Simon Kornblith, and Geoffrey Hinton, "When does label smoothing help?" *arXiv preprint arXiv:1906.02629*, 2019.

[214] Ilya Loshchilov and Frank Hutter, "Sgdr: Stochastic gradient descent with warm restarts," *arXiv preprint arXiv:1608.03983*, 2016.

[215] Shlok Mishra, Anshul Shah, Ankan Bansal, Jonghyun Choi, Abhinav Shrivastava, Abhishek Sharma, and David Jacobs, "Learning visual representations for transfer learning by suppressing texture," *arXiv preprint arXiv:2011.01901*, 2020.

[216] Yingwei Li, Qihang Yu, Mingxing Tan, Jieru Mei, Peng Tang, Wei Shen, Alan Yuille, and Cihang Xie, "Shape-texture debiased neural network training," in *ICLR*, 2021.

[217] Katherine L Hermann, Ting Chen, and Simon Kornblith, "The origins and prevalence of texture bias in convolutional neural networks," in *NeurIPS*, 2020.

[218] Xiaofeng Mao, Gege Qi, Yuefeng Chen, Xiaodan Li, Shaokai Ye, Yuan He, and Hui Xue, "Rethinking the design principles of robust vision transformer," *arXiv preprint arXiv:2105.07926*, 2021.

[219] 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.

[220] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," in *ICLR*, 2021.

[221] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *ICCV*, 2021.

[222] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie, "A convnet for the 2020s," in *CVPR*, 2022.

[223] Eric Mintun, Alexander Kirillov, and Saining Xie, "On interaction between augmentations and corruptions in natural corruption robustness," in *NeurIPS*, 2021.

[224] Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, *et al.*, "Wilds: A benchmark of in-the-wild distribution shifts," in *ICML*, 2021.

[225] Antonio Torralba and Alexei A Efros, "Unbiased look at dataset bias," in *CVPR*, 2011.

[226] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky, "Domain-adversarial training of neural networks," *JMLR*, 2016.

[227] Thomas Varsavsky, Mauricio Orbes-Arteaga, Carole H Sudre, Mark S Graham, Parashkev Nachev, and M Jorge Cardoso, "Test-time unsupervised domain adaptation," in *MICCAI*, 2020.

[228] Yusuke Iwasawa and Yutaka Matsuo, "Test-time classifier adjustment module for model-agnostic domain generalization," in *NeurIPS*, 2021.

[229] Marvin Zhang, Sergey Levine, and Chelsea Finn, "MEMO: Test time robustness via adaptation and augmentation," *arXiv preprint arXiv:2110.09506*, 2021.

[230] Cian Eastwood, Ian Mason, Chris Williams, and Bernhard Schölkopf, "Source-free adaptation to measurement shift via bottom-up feature restoration," in *ICLR*, 2021.

[231] Marvin Zhang, Henrik Marklund, Nikita Dhawan, Abhishek Gupta, Sergey Levine, and Chelsea Finn, "Adaptive risk minimization: Learning to adapt to domain shift," *arXiv preprint arXiv:2007.02931*, 2020.

[232] Jongmin Yoon, Sung Ju Hwang, and Juho Lee, "Adversarial purification with score-based generative models," in *ICML*, 2021.

[233] Erik Reinhard, Michael Adhikhmin, Bruce Gooch, and Peter Shirley, "Color transfer between images," *IEEE Computer graphics and applications*, vol. 21, no. 5, pp. 34–41, 2001.

[234] François Pitié, Anil C Kokaram, and Rozenn Dahyot, "Automated colour grading using colour distribution transfer," *Computer Vision and Image Understanding*, vol. 107, no. 1-2, pp. 123–137, 2007.

[235] Yijun Li, Ming-Yu Liu, Xueting Li, Ming-Hsuan Yang, and Jan Kautz, "A closed-form solution to photorealistic image stylization," in *ECCV*, 2018.

[236] Jaejun Yoo, Youngjung Uh, Sanghyuk Chun, Byeongkyu Kang, and Jung-Woo Ha, "Photorealistic style transfer via wavelet transforms," in *ICCV*, 2019.

[237] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *ICCV*, 2017.

[238] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz, "Multimodal unsupervised image-to-image translation," in *ECCV*, 2018.

[239] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu, "Semantic image synthesis with spatially-adaptive normalization," in *CVPR*, 2019.

[240] Taesung Park, Alexei A Efros, Richard Zhang, and Jun-Yan Zhu, "Contrastive learning for unpaired image-to-image translation," in *ECCV*, 2020.

[241] Liming Jiang, Changxu Zhang, Mingyang Huang, Chunxiao Liu, Jianping Shi, and Chen Change Loy, "Tsit: A simple and versatile framework for image-to-image translation," in *ECCV*, 2020.

[242] Stephan R Richter, Hassan Abu Al Haija, and Vladlen Koltun, "Enhancing photorealism enhancement," *PAMI*, 2022.

[243] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *ICML*, 2015.

[244] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer, "High-resolution image synthesis with latent diffusion models," in *CVPR*, 2022.

[245] Yang Song and Stefano Ermon, "Generative modeling by estimating gradients of the data distribution," in *NeurIPS*, 2019.

[246] Yang Song and Stefano Ermon, "Improved techniques for training score-based generative models," in *NeurIPS*, 2020.

[247] Alexander Quinn Nichol and Prafulla Dhariwal, "Improved denoising diffusion probabilistic models," in *ICML*, 2021.

[248] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole, "Score-based generative modeling through stochastic differential equations," in *ICLR*, 2021.

[249] Jonathan Ho, Ajay Jain, and Pieter Abbeel, "Denoising diffusion probabilistic models," in *NeurIPS*, 2020.

[250] Prafulla Dhariwal and Alexander Nichol, "Diffusion models beat gans on image synthesis," in *NeurIPS*, 2021.

[251] Jonathan Ho and Tim Salimans, "Classifier-free diffusion guidance," in *NeurIPS Workshop on Deep Generative Models and Downstream Applications*, 2021.

[252] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen, "Glide: Towards photorealistic image generation and editing with text-guided diffusion models," *arXiv preprint arXiv:2112.10741*, 2021.

[253] Xihui Liu, Dong Huk Park, Samaneh Azadi, Gong Zhang, Arman Chopikyan, Yuxiao Hu, Humphrey Shi, Anna Rohrbach, and Trevor Darrell, "More control for free! image synthesis with semantic diffusion guidance," *arXiv preprint arXiv:2112.05744*, 2021.

[254] Jooyoung Choi, Sungwon Kim, Yonghyun Jeong, Youngjune Gwon, and Sungroh Yoon, "Ilvr: Conditioning method for denoising diffusion probabilistic models," in *ICCV*, 2021.

[255] Duo Li, Jie Hu, Changhu Wang, Xiangtai Li, Qi She, Lei Zhu, Tong Zhang, and Qifeng Chen, "Involution: Inverting the inherence of convolution for visual recognition," in *CVPR*, 2021.

[256] Ananya Kumar, Tengyu Ma, and Percy Liang, "Understanding self-training for gradual domain adaptation," in *ICML*, 2020.

[257] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, *et al.*, "On the opportunities and risks of foundation models," *arXiv preprint arXiv:2108.07258*, 2021.

[258] Amir R Zamir, Alexander Sax, William Shen, Leonidas J Guibas, Jitendra Malik, and Silvio Savarese, "Taskonomy: Disentangling task transfer learning," in *CVPR*, 2018.

[259] Amir R Zamir, Alexander Sax, Nikhil Cheerla, Rohan Suri, Zhangjie Cao, Jitendra Malik, and Leonidas J Guibas, "Robust learning through cross-task consistency," in *CVPR*, 2020.